

Practical No. 11

Title: Study and Implementation of Data Transpose Operation in R

Introduction:

Data transposition is the process of flipping rows and columns in a dataset. It is often used when the orientation of the data needs to be changed to suit the requirements of analysis, visualization, or reporting. In R, transposition is particularly useful in reshaping data from wide format to long format or vice versa, and can be achieved through several functions from base R as well as packages like `data.table` and `tidyverse`.

Objective:

- To understand the concept and purpose of data transposition.
- To learn how to transpose data using base R (`t ()` function).
- To implement transposition in the context of data reshaping using `tidyr` and `data.table`.
- To apply transpose operations on both numeric matrices and data frames.

Tools Required:

- **Software:** R (version 4.0 or above)
- **IDE (Recommended):** RStudio
- **Libraries:** Base R, `tidyr`, `data.table` (for extended operations)

Theory:

What is Transposition?

Transposing data means converting:

- **Rows into columns**, or
- **Columns into rows**

In mathematical terms, the transpose of a matrix AAA is denoted ATA^T , where the element at row i and column j in AAA becomes the element at row j and column i in ATA^T .

Methods for Transposing Data in R

1. Using Base R – `t ()` Function

The simplest way to transpose a matrix or data frame is using the `t ()` function.

Syntax:

`t (x)`


Example 1: Transposing a matrix

```
mat <- matrix(1:9, nrow = 3, byrow = TRUE)
print("Original Matrix:")
print(mat)

transposed_mat <- t(mat)
print("Transposed Matrix:")
print(transposed_mat)
```

Example 2: Transposing a data frame

```
df <- data.frame(Name = c("A", "B"), Score1 = c(85, 90), Score2 = c(88, 92))
transposed_df <- t(df)
print(transposed_df)
```

 **Note:** When using `t()` on a data frame, the result becomes a matrix and loses original data types.

2. Using `tidyr::pivot_longer()` and `pivot_wider()`

These functions are used for reshaping tidy data, which is a form of transposition.

Wide to Long (rows instead of columns):

```
library(tidyr)

df <- data.frame(
  Student = c("Alice", "Bob"),
  Math = c(90, 85),
  English = c(95, 80)
)

long_df <- pivot_longer(df, cols = Math:English, names_to = "Subject", values_to = "Score")
print(long_df)
```

Long to Wide (columns instead of rows):

```
wide_df <- pivot_wider(long_df, names_from = Subject, values_from = Score)
print(wide_df)
```

3. Using `data.table::transpose()`




This function is helpful when you need to transpose data while keeping column names.

```
library(data.table)

dt <- data.table(
  Name = c("Alice", "Bob"),
  Math = c(90, 85),
  Science = c(92, 88)
)

# Transpose only the numeric part
transposed_dt <- transpose(dt[, -1], make.names = "Name")
print(transposed_dt)
```

Comparison of Methods

Method	Best Used For	Retains Data Types	Notes
<code>t()</code> (Base R)	Matrices or simple data frames	 No	Converts to matrix
<code>pivot_longer/wider</code>	Tidy data reshaping	 Yes	Great for analysis and plotting
<code>transpose()</code>	<code>data.table</code> workflows	 Yes	Retains data format and column names

Conclusion:

Transposition is a valuable operation in R for reshaping and reorganizing data. It is especially useful when preparing data for analysis, reporting, or visualization. R offers multiple ways to perform transposition—from the basic `t()` function to more advanced and tidy-friendly functions like `pivot_longer()` and `transpose()` from `data.table`. Understanding when and how to use these methods enables more flexible and efficient data handling in any project.