Lecture 6: Working with Vectors and Variables in R

Dr. Logan Kelly

2024-09-04

Overview

- In this lecture, we'll explore:
  - How to create and work with **vectors** in R.
  - Storing different types of **variables** (numeric, character, logical).
  - **Indexing** vectors to access or modify specific elements.

1. Creating Vectors in R

- **What is a vector?**
  - A **vector** is one of the most fundamental data structures in R.
  - A vector stores a collection of elements that are all of the same data type (e.g., all numbers or all characters).
- **How to create a vector**:
  - You can create a vector using the `c()` function, which stands for "combine."
- **Example: Creating a numeric vector**:

```
numbers <- c(10, 20, 30, 40, 50)  # Create a vector of numbers
```

- **Example: Creating a character vector**:

```
names <- c("Alice", "Bob", "Charlie", "David")  # Create a vector of names
```

- **Example: Creating a logical vector**:

```
logical_vector <- c(TRUE, FALSE, TRUE, TRUE)  # Create a vector of logical values
```

2. Storing Different Types of Variables

- **What is a variable in R?**
  - A **variable** is a storage location where you can keep a value (e.g., a number, a string of text, or a logical value).
- **Types of variables**:
  - **Numeric**: Numbers (e.g., `10`, `25.5`).
  - **Character**: Text strings (e.g., `"Alice"`, `"Data"`).
  - **Logical**: Boolean values, `TRUE` or `FALSE`.
- **Example: Storing a number as a variable**:

```
x <- 10  # Store the number 10 in the variable 'x'
```

- **Example: Storing text as a variable**:

```
name <- "Alice"  # Store the name 'Alice' in the variable 'name'
```

- **Example: Storing a logical value**:

```
is_active <- TRUE  # Store the value TRUE in the variable 'is_active'
```

3. Indexing Vectors in R

- **What is indexing?**
  - **Indexing** allows you to access or modify specific elements in a vector.
  - In R, indexing starts at **1**, unlike many programming languages where indexing starts at **0**.
- **Accessing elements in a vector**:
  - Use square brackets `[]` to access elements at specific positions in a vector.
- **Example: Accessing the first element of a vector**:

```
numbers <- c(10, 20, 30, 40, 50)  # Create a numeric vector
numbers[1]  # Access the first element (10)
```

```
[1] 10
```

- **Example: Accessing multiple elements**:

```r
numbers[2:4]  # Access elements from the second to the fourth position (20, 30, 40)
```

```
[1] 20 30 40
```

- **Modifying elements in a vector**:
  - You can **replace** elements in a vector by assigning new values to specific positions.
- **Example: Changing the third element**:

```r
numbers[3] <- 35  # Change the third element to 35
```

4. Performing Operations on Vectors

- **Element-wise operations**:
  - R can perform operations on **each element** of a vector automatically.
- **Example: Adding 10 to each element of a vector**:

```r
numbers <- c(10, 20, 30, 40, 50)
numbers + 10  # Add 10 to each element in the vector
```

```
[1] 20 30 40 50 60
```

- **Vector arithmetic**:
  - You can also perform arithmetic between two vectors of the same length.
- **Example: Adding two vectors**:

```r
numbers1 <- c(1, 2, 3)
numbers2 <- c(4, 5, 6)
result <- numbers1 + numbers2  # Add corresponding elements (1+4, 2+5, 3+6)
```

Key Takeaways

- You've learned how to create **vectors** in R and store different types of **variables**.
- **Indexing** allows you to access and modify specific elements within a vector.
- R performs operations on vectors in an **element-wise** manner, making it easy to manipulate data.

Looking Forward

- In the next lecture, we'll explore **data frames and lists**, two more advanced data structures that allow you to work with larger and more complex datasets in R.