# Module (3)

## Course Module Introduction: Arrays and Matrices in R

### Introduction

This module builds upon the previous module on vectors, introducing the next level of complexity in R data structures: arrays and matrices. Specifically, it focuses on matrices, which are two-dimensional vectors. Mastering matrices will pave the way for understanding and working with higher-dimensional arrays.

### Key Topics

1. **Creating matrices:** Different methods for constructing matrices in R.
2. **Operations on matrices:** Mathematical and statistical operations applicable to matrices.
3. **Combining matrices:** Merging matrices using various techniques.
4. **Plotting matrices:** Visualizing matrix data for analysis and interpretation.

### Upcoming Topics

- Introduction to matrices
- Matrix creation
- Matrix operations
- Combining matrices
- Plotting matrices

This module promises to be a comprehensive guide to working with matrices in R, equipping you with essential skills for data analysis and manipulation.

## Lecture Notes: Introduction to Matrices in R

### Matrices in R

- **Matrices** are two-dimensional arrays with data of a single type (homogeneous).
- They are a special case of **arrays**, which can have multiple dimensions.
- Matrices are built upon **atomic vectors**, the fundamental building block of R data structures.

### Key Concepts

- **Homogeneous:** All elements within a matrix must be of the same data type (e.g., numeric, character).
- **Length:** The total number of elements in a matrix.
- **Dimensions:** The size of the matrix, specified as (rows, columns).
- **Dim Function:** The `dim()` function is used to get the dimensions of a matrix.

### Example: 3 x 4 Matrix with Missing Values

Code snippet

```
matrix(data = NA, nrow = 3, ncol = 4)


     [,1] [,2] [,3] [,4]
[1,]   NA   NA   NA   NA
[2,]   NA   NA   NA   NA
[3,]   NA   NA   NA   NA
```

### Key Points

- A matrix is a rectangular arrangement of elements with rows and columns.
- Matrices have two dimensions, unlike vectors, which are one-dimensional.
- The `dim()` function provides information about the number of rows and columns in a matrix.

### Upcoming Topics

- Different ways to create matrices in R.
- Operations that can be performed on matrices.
- Combining matrices using various techniques.
- Visualizing matrix data through plotting.

## Lecture Notes: Creating Matrices in R

### Creating Matrices

Matrices in R are two-dimensional arrays with elements of the same data type. The `matrix()` function is the primary way to create matrices.

### The `matrix()` Function

Code snippet

```
matrix(data, nrow, ncol, byrow = FALSE, dimnames = NULL)
```

- **data:** The input vector of values to fill the matrix.
- **nrow:** The number of rows in the matrix.
- **ncol:** The number of columns in the matrix.
- **byrow:** A logical value indicating whether to fill the matrix by rows (TRUE) or by columns (FALSE). Default is FALSE (fill by columns).
- **dimnames:** A list of optional row and column names.

## Examples

Code snippet

```
# Create a 2x5 matrix filled by columns with values 1-10
m <- matrix(1:10, nrow = 2, ncol = 5)

# Create a 2x10 matrix filled with 99
matrix(99, nrow = 2, ncol = 10)

# Create a logical matrix filled with TRUE values
d <- matrix(TRUE, nrow = 2, ncol = 5)
```

## Matrix Attributes and Functions

- `dim()`: Returns the dimensions of a matrix (number of rows and columns).
- `nrow()`: Returns the number of rows.
- `ncol()`: Returns the number of columns.
- `length()`: Returns the total number of elements.
- `as.vector()`: Converts a matrix into a one-dimensional vector.

## Example: Converting Matrix to Vector and Back

Code snippet

```
v <- as.vector(m)
m_reconstructed <- matrix(v, nrow = 2, ncol = 5)
```

## Creating Matrices of Specific Types

- **Numeric:** Use numeric values in the `data` argument of the `matrix()` function.
- **Integer:** Use integer values (e.g., `1L`, `2L`) in the `data` argument.

