

# Rolling window features: part 2

---

Window features

# How to pick the window size & statistics?

Date	Sales
2020-02-12	<b>35</b>
2020-02-13	<b>30</b>
2020-02-14	<b>23</b>
2020-02-15	30
2020-02-16	34
2020-02-17	?

Date	Sales
2020-02-12	<b>35</b>
2020-02-13	<b>30</b>
2020-02-14	<b>23</b>
2020-02-15	<b>30</b>
2020-02-16	34
2020-02-17	?

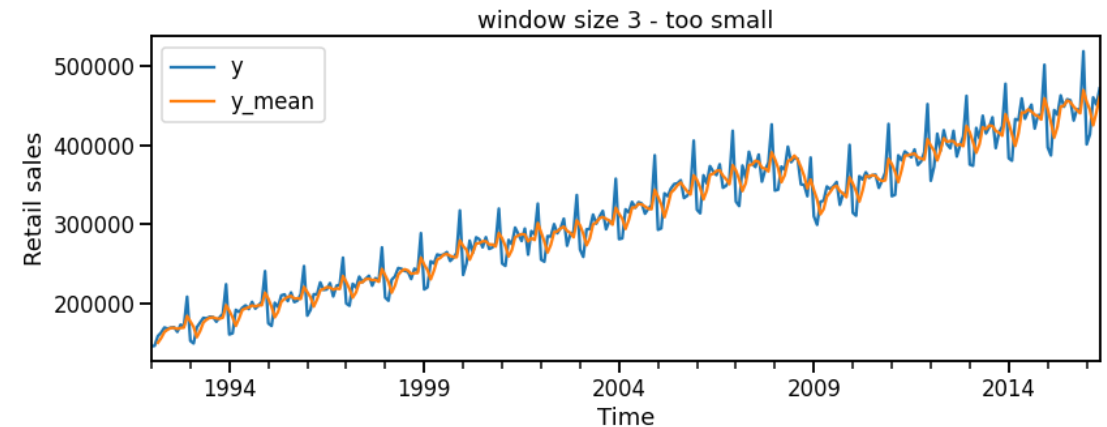
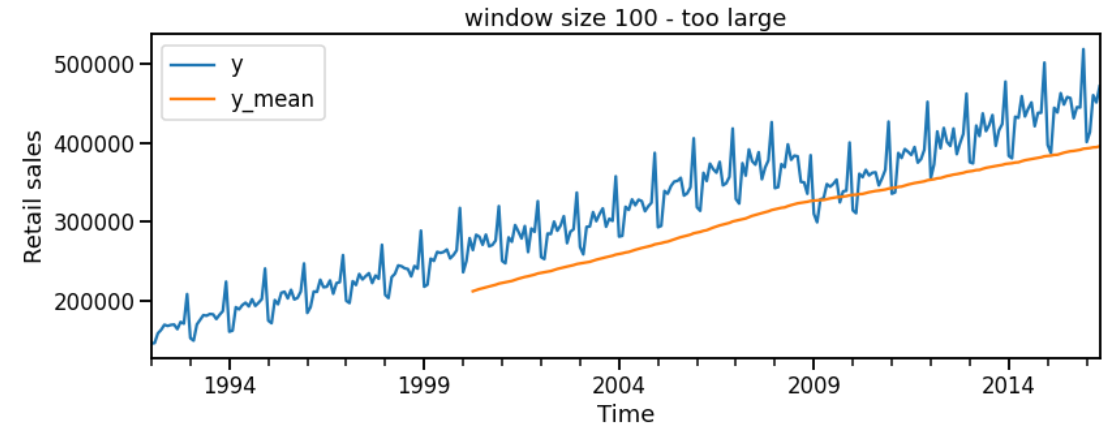
Mean?



Kurtosis?

