Rolling statistics

Outliers

Contents

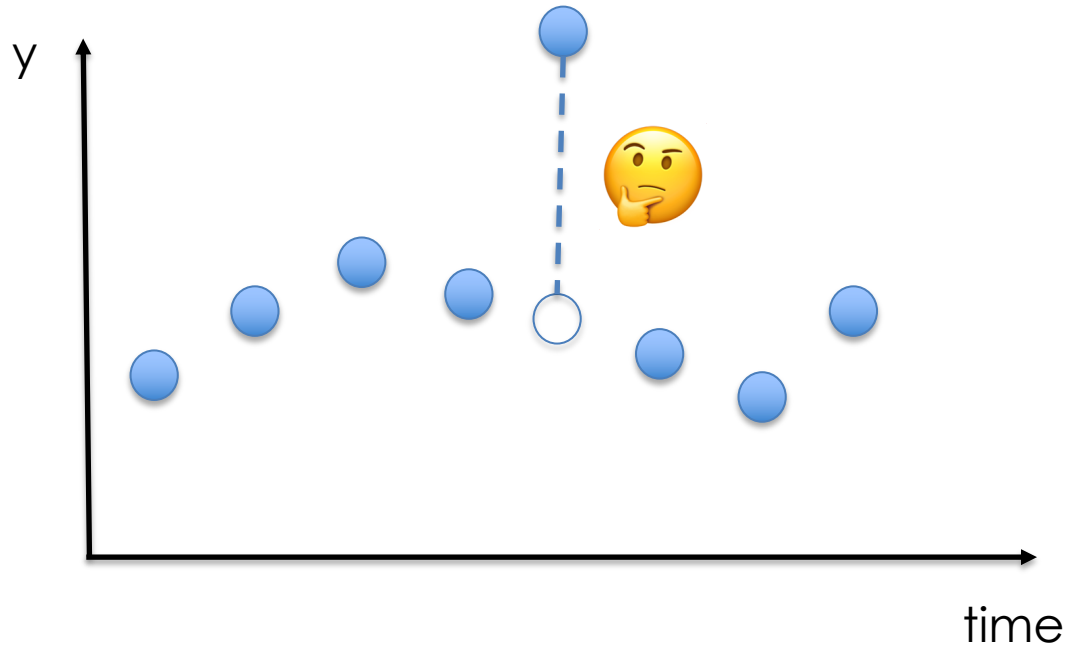
ESTIMATION METHODS

ROLLING STATISTICS

MOTIVATE ROLLING MEAN

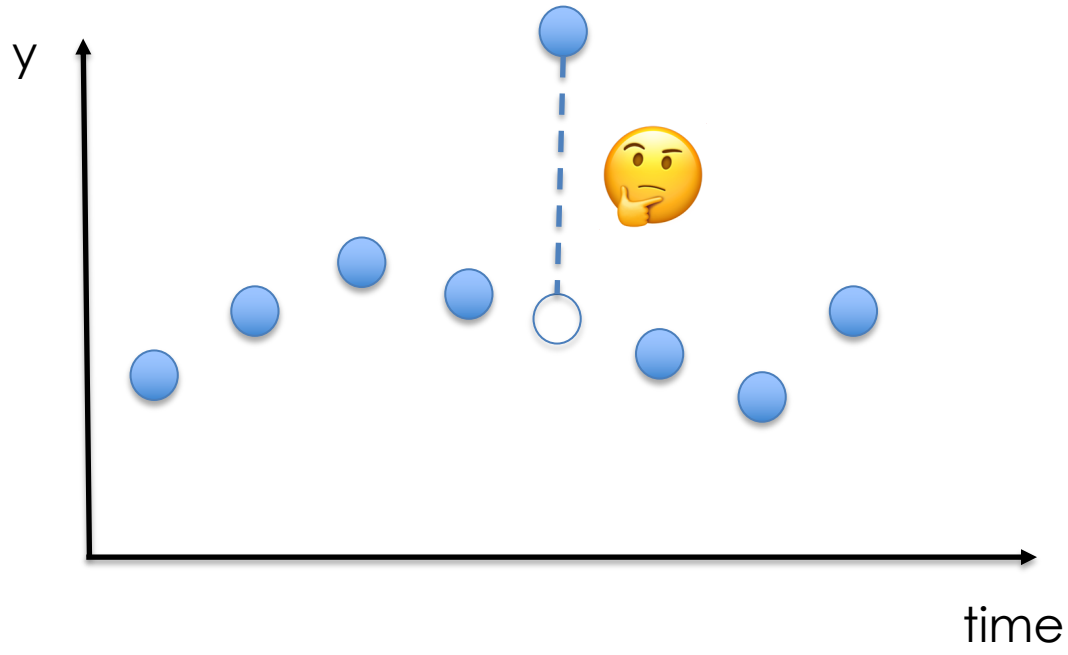
Estimation methods

- Examine each actual: y_t
- Compute expected value: $\hat{y}_t$
- Is actual very different than expected?
- If yes, flag as an outlier
- Formally: $|y_t - \hat{y}_t| > \delta$, where δ is a threshold to select outliers



Estimation methods

- How to determine $\hat{y}_t$?
- How to pick threshold δ?
- Different methods answer these questions
- Estimation methods use values before and after t to get $\hat{y}_t$
- Let's talk about rolling statistics (e.g., moving averages)



