Moving averages

Time series decomposition

Contents

INTRODUCE MOVING AVERAGES

PRACTICAL CONSIDERATIONS

Moving average example

- Consider window of size 3

Date	y	3-MA
2020-02-12	23	
2020-02-13	30	
2020-02-14	70	
2020-02-15	30	
2020-02-16	25	
2020-02-17	22	

Moving average example

- Consider window of size 3

Date	y	3-MA
2020-02-12	23	
2020-02-13	30	
2020-02-14	70	
2020-02-15	30	
2020-02-16	25	
2020-02-17	22	

Moving average example

- Consider window of size 3
- Compute at center of window
- Compute mean

Date	y	3-MA
2020-02-12	23	
2020-02-13	30	
2020-02-14	70	
2020-02-15	30	
2020-02-16	25	
2020-02-17	22	

Moving average example

- Consider window of size 3
- Compute at center of window
- Compute mean

Date	y	3-MA
2020-02-12	**23**	
2020-02-13	**30**	**41.0**
2020-02-14	**70**	
2020-02-15	30	
2020-02-16	25	
2020-02-17	22	

Moving average example

- Consider window of size 3
- Compute at center of window
- Compute mean
- Move window and iterate

Date	y	3-MA
2020-02-12	**23**	
2020-02-13	**30**	**41.0**
2020-02-14	**70**	
2020-02-15	30	
2020-02-16	25	
2020-02-17	22	

Moving average example

- Consider window of size 3
- Compute at center of window
- Compute mean
- Move window and iterate

Date	y	3-MA
2020-02-12	23	
2020-02-13	**30**	41.0
2020-02-14	**70**	**43.3**
2020-02-15	**30**	
2020-02-16	25	
2020-02-17	22	

Moving average example

- Consider window of size 3
- Compute at center of window
- Compute mean
- Move window and iterate

Date	y	3-MA
2020-02-12	23	
2020-02-13	30	41.0
2020-02-14	**70**	43.3
2020-02-15	**30**	**41.7**
2020-02-16	**25**	
2020-02-17	22	

Moving average example

- Consider window of size 3
- Compute at center of window
- Compute mean
- Move window and iterate

Date	y	3-MA
2020-02-12	23	
2020-02-13	30	41.0
2020-02-14	70	43.3
2020-02-15	**30**	41.7
2020-02-16	**25**	**25.7**
2020-02-17	**22**	

Moving average example

- Consider window of size 3
- Compute at center of window
- Compute mean
- Move window and iterate
- 3-MA is a shorter time series

Date	y	3-MA
2020-02-12	23	
2020-02-13	30	41.0
2020-02-14	70	43.3
2020-02-15	**30**	41.7
2020-02-16	**25**	**25.7**
2020-02-17	**22**	

Moving average example

- Consider window of size 3
- Compute at center of window
- Compute mean
- Move window and iterate
- 3-MA is a shorter time series

Date	y	3-MA
2020-02-12	23	NaN
2020-02-13	30	41.0
2020-02-14	70	43.3
2020-02-15	**30**	41.7
2020-02-16	**25**	**25.7**
2020-02-17	**22**	NaN

Moving average

- Moving average of order m (denoted m-MA):

$$Z_t = \frac{1}{m} \sum_{j=-k}^{j=k} y_{t+j}$$

- where m=2k+1 is the size of the window where k datapoints either side of t are included in the average

- Each data point in the window receives equal weight and the window is symmetric

Even window size

- Often the window size is selected to be the same as the seasonality to smooth out seasonal variation
- Example: Monthly data, yearly seasonality T=12
- With an even window, where do we compute the mean value?
- An odd window size would double count specific months

2020												2021	
JAN	FEB	MAR	APR	MAR	JUN	JUL	AUG	SEP	OCT	NOV	DEC	**JAN**	FEB

Even window size

- Problem: Where should the average value go as there is no obvious centre?
- We could think of the value belonging half way between two rows

Time index	y	4-MA
1	23	
2	30	
3	70	
4	30	
5	25	
6	22	

Even window size

- Problem: Where should the average value go as there is no obvious centre?
- We could think of the value belonging half way between two rows

Time index	y	4-MA
1	23	
1.5		
2	30	
2.5		
3	70	
3.5		
4	30	
4.5		
5	25	
5.5		
6	22	

Even window size

- Problem: Where should the average value go as there is no obvious centre?
- We could think of the value belonging half way between two rows

Time index	y	4-MA
1	**23**	
1.5		
2	**30**	
2.5		**38.25**
3	**70**	
3.5		
4	**30**	
4.5		
5	25	
5.5		
6	22	

Even window size

- Problem: Where should the average value go as there is no obvious centre?
- We could think of the value belonging half way between two rows

Time index	y	4-MA
1	23	
1.5		
2	**30**	
2.5		38.25
3	**70**	
3.5		**38.75**
4	**30**	
4.5		
5	**25**	
5.5		
6	22	

Even window size

- Problem: Where should the average value go as there is no obvious centre?
- We could think of the value belonging half way between two rows

Time index	y	4-MA
1	23	
1.5		
2	30	
2.5		38.25
3	**70**	
3.5		38.75
4	**30**	
4.5		**36.75**
5	**25**	
5.5		
6	**22**	

Even window size

- Problem: Where should the average value go as there is no obvious centre?
- We could think of the value belonging half way between two rows
- Apply another moving average of window size 2 to the 4-MA (aka 2 X 4-MA)

Time index	y	4-MA
1	23	
1.5		
2	30	
2.5		38.25
3	70	
3.5		38.75
4	30	
4.5		36.75
5	25	
5.5		
6	22	

Even window size

- Problem: Where should the average value go as there is no obvious centre?
- We could think of the value belonging half way between two rows
- Apply another moving average of window size 2 to the 4-MA (aka 2 X 4-MA)

Time index	y	4-MA	2 X 4-MA
1	23		
1.5			
2	30		
2.5		38.25	
3	70		
3.5		38.75	
4	30		
4.5		36.75	
5	25		
5.5			
6	22		

Even window size

- Problem: Where should the average value go as there is no obvious centre?
- We could think of the value belonging half way between two rows
- Apply another moving average of window size 2 to the 4-MA (aka 2 X 4-MA)

Time index	y	4-MA	2 X 4-MA
1	23		
1.5			
2	30		
2.5		**38.25**	
3	70		**38.5**
3.5		**38.75**	
4	30		
4.5		36.75	
5	25		
5.5			
6	22		

Even window size

- Problem: Where should the average value go as there is no obvious centre?
- We could think of the value belonging half way between two rows
- Apply another moving average of window size 2 to the 4-MA (aka 2 X 4-MA)

Time index	y	4-MA	2 X 4-MA
1	23		
1.5			
2	30		
2.5		38.25	
3	70		38.5
3.5		**38.75**	
4	30		**37.75**
4.5		**36.75**	
5	25		
5.5			
6	22		

Even window size

- This gives a symmetric window where the weights still sum to one

$$z_t = \frac{1}{2}\left[\frac{1}{4}(y_{t-2} + y_{t-1} + y_t + y_{t+1}) + \frac{1}{4}(y_{t-1} + y_t + y_{t+1} + y_{t+2})\right]$$

$$= \frac{1}{8}y_{t-2} + \frac{1}{4}y_{t-1} + \frac{1}{4}y_t + \frac{1}{4}y_{t+1} + \frac{1}{8}y_{t+2}$$

- The edges of the window share half the weight, this mitigates the double counting discussed earlier

Time index	y	4-MA	2 X 4-MA
1	23		
1.5			
2	30		
2.5		38.25	
3	70		38.5
3.5		38.75	
4	30		37.75
4.5		36.75	
5	25		
5.5			
6	22		

Even window size

- This gives a symmetric window where the weights still sum to one

$$z_t = \frac{1}{2}\left[\frac{1}{4}(y_{t-2} + y_{t-1} + y_t + y_{t+1}) + \frac{1}{4}(y_{t-1} + y_t + y_{t+1} + y_{t+2})\right]$$

$$= \frac{1}{8}y_{t-2} + \frac{1}{4}y_{t-1} + \frac{1}{4}y_t + \frac{1}{4}y_{t+1} + \frac{1}{8}y_{t+2}$$

- The edges of the window share half the weight, this mitigates the double counting discussed earlier

Time index	y	4-MA	2 X 4-MA
1	23		NaN
1.5			
2	30		NaN
2.5		38.25	
3	70		38.5
3.5		38.75	
4	30		37.75
4.5		36.75	
5	25		NaN
5.5			
6	22		NaN

Even window size

- Any even order centered MA can be dealt with by applying an additional 2-MA
- Example: If we wanted m=6 we can compute a 2 x 6-MA to get:

$$z_t = \frac{1}{12}y_{t-3} + \frac{1}{6}y_{t-2} + \frac{1}{6}y_{t-1} + \frac{1}{6}y_t + \frac{1}{6}y_{t+1} + \frac{1}{6}y_{t+2} + \frac{1}{12}y_{t+3}$$

Time index	y	4-MA	2 X 4-MA
1	23		NaN
1.5			
2	30		NaN
2.5		38.25	
3	70		38.5
3.5		38.75	
4	30		37.75
4.5		36.75	
5	25		NaN
5.5			
6	22		NaN

Moving average implementation

pandas.DataFrame.rolling

`DataFrame.rolling`(*window, min_periods=None, center=False, win_type=None, on=None, axis=0, closed=None*) [source]

Provide rolling window calculations.

Parameters: **window** : ***int, offset, or BaseIndexer subclass***

Size of the moving window. This is the number of observations used for calculating the statistic. Each window will be a fixed size.

If its an offset then this will be the time period of each window. Each window will be a variable sized based on the observations included in the time-period. This is only valid for datetimelike indexes.

If a BaseIndexer subclass is passed, calculates the window boundaries based on the defined `get_window_bounds` method. Additional rolling keyword arguments, namely *min_periods*, *center*, and *closed* will be passed to *get_window_bounds*.

min_periods : ***int, default None***

Minimum number of observations in window required to have a value (otherwise result is NA). For a window that is specified by an offset, *min_periods* will default to 1. Otherwise, *min_periods* will default to the size of the window.

center : ***bool, default False***

Set the labels at the center of the window.

Moving average implementation

```
# Compute 3-MA
window_size = 3
df.rolling(window=window_size, center=True).mean()
```

	y
0	NaN
1	41.000000
2	43.333333
3	41.666667
4	25.666667
5	NaN

Moving average implementation

```python
# Compute 2 X 4-MA
window_size = 4
(df.rolling(window=window_size).mean() # Apply the 4-MA without a centered window
                                       # The average is computed at the end of the window
   .rolling(window=2).mean() # Apply the 2-MA without a centred window
                             # The average is computed at the end of the window
   .shift(-window_size // 2) # Shift is required to align the 2x4-MA to what a centered window would have produced
                             # Integer division is used as shift() requires an int

)
```

	y
0	NaN
1	NaN
2	38.50
3	37.75
4	NaN
5	NaN

Summary

Moving average computes the mean of the data over a window across a time series

The window size, m, defines the order of the moving average denoted as m-MA

An even ordered centered moving average can be obtained by applying an additional 2-MA