Lecture 10: Programming in R – Scripts, Loops, and Conditional Statements

Dr. Logan Kelly

2024-09-04

Overview

- In this lecture, we'll cover:
  - How to write and run **R scripts**.
  - Using **loops** to automate repetitive tasks.
  - Applying **conditional statements** to control the flow of your programs.
  - Writing **custom functions** for reusable code.

1. Writing and Running R Scripts

- **What is an R script?**
  - An **R script** is a file containing a series of R commands that you can run together. Scripts are useful for automating tasks and organizing your code.
- **How to create a new script**:
  - In RStudio, navigate to `File -> New File -> R Script`. This opens a new script file where you can write and save your code.
- **Running a script**:
  - You can run individual lines of code in a script by selecting the line and pressing `Ctrl + Enter` (Windows) or `Cmd + Enter` (Mac).
  - To run the entire script, click on the **Source** button or use `Ctrl + Shift + Enter`.
- **Example: A simple R script**:

```r
# This is a simple script that calculates the square of a number
number <- 5
square <- number^2
print(square)  # Output the result
```

```
[1] 25
```

- **Saving the script**:
  - Save your script by going to `File -> Save As` and choosing a file name with the `.R` extension (e.g., `my_script.R`).

2. Using Loops in R

- **What is a loop?**
  - A **loop** allows you to repeat a block of code multiple times. Loops are useful for automating repetitive tasks.
- **For loop**:
  - The **for loop** iterates over a sequence of values, running the code inside the loop for each value.
- **Example: A simple for loop**:

```r
# A for loop that prints numbers from 1 to 5
for (i in 1:5) {
  print(i)
}
```

```
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
```

- **While loop**:
  - The **while loop** repeats a block of code as long as a specified condition is `TRUE`.
- **Example: A simple while loop**:

```r
# A while loop that prints numbers from 1 to 5
i <- 1
while (i <= 5) {
  print(i)
  i <- i + 1  # Increment the value of i
}
```

```
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
```

3. Using Conditional Statements in R

- **What is a conditional statement?**
  - A **conditional statement** allows you to execute certain blocks of code based on whether a condition is `TRUE` or `FALSE`.
- **If-else statements**:
  - The **if-else** statement runs code when a condition is true and optionally runs different code when the condition is false.
- **Example: Using an if-else statement**:

```r
x <- 10

if (x > 5) {
  print("x is greater than 5")
} else {
  print("x is less than or equal to 5")
}
```

```
[1] "x is greater than 5"
```

- **Nested if-else statements**:
  - You can nest multiple if-else statements to check for multiple conditions.
- **Example: Checking multiple conditions**:

```r
x <- 10

if (x > 15) {
  print("x is greater than 15")
} else if (x > 5) {
  print("x is greater than 5 but less than or equal to 15")
} else {
  print("x is 5 or less")
}
```

```
[1] "x is greater than 5 but less than or equal to 15"
```

4. Writing Custom Functions

- **Why write custom functions?**
  - Writing your own **functions** allows you to organize and reuse code. A function can accept input, process the data, and return a result.
- **Structure of a function**:
  - The general structure of a function in R is:

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

- **Example: Writing a custom function**:

```r
# A function to calculate the square of a number
square <- function(x) {
  return(x^2)  # Return the square of the input
}
```

- **Calling the function**:

```r
square(4)  # Call the function with input 4
```

```
[1] 16
```

- **Using multiple arguments**:
  - Functions can take multiple arguments, and you can define default values for these arguments.
- **Example: A function with multiple arguments**:

```r
add_numbers <- function(a = 1, b = 2) {
  return(a + b)
}

add_numbers(3, 5)  # Call the function with specific values for a and b
```

```
[1] 8
```

Key Takeaways

- You can write and run **R scripts** to organize and automate tasks.
- **Loops** and **conditional statements** allow you to control the flow of your code and automate repetitive tasks.
- Writing **custom functions** helps make your code more modular and reusable.

Looking Forward

- In the next lecture, we'll dive into **more advanced programming techniques** in R, including working with **apply functions** and exploring **vectorized operations** for efficient coding.