Rolling statistics

- Apply a window to data
- Compute statistics from the data inside the window
- Move window and iterate through all the data

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

Rolling statistics example

- Consider window of size 3

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

Rolling statistics example

- Consider window of size 3

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

Rolling statistics example

- Consider window of size 3
- Compute at center of window
- Compute mean and standard deviation

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

Rolling statistics example

- Consider window of size 3
- Compute at center of window
- Compute mean and standard deviation

Date	y	mean	std
2020-02-12	**23**		
2020-02-13	**30**	**41.0**	**25.4**
2020-02-14	**70**		
2020-02-15	30		
2020-02-16	25		
2020-02-17	22		

Rolling statistics example

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

Date	y	mean	std
2020-02-12	**23**		
2020-02-13	**30**	**41.0**	**25.4**
2020-02-14	**70**		
2020-02-15	30		
2020-02-16	25		
2020-02-17	22		

Rolling statistics example

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

Date	y	mean	std
2020-02-12	23		
2020-02-13	**30**	41.0	25.4
2020-02-14	**70**	**43.3**	**23.1**
2020-02-15	**30**		
2020-02-16	25		
2020-02-17	22		

Rolling statistics example

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

Date	y	mean	std
2020-02-12	23		
2020-02-13	30	41.0	25.4
2020-02-14	70	43.3	23.1
2020-02-15	**30**	41.7	24.7
2020-02-16	**25**	**25.7**	**4.04**
2020-02-17	**22**		

Rolling statistics example

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

Date	y	mean	std
2020-02-12	23		
2020-02-13	30	41.0	25.4
2020-02-14	70	43.3	23.1
2020-02-15	**30**	41.7	24.7
2020-02-16	**25**	**25.7**	**4.04**
2020-02-17	**22**		

Rolling statistics example

- What about edge cases?

Date	y	mean	std
2020-02-12	23		
2020-02-13	30	41.0	25.4
2020-02-14	70	43.3	23.1
2020-02-15	30	41.7	24.7
2020-02-16	25	25.7	4.04
2020-02-17	22		

Rolling statistics example

- What about edge cases?
- Option 1: Treat as missing data

Date	y	mean	std
2020-02-12	23	NaN	NaN
2020-02-13	30	41.0	25.4
2020-02-14	70	43.3	23.1
2020-02-15	30	41.7	24.7
2020-02-16	25	25.7	4.04
2020-02-17	22	NaN	NaN

Rolling statistics example

- Pros: All rolling statistics have same window size
- Cons: May need to truncate data depending on application due to missing data at edges

Date	y	mean	std
2020-02-12	23	NaN	NaN
2020-02-13	30	41.0	25.4
2020-02-14	70	43.3	23.1
2020-02-15	30	41.7	24.7
2020-02-16	25	25.7	4.04
2020-02-17	22	NaN	NaN

Rolling statistics example

- What about edge cases?

Date	y	mean	std
2020-02-12	23		
2020-02-13	30	41.0	25.4
2020-02-14	70	43.3	23.1
2020-02-15	30	41.7	24.7
2020-02-16	25	25.7	4.04
2020-02-17	22		

Rolling statistics example

- What about edge cases?
- Option 2: Use smaller window at edges and don't center window

Date	y	mean	std
2020-02-12	**23**	**26.5**	**4.95**
2020-02-13	**30**	41.0	25.4
2020-02-14	70	43.3	23.1
2020-02-15	30	41.7	24.7
2020-02-16	**25**	25.7	4.04
2020-02-17	**22**	**23.5**	**2.12**

Rolling statistics example

- **Pro**: No missing data
- **Con**: Statistics at edges are based on smaller sample sizes and asymmetric window, resulting in overfitting at the edges

Date	y	mean	std
2020-02-12	**23**	**26.5**	**4.95**
2020-02-13	**30**	41.0	25.4
2020-02-14	70	43.3	23.1
2020-02-15	30	41.7	24.7
2020-02-16	**25**	25.7	4.04
2020-02-17	**22**	**23.5**	**2.12**

Rolling statistics 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.

Rolling statistics implementation

```
df = pd.DataFrame(data=[23, 30, 70, 30, 25, 22], columns=['y'])
```

```
df.rolling(window=3, center=True).agg(['mean', 'std'])
```

	y	
	mean	std
0	NaN	NaN
1	41.000000	25.357445
2	43.333333	23.094011
3	41.666667	24.664414
4	25.666667	4.041452
5	NaN	NaN

Rolling statistics implementation

```
df = pd.DataFrame(data=[23, 30, 70, 30, 25, 22], columns=['y'])
```

```
df.rolling(window=3, center=True, min_periods=1).agg(['mean', 'std'])
```

	y	
	mean	std
0	26.500000	4.949747
1	41.000000	25.357445
2	43.333333	23.094011
3	41.666667	24.664414
4	25.666667	4.041452
5	23.500000	2.121320

Returning to estimation methods

- Expected value, $\hat{y}_t$: Use the rolling mean
- Threshold, δ: Use the rolling standard deviation
- Recap: Values outside mean ± 3 × standard deviations can be considered outliers
- $|y_t - \hat{y}_t| > \alpha\hat{\sigma}_t$ where $\alpha = 3$