- **Character:** Use character values (e.g., `"a"`, `"b"`) in the `data` argument.
- **Logical:** Use logical values (`TRUE`, `FALSE`) in the `data` argument.

### Checking Matrix Type

The `is.*()` functions can be used to check the type of a matrix (e.g., `is.numeric()`, `is.character()`, `is.logical()`).

### Understanding `byrow`

- `byrow = FALSE` (Default): Fills the matrix column by column.
- `byrow = TRUE`: Fills the matrix row by row.

### Key Points

- Matrices can be created using the `matrix()` function.
- The `byrow` argument controls the filling order of the matrix.
- Matrices can be converted to vectors and back using `as.vector()` and `matrix()`.
- You can create matrices of different data types using appropriate values in the `data` argument.

### Upcoming Topics

- Matrix functions for manipulation and analysis.

# Lecture Notes: Matrix Functions in R

## Matrix Functions in R

R provides a variety of functions for working with matrices, enabling you to perform calculations, extract information, and manipulate the data.

### Key Points

- Matrices are based on vectors, so you can use vector functions on matrices as well.
- R has specific functions tailored for matrix operations.
- Understanding matrix functions is crucial for data analysis and manipulation.

### Examples

Code snippet

```
m <- matrix(1:10, nrow = 2, ncol = 5)
log(m)    # Calculate logarithms of each element
all(m > 0)  # Check if all elements are positive (TRUE)
```

### Accessing and Modifying Matrix Elements

Code snippet

```
m[1, 1]   # Access element in the 1st row, 1st column
m[1, 3] <- 6  # Modify element in the 1st row, 3rd column
```

### `head()` Function

- `head(m)`: Displays the entire matrix (or the first few rows if it's large).
- `head(m, n)`: Displays the first `n` rows of the matrix.

### `summary()` Function

- Provides summary statistics for each column of the matrix, including minimum, first quartile, median, mean, third quartile, and maximum.
- Useful for quickly understanding the distribution of data in each column.

### Upcoming Topics

- More advanced matrix functions.
- Special functions for matrices in R.
- Applications of matrices in data analysis.

### Additional Notes

- Explore the R documentation for a comprehensive list of matrix functions and their usage.
- Experiment with different functions to gain hands-on experience with matrix manipulation in R.

# Lecture Notes: Mathematical Operations on Matrices in R

## Mathematical Operations on Matrices

Matrices in R can be used for various arithmetic operations, such as solving linear equations and other mathematical problems. All operations applicable to vectors can also be applied to matrices.

## Scalar Operations

You can perform arithmetic operations between a matrix and a scalar (single value). The scalar is applied to each element of the matrix.

Code snippet

```
x <- matrix(1:4, nrow = 2, ncol = 2)
x + 2
#     [,1] [,2]
#[1,]    3    5
#[2,]    4    6
x * 3
#     [,1] [,2]
#[1,]    3    9
#[2,]    6   12
```

## Matrix-Vector Multiplication

You can multiply a matrix by a vector, resulting in a new vector.

Code snippet

```
y <- c(5, 20)
x * y  # Element-wise multiplication of each column of x by y
#     [,1] [,2]
#[1,]    5   60
#[2,]   10   80
```

## Matrix-Matrix Multiplication

You can multiply two matrices if they have compatible dimensions (the number of columns in the first matrix must equal the number of rows in the second matrix).

Code snippet

```
z <- matrix(rep(y, each = 2), nrow = 2) # Create a matrix by repeating the vector y
x * z  # Same as x * y
```

## Element-wise Matrix Operations

You can perform arithmetic operations between two matrices of the same dimensions element-wise.

Code snippet

```
y <- matrix(10:13, nrow = 2, ncol = 2)
x + y
#     [,1] [,2]
#[1,]   11   33
#[2,]   14   44
x / y
#      [,1] [,2]
#[1,]  0.1  0.1
#[2,]  0.1  0.1
```

## Other Matrix Functions

- **`t(x)`**: Transposes the matrix `x`.
- **`diag(x)`**: Extracts the diagonal elements of the matrix `x`.
- **`det(x)`**: Calculates the determinant of the matrix `x`.

## Examples

Code snippet

```
t(x)   # Transpose of x
diag(x) # Diagonal elements of x
det(x)  # Determinant of x
```

## Key Points

- Matrices can be used in various arithmetic operations.
- Operations can be performed between matrices and scalars, vectors, or other matrices.
- R provides functions for common matrix operations like transpose, diagonal extraction, and determinant calculation.
- Understanding matrix operations is crucial for data analysis and solving mathematical problems in R.

## Upcoming Topics

- More advanced matrix operations.

- Applications of matrices in data analysis.

# Lecture Notes: Matrix Attributes in R

## Matrix Attributes

Matrices in R have both mandatory and optional attributes. These attributes provide metadata about the matrix structure and data types.

## Mandatory Attributes

- **`class`**: Indicates the class of the object (e.g., "matrix", "array").
- **`dim`**: Specifies the dimensions of the matrix as a two-element integer vector (number of rows, number of columns).
- **`length`**: The total number of elements in the matrix.

## Checking Attributes

- **`attributes()`**: Returns a list of all attributes of a matrix.
- **`dim()`**: Returns the dimensions of a matrix.
- **`nrow()`**: Returns the number of rows.
- **`ncol()`**: Returns the number of columns.
- **`length()`**: Returns the total number of elements.
- **`class()`**: Returns the class of the matrix (e.g., "matrix", "array").
- **`typeof()`**: Returns the internal storage mode of the matrix (more specific than `class()`).

## Examples

Code snippet

```
x <- matrix(NA, nrow = 2, ncol = 10)
attributes(x)  # Output: $dim [1]  2 10

# Creating a vector with the same dimensions
v <- c(nrow(x), ncol(x))
attributes(v)  # Output: $names [1] "nrow(x)" "ncol(x)"
```

## Data Types and Classes

- The `class()` function primarily reveals the matrix structure (e.g., "matrix", "array").
- For vectors, `class()` indicates the data type (e.g., "numeric", "integer").

Code snippet

```
class(x)     # Output: "matrix"   "array"
typeof(x)    # Output: "integer"

v <- c(1, 2)
class(v)     # Output: "numeric"
typeof(v)    # Output: "double"
```

## Row and Column Names

- **`rownames()`**: Retrieves or sets the names of the rows in a matrix.
- **`colnames()`**: Retrieves or sets the names of the columns in a matrix.

Code snippet

```
rownames(x) <- c("Row1", "Row2")
colnames(x) <- paste0("Col", 1:10) # Assign "Col1", "Col2", ... "Col10" as column names

# Changing column names
colnames(x)[2] <- "Coursera"
```

## Creating Named Matrices

You can create matrices with row and column names directly using the `dimnames` argument of the `matrix()` function.

## Key Points

- Matrix attributes provide metadata about the matrix, including dimensions, data types, and names.
- You can manipulate and access attributes using various functions like `attributes()`, `dim()`, `rownames()`, and `colnames()`.
- Understanding matrix attributes is essential for effective data manipulation and analysis in R.

## Upcoming Topics

- Combining different objects in R.

# Lecture Notes: Combining Objects in R

## Combining Objects in R

R allows combining vectors and matrices to create new matrices, either row-wise or column-wise.

## Row-wise Binding (`rbind()`)

The `rbind()` function combines vectors or matrices by rows.

Code snippet

```
x <- c(5, 5, 5)
y <- c(11, 22, 33)
z <- rbind(x, y)

# Output:
# [,1] [,2] [,3]
# x    5    5    5
# y   11   22   33
```

## Column-wise Binding (`cbind()`)

The `cbind()` function combines vectors or matrices by columns.