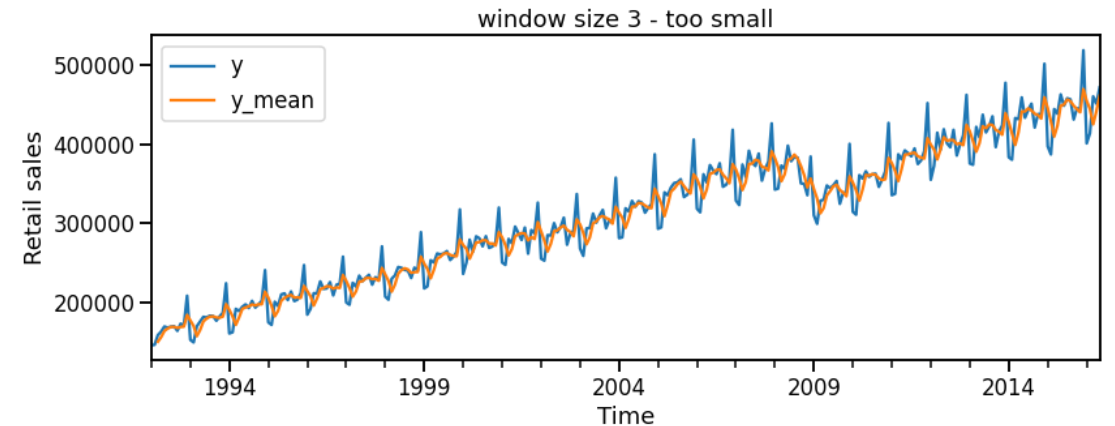
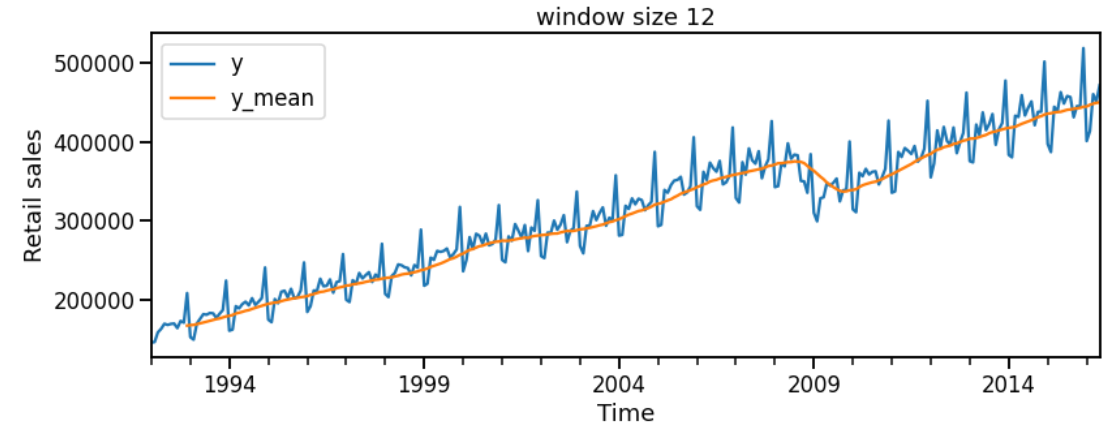
# How to pick the window size?

- If the window size is too large:
  - Won't capture the local behavior of the time series.
  - Potentially a lot of missing data at the edges.
- If the window size is too small:
  - The summary statistic is computed over a small set of numbers and may not add much additional information.
- Different window sizes can capture different behaviors in the data (e.g., short term vs long term trends) → try different window sizes.



