How can I use dplyr to arrange variables by group based on their rank within each group?

Authored by

stats writer

May 5, 2024

Dplyr is a popular data manipulation package in R that offers a variety of functions for efficient data manipulation. One of its useful functions is the ability to arrange variables by group based on their rank within each group. This means that the variables will be sorted in ascending or descending order based on their values within each group. This can be achieved by using the "arrange" function in dplyr, which allows for easy sorting and reordering of data within specific groups. This feature is particularly beneficial in situations where grouping and ranking of data is required, such as in statistical analysis or creating visualizations. By using dplyr to arrange variables by group based on their rank, users can easily organize and analyze their data in a more organized and efficient manner.

Rank Variables by Group Using dplyr

You can use the following basic syntax to rank variables by group in dplyr:

```
df %>% arrange(group_var, numeric_var) %>%
group_by(group_var) %>%
mutate(rank = rank(numeric_var))
```

The following examples show how to use this syntax in practice with the following data frame:

```
#create data frame
df <- data.frame(team = c('A', 'A', 'A', 'A', 'B', 'B', 'B', 'C', 'C', 'C'),
points = c(12, 28, 19, 22, 32, 45, 22, 28, 13, 19),
rebounds = c(5, 7, 7, 12, 11, 4, 10, 7, 8, 8))
```

#view data frame

df

team points rebounds

1 A 12 5

2 A 28 7

3 A 19 7

4 A 22 12

5 B 32 11

6 B 45 4

7 B 22 10

8 C 28 7

9 C 13 8

10 C 19 8

Example 1: Rank in Ascending Order

The following code shows how to rank the points scored by players in ascending order, grouped by team:

```
library(dplyr)#rank points scored, grouped by team
df %>% arrange(team, points) %>%
group_by(team) %>%
mutate(rank = rank(points))
```

A tibble: 10 x 4

Groups: team

team points rebounds rank

1 A 12 5 1

2 A 19 7 2

3 A 22 12 3

4 A 28 7 4

5 B 22 10 1

6 B 32 11 2

7 B 45 4 3

8 C 13 8 1

9 C 19 8 2

10 C 28 7 3

Example 2: Rank in Descending Order

We can also rank the points scored in descending order by group, using a negative sign within the rank() function:

```
library(dplyr)#rank points scored in reverse, grouped by team
df %>% arrange(team, points) %>%
group_by(team) %>%
```

mutate(rank = rank(-points))

A tibble: 10 x 4
Groups: team
team points rebounds rank

1 A 12 5 4
2 A 19 7 3
3 A 22 12 2
4 A 28 7 1
5 B 22 10 3
6 B 32 11 2
7 B 45 4 1
8 C 13 8 3
9 C 19 8 2
10 C 28 7 1

How to Handle Ties in Ranking

We can use the ties.method argument to specify how we should handle ties when ranking numerical values.

rank(points, ties.method='average')

You can use one of the following options to specify how

to handle ties:

average: (Default) Assigns each tied element to the average rank (elements ranked in the 3rd and 4th position would both receive a rank of 3.5)first: Assigns the first tied element to the lowest rank (elements ranked in the 3rd and 4th positions would receive ranks 3 and 4 respectively)min: Assigns every tied element to the lowest rank (elements ranked in the 3rd and 4th position would both receive a rank of 3)max: Assigns every tied element to the highest rank (elements ranked in the 3rd and 4th position would both receive a rank of 4)random: Assigns every tied element to a random rank (either element tied for the 3rd and 4th position could receive either rank)

The following tutorials explain how to perform other common functions in dplyr: