Data Manipulation with `dplyr` and `tidyr`

1 Introduction

The `dplyr` and `tidyr` packages are essential tools in R for efficient data manipulation and tidying. This document introduces key functionalities of these packages, including filtering, summarizing, grouping, and reshaping data. Each section includes explanations, examples, best practices, and exercises.

2 Filtering Data with `dplyr`

2.1 Overview

Filtering involves selecting rows from a dataset based on specific conditions. The `filter()` function in `dplyr` allows you to extract subsets of data.

2.2 Examples

```
# Load dplyr
library(dplyr)

# Example dataset: iris
data(iris)

# Filter rows where Species is "setosa"
setosa_data <- filter(iris, Species == "setosa")

# Filter rows where Sepal.Length > 5
long_sepals <- filter(iris, Sepal.Length > 5)

# Combine multiple conditions
filtered_data <- filter(iris, Sepal.Length > 5, Petal.
   Length < 2)
```

2.3 Best Practices

- Use logical operators (`&`, `|`, `!`) for complex conditions.
- Avoid filtering directly in your original dataset; store the filtered result in a new variable.

3 Summarizing Data with dplyr

3.1 Overview

The summarize() function computes summary statistics (mean, median, sum, etc.) for datasets.

3.2 Examples

```
# Compute mean of Sepal.Length
mean_sepal <- summarize(iris, mean_sepal_length = mean(
   Sepal.Length))

# Compute multiple statistics
stats <- summarize(iris,
                   mean_sepal = mean(Sepal.Length),
                   max_sepal = max(Sepal.Length))
```

3.3 Best Practices

- Use na.rm = TRUE to handle missing values in summary functions.
- Combine group_by() with summarize() for grouped summaries.

4 Grouping Data with dplyr

4.1 Overview

The group_by() function divides a dataset into groups, allowing for grouped operations.

4.2 Examples

```
# Group by Species and compute mean Sepal.Length
grouped_stats <- iris %>%
  group_by(Species) %>%
  summarize(mean_sepal = mean(Sepal.Length))
```

4.3 Best Practices

- Always use `ungroup()` after grouped operations to avoid unintended behavior.
- Combine `group_by()` with other `dplyr` functions like `mutate()` or `summarize()`.

5 Reshaping Data with `tidyr`

5.1 Overview

The `tidyr` package provides functions to transform data between wide and long formats.

5.2 Examples

```
# Example dataset: pivot_wider
library(tidyr)
data <- data.frame(
  id = c(1, 1, 2, 2),
  variable = c("A", "B", "A", "B"),
  value = c(10, 20, 15, 25)
)

# Wide format
wide_data <- pivot_wider(data, names_from = variable,
   values_from = value)

# Long format
long_data <- pivot_longer(wide_data, cols = A:B, names_
   to = "variable", values_to = "value")
```

5.3 Best Practices

- Ensure the dataset is in the desired structure (wide or long) before analysis.
- Use `gather()` and `spread()` (deprecated) for backward compatibility.

6 Best Practices for Data Manipulation

- Always inspect your data before and after applying transformations.
- Use the pipe operator (`%>%`) for readable and efficient workflows.
- Document your data manipulation steps for reproducibility.

7 Practice Exercises

7.1 Datasets

Use datasets from the `datasets` package in R (`iris`, `mtcars`, `airquality`, etc.) for these exercises.

7.2 Exercises

1. Filter rows in the `mtcars` dataset where `mpg` is greater than 20.
2. Compute the mean of `Sepal.Length` for each species in the `iris` dataset.
3. Group the `mtcars` dataset by the number of cylinders (`cyl`) and calculate the average horsepower (`hp`) for each group.
4. Reshape the `airquality` dataset to a long format with `month` as a key column.
5. Use the `group_by()` and `summarize()` functions to compute the maximum wind speed for each month in the `airquality` dataset.
6. Filter the `iris` dataset for rows where `Sepal.Width` is less than 3 and `Species` is not `setosa`.
7. Reshape the `mtcars` dataset into a long format and back to wide format using `pivot_longer()` and `pivot_wider()`.
8. Create a new column in the `iris` dataset that contains the ratio of `Sepal.Length` to `Sepal.Width`.
9. Use `summarize()` to compute the total number of observations for each species in the `iris` dataset.
10. Combine filtering and summarizing to calculate the average mileage (`mpg`) for cars with more than 100 horsepower in the `mtcars` dataset.

8 Conclusion

The `dplyr` and `tidyr` packages provide powerful tools for efficient data manipulation in R. Mastering these functions will significantly enhance your data analysis capabilities.