How can I arrange rows by group using dplyr, and what are some examples of how to do so?

Authored by

stats writer

June 27, 2024

Dplyr is a popular R package used for data manipulation and analysis. One of its useful functions is the ability to arrange rows by group. This can be done using the "group_by" and "arrange" functions. First, the data must be grouped based on a specific variable using the "group_by" function. Then, the "arrange" function can be used to sort the data within each group based on a chosen variable. This allows for easy comparison and analysis of the data within each group. For example, if we have a dataset containing sales data for different products, we can use dplyr to group the data by product category and then arrange the data within each category by sales amount in descending order. This will help us identify the top-selling products within each category. In summary, dplyr's "group_by" and "arrange" functions provide a convenient and efficient way to organize and analyze data by groups.

Arrange Rows by Group Using dplyr (With Examples)

You can use the following methods to arrange rows by group in dplyr:

Method 1: Arrange Rows in Ascending Order by Group

```
library(dplyr)

#arrange rows in ascending order based on col2, grouped by col1
df %>%
group_by(col1) %>%
arrange(col2, .by_group=TRUE)
```

Method 2: Arrange Rows in Descending Order by Group

```
library(dplyr)
```

```
#arrange rows in descending order based on col2, grouped by col1
df %>%
group_by(col1) %>%
arrange(desc(col2), .by_group=TRUE)
```

Method 3: Arrange Rows by Multiple Groups

```
library(dplyr)

#arrange rows based on col3, grouped by col1 and col2
df %>%
group_by(col1, col2) %>%
arrange(col3, .by_group=TRUE)
```

This tutorial explains how to use each method in practice with the following data frame:

```
#create data frame
df <- data.frame(team=c('A', 'A', 'A', 'A', 'B', 'B', 'B', 'B'),
position=c('G', 'G', 'F', 'F', 'G', 'G', 'F', 'F'),
points=c(10, 12, 3, 14, 22, 15, 17, 17))

#view data frame
df
```

team position points
1 A G 10
2 A G 12
3 A F 3
4 A F 14
5 B G 22
6 B G 15
7 B F 17
8 B F 17

Example 1: Arrange Rows in Ascending Order by Group

The following code shows how to arrange the rows in ascending order based on points, grouped by the team column:

```
library(dplyr)

#arrange rows in ascending order by points, grouped by team
df %>%
group_by(team) %>%
arrange(points, .by_group=TRUE)

# A tibble: 8 x 3
```

Groups: team
team position points

1 A F 3
2 A G 10
3 A G 12
4 A F 14
5 B G 15
6 B F 17
7 B F 17
8 B G 22

The rows are arranged in ascending order (smallest to largest) by points, grouped by the team column.

Example 2: Arrange Rows in Descending Order by Group

The following code shows how to arrange the rows in descending order based on points, grouped by the team column:

```
library(dplyr)

#arrange rows in descending order by points, grouped by team
df %>%
```

```
group_by(team) %>%
arrange(desc(points), .by_group=TRUE)

# A tibble: 8 x 3
# Groups: team
team position points

1 A F 14
2 A G 12
3 A G 10
4 A F 3
5 B G 22
6 B F 17
7 B F 17
8 B G 15
```

The rows are arranged in descending order (largest to smallest) by points, grouped by the team column.

Example 3: Arrange Rows by Multiple Groups

The following code shows how to arrange the rows in ascending order based on points, grouped by the team*and*position columns:

```
library(dplyr)
```

```
#arrange rows in descending order by points, grouped by team and position
df %>%
group_by(team, position) %>%
arrange(points, .by_group=TRUE)

# A tibble: 8 x 3
# Groups: team, position
team position points

1 A F 3
2 A F 14
3 A G 10
4 A G 12
5 B F 17
6 B F 17
7 B G 15
8 B G 22
```

The rows are arranged in ascending order (smallest to largest) by points, grouped by the team*and*position columns.