

How to Easily Create New Pandas Columns Like R's mutate()

Authored by
stats writer

November 24, 2025

RECOMMENDED CITATION

stats writer (2025). *How to Easily Create New Pandas Columns Like R's mutate()*.
PSYCHOLOGICAL SCALES. Retrieved from <https://scales.arabpsychology.com/?p=100452>

The `mutate()` function in Pandas is equivalent to the `mutate()` function in R and can be used to create new columns in a `DataFrame` by applying a function to existing columns. This function works similarly to the `apply()` function and can be used to add, update, or remove columns from a `DataFrame`. It can also be used to transform existing columns, such as creating a new column based on the value of an existing column. The `mutate()` function has powerful capabilities to manipulate `DataFrames` and can be used in many different ways to achieve specific goals.

In the R programming language, we can use the **`mutate()`** function from the **`dplyr`** package to quickly add new columns to a data frame that are calculated from existing columns.

For example, the following code shows how to calculate the mean value of a specific column in R and add that value as a new column in a data frame:

```
library(dplyr)
```

```
#create data frame
```

```
df <- data.frame(team=c('A', 'A', 'A', 'A', 'B', 'B', 'B', 'B'),  
points=c(30, 22, 19, 14, 14, 11, 20, 28))
```

```
#add new column that shows mean points by team
```

```
df <- df %>%
```

```
group_by(team) %>%
```

```
mutate(mean_points = mean(points))
```

```
#view updated data frame
```

```
df
```

```
team points mean_points
```

```
1 A 30 21.2
```

```
2 A 22 21.2
```

```
3 A 19 21.2
```

```
4 A 14 21.2
```

```
5 B 14 18.2
```

```
6 B 11 18.2
```

```
7 B 20 18.2
```

```
8 B 28 18.2
```

The equivalent of the **`mutate()`** function in pandas is the **`transform()`** function.

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

Example: Using transform() in pandas to Replicate mutate() in R

Suppose we have the following pandas DataFrame that shows the points scored by basketball players on various teams:

```
import pandas as pd
```

```
#create DataFrame  
df = pd.DataFrame({'team': ,  
'points': })
```

```
#view DataFrame  
print(df)
```

```
team points
```

```
0 A 30
```

```
1 A 22
```

```
2 A 19
```

```
3 A 14
```

```
4 B 14
```

```
5 B 11
```

```
6 B 20
```

```
7 B 28
```

We can use the **transform()** function to add a new column called **mean_points** that shows the mean points scored by each team:

```
#add new column to DataFrame that shows mean points by team
```

```
df = df.groupby('team').transform('mean')
```

```
#view updated DataFrame  
print(df)
```

```
team points mean_points
```

```
0 A 30 21.25
```

```
1 A 22 21.25
```

```
2 A 19 21.25
```

```
3 A 14 21.25
```

```
4 B 14 18.25
```

```
5 B 11 18.25
```

```
6 B 20 18.25
```

7 B 28 18.25

The mean points value for players on team A was **21.25** and the mean points value for players on team B was **18.25**, so these values were assigned accordingly to each player in a new column.

Notice that this matches the results we got from using the **mutate()** function in the introductory example.

It's worth noting that you can also use **lambda** to perform some custom calculation within the **transform()** function.

For example, the following code shows how to use **lambda** to calculate the percentage of total points scored by each player on their respective teams:

```
#create new column called percent_of_points
df = df.groupby('team').transform(lambda x: x/x.sum())

#view updated DataFrame
print(df)
```

```
team points percent_of_points
```

```
0 A 30 0.352941
```

```
1 A 22 0.258824
```

```
2 A 19 0.223529
```

```
3 A 14 0.164706
```

```
4 B 14 0.191781
```

```
5 B 11 0.150685
```

```
6 B 20 0.273973
```

```
7 B 28 0.383562
```

Here's how to interpret the output:

The first player on team A scored 30 out of 85 total points among team A players. Thus, his percentage of total points scored was $30/85 = 0.352941$.

The second player on team A scored 22 out of 85 total points among team A players. Thus, his percentage of total points scored was $22/85 = 0.258824$.

Note that we can use the **lambda** argument within the **transform()** function to perform any custom calculation that we'd like.