Lecture 6: Working with Data Frames in R

Dr. Logan Kelly

2024-09-04

Overview

- In this lecture, we'll cover:
  - Introduction to **data frames**, a widely used data structure in R.
  - How to create, access, and modify **data frames**.
  - Practical examples of working with **business datasets** in data frames.

1. Introduction to Data Frames

- A **data frame** is a two-dimensional data structure in R where:
  - Each **column** represents a different variable, and columns can store different data types (numeric, character, logical, etc.).
  - Each **row** represents an observation or record.
- Data frames are similar to tables in a spreadsheet or SQL database, making them ideal for business data analysis.

Key Uses of Data Frames:

- **Organizing datasets** for analysis.
- **Storing business data** such as sales records, employee details, customer information, etc.
- **Data cleaning and transformation** for analysis.

2. Creating Data Frames

You can create a data frame using the `data.frame()` function, where each argument represents a column in the data frame.

Example: Creating a Sales Data Frame

```
# Creating a data frame for sales data
sales_data <- data.frame(
  Product = c("A", "B", "C", "D", "E"),
  Sales_Q1 = c(120, 150, 90, 100, 130),
  Sales_Q2 = c(170, 200, 140, 180, 190)
)
sales_data
```

```
  Product Sales_Q1 Sales_Q2
1       A      120      170
2       B      150      200
3       C       90      140
4       D      100      180
5       E      130      190
```

- **Explanation**: In this example, `sales_data` is a data frame where:
  - The `Product` column contains product names.
  - `Sales_Q1` and `Sales_Q2` columns contain sales figures for the first and second quarters.

Checking the Structure of a Data Frame

You can inspect the structure of a data frame using the `str()` function, which shows the types of each column.

```
str(sales_data)
```

```
'data.frame':   5 obs. of  3 variables:
 $ Product : chr  "A" "B" "C" "D" ...
 $ Sales_Q1: num  120 150 90 100 130
 $ Sales_Q2: num  170 200 140 180 190
```

- **Explanation**: This code reveals the structure of `sales_data`, showing that `Product` is a character column, while `Sales_Q1` and `Sales_Q2` are numeric columns.

3. Accessing Data in a Data Frame

Accessing Columns in a Data Frame

You can access individual columns in a data frame using the `$` operator or square brackets.

Example: Accessing a Single Column

```
# Accessing the Sales_Q1 column
sales_data$Sales_Q1
```

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

- **Explanation**: This retrieves the `Sales_Q1` column from the `sales_data` data frame.

Example: Accessing Multiple Columns

```
# Accessing the Product and Sales_Q2 columns
sales_data[, c("Product", "Sales_Q2")]
```

```
  Product Sales_Q2
1       A      170
2       B      200
3       C      140
4       D      180
5       E      190
```

- **Explanation**: This extracts the `Product` and `Sales_Q2` columns, showing both in the output.

Accessing Rows in a Data Frame

You can access rows by specifying row indices using square brackets `[row, ]`.

Example: Accessing a Specific Row

```
# Accessing the first row of the data frame
sales_data[1, ]
```

```
  Product Sales_Q1 Sales_Q2
1       A      120      170
```

- **Explanation**: This code retrieves the first row, showing all the details for Product `A` in the first quarter (`Q1`) and second quarter (`Q2`).

4. Modifying Data in a Data Frame

You can update specific elements, rows, or columns of a data frame by assigning new values.

Example: Modifying an Element in a Data Frame

```
# Changing the sales value for Product B in Q2
sales_data[2, "Sales_Q2"] <- 210
sales_data
```

```
  Product Sales_Q1 Sales_Q2
1       A      120      170
2       B      150      210
3       C       90      140
4       D      100      180
5       E      130      190
```

- **Explanation**: This updates the sales figure for Product `B` in the second quarter from 200 to 210.

Example: Adding a New Column

```
# Adding a new column for Sales in Q3
sales_data$Sales_Q3 <- c(190, 220, 150, 200, 210)
sales_data
```

```
  Product Sales_Q1 Sales_Q2 Sales_Q3
1       A      120      170      190
2       B      150      210      220
3       C       90      140      150
4       D      100      180      200
5       E      130      190      210
```

- **Explanation**: This adds a new column, `Sales_Q3`, to the `sales_data` data frame with sales figures for the third quarter.

5. Practical Example: Analyzing Sales Data

Let's calculate the **total sales** across all three quarters for each product by creating a new column.

Example: Calculating Total Sales

```
# Creating a new column for total sales across all quarters
sales_data$Total_Sales <- sales_data$Sales_Q1 + sales_data$Sales_Q2 + sales_data$Sales_Q3
sales_data
```

```
  Product Sales_Q1 Sales_Q2 Sales_Q3 Total_Sales
1       A      120      170      190         480
2       B      150      210      220         580
3       C       90      140      150         380
4       D      100      180      200         480
5       E      130      190      210         530
```

- **Explanation**: This code adds a new column `Total_Sales` that sums up sales from `Sales_Q1`, `Sales_Q2`, and `Sales_Q3` for each product.

6. Sorting and Filtering Data in a Data Frame

Sorting Data Frames

You can sort a data frame by one or more columns using the `order()` function.

Example: Sorting by Total Sales in Descending Order

```r
# Sorting the data frame by Total_Sales in descending order
sales_data_sorted <- sales_data[order(-sales_data$Total_Sales), ]
sales_data_sorted
```

```
  Product Sales_Q1 Sales_Q2 Sales_Q3 Total_Sales
2       B      150      210      220         580
5       E      130      190      210         530
1       A      120      170      190         480
4       D      100      180      200         480
3       C       90      140      150         380
```

- **Explanation**: This code sorts the `sales_data` by `Total_Sales` in descending order, with the highest sales appearing first.

Filtering Data Frames

You can filter rows of a data frame based on specific conditions.

Example: Filtering Products with Sales Greater Than 150 in Q1

```r
# Filtering products with Sales_Q1 greater than 150
high_sales_q1 <- sales_data[sales_data$Sales_Q1 > 150, ]
high_sales_q1
```

```
[1] Product     Sales_Q1    Sales_Q2    Sales_Q3    Total_Sales
<0 rows> (or 0-length row.names)
```

- **Explanation**: This code filters the `sales_data` to show only the products with sales greater than `150` in the first quarter.

Key Takeaways

- **Data frames** are one of the most important data structures in R for handling datasets with different types of data in columns.
- You can easily **create**, **access**, **modify**, **sort**, and **filter** data frames in R.
- Data frames are widely used in business analytics for **organizing**, **analyzing**, and **transforming** data.

Looking Forward

- In the next lecture, we'll explore **lists** and **factors**, two other important data structures in R, and how to work with categorical data and complex collections of data.