

Lab Program - 2

R V Abhishek

2025-08-20

Lab-Program 2: Creating and Manipulating Data Structures

This program evaluates the student's understanding of different data structures (vectors, matrices, lists, and data frames) in R and how to manipulate them.

1. Create a vector of random numbers and apply operations such as sorting and searching

```
set.seed(42) # For reproducibility
random_vector <- runif(20, min = 1, max = 100) # Vector of 20 random numbers between 1 and 100
cat("Original random vector:\n", random_vector, "\n")
```

Original random vector:

91.5658 93.77047 29.32781 83.21431 64.53281 52.3905 73.92224 14.33199 66.04224 70.80141 46.31644 72.19211 93.53255 94.06144 97.84442 12.63125 14.33199 26.28745 29.32781 46.31644 46.76699 48.02471 52.3905 56.47294 64.53281 66.04224 70.80141

Sort the vector

```
sorted_vector <- sort(random_vector)
cat("Sorted vector:\n", sorted_vector, "\n")
```

Sorted vector:

12.63125 14.33199 26.28745 29.32781 46.31644 46.76699 48.02471 52.3905 56.47294 64.53281 66.04224 70.80141 72.19211 93.53255 94.06144 97.84442

Search for a specific value (check if a number is present)

```
search_value <- 50
is_value_present <- any(random_vector == search_value)
cat("Is", search_value, "present in the vector?", is_value_present, "\n")
```

Is 50 present in the vector? FALSE

Find values in the vector greater than 60

```
values_greater_than_60 <- random_vector[random_vector > 60]
cat("Values greater than 60:\n", values_greater_than_60, "\n")
```

Values greater than 60:

91.5658 93.77047 83.21431 64.53281 73.92224 66.04224 70.80141 72.19211 93.53255 94.06144 97.84442

2. Convert the vector into a matrix and perform matrix multiplication

Create a 4x5 matrix from the vector

```
matrix_from_vector <- matrix(random_vector, nrow = 4, ncol = 5)
cat("Matrix from vector:\n")
```

Matrix from vector:

```
print(matrix_from_vector)
```

```
      [,1]      [,2]      [,3]      [,4]      [,5]
[1,] 91.56580 64.53281 66.04224 93.53255 97.84442
[2,] 93.77047 52.39050 70.80141 26.28745 12.63125
[3,] 29.32781 73.92224 46.31644 46.76699 48.02471
[4,] 83.21431 14.33199 72.19211 94.06144 56.47294
```

```
# Perform matrix multiplication (matrix with its transpose)
matrix_transpose <- t(matrix_from_vector)
matrix_multiplication_result <- matrix_from_vector %*% matrix_transpose
cat("Matrix multiplication result:\n")
```

Matrix multiplication result:

```
print(matrix_multiplication_result)
```

```
      [,1]      [,2]      [,3]      [,4]
[1,] 35232.22 20337.59 19587.86 27635.57
[2,] 20337.59 17401.08 11738.17 16851.17
[3,] 19587.86 11738.17 12963.36 13954.70
[4,] 27635.57 16851.17 13954.70 24378.48
```

```
# Element-wise matrix multiplication (Hadamard product)
elementwise_multiplication_result <- matrix_from_vector * matrix_from_vector
cat("Element-wise matrix multiplication result:\n")
```

Element-wise matrix multiplication result:

```
print(elementwise_multiplication_result)
```

```
      [,1]      [,2]      [,3]      [,4]      [,5]
[1,] 8384.2954 4164.483 4361.577 8748.3384 9573.5298
[2,] 8792.9003 2744.764 5012.840 691.0302 159.5484
[3,] 860.1207 5464.498 2145.212 2187.1513 2306.3729
[4,] 6924.6222 205.406 5211.701 8847.5541 3189.1932
```

```
# 3. Create a list containing different types of elements and perform subsetting
```

```
my_list <- list(
  numbers = random_vector,
  characters = c("A", "B", "C", "D"),
  logical_values = c(TRUE, FALSE, TRUE),
  matrix = matrix_from_vector
)
cat("List:\n")
```

