How can I use the ungroup() function in dplyr, and what are some examples of its implementation?

Authored by

stats writer

June 27, 2024

The ungroup() function is a useful tool in the dplyr package for data manipulation in R. It allows the user to remove any grouping variables that have been previously applied to a data frame. This function is particularly helpful when working with grouped data frames, as it allows for easier analysis and visualization of the data.

To use the ungroup() function, simply specify the data frame you want to remove grouping from as the argument. The function will then return a data frame with the grouping variables removed, allowing for individual observations to be analyzed separately.

For example, if you have a data frame of sales data grouped by region and you want to analyze the data for each individual store, you can use the ungroup() function to remove the grouping by region and obtain a data frame with all stores listed separately.

Another example is when working with merged data frames, the ungroup() function can be used to remove any grouping that may have been inherited from the original data frames.

In summary, the ungroup() function is a versatile tool in dplyr that simplifies the manipulation and analysis of data frames by removing any grouping variables. Its implementation can be seen in various scenarios such as working with grouped data frames or merged data frames.

Use ungroup() in dplyr (With Examples)

You can use the ungroup() function in dplyr to ungroup rows after using the group_by() function to summarize a variable by group.

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

Example: How to Use ungroup() in dplyr

Suppose we have the following data frame in R:

#create data frame

```
df <- data.frame(team=c('A', 'A', 'A', 'B', 'B', 'B'),
points=c(14, 18, 22, 26, 36, 34),
assists=c(5, 4, 4, 8, 7, 3))

#view data frame
df
```

```
team points assists
1 A 14 5
2 A 18 4
3 A 22 4
4 B 26 8
5 B 36 7
6 B 34 3
```

Now suppose we use the following code to calculate the mean value of points, grouped by team:

```
library(dplyr)

#calculate mean of points, grouped by team
df_new <- df %>%
group_by(team) %>%
summarize(mean_points = mean(points)) %>%
ungroup()
```

```
#view results
df_new

# A tibble: 2 x 2
team mean_points

1 A 18
2 B 32
```

Using this syntax, we're able to calculate the mean value of points grouped by team, but we've lost the assists column.

To retain the assists column, we can use mutate() instead of summarize() and still use ungroup() at the end:

```
library(dplyr)

#calculate mean of points, grouped by team
df_new <- df %>%
group_by(team) %>%
mutate(mean_points = mean(points)) %>%
ungroup()

#view results
```

df_new

A tibble: 6 x 4
team points assists mean_points

1 A 14 5 18
2 A 18 4 18
3 A 22 4 18
4 B 26 8 32
5 B 36 7 32
6 B 34 3 32

This time we're able to retain the assists column and by using the mutate() function we've simply added a new column called mean_points that shows the mean points value for each team.

Since we used the ungroup() function as well, we can perform calculations on this new data frame without worrying about any calculations being affected by any grouped variables.

If we didn't use the ungroup() function then the rows of the data frame would still be grouped, which could have unintended consequences when we perform more

calculations later on.