Session 3 : Vectors, Data Frames, Factors, Sorting Numeric, Character, and Factor Vectors, Special Values in R

Jamuna S Murthy
Assistant Professor
Department of CSE

Academic Year - 2025-26

1 Vectors in R

A **vector** is a basic data structure in R that contains elements of the same type.

1.1 Creating Vectors

Vectors can be created using the `c()` function.

```r
# Numeric Vector
num_vector <- c(10, 20, 30, 40)

# Character Vector
char_vector <- c("apple", "banana", "cherry")

# Logical Vector
log_vector <- c(TRUE, FALSE, TRUE)
```

1.2 Sequence Vectors

We can generate a sequence of numbers using `:` or `seq()`.

```r
# Sequence using colon (:)
seq_vec1 <- 1:10  # Generates 1 to 10

# Sequence using seq()
seq_vec2 <- seq(from = 1, to = 10, by = 2)  # Generates 1, 3, 5, 7, 9
```

1.3 Repeating Elements

To create repeated elements, use `rep()`.

```
# Repeat a single number
rep_vec1 <- rep(5, times = 4)   # Output: 5 5 5 5

# Repeat a sequence
rep_vec2 <- rep(c(1, 2), times = 3)  # Output: 1 2 1 2 1 2

# Repeat each element a specific number of times
rep_vec3 <- rep(c(1, 2), each = 3)  # Output: 1 1 1 2 2 2
```

1.4 Accessing Elements in Vectors

You can access elements using **indexing** (1-based indexing in R).

```
vec <- c(10, 20, 30, 40, 50)

# Access the 3rd element
print(vec[3])  # Output: 30

# Access multiple elements
print(vec[c(2, 4)])  # Output: 20 40

# Exclude an element using negative index
print(vec[-1])  # Output: 20 30 40 50
```

1.5 Vector Operations

Vectorized operations apply to all elements in the vector.

```
x <- c(1, 2, 3)
y <- c(4, 5, 6)

# Element-wise Addition
z <- x + y  # Output: 5 7 9

# Element-wise Multiplication
w <- x * y  # Output: 4 10 18
```

1.6 Logical Operations on Vectors

Logical comparisons return a vector of Boolean values.

```
nums <- c(10, 20, 30, 40, 50)

# Check which elements are greater than 25
result <- nums > 25
print(result)  # Output: FALSE FALSE TRUE TRUE TRUE

# Filter elements based on a condition
filtered <- nums[nums > 25]
print(filtered)  # Output: 30 40 50
```

1.7 Handling Missing Values (NA)

Missing values (NA) can appear in vectors.

```r
vec_with_na <- c(1, 2, NA, 4, 5)

# Check for NA values
print(is.na(vec_with_na))  # Output: FALSE FALSE TRUE FALSE FALSE

# Remove NA values
clean_vec <- vec_with_na[!is.na(vec_with_na)]
print(clean_vec)  # Output: 1 2 4 5
```

1.8 Sorting a Vector

Sorting can be done using `sort()`.

```r
num_vec <- c(4, 1, 8, 2)

# Ascending Order
sorted_vec <- sort(num_vec)  # Output: 1 2 4 8

# Descending Order
sorted_desc <- sort(num_vec, decreasing = TRUE)  # Output: 8 4 2 1
```

1.9 Vector Length and Type

You can check the length and type of a vector.

```r
vec <- c(1, 2, 3, 4)

# Get the length of the vector
print(length(vec))  # Output: 4

# Check the type of vector
print(typeof(vec))  # Output: "double"
```

1.10 Converting Between Types

Use `as.numeric()`, `as.character()`, or `as.logical()`.

```r
char_vec <- c("1", "2", "3")

# Convert to Numeric
num_vec <- as.numeric(char_vec)  # Output: 1 2 3
```

Practice Questions

1. Create a numeric vector with 10 elements and sort it in descending order.
2. Create a character vector containing names of 5 cities and sort it alphabetically.
3. Generate a vector with numbers from 1 to 100 but only selecting multiples of 5.
4. Create a logical vector based on whether numbers in a given vector are greater than 50.
5. Create a vector with missing values (`NA`) and replace them with zero.

2 Factors in R

Factors are used to represent categorical data and store it efficiently.

