How can a rolling maximum be calculated in Pandas, and can you provide some examples?

Authored by

stats writer

June 27, 2024

A rolling maximum in Pandas refers to the process of calculating the maximum value of a specific column or series over a specified window of time. This can be achieved using the "rolling" function in Pandas, which applies a rolling calculation to the data based on a defined window size. This function allows for the calculation of rolling maximums over various time intervals, such as days, weeks, or months.

For example, if we have a dataset containing daily stock prices for a company, we can use the rolling maximum function to calculate the maximum price over a window of, say, 30 days. This would give us the highest price the stock has reached in the previous 30 days, and this value would be updated as we move forward in time.

Another example could be using the rolling maximum function to calculate the maximum temperature recorded in a city over a period of 7 days. This would give us the highest temperature in the previous week, and this value would be updated as we move on to the next day.

In summary, the rolling maximum function in Pandas is a useful tool for analyzing trends and patterns in time-series data, allowing for the calculation of maximum values over a specified time window.

Calculate a Rolling Maximum in Pandas (With Examples)

You can use the following methods to calculate a rolling maximum value in a pandas DataFrame:

Method 1: Calculate Rolling Maximum

df = df.values_column.cummax()

Method 2: Calculate Rolling Maximum by Group

df = df.groupby('group_column').values_column.cummax()

The following examples show how to use each method in practice.

Example 1: Calculate Rolling Maximum

Suppose we have the following pandas DataFrame that shows the sales made each day at some store:

import pandas as pd

#create DataFrame
df = pd.DataFrame({'day': ,
'sales': })

#view DataFrame
print(df)

day sales
0 1 4
1 2 6
2 3 5
3 4 8
4 5 14
5 6 13
6 7 13
7 8 12

8 9 9

9 10 8

10 11 19

11 12 14

We can use the following syntax to create a new column that displays the rolling maximum value of sales:

```
#add column that displays rolling maximum of sales
df = df.sales.cummax()

#view updated DataFrame
print(df)
```

day sales rolling_max

0 1 4 4

1 2 6 6

2 3 5 6

3 4 8 8

4 5 14 14

5 6 13 14

6 7 13 14

7 8 12 14

8 9 9 14

9 10 8 14

10 11 19 19

11 12 14 19

The new column titled rolling_max displays the rolling maximum value of sales.

Example 2: Calculate Rolling Maximum by Group

Suppose we have the following pandas DataFrame that shows the sales made each day at two different stores:

```
import pandas as pd

#create DataFrame
df = pd.DataFrame({'store': ,
'day': ,
'sales': })

#view DataFrame
print(df)
```

store day sales

0 A 1 4

1 A 2 6

2 A 3 5

3 A 4 8

4 A 5 14

5 A 6 13

6 B 7 13

7 B 8 12

8 B 9 9

9 B 10 8

10 B 11 19

11 B 12 14

We can use the following syntax to create a new column that displays the rolling maximum value of sales grouped by store:

```
#add column that displays rolling maximum of sales grouped by store
df = df.groupby('store').sales.cummax()

#view updated DataFrame
print(df)
```

store day sales rolling_max

0 A 1 4 4

1 A 2 6 6

2 A 3 5 6

3 A 4 8 8

4 A 5 14 14

5 A 6 13 14

6 B 7 13 13

7 B 8 12 13

8 B 9 9 13

9 B 10 8 13

10 B 11 19 19

11 B 12 14 19

The following tutorials explain how to perform other common operations in pandas: