Lecture 5: Introduction to Functions in R

Dr. Logan Kelly

2024-09-04

Overview

- In this lecture, we'll explore:
  - What **functions** are and how to use them.
  - How to **write your own functions** in R.
  - The importance of **comments** in code to improve readability.

1. What are Functions in R?

- **Functions** in R are blocks of code designed to perform specific tasks.
- They can take **input**, process the data, and return a **result**.
- R includes many **built-in functions** that you can use right away, such as `mean()`, `sum()`, and `sqrt()`.
- **Example: Using the `mean()` function**:

```r
numbers <- c(1, 2, 3, 4, 5)  # Create a vector of numbers
mean(numbers)  # Calculate the mean of the numbers
```

```
[1] 3
```

- Functions can take **arguments** (input values), perform an operation, and then return a result.

2. Writing Your Own Functions in R

- **Why write your own functions?**
  - Custom functions allow you to encapsulate a series of commands into a reusable block of code.
  - They make your code **modular**, **efficient**, and easier to maintain.
- **Structure of a function in R**:
  - Functions in R follow this structure:

```r
function_name <- function(argument1, argument2, ...) {
    # Code that performs some task
    result <- argument1 + argument2  # Example of an operation
    return(result)  # Return the result
}
```

- **Example: Creating a function to add two numbers**:

```r
add_numbers <- function(a, b) {
    sum <- a + b  # Add the two numbers
    return(sum)  # Return the result
}
```

- **Using the function**:

```r
add_numbers(3, 5)  # Call the function with two arguments
```

```
[1] 8
```

- **Explanation**:
  - The function `add_numbers()` takes two inputs, `a` and `b`.
  - Inside the function, the numbers are added together, and the result is returned.
  - The function can now be reused whenever you need to add two numbers.

3. Adding Comments to Your Code

- **Why are comments important?**
  - Comments are notes you add to your code to **explain** what it's doing.
  - They make your code easier to understand, especially when sharing with others or revisiting it after some time.
- **How to add comments in R**:
  - Use the `#` symbol to add a comment. Everything after `#` on the same line is ignored by R.
- **Example: Adding comments to a function**:

```
# This function adds two numbers together
add_numbers <- function(a, b) {
    sum <- a + b  # Add the two input numbers
    return(sum)  # Return the result
}

# Call the function with arguments 3 and 5
add_numbers(3, 5)
```

```
[1] 8
```

- **Best practices for commenting**:
  - Add comments to **describe the purpose** of your function.
  - Use comments to explain **complex logic** or calculations.
  - Don't over-comment; focus on clarity.

4. Returning Values from Functions

- Functions in R often **return** a value after performing their operations.
- The `return()` statement is used to **output** the result from a function.
- If no `return()` is specified, R automatically returns the last expression evaluated.
- **Example: A function that multiplies two numbers**:

```
multiply_numbers <- function(x, y) {
    product <- x * y  # Multiply the inputs
    return(product)  # Return the result
}
```

- **Using the function**:

```r
multiply_numbers(4, 6)
```

```
[1] 24
```

Key Takeaways

- You've learned how to use built-in functions like `mean()` and how to create your own custom functions in R.
- Writing custom functions helps make your code modular and reusable.
- Comments in R are essential for making your code easier to understand and maintain.

Looking Forward

- In the next lecture, we'll explore **working with vectors and variables** in more detail, including how to manipulate and index data in R.