Lecture 4: Introduction to Data Structures in R

Dr. Logan Kelly

2024-09-04

Overview

- In this lecture, we'll cover:
  - An overview of **data structures** in R.
  - How to create and manipulate **vectors** in R.
  - Practical examples of accessing data within **vectors**.
  - Introduction to other data structures: **matrices**, **data frames**, **lists**, and **factors**.

1. Overview of Data Structures in R

R provides several **data structures** that are used to store, manage, and manipulate data efficiently. Each structure has its specific use case and characteristics:

- **Vectors**: One-dimensional arrays containing elements of the same type.
- **Matrices**: Two-dimensional arrays where each element is of the same type.
- **Data Frames**: Two-dimensional structures where each column can have different types of data (similar to a spreadsheet).
- **Lists**: Collections of elements that can hold different types of objects.
- **Factors**: Specialized vectors for storing categorical data.

Choosing the appropriate data structure is crucial for effective analysis, as each structure is suited for different types of operations and datasets.

2. Vectors in R

What is a Vector?

- A **vector** is the simplest and most common data structure in R. It is a one-dimensional array that holds data of a single type, such as numeric, character, or logical.

Creating a Vector

You can create a vector using the `c()` function, which stands for "combine" or "concatenate." This function allows you to combine multiple elements into a single vector.

Example: Creating a Numeric Vector

```
# Creating a numeric vector
sales_vector <- c(120, 150, 90, 100, 130, 170, 200)
sales_vector
```

```
[1] 120 150  90 100 130 170 200
```

- **Explanation**: This code creates a numeric vector containing sales data. The `c()` function combines the sales figures into a single vector called `sales_vector`.

Example: Creating a Character Vector

```
# Creating a character vector
product_vector <- c("Product A", "Product B", "Product C", "Product D")
product_vector
```

```
[1] "Product A" "Product B" "Product C" "Product D"
```

- **Explanation**: This example creates a vector of product names using the `c()` function to combine the names into a single character vector.

Accessing Elements in a Vector

You can access individual elements in a vector using **indexing**. R indexing starts at **1**, meaning the first element of the vector is accessed with `vector_name[1]`.

Example: Accessing the First Element of a Vector

```
# Accessing the first element of the sales vector
sales_vector[1]  # Returns 120
```

```
[1] 120
```

- **Explanation**: This code retrieves the first element of the `sales_vector`, which is the value `120`.

Example: Accessing Multiple Elements

You can access multiple elements of a vector using a range of indices.

```
# Accessing the first three elements of the sales vector
sales_vector[1:3]  # Returns 120, 150, and 90
```

```
[1] 120 150  90
```

- **Explanation**: The range `1:3` retrieves the first three elements of the `sales_vector`, returning the sales figures `120`, `150`, and `90`.

Modifying Elements in a Vector

You can modify specific elements of a vector by assigning new values to a specific index.

Example: Modifying the Second Element of a Vector

```
# Modifying the second element of the sales vector
sales_vector[2] <- 160
sales_vector
```

```
[1] 120 160  90 100 130 170 200
```

- **Explanation**: This example updates the second element of the `sales_vector` from `150` to `160`.

3. Introduction to Other Data Structures in R

Matrices

- A **matrix** is a two-dimensional array where each element is of the same type. Matrices are useful for mathematical operations across rows and columns.
- **Example**: Creating a matrix of sales data for different products across different quarters.

```
# Creating a numeric matrix
sales_matrix <- matrix(c(120, 150, 90, 100, 130, 170, 200, 210, 180),
                       nrow = 3,
                       ncol = 3,
                       byrow = TRUE)
sales_matrix
```

```
     [,1] [,2] [,3]
[1,]  120  150   90
[2,]  100  130  170
[3,]  200  210  180
```

- **Explanation**: This matrix contains sales data for three products across three quarters. Each row represents a different product, and each column represents a quarter.

Data Frames

- **Data frames** are two-dimensional structures that allow you to store data of different types (e.g., numeric and character) in columns. This is one of the most common data structures in R for business datasets.

```
# Creating a data frame
sales_data <- data.frame(
  Product = c("A", "B", "C"),
  Sales_Q1 = c(120, 150, 90),
  Sales_Q2 = c(170, 200, 140)
)
sales_data
```

```
  Product Sales_Q1 Sales_Q2
1       A      120      170
2       B      150      200
3       C       90      140
```

- **Explanation**: This data frame contains sales data for three products (`A`, `B`, and `C`) across two quarters (`Q1` and `Q2`).

Lists

- A **list** can contain elements of different types, including vectors, matrices, and even other lists. Lists are highly flexible and can be used to store complex data structures.

```
# Creating a list
sales_list <- list(
  Products = c("A", "B", "C"),
  Sales = sales_vector,
  Sales_Matrix = sales_matrix
)
sales_list
```

```
$Products
[1] "A" "B" "C"

$Sales
[1] 120 160  90 100 130 170 200

$Sales_Matrix
     [,1] [,2] [,3]
[1,]  120  150   90
[2,]  100  130  170
[3,]  200  210  180
```

- **Explanation**: This list contains three elements: a vector of product names, a vector of sales, and a matrix of sales data.

Factors

- **Factors** are used for storing categorical data, such as product categories or customer satisfaction levels. They ensure proper handling of categorical variables in statistical models.

```
# Creating a factor
satisfaction <- factor(c("High", "Medium", "Low", "Medium", "High"))
satisfaction
```

```
[1] High   Medium Low    Medium High
Levels: High Low Medium
```

- **Explanation**: This factor represents customer satisfaction levels, with three categories: "High", "Medium", and "Low".

Key Takeaways

- **Vectors** are the simplest data structures in R, storing data of a single type. You can easily access and modify elements using indexing.
- **Matrices**, **data frames**, **lists**, and **factors** allow you to work with more complex data structures.
- You now know how to create and work with basic data structures in R.

Looking Forward

- In the next lecture, we'll dive deeper into working with **matrices** in R, including creating, accessing, and manipulating matrices for business analysis.