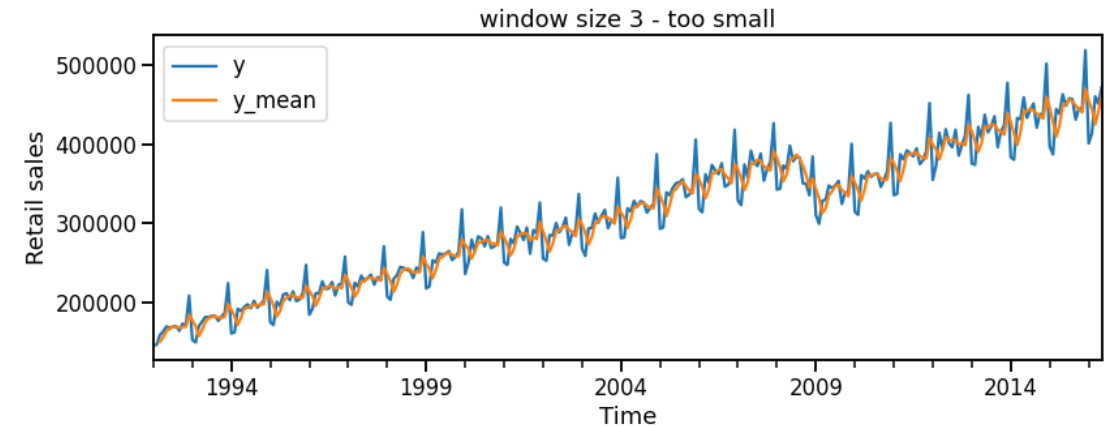
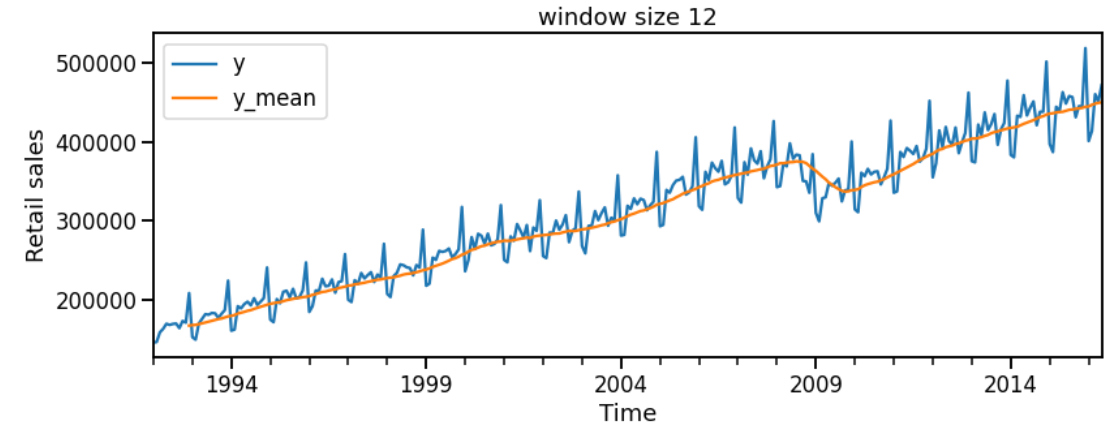
# How to pick the window size?

- If the window size is too large:
  - Won't capture the local behavior of the time series.
  - Potentially a lot of missing data at the edges.
- If the window size is too small:
  - The summary statistic is computed over a small set of numbers and may not add much additional information.
- Different window sizes can capture different behaviors in the data (e.g., short term vs long term trends) → try different window sizes.



# How to pick the window size?

- Heuristic: Set the window size(s) to be the same as the seasonal period(s) to smooth over the multiple seasonalities.
- Try multiple window sizes, use feature selection methods, and assess the performance of the model.
- Try nested window features where we use multiple window sizes on different time scales [1, 2].



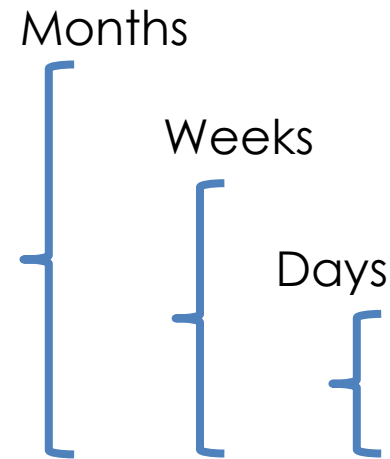
[1] Lazzeri, F., 2020. *Machine learning for time series forecasting with Python*. John Wiley & Sons.

[2] Lazzeri, F., 2020 [Introduction to feature engineering for time series forecasting](#)

# How to pick the window size?

- Heuristic: Set the window size(s) to be the same as the seasonal period(s) to smooth over the multiple seasonalities.
- Try multiple window sizes, use feature selection methods, and assess the performance of the model.
- Try nested window features where we use multiple window sizes on different time scales [1, 2].

Months  
Weeks  
Days



Date	Sales
2020-02-12	35
2020-02-13	30
2020-02-14	23
2020-02-15	30
2020-02-16	34
2020-02-17	?

[1] Lazzeri, F., 2020. *Machine learning for time series forecasting with Python*. John Wiley & Sons.

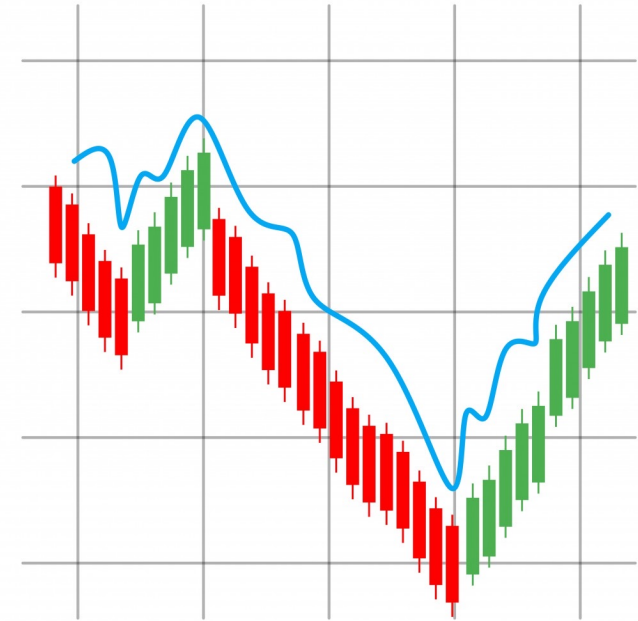
[2] Lazzeri, F., 2020 [Introduction to feature engineering for time series forecasting](#)

# Nested rolling window features

Date	Sales	Rolling mean Sales (days)	Rolling mean Sales (weeks)	Rolling mean Sales (months)
2020-02-12	35	33.2	29.2	29.2
2020-02-13	30	29.1	28.4	28.4
2020-02-14	23	24.5	20.5	20.5
2020-02-15	30	30.1	27.1	27.1
2020-02-16	34	31.5	28.5	28.5
2020-02-17	?	31.2	31.2	23.5

# Why use nested rolling window features?

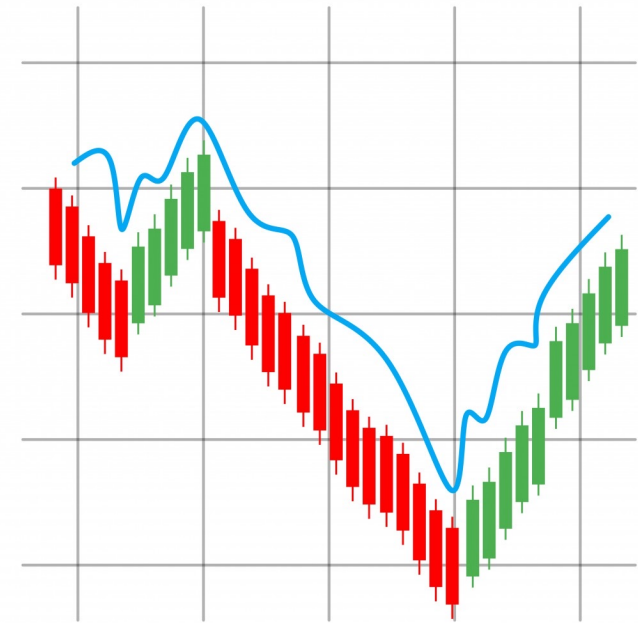
- Nested rolling windows can help capture information at different time scales:
  - Trend in short term (weeks) vs long term (months) in the target and features.
- Example from finance:
  - Momentum: Tendency of an increasing/decreasing stock price to continue to increase/decrease.
  - One indicator of momentum is the difference between a shorter term moving average and a longer term moving average.





