How can mutate() be used with multiple conditions in dplyr?

Authored by

stats writer

June 24, 2024

The mutate() function in the dplyr package allows for the creation of new columns in a data frame based on specified conditions. This function can be used with multiple conditions, where the new column values are determined by the outcome of these conditions. This allows for efficient and streamlined data manipulation, as it eliminates the need for multiple separate steps. Additionally, the use of multiple conditions in mutate() allows for more complex and precise data transformations to be performed. Overall, the ability to use mutate() with multiple conditions in dplyr offers a flexible and powerful tool for data analysts and scientists.

dplyr: Use mutate() with Multiple Conditions

You can use the following basic syntax in to use the mutate() function to create a new column based on multiple conditions:

```
library(dplyr)

df <- df%>% mutate(class = case_when((team == 'A' & points >= 20) ~ 'A_Good',
(team == 'A' & points < 20) ~ 'A_Bad',
(team == 'B' & points >= 20) ~ 'B_Good',
TRUE ~ 'B_Bad'))
```

This particular syntax creates a new column called class that takes on the following values:

A_Good if team is equal to A and points is greater than or equal to 20.A_Bad if team is equal to A and points is less than 20.B_Good if team is equal to B and points is

greater than or equal to 20.B_Bad if none of the previous conditions are met.

The following example shows how to use this syntax in practice.

Example: Use mutate() in dplyr with Multiple Conditions

Suppose we have the following data frame in R that contains information about various basketball players:

```
#create data frame
df <- data.frame(team=c('A', 'A', 'A', 'A', 'A', 'B', 'B', 'B', 'B', 'B'),
points=c(22, 30, 34, 19, 14, 12, 39, 15, 22, 25))

#view data frame
df

team points
1 A 22
2 A 30
3 A 34
4 A 19
5 A 14
```

6 B 12
7 B 39
8 B 15
9 B 22
10 B 25

We can use the following syntax with the mutate() function to create a new column called class whose values are based on the values of the team and points columns:

```
library(dplyr)
#add new column based on values in team and points columns
df <- df%>% mutate(class = case_when((team == 'A' & points >= 20) ~ 'A_Good',
(team == 'A' & points < 20) ~ 'A_Bad',
(team == 'B' & points >= 20) ~ 'B_Good',
TRUE ~ 'B_Bad'))

#view updated data frame
df

team points class
1 A 22 A_Good
```

2 A 30 A_Good

3 A 34 A_Good

4 A 19 A_Bad

5 A 14 A_Bad

6 B 12 B_Bad

7 B 39 B_Good

8 B 15 B_Bad

9 B 22 B_Good

10 B 25 B_Good

The new class column takes on values based on the values in the team and points columns.

For example, the first row had a value of A in the team column and a points value greater than or equal to 20, so it received a value of A_Good in the new class column.

Note that in this example we used the & symbol as an "AND" operator to check if two conditions were both met before assigning a value in the class column.

However, we could have used the | symbol as an "OR" operator to instead check if either one of two conditions were met before assigning a value in the class column.

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