R Basics

R is a powerful language and environment for statistical computing and graphics. It is widely used for data analysis, visualization, and statistical modeling. Here's a guide to understanding the basics of R, including its syntax, data types, and fundamental data structures like vectors, lists, and data frames.

1. R Syntax Basics

a. The R Console

- You can run R interactively using the R console or in scripts (`.R` files).
- In the console, you type commands and press Enter to execute them.

b. Comments

- Comments in R start with the `#` symbol. Anything following `#` on the same line is ignored by the interpreter.

r

```r
# This is a comment
```

c. Assignment

- The assignment operator in R is `<-`, though `=` can also be used. However, `<-` is the traditional operator used by R programmers.

r

```r
x <- 5  # Assigns the value 5 to x
y = 10  # Also assigns the value 10 to y
```

d. Printing Values

- You can print values to the console using `print()` or by simply typing the variable name.

r

```r
print(x)  # Prints the value of x
x         # Another way to print the value of x
```

2. Data Types in R

R supports several basic data types:

a. Numeric

- Represents real numbers.

r

```r
num <- 42.5
```

b. Integer

- Represents integer values. Specify integers with an `L` suffix.

r

```r
int <- 42L
```

c. Character

- Represents text or strings.

r

```r
char <- "Hello, R!"
```

d. Logical

- Represents boolean values: `TRUE` or `FALSE`.

r

```r
bool <- TRUE
```

e. Complex

- Represents complex numbers with real and imaginary parts.

r

```r
comp <- 4 + 3i
```

3. Basic Data Structures in R

a. Vectors

- Vectors are the most basic data structures in R. They are sequences of elements of the same type.

Creating Vectors:

r

```r
# Numeric vector
num_vector <- c(1, 2, 3, 4, 5)

# Character vector
char_vector <- c("apple", "banana", "cherry")

# Logical vector
log_vector <- c(TRUE, FALSE, TRUE)
```

Vector Operations:

- Vectors in R support element-wise operations.

r

```r
x <- c(1, 2, 3)
y <- c(4, 5, 6)
```

```
sum_vector <- x + y  # Adds corresponding elements of x and y
```

Accessing Elements:

- Elements in a vector are accessed using square brackets `[]`.

r

```
num_vector[1]  # Accesses the first element of num_vector
```

Vector Functions:

- Common functions that operate on vectors include `length()`, `sum()`, `mean()`, and `min()`.

r

```
length(num_vector)  # Returns the number of elements in
num_vector
mean(num_vector)    # Calculates the average of num_vector
```

b. Lists

- Lists are versatile data structures that can hold elements of different types, including other lists.

Creating Lists:

r

```
my_list <- list(42, "apple", TRUE, c(1, 2, 3))
```

Accessing List Elements:

- Use double square brackets `[[]]` to access individual elements of a list.

r

```r
my_list[[1]]  # Accesses the first element of the list
```

- Use single square brackets `[ ]` to return a subset of the list as a list.

r

```r
my_list[1]  # Returns the first element as a list
```

Named Lists:

- You can name the elements of a list.

r

```r
named_list <- list(age = 25, name = "John", married = FALSE)

# Access by name
named_list$name
named_list[["age"]]
```

c. Data Frames

- Data frames are 2-dimensional, table-like structures where each column can contain different types of data. Data frames are a fundamental data structure for most data analysis in R.

Creating Data Frames:

r

```r
df <- data.frame(
  Name = c("Alice", "Bob", "Charlie"),
  Age = c(25, 30, 35),
  Married = c(TRUE, FALSE, TRUE)
)
```

Accessing Data Frame Elements:

- Use the `$` operator to access columns by name.

r

```
df$Name  # Accesses the "Name" column
```

- Use square brackets for more general indexing.

r

```
df[1, ]    # Accesses the first row
df[, "Age"]  # Accesses the "Age" column
df[2, 3]  # Accesses the element at the second row and third
column
```

Adding and Removing Columns:

- You can add new columns to a data frame.

r

```
df$Height <- c(160, 175, 180)  # Adds a new column "Height"
```

- You can remove columns by setting them to `NULL`.

r

```
df$Height <- NULL  # Removes the "Height" column
```

Basic Data Frame Operations:

- `head(df)` and `tail(df)` return the first and last few rows of the data frame.
- `dim(df)` returns the dimensions of the data frame (rows and columns).
- `summary(df)` provides a summary of each column in the data frame.

Summary

- **R Syntax**: Learn basic syntax like assignments (`<-`), comments (`#`), and how to run R code.
- **Data Types**: Understand the basic data types: numeric, integer, character, logical, and complex.
- **Vectors**: Learn how to create and manipulate vectors, which are sequences of elements of the same type.
- **Lists**: Use lists to store collections of elements of different types.
- **Data Frames**: Utilize data frames for table-like data structures where each column can contain different types.