Functions in R Programming

Introduction

Functions in R are a fundamental part of programming, enabling code reuse, modularity, and efficiency. Functions help encapsulate operations, making code easier to read and maintain.

Defining a Function

A function in R is created using the function keyword. The basic syntax is:

```
function_name <- function(arg1, arg2, ...) {
  # Function body
  return(value)
}
```

Example

```
add_numbers <- function(a, b) {
  result <- a + b
  return(result)
}


# Calling the function
sum_value <- add_numbers(5, 7)
print(sum_value)  # Output: 12
```

Types of Functions

1. **Built-in Functions**: R provides a wide range of built-in functions, such as sum(), mean(), sd(), and length().
    1. x <- c(1, 2, 3, 4, 5)
    2. mean_value <- mean(x)
    3. print(mean_value) # Output: 3
2. **User-defined Functions**: Custom functions created by the user for specific tasks.
3. **Anonymous Functions (Lambda Functions)**: Functions without a name, often used within apply() family functions.
    1. (function(x) x^2)(4) # Output: 16

Function Arguments

Functions in R can take various types of arguments:

- **Required Arguments**: Must be provided.
- **Default Arguments**: Assigned default values.
- **Variable Arguments**: ... allows passing multiple arguments.

Example with default arguments:

```
power_function <- function(x, power=2) {
  return(x^power)
}
print(power_function(3))  # Output: 9 (default power=2)
print(power_function(3, 3))  # Output: 27
```

Scope of Variables

R has two types of variable scopes:

- **Local Scope**: Variables defined within a function are not accessible outside.
- **Global Scope**: Variables defined outside functions are accessible globally.

Example:

```
my_function <- function() {
  local_var <- 10
  return(local_var)
}
print(my_function())  # Output: 10
print(local_var)  # Error: object 'local_var' not found
```

Recursive Functions

A function can call itself, useful for tasks like computing factorial.

```
factorial_func <- function(n) {
  if (n == 0) return(1)
  return(n * factorial_func(n - 1))
}
print(factorial_func(5))  # Output: 120
```

Problems to Solve

1. **Write a function in R that calculates the Fibonacci sequence up to a given number n.**
2. **Create a function that takes a vector and returns the sum of its squares.**
3. **Write a function that checks if a number is prime.**
4. **Implement a function that normalizes a numeric vector (scales values between 0 and 1).**
5. **Write a recursive function to compute the greatest common divisor (GCD) of two numbers.**