Lecture 7: Working with Data Frames and Lists in R

Dr. Logan Kelly

2024-09-04

Overview

- In this lecture, we'll explore:
  - How to create and work with **data frames**, one of R's most important data structures.
  - How to use **lists** to store complex data types.
  - Practical examples of **manipulating** and **accessing** data within these structures.

1. Introduction to Data Frames

- **What is a data frame?**
  - A **data frame** is a table-like structure where each column is a **vector** that holds data of the same type (e.g., numeric, character), but different columns can store different types of data.
  - It is one of the most commonly used data structures in R for storing datasets.
- **How to create a data frame**:
  - You can create a data frame using the `data.frame()` function.
- **Example: Creating a data frame**:

```r
df <- data.frame(
  name = c("Alice", "Bob", "Charlie"),  # Character vector
  age = c(25, 30, 35),  # Numeric vector
  salary = c(50000, 60000, 70000)  # Numeric vector
)
```

- **Accessing columns in a data frame**:
  - You can access a specific column using the `$` operator.
- **Example: Accessing the `name` column**:

```
df$name
```

```
[1] "Alice"   "Bob"     "Charlie"
```

- **Adding a new column to a data frame**:
  - You can add a new column to a data frame by creating a new variable and assigning it to the data frame.
- **Example: Adding a `department` column**:

```
df$department <- c("HR", "Finance", "IT")
```

2. Introduction to Lists

- **What is a list?**
  - A **list** is a versatile data structure in R that can store elements of **different types** and sizes, such as vectors, data frames, or even other lists.
  - Lists allow you to group related objects together, even if they are of different types.
- **How to create a list**:
  - Use the `list()` function to create a list.
- **Example: Creating a list**:

```
my_list <- list(
  name = "Alice",
  age = 25,
  salary = 50000,
  skills = c("R", "Python", "SQL")
)
```

- **Accessing elements in a list**:
  - You can access elements in a list using double square brackets `[[ ]]` or by using the `$` operator if the elements are named.
- **Example: Accessing the `name` element**:

```r
my_list$name
```

```
[1] "Alice"
```

- **Example: Accessing the `skills` vector**:

```r
my_list$skills
```

```
[1] "R"      "Python" "SQL"
```

3. Manipulating Data Frames

- **Filtering rows in a data frame**:
  - You can **filter** rows in a data frame by specifying conditions.
- **Example: Filtering rows where `age` is greater than 25**:

```r
df[df$age > 25, ]
```

```
     name age salary department
2     Bob  30  60000    Finance
3 Charlie  35  70000         IT
```

- **Selecting specific columns**:
  - You can select specific columns from a data frame by specifying their names in square brackets.
- **Example: Selecting the `name` and `salary` columns**:

```r
df[, c("name", "salary")]
```

```
     name salary
1   Alice  50000
2     Bob  60000
3 Charlie  70000
```

4. Manipulating Lists

- **Adding new elements to a list**:
  - You can add new elements to a list by assigning them to a new slot.
- **Example: Adding a new `department` element**:

```
my_list$department <- "HR"
```

- **Modifying elements in a list**:
  - You can modify existing elements by assigning new values to them.
- **Example: Changing the `salary` element**:

```
my_list$salary <- 55000
```

5. Combining Data Frames and Lists

- **Lists of data frames**:
  - You can store multiple data frames in a list, allowing you to work with complex, multi-dimensional data.
- **Example: Creating a list of data frames**:

```
df1 <- data.frame(name = c("Alice", "Bob"), age = c(25, 30))
df2 <- data.frame(name = c("Charlie", "David"), age = c(35, 40))

data_list <- list(df1, df2)
```

- **Accessing a data frame from the list**:

```
data_list[[1]]  # Access the first data frame in the list
```

```
   name age
1 Alice  25
2   Bob  30
```

Key Takeaways

- Data frames are essential for organizing and analyzing **tabular data** in R, allowing you to store data in columns of different types.
- Lists are powerful structures for storing complex and heterogeneous data, allowing you to group different types of objects.
- You've learned how to manipulate and access both **data frames** and **lists** in R.

Looking Forward

- In the next lecture, we'll explore **packages and comments** in R, focusing on how to install, manage, and utilize packages for extended functionality, and how to properly document your code using comments.