Session 1 : Variabls, Datatypes, Arithmetic and Logical Operators, User Input

Jamuna S Murthy
Assistant Professor
Department of CSE

Academic Year - 2025-26

1 What is R?

R is a programming language and environment specifically designed for statistical computing and graphics. It is widely used in data science, statistical modeling, and machine learning.

1.1 Key Features of R:

- Open-source and free.
- Strong statistical analysis capabilities.
- Extensive library of packages.
- Excellent data visualization tools.
- Cross-platform compatibility.

2 Installing R and R Studio

To start working with R, you need to install:

1. **R**: The core programming language. Download from CRAN.
2. **RStudio**: An IDE (Integrated Development Environment) for R. Download from RStudio.

3 RStudio Overview

RStudio has four key panels:

- **Source Panel**: Where you write and edit scripts.
- **Console**: Where R commands are executed.
- **Environment/History**: Stores variables and previous commands.
- **Files/Plots/Packages/Help**: Manages files, graphs, packages, and documentation.

4 Working in the Console

The console allows you to execute R commands directly. You can type a command and press **Enter** to execute it.

```
print("Hello, R!")
```

5 Variables in R

Variables in R are used to store data values. R allows dynamic typing, meaning you do not need to declare the type of a variable explicitly.

5.1 Assigning Values to Variables

In R, variables are assigned using the assignment operator `<-` or `=`.

```
x <- 10   # Assigning value using <-
y = 20    # Assigning value using =
print(x)
print(y)
```

5.2 Variable Naming Rules

- Variable names must start with a letter.
- Can contain letters, numbers, and underscores.
- Cannot contain spaces or special characters.
- Case-sensitive (`myVar` and `myvar` are different).

Example:

```
my_var <- 100   # Valid
_myVar <- 200   # Valid
2var <- 300     # Invalid (cannot start with a number)
```

6 Data Types in R

R has several built-in data types:

6.1 Numeric Data Type

Numeric values include integers and floating-point numbers.

```
a <- 10.5  # Floating-point number
b <- 100   # Integer
print(class(a))  # Output: "numeric"
print(class(b))  # Output: "numeric"
```

6.2 Integer Data Type

To explicitly define an integer, use `L` after the number.

```
c <- 50L   # Integer
print(class(c))  # Output: "integer"
```

6.3 Character (String) Data Type

Character data stores text.

```
name <- "R Programming"
print(class(name))  # Output: "character"
```

6.4 Logical (Boolean) Data Type

Logical values store `TRUE` or `FALSE`.

```
flag <- TRUE
print(class(flag))  # Output: "logical"
```

6.5 Complex Data Type

R supports complex numbers with real and imaginary parts.

```
comp <- 3 + 2i
print(class(comp))  # Output: "complex"
```

6.6 Raw Data Type

Raw data type is rarely used and stores raw bytes.

```
x <- charToRaw("Hello")
print(class(x))  # Output: "raw"
```

7 Checking Data Types

To check the data type of a variable, use the `class()` function.

```
x <- 10
print(class(x))  # Output: "numeric"
```

8 Type Conversion in R

R allows type conversion using functions like `as.numeric()`, `as.character()`, etc.

```
num <- "100"
num <- as.numeric(num)  # Convert string to numeric
print(class(num))  # Output: "numeric"
```

9 Practice Questions

1. Declare three different variables in R (numeric, character, and logical) and print their types.
2. Convert an integer variable into a character and print its type.
3. Take user input and check whether the entered value is numeric, character, or logical.
4. Create a program that assigns a complex number to a variable and prints its type.
5. Write an R script to check if a given variable is of type integer or not.

10 Arithmetic Operators in R

R supports various arithmetic operations:

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
/	Division
^ or **	Exponentiation
%%	Modulus (Remainder)
%/%	Integer Division

Table 1: Arithmetic Operators in R

10.1 Examples

```
# Example 1: Addition
x <- 15
y <- 5
sum <- x + y
print(sum)

# Example 2: Subtraction
diff <- x - y
print(diff)

# Example 3: Multiplication
prod <- x * y
print(prod)

# Example 4: Division
div <- x / y
print(div)

# Example 5: Exponentiation
exp <- x ^ y
print(exp)
```

11 Logical Operations in R

Logical operators compare values and return **TRUE** or **FALSE**.

Operator	Description
==	Equal to
!=	Not equal to
<	Less than
>	Greater than
≤	Less than or equal to
≥	Greater than or equal to
&	Logical AND
|	Logical OR
!	Logical NOT

Table 2: Logical Operators in R

Examples

```
# Example 1: Equality Check
x <- 10
y <- 20
print(x == y)  # FALSE

# Example 2: Inequality Check
print(x != y)  # TRUE

# Example 3: Logical AND
print(x > 5 & y < 25)  # TRUE

# Example 4: Logical OR
print(x > 15 | y < 25)  # TRUE

# Example 5: Logical NOT
print(!TRUE)  # FALSE
```

12 User Input in R

R allows users to take input using the `readline()` function for text-based input and `scan()` for multiple inputs.

Arithmetic Operations Using User Input We can take user input and perform arithmetic operations such as addition, subtraction, multiplication, and division.

Example: Basic Arithmetic Operations

```
# Taking user input for two numbers
num1 <- as.numeric(readline(prompt = "Enter first number: "))
num2 <- as.numeric(readline(prompt = "Enter second number: "))

# Performing arithmetic operations
sum_result <- num1 + num2
sub_result <- num1 - num2
mul_result <- num1 * num2
div_result <- num1 / num2

# Displaying results
print(paste("Sum:", sum_result))
print(paste("Difference:", sub_result))
print(paste("Product:", mul_result))
print(paste("Quotient:", div_result))
```

Practice Questions

1. Write an R script to take two numbers as input and calculate their modulus.
2. Write an R program to take three numbers as input and find their average.
3. Modify the above example to include exponentiation (power operation).
4. Extend the program to take user input for the operation type (addition, subtraction, etc.) and perform the selected operation.
5. Write an R script that takes a number as input and checks whether it is positive, negative, or zero.

Logical Operations Using User Input Logical operations can be performed by taking user input and comparing values.

Example: Checking Conditions Using User Input

```
# Taking user input for two numbers
num1 <- as.numeric(readline(prompt = "Enter first number: "))
num2 <- as.numeric(readline(prompt = "Enter second number: "))
```

```
# Performing logical operations
print(paste("Is num1 equal to num2?", num1 == num2))
print(paste("Is num1 not equal to num2?", num1 != num2))
print(paste("Is num1 greater than num2?", num1 > num2))
print(paste("Is num1 less than or equal to num2?", num1 <= num2))
print(paste("Both numbers are positive:", num1 > 0 & num2 > 0))
print(paste("At least one number is positive:", num1 > 0 | num2 > 0))
```

Practice Questions

1. Write an R script to check whether an entered number is even or odd.
2. Write an R program to check if two entered numbers are both positive, both negative, or mixed.
3. Modify the above example to check whether a number is in the range 1-100.
4. Write an R function that takes two numbers as input and checks if they are both greater than a given threshold.
5. Extend the program to take a third number and check if all three numbers are equal.