Experiment No 7

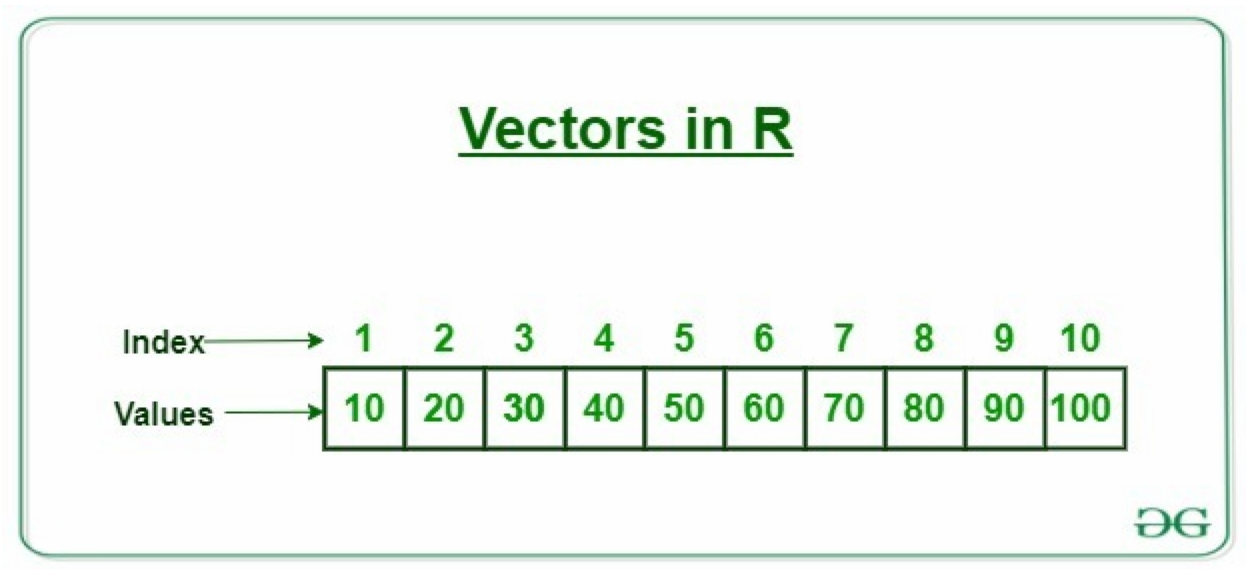
Title: Write an R Program to create a vector, add two vectors of integer type, and find sum, mean and product of a vector.

Objective:

The objective of this practical is to practice using basic operations with vectors in R, like creating vectors, adding two vectors together, and calculating the sum, mean, and product of a vector. This will help in understanding basic math operations with data in R.

Theory:

R Vectors are the same as the arrays in R language which are used to hold multiple data values of the same type. One major key point is that in R Programming Language the indexing of the vector will start from '1' and not from '0'. We can create numeric vectors and character vectors as well.



Types of Vectors

Vectors can be created to store various types of data. Each vector must hold elements of the same type:

Numeric vectors: Hold numeric data like integers or real numbers (e.g., c(1, 2.5, 4)).

Integer vectors: Hold integer data, created using the L suffix (e.g., c(1L, 2L, 3L)).

Character vectors: Hold text strings (e.g., c("apple", "banana")).

Logical vectors: Contain TRUE or FALSE values (e.g., c(TRUE, FALSE, TRUE)).

Complex vectors: Contain complex numbers (e.g., c(3+2i, 4+1i)).

Raw vectors: Used to store raw bytes, usually in advanced programming tasks.

Creating Vectors

c() function: The primary way to create a vector by combining elements, as in c(1, 2, 3).

Sequence generation: Use the colon operator : or seq() for numeric sequences, e.g., 1:5 (1, 2, 3, 4, 5) or seq(1, 10, by=2).

Replicating values: Use rep() to repeat elements, e.g., rep(1, times=5) creates c(1, 1, 1, 1, 1).

Vector Operations

Vectors in R support many operations, both element-wise and as a whole:

Element-wise arithmetic: When performing arithmetic on two vectors, R operates element-by-element if they are of the same length. E.g., c(1, 2) + c(3, 4) results in c(4, 6).

Vector recycling: If vectors are of different lengths, R "recycles" the shorter one. For example, c(1, 2, 3) + c(4, 5) results in c(5, 7, 7).

Math functions: Functions like sum(), mean(), prod(), max(), and min() operate on all elements in the vector and return a single result.

Logical operations: Comparisons (e.g., >, <, ==) between vectors produce logical vectors, useful for subsetting and conditional operations.

Vector Indexing and Subsetting

Vectors support multiple ways of accessing and manipulating their elements:

Position-based indexing: Use square brackets with indices, e.g., vec[1:3].

Negative indexing: Excludes specified indices, e.g., vec[-2] removes the second element.

Logical indexing: Subsets based on a logical condition, e.g., vec[vec > 3].

Name-based indexing: If a vector has named elements (e.g., c(a=1, b=2)), you can access elements by name: vec["a"].

Practical Approach:

- **R Program to create a vector, add two vectors of integer type,.**

```
# Create a vector 'x' of integer type and length 3

x = c(10, 20, 30)


# Create another vector 'y' of integer type and length 3

y = c(20, 10, 40)


# Print message indicating the original vectors

print("Original Vectors:")


# Print the contents of vector 'x'

print(x)


# Print the contents of vector 'y'

print(y)


# Print message indicating the result after adding the vectors

print("After adding two Vectors:")


# Add vectors 'x' and 'y' element-wise and store in 'z'

z = x + y
```

# Print the resulting vector 'z'

print(z)

Output:

```
[1] "Original Vectors:"
[1] 10 20 30
[1] 20 10 40
[1] "After adding two Vectors:"
[1] 30 30 70
```

- **R program to find sum, mean and product of a vector.**

# Create a numeric vector 'x' with elements 10, 20, and 30

x = c(10, 20, 30)

# Print a message indicating the calculation of the sum

print("Sum:")

# Calculate and print the sum of the elements in vector 'x'

print(sum(x))

# Print a message indicating the calculation of the mean

print("Mean:")

# Calculate and print the mean of the elements in vector 'x'

print(mean(x))

# Print a message indicating the calculation of the product

print("Product:")

# Calculate and print the product of the elements in vector 'x'

print(prod(x))

Output:

```
[1] "Sum:"
[1] 60
[1] "Mean:"
[1] 20
[1] "Product:"
[1] 6000
```

Outcome:

Students will learn how to:

Make Vectors: Create lists of numbers or words in R.

Do Calculations: Add vectors together and find the total, average, and product of a vector.

Use Basic R Functions: Use simple R functions to quickly get insights from data in a vector.

This helps build a basic understanding of handling data in R, which is essential for more advanced data analysis.

Online Reference Websites:

- **https://www.w3resource.com/r-programming-exercises/vector/index.php**
- **https://jrnold.github.io/r4ds-exercise-solutions/vectors.html**

- https://www.geeksforgeeks.org/r-programming-exercises-practice-questions-and-solutions/
- https://mathcenter.oxford.emory.edu/site/math117/probSetVectors/

Viva Questions:

1.	What is a vector in R, and why is it important?
2.	How do you create a vector in R? Can you give an example?
3.	What types of data can be stored in a vector?
4.	Explain how to add two vectors together in R.
5.	What happens if you try to add two vectors of different lengths?
6.	How do you calculate the sum, mean, and product of a vector?
7.	What function would you use to find the length (number of elements) in a vector?
8.	How does R handle a mix of data types within a single vector?
9.	Can you explain what vector recycling is and give an example?
10.	Why is it useful to perform vectorized operations in R instead of using loops?

Conclusion:

In this practical, we learned how to create vectors in R, add two vectors, and calculate the sum, mean, and product of a vector.