2.1 Creating Factor Vectors

```r
# Creating a factor vector
categories <- factor(c("Low", "Medium", "High", "Medium", "Low"))
print(categories)

# Checking the levels
print(levels(categories))

table(categories)  # Count of each category
```

2.2 Ordered Factors

Ordered factors are useful when categorical data has a meaningful order.

```r
# Creating an ordered factor
ordered_categories <- factor(c("Low", "Medium", "High", "Medium", "Low"),
                             levels = c("Low", "Medium", "High"),
                             ordered = TRUE)
print(ordered_categories)
```

3 Special Values in R

R includes several special values for handling undefined or missing data.

3.1 NA (Not Available)

Represents missing values in vectors and data frames.

```r
data <- c(10, 20, NA, 40, 50)
print(is.na(data))  # TRUE for missing values
```

3.2 NaN (Not a Number)

Occurs when performing undefined mathematical operations.

```r
result <- 0 / 0  # NaN
print(is.nan(result))
```

3.3 Inf and -Inf (Infinity)

Represents positive and negative infinity.

```r
inf_value <- 1 / 0  # Inf
neg_inf <- -1 / 0  # -Inf
print(inf_value)
print(neg_inf)
```

3.4 Removing Special Values

You can remove NA, NaN, and Inf from vectors.

```r
data <- c(10, 20, NA, NaN, Inf, 30)
clean_data <- data[!is.na(data) & !is.nan(data) & is.finite(data)]
print(clean_data)  # Output: 10 20 30
```

Practice Questions

1. Create a factor vector representing colors (Red, Blue, Green, Red, Blue) and display its levels.

2. Convert an ordered factor for education levels (High School, Bachelor's, Master's, PhD) and print it.

3. Create a vector containing special values (NA, NaN, Inf) and filter out non-finite values.

4. Given a categorical dataset, convert it into a factor and display a frequency table.

5. Implement a script to replace NA values in a numeric vector with the mean of the available values.

Solutions

1. Create a numeric vector with 10 elements and sort it in descending order.

```r
numeric_vector <- c(12, 45, 3, 67, 34, 89, 23, 56, 78, 90)
sorted_vector <- sort(numeric_vector, decreasing = TRUE)
print(sorted_vector)
```

2. Create a character vector containing names of 5 cities and sort it alphabetically.

```r
cities <- c("New␣York", "Paris", "London", "Tokyo", "Sydney")
sorted_cities <- sort(cities)
print(sorted_cities)
```

3. Generate a vector with numbers from 1 to 100 but only selecting multiples of 5.

```r
multiples_of_five <- seq(from = 5, to = 100, by = 5)
print(multiples_of_five)
```

4. Create a logical vector based on whether numbers in a given vector are greater than 50.

```r
num_vector <- c(12, 65, 23, 89, 45, 90, 34, 56, 78, 10)
logical_vector <- num_vector > 50
print(logical_vector)
```

5. Create a vector with missing values (NA) and replace them with zero.

```r
data <- c(10, 20, NA, 40, NA, 50)
data[is.na(data)] <- 0
print(data)
```

6. Create a factor vector representing colors (Red, Blue, Green, Red, Blue) and display its levels.

```r
colors <- factor(c("Red", "Blue", "Green", "Red", "Blue"))
print(levels(colors))
```

7. Convert an ordered factor for education levels (High School, Bachelor's, Master's, PhD) and print it.

```r
education_levels <- factor(c("High␣School", "Bachelor's", "Master's", "PhD", "Bachelor's"),
                           levels = c("High␣School", "Bachelor's", "Master's", "PhD"),
                           ordered = TRUE)
print(education_levels)
```

8. Create a vector containing special values (NA, NaN, Inf) and filter out non-finite values.

```r
special_values <- c(10, 20, NA, NaN, Inf, 30)
filtered_values <- special_values[!is.na(special_values) & !is.nan(special_values) & is.finite(special_values)]
print(filtered_values)
```

9. Given a categorical dataset, convert it into a factor and display a frequency table.

```r
categories <- factor(c("Apple", "Banana", "Apple", "Cherry", "Banana", "Cherry", "Apple"))
print(table(categories))
```

10. Implement a script to replace NA values in a numeric vector with the mean of the available values.

```r
data <- c(10, 20, NA, 40, 50, NA)
mean_value <- mean(data, na.rm = TRUE)
data[is.na(data)] <- mean_value
print(data)
```