Code snippet

```
u <- cbind(x, y)
# Output:
#      x  y
# [1,] 5 11
# [2,] 5 22
# [3,] 5 33
```

## Assigning Dimension Names

You can assign names to rows or columns while using `rbind()` or `cbind()`, respectively.

Code snippet

```
par_age <- c(45, 38)
par_height <- c(1.78, 1.65)
par <- rbind(age = par_age, height = par_height)
print(par)
# Output:
```

```
#        [,1]  [,2]
# age    45.00 38.00
# height  1.78  1.65
```

## Combining Matrices

You can also combine matrices row-wise or column-wise, as long as the dimensions are compatible.

Code snippet

```
x1 <- matrix(1:4, nrow = 2)
x2 <- matrix(c(1,0,1,1,0,2), nrow = 2)
cbind(x1, x2)  # Column-wise binding of matrices
#Output:
#     [,1] [,2] [,3]
#[1,]    1    3    1
#[2,]    2    4    0
rbind(x1[,1], x2[,1])  # Row-wise binding of specific columns
#Output:
#     [,1]
#[1,]    1
#[2,]    2
#[3,]    1
#[4,]    1
```

## Combining Vectors and Matrices

You can combine vectors and matrices using `rbind()` or `cbind()`. Ensure the dimensions are compatible for successful binding.

Code snippet

```
# Example (try at home)
x <- matrix(1:4, nrow = 2)
y <- c(5, 6)
rbind(x, y)
```

## Key Points

- `rbind()` and `cbind()` are versatile functions for combining objects in R.
- You can bind vectors and matrices, but ensure the dimensions match for row-

wise and column-wise operations.
- Assigning dimension names can improve the readability and interpretation of the resulting matrix.

# Lecture Notes: Subsetting Matrices in R

## Subsetting Matrices

Subsetting allows you to extract specific elements or groups of elements from a matrix. Matrices are two-dimensional, so you need to specify both row and column indices when subsetting.

## Matrix Structure

```
     [,1] [,2] [,3] [,4]
[1,] x[1,1] x[1,2] x[1,3] x[1,4]
[2,] x[2,1] x[2,2] x[2,3] x[2,4]
[3,] x[3,1] x[3,2] x[3,3] x[3,4]
```

## Creating a Matrix for Visualization

Code snippet

```
x <- matrix(sprintf("x[%d,%d]", row(matrix(NA, 3, 4)), col(matrix(NA, 3, 4))), 3, 4)
print(x)
#Output:
     [,1]    [,2]    [,3]    [,4]
[1,] "x[1,1]" "x[1,2]" "x[1,3]" "x[1,4]"
[2,] "x[2,1]" "x[2,2]" "x[2,3]" "x[2,4]"
[3,] "x[3,1]" "x[3,2]" "x[3,3]" "x[3,4]"
```

## Subsetting by Index

Code snippet

```
x[3, 1]     # Extract element at row 3, column 1
x[3,]       # Extract the entire 3rd row
x[,3]       # Extract the entire 3rd column
x[c(4, 2), 2]   # Extract elements at (row 4, column 2) and (row 2, column 2)
x[1, 1:3]    # Extract elements from row 1, columns 1 to 3
x[c(2,3),1:2] # Extract a block of elements from rows 2 and 3, columns 1 and 2
```

## Subsetting with Names

Code snippet

```
medals <- matrix(c(4,3,2,1,2,5), nrow = 3,
                 dimnames = list(c("Canada", "US", "England"),
                                 c("Gold", "Silver")))
medals["Canada", "Gold"]  # Extract the value for Canada's gold medals
medals["England", ] # Extract all medal values for England (as a named vector)
medals["England", , drop = FALSE] # Extract all medal values for England (as a matrix)
```

## Logical Vector Subsetting

Code snippet

```
gold_greater_than_2 <- medals[,"Gold"] > 2
medals[gold_greater_than_2,]
```

## Search and Replace using Logical Vectors

Code snippet

```
x <- matrix(runif(6), nrow = 3)
x[x > 0.5] <- 999
```

## Mixing Subsetting Methods

Code snippet

```
medals[1, "Silver"]
```

## Error Handling

Accessing undefined elements will result in an error.

