# Lecture 5: Working with Matrices in R

Dr. Logan Kelly

2024-09-04

### Overview

- In this lecture, we'll cover:
  - Introduction to **matrices** in R.
  - How to create and manipulate **matrices** for data analysis.
  - Accessing and modifying matrix elements.
  - Practical examples of using matrices for business analytics.

## 1. Introduction to Matrices

- A **matrix** is a two-dimensional array in R where all elements must be of the same type (e.g., numeric, character, or logical).
- Matrices are particularly useful for storing and performing mathematical operations on data structured in rows and columns.
- Each element in a matrix can be accessed by specifying its **row** and **column** indices.

#### Example: Creating a Matrix

You can create a matrix using the `matrix()` function, where you define the elements, number of rows (`nrow`), and number of columns (`ncol`).

#### Example: Creating a Sales Matrix

```
# Creating a numeric matrix for sales data
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 code creates a matrix representing sales data for three products across three quarters. The argument `byrow = TRUE` fills the matrix by rows.

#### Key Matrix Operations:

- **Dimensions**: Matrices have a specific number of rows and columns. The `dim()` function returns the dimensions of a matrix.
- **Rows and Columns**: You can extract specific rows or columns from a matrix for analysis.

```
dim(sales_matrix)  # Check the dimensions of the matrix (3 rows, 3 columns)
```

```
[1] 3 3
```

### 2. Accessing Elements in a Matrix

You can access elements within a matrix by specifying their row and column indices using square brackets `[row, column]`.

#### Example: Accessing an Element in the Matrix

```
# Accessing the element in the first row, second column
element <- sales_matrix[1, 2]  # Returns 150
element
```

```
[1] 150
```

- **Explanation**: This code retrieves the element located in the first row and second column of the matrix, which corresponds to the value `150`.

#### Example: Accessing an Entire Row or Column

```
# Accessing the first row of the matrix
first_row <- sales_matrix[1, ]
first_row
```

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

```
# Accessing the second column of the matrix
second_column <- sales_matrix[, 2]
second_column
```

```
[1] 150 130 210
```

- **Explanation**: The first line retrieves the entire first row, while the second line retrieves the entire second column. This is useful for analyzing specific rows or columns in the data.

### 3. Modifying Matrix Elements

Matrix elements can be updated by assigning new values to specific positions using the same row and column indexing.

#### Example: Modifying a Matrix Element

```
# Changing the value of the element in the second row, third column
sales_matrix[2, 3] <- 180
sales_matrix
```

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

- **Explanation**: This example updates the value at the second row, third column of the matrix from 170 to 180.

### 4. Matrix Arithmetic Operations

Matrices support various arithmetic operations such as addition, subtraction, multiplication, and division. These operations are performed element-wise.

### Example: Adding Two Matrices

```
# Creating a second matrix for addition
additional_sales <- matrix(c(10, 15, 5, 10, 20, 15, 10, 5, 20),
                           nrow = 3,
                           ncol = 3)

# Adding two matrices
total_sales <- sales_matrix + additional_sales
total_sales
```

```
     [,1] [,2] [,3]
[1,]  130  160  100
[2,]  115  150  185
[3,]  205  225  200
```

- **Explanation**: This example adds the `sales_matrix` and `additional_sales` matrices element-wise, resulting in a new matrix `total_sales` with the combined sales data.

### Example: Matrix Multiplication

```
# Matrix multiplication (element-wise multiplication)
sales_growth <- sales_matrix * 1.1  # Increase each value by 10%
sales_growth
```

```
     [,1] [,2] [,3]
[1,]  132  165   99
[2,]  110  143  198
[3,]  220  231  198
```

- **Explanation**: This example multiplies each element in the `sales_matrix` by `1.1`, increasing sales by 10%. Matrix multiplication is useful when applying transformations or adjustments across all elements.

## 5. Practical Applications of Matrices in Business Analytics

Matrices are often used in business analytics to represent: - **Sales data**: Organizing data for multiple products across different time periods. - **Cost and revenue analysis**: Using matrices to compute profits by subtracting costs from revenue data. - **Forecasting**: Applying growth rates or trends to matrices for future projections.

**Example: Computing Profit from Sales and Cost Matrices**

```
# Defining a cost matrix
cost_matrix <- matrix(c(70, 90, 50, 60, 80, 120, 150, 160, 140),
                      nrow = 3,
                      ncol = 3)

# Calculating profit by subtracting cost from sales
profit_matrix <- sales_matrix - cost_matrix
profit_matrix
```

```
     [,1] [,2] [,3]
[1,]   50   90  -60
[2,]   10   50   20
[3,]  150   90   40
```

- **Explanation**: This code calculates profit by subtracting a cost matrix from the sales matrix, resulting in a matrix of profits for each product across different periods.

## Key Takeaways

- **Matrices** are two-dimensional arrays that hold elements of the same type and are useful for performing mathematical operations across rows and columns.
- You can create, access, and modify **matrix elements** in R using indexing.
- **Matrix arithmetic** (e.g., addition and multiplication) allows for efficient calculations, especially when working with datasets like sales or revenue.

## Looking Forward

- In the next lecture, we'll dive into working with **data frames**, one of the most common and flexible data structures in R, allowing you to handle datasets with different types of data in each column.