Introduction to R Programming

Laxmikant Soni

2025-09-07

R Overview

R is a comprehensive **statistical** and **graphical** programming language and is a dialect of the **S language**.

History of S Language

- **1988 - S2**: RA Becker, JM Chambers, A Wilks
- **1992 - S3**: JM Chambers, TJ Hastie
- **1998 - S4**: JM Chambers

Development of R

- **Origin**: Initially written by *Ross Ihaka* and *Robert Gentleman* at the Department of Statistics, University of Auckland, New Zealand during the 1990s.
- **Since 1997**: Maintained by an international **R-core team** of 15 people with access to a common CVS archive.

Timeline of S and R

Year	Event
1988	S2 (RA Becker, JM Chambers, A Wilks)
1992	S3 (JM Chambers, TJ Hastie)
1998	S4 (JM Chambers)
1990s	R by Ross Ihaka & Robert Gentleman
1997+	Maintained by R-core team

What is R?

- **Open-source** programming language for statistical computing and graphics.
- Widely used by **statisticians, data scientists, and researchers**.
- Provides a rich ecosystem of **packages and libraries**.
- Highly extensible for **data analysis, visualization, and machine learning**.

Advantages of R

- **Free and Open-source**: No licensing costs, accessible to everyone.
- **Cross-platform**: Works on Windows, macOS, and Linux.
- **Strong Community Support**: Thousands of contributors and active forums.
- **CRAN Repository**: 18,000+ packages for diverse applications.
- **Excellent Visualization**: High-quality graphics with `ggplot2`, `plotly`, etc.
- **Integration**: Can work with Python, C/C++, Java, and databases.

R vs Python

- **R**
  - Best for **statistical analysis** and advanced data visualization.
  - Rich ecosystem for **research, academia, and data exploration**.
  - Strong libraries: `ggplot2`, `dplyr`, `caret`, `shiny`.
- **Python**
  - General-purpose programming with **broader applications** (web, ML, AI).
  - Widely adopted in **industry and production systems**.
  - Strong libraries: `pandas`, `scikit-learn`, `TensorFlow`, `matplotlib`.
- **Takeaway**
  - Use **R** when the focus is on **statistics & visualization**.
  - Use **Python** when the focus is on **scalability, AI, and deployment**.

Limitations of R

- **Memory Intensive**: Stores all objects in memory, may struggle with very large datasets.
- **Speed Issues**: Slower than languages like C++, Java, or even Python in some cases.
- **Steep Learning Curve**: Syntax and advanced features can be challenging for beginners.
- **Less Suitable for Production**: Primarily designed for analysis, not large-scale deployment.
- **Package Overlap**: Thousands of packages, sometimes overlapping in functionality, can cause confusion.

R Resources

- **Official Website**: https://www.r-project.org
- **CRAN (Packages Repository)**: https://cran.r-project.org
- **RStudio IDE**: https://posit.co/download/rstudio-desktop/
- **Documentation & Manuals**:
  https://cran.r-project.org/manuals.html
- **Cheat Sheets**: https://posit.co/resources/cheatsheets/
- **Online Tutorials**:
  - R for Data Science (book)
  - Quick-R
- **Community & Help**:
  - RStudio Community
  - Stack Overflow R Tag

Arithmetic in R

R can perform all basic arithmetic operations:

- **Addition (+)**: `2 + 3` $\rightarrow 5$
- **Subtraction (-)**: `7 - 4` $\rightarrow 3$
- **Multiplication (*)**: `6 * 3` $\rightarrow 18$
- **Division (/)**: `10 / 2` $\rightarrow 5$
- **Exponentiation (^)**: `2 ^ 4` $\rightarrow 16$
- **Modulo (%%)**: `17 %% 3` $\rightarrow 2$ (remainder)
- **Integer Division (%/%)**: `17 %/% 3` $\rightarrow 5$

Example in R

```r
# Basic arithmetic
2 + 3
7 - 4
6 * 3
10 / 2

# Advanced operations
2 ^ 4       # exponentiation
17 %% 3     # modulo
17 %/% 3    # integer division
```

Objects in R

- In R, **everything is an object** (numbers, text, data, functions).
- Objects are created using the **assignment operator** `<-` (or `=`).
- Object names must start with a **letter** (not a number) and can include letters, numbers, `.` or `_`.
- You can check existing objects using `ls()` and remove with `rm()`.

Example: Creating Objects

```r
# Assigning values
x <- 10
y <- 20
name <- "R Programming"

# Using objects
sum <- x + y
sum

# Listing all objects
ls()

# Removing an object
rm(sum)
```

Object Types in R

R supports different **data types**:

- **Numeric**: numbers with or without decimals
  - Example: `x <- 10`, `y <- 3.14`
- **Integer**: whole numbers
  - Example: `z <- 5L`
- **Character (string)**: text values
  - Example: `name <- "R Language"`
- **Logical (boolean)**: TRUE or FALSE
  - Example: `flag <- TRUE`
- **Complex**: numbers with real and imaginary parts
  - Example: `cnum <- 3 + 2i`

Example in R

```r
# Numeric
x <- 10.5
class(x)

# Integer
y <- 10L
class(y)

# Character
name <- "Hello R"
class(name)
```

Example in R

```r
# Logical
flag <- FALSE
class(flag)

# Complex
z <- 2 + 3i
class(z)
```

Why Math in R?

R is built for numerical computing: arithmetic, algebra, optimization, statistics, and simulation.
This note gives a quick, example-driven tour.

Arithmetic Operators in R

Operator	Meaning	Example Code	Result
+	Addition	`5 + 3 * 2`	11
()	Precedence grouping	`(5 + 3) * 2`	16
^	Exponentiation	`2^3^2`	512
%%	Modulus (remainder)	`17 %% 5`	2
%/%	Integer division	`17 %/% 5`	3

Note: *R evaluates exponentiation right-to-left.*
Example: `2^3^2 = 2^(3^2) = 2^9 = 512`.

Rounding & Absolute Values

R provides functions for rounding numbers and working with absolute values:

- `round(x, digits)` : Rounds to the specified number of decimal places
- `floor(x)` : Largest integer less than or equal to x
- `ceiling(x)` : Smallest integer greater than or equal to x
- `trunc(x)` : Truncates toward zero
- `sign(x)` : Returns -1, 0, or 1 depending on the sign of x
- `abs(x)` : Absolute value

Examples

```r
round(3.14159, 3)          # 3.142
floor(2.9)                 # 2
ceiling(2.1)               # 3
trunc(-2.9)                # -2
sign(-10)                  # -1
abs(-7.5)                  # 7.5
```

Variables and Strings in R

In R, variables are used to store data, and strings are text values enclosed in quotes.

Variables in R

- A variable is created when a value is assigned to it.
- Assignment operators: <-, =, and ->

–

Examples

```
x <- 10          # using <-
y = 20           # using =
30 -> z          # using ->

x; y; z
# 10 20 30
```

Variable Naming Rules in R

- Must start with a **letter** or a **dot (.)** not followed by a number.
- Can contain **letters, numbers, dots (.)**, and **underscores (_)**
- R is **case sensitive** $\rightarrow$ `Var` and `var` are different.

Examples

Valid names

```
x
total_sum
.data1
value123
student_name <- "Laxmi"
Student_Name <- "Kag"
student_name == Student_Name
```

Strings in R

Strings are sequences of characters enclosed in either **single** or **double quotes**.

```
str1 <- "Hello"
str2 <- 'World'
paste(str1, str2)
```