Practical No. 6

Title: Creating Sample Data and Performing Data Manipulation in R

Introduction:

R is a powerful statistical programming language used widely in data analysis and statistical computing. Before working on real-world datasets, it's essential to understand how to create sample data and manipulate it effectively. Data manipulation involves organizing and transforming data into a useful format for analysis, often using packages like `dplyr` and `tidyr`.

Objective:

The main objectives of this practical are:

- To learn how to create sample data in R.
- To perform basic data manipulation tasks such as filtering, selecting, mutating, summarizing, and arranging using R functions.
- To understand the use of `dplyr` for efficient data handling.

Tools Required:

- R (version 4.0 or later)
- RStudio (recommended IDE)
- `dplyr` and `tidyverse` packages (for data manipulation)

Theory:

1. Creating Sample Data

In R, sample data can be created using functions like `data.frame()`, `tibble()`, `sample()`, and `seq()`. Example:

```
r
CopyEdit
# Create sample employee data
employee_data <- data.frame(
  ID = 1:10,
  Name = paste("Employee", 1:10),
  Age = sample(25:50, 10, replace = TRUE),
  Department = sample(c("HR", "Finance", "IT", "Sales"), 10, replace = TRUE),
  Salary = sample(seq(30000, 70000, by = 5000), 10, replace = TRUE)
)
```

2. Data Manipulation with dplyr

`dplyr` provides a grammar for data manipulation with a consistent set of verbs:

- `select()`: Choose specific columns
- `filter()`: Filter rows based on conditions
- `mutate()`: Add or modify columns

- `arrange()`: Sort data
- `summarise()`: Aggregate data
- `group_by()`: Group data for summary operations

Detailed Data Manipulation Operations in R (`dplyr`)

1. `select()` – Selecting Columns

Purpose:
Used to select specific columns (variables) from a data frame.

Syntax:

```
r
CopyEdit
select(data, column1, column2, ...)
```

Example:

```
r
CopyEdit
# Select only Name and Salary columns
select(employee_data, Name, Salary)
```

2. `filter()` – Filtering Rows

Purpose:
Used to extract rows that meet certain logical conditions.

Syntax:

```
r
CopyEdit
filter(data, condition)
```

Example:

```
r
CopyEdit
# Filter employees in the IT department
filter(employee_data, Department == "IT")

# Filter employees with Salary > 50000
filter(employee_data, Salary > 50000)
```

3. `mutate()` – Creating or Modifying Columns

Purpose:
Adds new variables or modifies existing ones.

Syntax:

```
r
CopyEdit
mutate(data, new_column = expression)
```

Example:

```
r
CopyEdit
# Add a column 'Bonus' which is 10% of Salary
mutate(employee_data, Bonus = Salary * 0.10)

# Create a new column 'Age_Group'
mutate(employee_data, Age_Group = ifelse(Age < 35, "Young", "Senior"))
```

4. `arrange()` – Sorting Data

Purpose:
Reorders rows of a data frame by the values of one or more columns.

Syntax:

```
r
CopyEdit
arrange(data, column1, column2, ...)
```

Example:

```
r
CopyEdit
# Sort by Salary ascending
arrange(employee_data, Salary)

# Sort by Salary descending
arrange(employee_data, desc(Salary))
```

5. `summarise()` – Summarizing Data

Purpose:
Reduces multiple values to a single summary, like mean, sum, count, etc.

Syntax:

```
r
CopyEdit
summarise(data, summary_column = function(column))
```

Example:

```
r
CopyEdit
# Calculate average salary
summarise(employee_data, Avg_Salary = mean(Salary))
```

6. `group_by()` – Grouping Data

Purpose:
Used to group data by one or more variables, typically before summarising.

Syntax:

```
r
CopyEdit
group_by(data, column1, column2, ...)
```

Combined with `summarise()`:

```
r
CopyEdit
# Average salary by department
employee_data %>%
  group_by(Department) %>%
  summarise(Average_Salary = mean(Salary))
```

7. `rename()` – Renaming Columns

Purpose:
Changes column names in a data frame.

Syntax:

```
r
CopyEdit
rename(data, new_name = old_name)
```

Example:

```
r
CopyEdit
rename(employee_data, Emp_Name = Name)
```

8. `distinct()` – Removing Duplicate Rows

Purpose:
Keeps only unique rows or combinations of columns.

Syntax:

```
r
CopyEdit
distinct(data, column1, column2, ...)
```

Example:

```
r
CopyEdit
# Get unique departments
distinct(employee_data, Department)
```

Conclusion:

In this practical, we learned how to generate sample data and apply core data manipulation techniques using the `dplyr` package in R. These operations—filtering, selecting, creating new variables, summarizing, sorting, and grouping—form the backbone of data preprocessing in R. Mastery of these techniques is essential for effective data analysis and will greatly enhance your productivity in future projects.