## Sorting Matrices

- You can sort a matrix by a specific row or column using `sort()`.
- The `order()` function can be used to obtain the order of elements for sorting.

## Summary Table of Subsetting Methods

| Method | Description | Example |
|---|---|---|
| Index | Numerical position of elements (starts at 1) | `x[row, column]` |
| Logical Vector | TRUE/FALSE vector to select elements | `x[logical_vector]` |
| Names | Using names of elements in named matrices | `x["rowname", "colname"]` |
| Negative Index | Exclude elements by index | `x[-1, ]` |
| Combinations | Combine different methods | `x[1, c("colname1", "colname2")]` |
| `which()` Function | Find indices of elements meeting a condition | `which(x > 5)` |
| Double Square Brackets (`[[]]`) | Extract a single element directly (no vector or name returned) | `x[[1]]` |

# Lecture Notes: Plotting Matrices in R

## Plotting Matrices

R offers several functions to visualize matrix data, aiding in understanding and interpreting the underlying patterns and relationships.

## Plotting Functions for Matrices

- **`plot()`**: Generic function for creating scatterplots or line plots. Can plot two columns of a matrix by default.
- **`matplot()`**: Plots the columns of a matrix against their row indices, with each column having a different color and line type.
- **`image()`**: Displays a matrix as an image, where the values are represented by colors.
- **`contour()`**: Creates a 2D contour plot, showing lines connecting points of equal value.
- **`filled.contour()`**: Generates a level plot by filling the areas between contours with colors.

## Examples

Code snippet

```
x <- seq(0, 2*pi, length.out = 200)
m <- cbind(sin(x), sin(x + pi))
```

```
plot(m)  # Scatterplot of the two columns
matplot(x, m, type = "l", lty = 1, col = c("blue", "red")) # Line plot of both columns

# Adding legend
legend("topright", legend = c("sin(x)", "sin(x + pi)"), col = c("blue", "red"), lty = 1)

# Plotting image
image(volcano)
```

## Customizing Plots

- Plots can be customized with various arguments like `main` for title, `xlab` and `ylab` for axis labels, `col` for colors, and `type` for plot type.
- Explore R documentation for detailed options and customization possibilities for each plotting function.

## Key Points

- R provides a versatile set of functions for plotting matrix data.
- `plot()` and `matplot()` are useful for visualizing the relationships between columns or variables.
- `image()`, `contour()`, and `filled.contour()` are effective for displaying two-dimensional data.
- Visualizing matrices can reveal patterns, trends, and anomalies in the data, aiding in analysis and interpretation.

# Course Module Summary: Arrays and Matrices in R

## Summary

This module began by building upon the concept of vectors, the fundamental data structure in R, and expanded to explore the world of matrices – two-dimensional arrays with a wide range of applications in data analysis and manipulation.

## Key Takeaways

- **Matrices** are essential tools in R for organizing and working with structured data.
- The **matrix function** allows for flexible creation of matrices with various dimensions and data types.
- Matrices can be manipulated using a variety of **functions and operations**, including mathematical calculations, subsetting, and plotting.

- Understanding matrix attributes, such as **dimensions and data types**, is crucial for effective data handling.

## Next Steps

The knowledge gained in this module sets the stage for delving deeper into R programming. The upcoming module will focus on functions, a fundamental aspect of R that allows for modular and reusable code.

Continue practicing with matrices, experimenting with different operations and visualizations. As you explore further, you'll discover the immense potential of matrices for solving complex problems and gaining valuable insights from your data.