Data Manipulation in R

Data manipulation in R is a crucial part of data analysis, allowing you to clean, transform, and prepare your data for further analysis or visualization. Two of the most powerful and popular packages for data manipulation in R are `dplyr` and `tidyr`, both of which are part of the tidyverse ecosystem.

1. Installing and Loading the Packages

Before using `dplyr` and `tidyr`, you need to install and load them.

r

```
# Install the packages
install.packages("dplyr")
install.packages("tidyr")

# Load the packages
library(dplyr)
library(tidyr)
```

2. Using `dplyr` for Data Manipulation

`dplyr` is designed for data manipulation tasks such as selecting, filtering, mutating, and summarizing data.

a. Selecting Columns with `select()`

The `select()` function allows you to choose specific columns from a data frame.

r

```
# Example data frame
data <- data.frame(
  Name = c("Alice", "Bob", "Charlie"),
  Age = c(25, 30, 35),
```

```
  Score = c(90, 85, 88)
)

# Select the Name and Score columns
selected_data <- select(data, Name, Score)
print(selected_data)
```

b. Filtering Rows with `filter()`

The `filter()` function is used to filter rows based on conditions.

r

```
# Filter rows where Age is greater than 28
filtered_data <- filter(data, Age > 28)
print(filtered_data)
```

c. Arranging Rows with `arrange()`

The `arrange()` function sorts the rows of a data frame by one or more columns.

r

```
# Arrange rows by Age in ascending order
arranged_data <- arrange(data, Age)
print(arranged_data)

# Arrange rows by Score in descending order
arranged_data_desc <- arrange(data, desc(Score))
print(arranged_data_desc)
```

d. Mutating (Adding or Transforming) Columns with `mutate()`

The `mutate()` function allows you to create new columns or transform existing ones.

r

```r
# Add a new column with the squared Age
mutated_data <- mutate(data, Age_Squared = Age^2)
print(mutated_data)
```

e. Summarizing Data with `summarize()`

The `summarize()` function is used to compute summary statistics for one or more variables.

r

```r
# Summarize data to calculate the average Score
summary_data <- summarize(data, Average_Score = mean(Score))
print(summary_data)
```

f. Grouping Data with `group_by()`

The `group_by()` function is often used in conjunction with `summarize()` to perform group-wise operations.

r

```r
# Example data frame
data <- data.frame(
  Name = c("Alice", "Bob", "Charlie", "Alice", "Bob"),
  Age = c(25, 30, 35, 28, 33),
  Score = c(90, 85, 88, 92, 80)
)

# Group data by Name and summarize to calculate the average
Score for each group
grouped_data <- data %>%
  group_by(Name) %>%
  summarize(Average_Score = mean(Score))
print(grouped_data)
```

3. Using `tidyr` for Data Cleaning and Transformation

`tidyr` is focused on tidying data, which means ensuring that data is in a consistent, rectangular format that is easy to analyze.

a. Gathering Columns with `gather()`

The `gather()` function reshapes data from wide to long format by collapsing multiple columns into key-value pairs.

r

```
# Example data frame in wide format
wide_data <- data.frame(
  Name = c("Alice", "Bob", "Charlie"),
  Score1 = c(90, 85, 88),
  Score2 = c(92, 80, 91)
)

# Gather Score1 and Score2 into a single column
long_data <- gather(wide_data, key = "Test", value = "Score",
Score1, Score2)
print(long_data)
```

b. Spreading Rows with `spread()`

The `spread()` function is the opposite of `gather()`, reshaping data from long to wide format by spreading key-value pairs into separate columns.

r

```
# Example data frame in long format
long_data <- data.frame(
  Name = c("Alice", "Alice", "Bob", "Bob", "Charlie",
"Charlie"),
  Test = c("Score1", "Score2", "Score1", "Score2", "Score1",
"Score2"),
  Score = c(90, 92, 85, 80, 88, 91)
```

```
)

# Spread the Test column back into separate columns
wide_data <- spread(long_data, key = "Test", value = "Score")
print(wide_data)
```

c. Separating and Uniting Columns with `separate()` and `unite()`

The `separate()` function splits a single column into multiple columns based on a separator, while `unite()` does the opposite by combining multiple columns into one.

r

```
# Example data frame
data <- data.frame(
  FullName = c("Alice Johnson", "Bob Smith", "Charlie Brown")
)

# Separate FullName into FirstName and LastName
separated_data <- separate(data, FullName, into = c("FirstName",
"LastName"), sep = " ")
print(separated_data)

# Unite FirstName and LastName back into FullName
united_data <- unite(separated_data, FullName, FirstName,
LastName, sep = " ")
print(united_data)
```

d. Handling Missing Values with `drop_na()` and `replace_na()`

- **`drop_na()`:** Removes rows with missing values.
- **`replace_na()`:** Replaces missing values with specified values.

r

```
# Example data frame with missing values
```

```r
data <- data.frame(
  Name = c("Alice", "Bob", "Charlie", NA),
  Score = c(90, 85, NA, 88)
)

# Drop rows with missing values
dropped_data <- drop_na(data)
print(dropped_data)

# Replace missing values in Score with 0
replaced_data <- replace_na(data, list(Score = 0))
print(replaced_data)
```

4. Combining `dplyr` and `tidyr`

Often, you'll use `dplyr` and `tidyr` together to clean and transform data.

Example: A Complete Workflow

r

```r
# Example messy data frame
data <- data.frame(
  Name = c("Alice", "Bob", "Charlie"),
  Test1 = c(90, 85, NA),
  Test2 = c(92, NA, 88)
)

# Workflow to clean and transform data
cleaned_data <- data %>%
  gather(key = "Test", value = "Score", Test1, Test2) %>%  # Convert to long format
  drop_na() %>%  # Drop missing values
  group_by(Name) %>%  # Group by Name
  summarize(Average_Score = mean(Score))  # Calculate average Score
```

```
print(cleaned_data)
```

In this example, the data is first reshaped into long format, missing values are removed, and then the data is grouped by `Name` and summarized to calculate the average score.

Summary

- **`dplyr`:** Provides functions like `select()`, `filter()`, `arrange()`, `mutate()`, `summarize()`, and `group_by()` for data manipulation.
- **`tidyr`:** Focuses on tidying data using functions like `gather()`, `spread()`, `separate()`, `unite()`, `drop_na()`, and `replace_na()`.
- **Combining `dplyr` and `tidyr`:** These packages are often used together to efficiently clean, transform, and manipulate data in R.