# When to use nested window features?

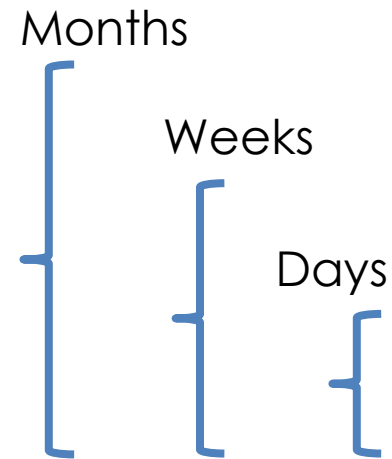
- No theoretically correct answer.
- Experiment, try feature selection, and check model performance.



# How to pick the window size?

- Heuristic: Set the window size(s) to be the same as the seasonal period(s) to smooth over the multiple seasonalities.
- Try multiple window sizes, use feature selection methods, and assess the performance of the model.
- Try nested window features where we use multiple window sizes on different time scales [1, 2].

Months  
Weeks  
Days

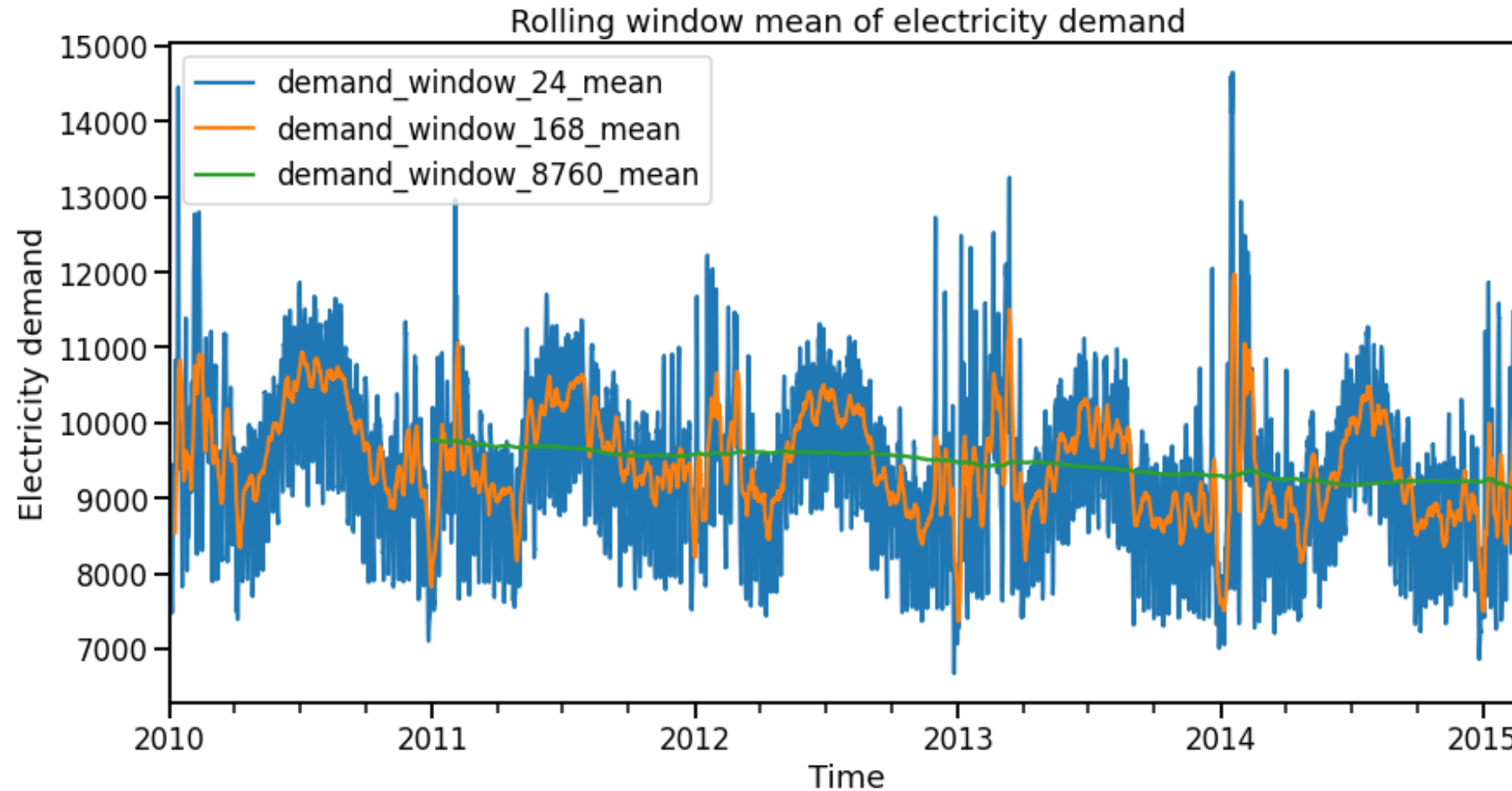


Date	Sales
2020-02-12	35
2020-02-13	30
2020-02-14	23
2020-02-15	30
2020-02-16	34
2020-02-17	?

[1] Lazzeri, F., 2020. *Machine learning for time series forecasting with Python*. John Wiley & Sons.

[2] Lazzeri, F., 2020 [Introduction to feature engineering for time series forecasting](#)

# Example: Hourly electricity demand



Moving averages over different time scales capture different behaviours (e.g., the yearly average captures long term trends).

# How to pick which statistics?

- Mean is very common to smooth the data.
- Standard deviation measures volatility which can sometimes be predictive.
- Without prior knowledge, it is difficult to know if other summary stats is useful.



## Rolling window functions

<code>Rolling.count()</code>	Calculate the rolling count of non NaN observations.
<code>Rolling.sum(*args[, engine, engine_kwargs])</code>	Calculate the rolling sum.
<code>Rolling.mean(*args[, engine, engine_kwargs])</code>	Calculate the rolling mean.
<code>Rolling.median([engine, engine_kwargs])</code>	Calculate the rolling median.
<code>Rolling.var([ddof, engine, engine_kwargs])</code>	Calculate the rolling variance.
<code>Rolling.std([ddof, engine, engine_kwargs])</code>	Calculate the rolling standard deviation.
