we create a sample data frame with two columns: 'Group' and 'Value'. We then measure the execution time of the grouping and summarizing operation and the splitting operation.

By using system.time() in R and timeit.timeit() in Python, we can obtain the elapsed time for the respective operations. The number parameter specifies the number of times the operation should be executed to obtain an average execution time.

Using R

In [14]:
```
library(dplyr)
#library(microbenchmark)

# Create a sample data frame
df <- data.frame(Group = rep(letters[1:3], each = 100000),
                 Value = rnorm(300000))

# Measure the time taken for grouping and summarizing

grouping_time <- mean(replicate(10, {
  system.time({
    df_grouped <- df %>% group_by(Group) %>%
                  summarise(Sum = sum(Value), Mean = mean(Value))
  })["elapsed"]
}))

# Measure the time taken for splitting


splitting_time <- mean(replicate(10, {
  system.time({
    df_split <- split(df, df$Group)
  })["elapsed"]
}))
```

```
# Print the results
cat("Grouping Time:", grouping_time, "seconds\n")
cat("Splitting Time:", splitting_time, "seconds\n")
```

```
Grouping Time: 0.0117 seconds
Splitting Time: 0.0848 seconds
```

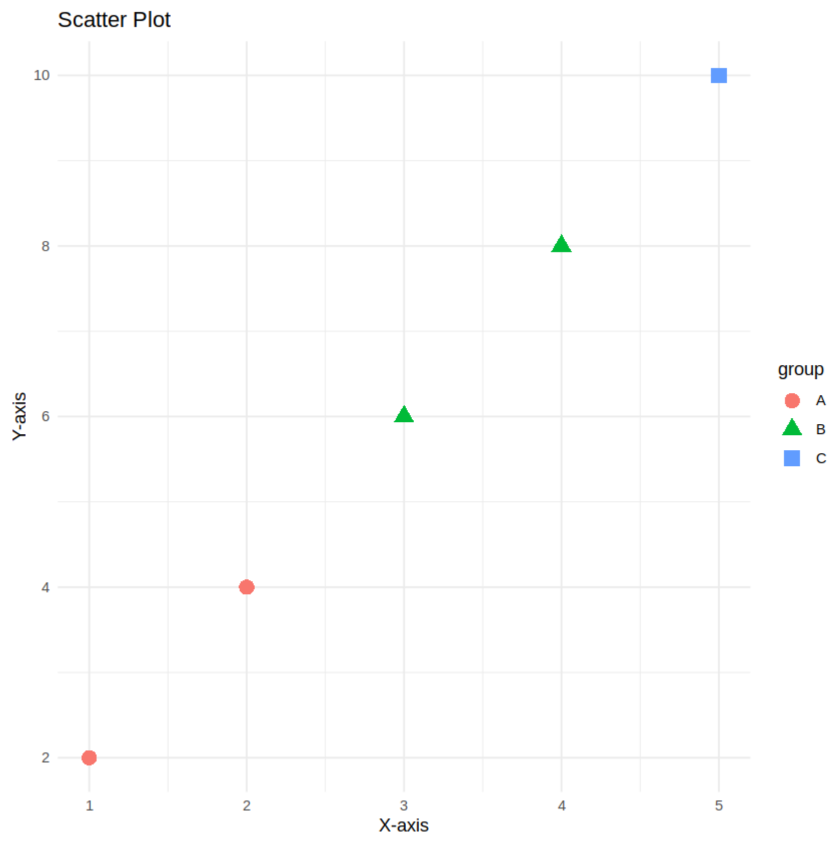
the number parameter is used in the timeit.timeit() function in Python and the system.time() function in R. This parameter specifies the number of times the given operation or function should be executed in order to calculate the average execution time.

By repeating the execution of the operation multiple times, we can obtain a more accurate measurement of the execution time, reducing the impact of any fluctuations caused by external factors such as system load or resource allocation.

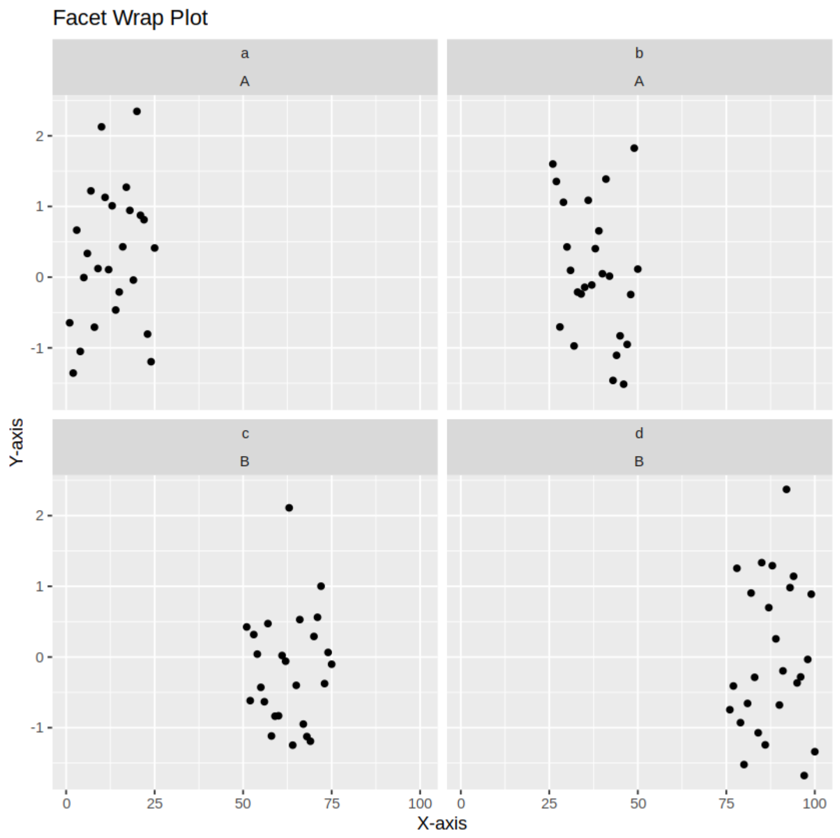
In the code snippets, the number parameter is set to 10, indicating that the operation will be executed 10 times in both R and Python. The execution times for each repetition are recorded, and the average execution time is calculated based on those results.

Adjusting the number parameter allows you to control the trade-off between accuracy and execution time. Higher values of number provide more precise results but also increase the overall time taken to measure the execution time. Lower values may introduce more variability in the measurements but lead to quicker results.

In [17]:
```
library(ggplot2)

# Create a sample data frame
df <- data.frame(
  x = c(1, 2, 3, 4, 5),
  y = c(2, 4, 6, 8, 10),
  group = c('A', 'A', 'B', 'B', 'C')
)

# Create a scatter plot with different colors and shapes for each group
ggplot(df, aes(x, y, color = group, shape = group),width=100) +
  geom_point(size = 4) +
  labs(x = 'X-axis', y = 'Y-axis', title = 'Scatter Plot')+
    theme_minimal()
```



In [23]:
```
library(ggplot2)

# Create a sample data frame
df <- data.frame(
  x = 1:100,
  y = rnorm(100),
  group = rep(letters[1:4], each = 25),
  category = rep(c("A", "B"), each = 50)
)

# Create a scatter plot with facet wrapping
ggplot(df, aes(x, y)) +
  geom_point() +
  facet_wrap(~ group + category, ncol = 2) +
  labs(x = "X-axis", y = "Y-axis", title = "Facet Wrap Plot")
```


Facet Wrap Plot
a
A
b
A
c
B
d
B
2
1
0
-1
0
25
50
75
100
Y-axis
X-axis