List:

```
print(my_list)
```

```
$numbers
```

```
[1] 91.56580 93.77047 29.32781 83.21431 64.53281 52.39050 73.92224 14.33199  
[9] 66.04224 70.80141 46.31644 72.19211 93.53255 26.28745 46.76699 94.06144  
[17] 97.84442 12.63125 48.02471 56.47294
```

```
$characters
```

```
[1] "A" "B" "C" "D"
```

```
$logical_values
```

```
[1] TRUE FALSE TRUE
```

```
$matrix
```

```
      [,1]      [,2]      [,3]      [,4]      [,5]  
[1,] 91.56580 64.53281 66.04224 93.53255 97.84442  
[2,] 93.77047 52.39050 70.80141 26.28745 12.63125  
[3,] 29.32781 73.92224 46.31644 46.76699 48.02471  
[4,] 83.21431 14.33199 72.19211 94.06144 56.47294
```

```
# Subsetting the list (extracting numeric and logical parts)
```

```
subset_numeric <- my_list$numbers
```

```
cat("Subset (numeric part of the list):\n", subset_numeric, "\n")
```

```
Subset (numeric part of the list):
```

```
91.5658 93.77047 29.32781 83.21431 64.53281 52.3905 73.92224 14.33199 66.04224 70.80141 46.31644 72.19211
```

```
subset_logical <- my_list$logical_values
```

```
cat("Subset (logical part of the list):\n", subset_logical, "\n")
```

```
Subset (logical part of the list):
```

```
TRUE FALSE TRUE
```

```
# Modify elements in the list (replace the second character with "Z")
```

```
my_list$characters[2] <- "Z"
```

```
cat("Modified list of characters:\n", my_list$characters, "\n")
```

```
Modified list of characters:
```

```
A Z C D
```

```
# Apply a function to the numeric part of the list (e.g., calculate the square of the numbers)
```

```
squared_numbers <- my_list$numbers ^ 2
```

```
cat("Squared numbers:\n", squared_numbers, "\n")
```

```
Squared numbers:
```

```
8384.295 8792.9 860.1207 6924.622 4164.483 2744.764 5464.498 205.406 4361.577 5012.84 2145.212 5211.70
```

4. Create a data frame and perform operations such as filtering, summarizing, and handling missing values
Create a data frame

```
df <- data.frame(  
  ID = 1:20,  
  Age = sample(18:65, 20, replace = TRUE),  
  Score = runif(20, min = 50, max = 100),  
  Passed = sample(c(TRUE, FALSE), 20, replace = TRUE)  
)  
cat("Data frame:\n")
```

Data frame:

```
print(df)
```

	ID	Age	Score	Passed
1	1	64	71.78858	FALSE
2	2	20	51.87155	FALSE
3	3	58	98.67700	FALSE
4	4	42	71.58756	FALSE
5	5	44	97.87883	FALSE
6	6	53	94.38775	FALSE
7	7	54	81.99894	TRUE
8	8	48	98.54833	FALSE
9	9	62	80.94191	TRUE
10	10	22	66.67136	FALSE
11	11	37	67.33741	TRUE
12	12	51	69.92427	FALSE
13	13	45	89.23464	FALSE
14	14	57	51.94682	FALSE
15	15	20	87.43977	FALSE
16	16	50	83.86384	TRUE
17	17	59	58.56322	TRUE
18	18	41	63.05440	TRUE
19	19	47	75.72065	TRUE
20	20	60	83.78036	FALSE

Filter the data frame (rows where Age > 30 and Score > 70)

```
filtered_df <- subset(df, Age > 30 & Score > 70)  
cat("Filtered data frame (Age > 30 and Score > 70):\n")
```

Filtered data frame (Age > 30 and Score > 70):

```
print(filtered_df)
```

	ID	Age	Score	Passed
1	1	64	71.78858	FALSE
3	3	58	98.67700	FALSE
4	4	42	71.58756	FALSE
5	5	44	97.87883	FALSE
6	6	53	94.38775	FALSE
7	7	54	81.99894	TRUE

```

8 8 48 98.54833 FALSE
9 9 62 80.94191 TRUE
13 13 45 89.23464 FALSE
16 16 50 83.86384 TRUE
19 19 47 75.72065 TRUE
20 20 60 83.78036 FALSE