<code>Rolling.min(*args[, engine, engine_kwargs])</code>	Calculate the rolling minimum.
<code>Rolling.max(*args[, engine, engine_kwargs])</code>	Calculate the rolling maximum.
<code>Rolling.corr([other, pairwise, ddof])</code>	Calculate the rolling correlation.
<code>Rolling.cov([other, pairwise, ddof])</code>	Calculate the rolling sample covariance.
<code>Rolling.skew(**kwargs)</code>	Calculate the rolling unbiased skewness.
<code>Rolling.kurt(**kwargs)</code>	Calculate the rolling Fisher's definition of kurtosis without bias.

# How to pick which statistics?

- Keep it simple and use mean and standard deviation.
- Is the accuracy good enough? Then don't add more features than needed.
- Try a variety of features and use feature selection & modelling techniques (e.g., LASSO to reduce number of features).



## Rolling window functions

<code>Rolling.count()</code>	Calculate the rolling count of non NaN observations.
<code>Rolling.sum(*args[, engine, engine_kwargs])</code>	Calculate the rolling sum.
<code>Rolling.mean(*args[, engine, engine_kwargs])</code>	Calculate the rolling mean.
<code>Rolling.median([engine, engine_kwargs])</code>	Calculate the rolling median.
<code>Rolling.var([ddof, engine, engine_kwargs])</code>	Calculate the rolling variance.
<code>Rolling.std([ddof, engine, engine_kwargs])</code>	Calculate the rolling standard deviation.
<code>Rolling.min(*args[, engine, engine_kwargs])</code>	Calculate the rolling minimum.
<code>Rolling.max(*args[, engine, engine_kwargs])</code>	Calculate the rolling maximum.
<code>Rolling.corr([other, pairwise, ddof])</code>	Calculate the rolling correlation.
<code>Rolling.cov([other, pairwise, ddof])</code>	Calculate the rolling sample covariance.
<code>Rolling.skew(**kwargs)</code>	Calculate the rolling unbiased skewness.
<code>Rolling.kurt(**kwargs)</code>	Calculate the rolling Fisher's definition of kurtosis without bias.