```

```

# Calculate mean, sum, and variance of numerical columns (Age and Score)
mean_age <- mean(df$Age)
sum_age <- sum(df$Age)
var_age <- var(df$Age)

mean_score <- mean(df$Score)
sum_score <- sum(df$Score)
var_score <- var(df$Score)

cat("Summary statistics for Age column:\n")

```

Summary statistics for Age column:

```
cat("Mean Age:", mean_age, "\n")
```

Mean Age: 46.7

```
cat("Sum of Age:", sum_age, "\n")
```

Sum of Age: 934

```
cat("Variance of Age:", var_age, "\n")
```

Variance of Age: 179.6947

```
cat("Summary statistics for Score column:\n")
```

Summary statistics for Score column:

```
cat("Mean Score:", mean_score, "\n")
```

Mean Score: 77.26086

```
cat("Sum of Score:", sum_score, "\n")
```

Sum of Score: 1545.217

```
cat("Variance of Score:", var_score, "\n")
```

Variance of Score: 219.2162

```
# 5. Handling missing values in the data frame
```

```
# Introduce some NA values in the Score column
```

```
df$Score[sample(1:20, 5)] <- NA
```

```
cat("Data frame with missing values:\n")
```

Data frame with missing values:

```
print(df)
```

	ID	Age	Score	Passed
1	1	64	71.78858	FALSE
2	2	20	51.87155	FALSE
3	3	58	98.67700	FALSE
4	4	42	NA	FALSE
5	5	44	97.87883	FALSE
6	6	53	94.38775	FALSE
7	7	54	81.99894	TRUE
8	8	48	98.54833	FALSE
9	9	62	NA	TRUE
10	10	22	NA	FALSE
11	11	37	67.33741	TRUE
12	12	51	NA	FALSE
13	13	45	NA	FALSE
14	14	57	51.94682	FALSE
15	15	20	87.43977	FALSE
16	16	50	83.86384	TRUE
17	17	59	58.56322	TRUE
18	18	41	63.05440	TRUE
19	19	47	75.72065	TRUE
20	20	60	83.78036	FALSE

```
# Replace NA values with the mean of the Score column
```

```
df$Score[is.na(df$Score)] <- mean(df$Score, na.rm = TRUE)
```

```
cat("Data frame after imputation of missing values:\n")
```

Data frame after imputation of missing values:

```
print(df)
```

	ID	Age	Score	Passed
1	1	64	71.78858	FALSE
2	2	20	51.87155	FALSE
3	3	58	98.67700	FALSE
4	4	42	77.79050	FALSE
5	5	44	97.87883	FALSE
6	6	53	94.38775	FALSE
7	7	54	81.99894	TRUE
8	8	48	98.54833	FALSE
9	9	62	77.79050	TRUE
10	10	22	77.79050	FALSE

```

11 11 37 67.33741 TRUE
12 12 51 77.79050 FALSE
13 13 45 77.79050 FALSE
14 14 57 51.94682 FALSE
15 15 20 87.43977 FALSE
16 16 50 83.86384 TRUE
17 17 59 58.56322 TRUE
18 18 41 63.05440 TRUE
19 19 47 75.72065 TRUE
20 20 60 83.78036 FALSE

```

```

# Grouping the data by Passed status and calculating group-wise statistics
library(dplyr)

```

Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

```
filter, lag
```

The following objects are masked from 'package:base':

```
intersect, setdiff, setequal, union
```

```

grouped_stats <- df %>%
  group_by(Passed) %>%
  summarise(
    mean_score = mean(Score, na.rm = TRUE),
    mean_age = mean(Age)
  )
cat("Grouped statistics by Passed status:\n")

```

Grouped statistics by Passed status:

```
print(grouped_stats)
```

```

# A tibble: 2 x 3
  Passed mean_score mean_age
  <lgl>      <dbl>      <dbl>
1 FALSE      80.6      44.9
2 TRUE       72.6      50

```