Calculating The Moving Average Prices And Volatilities

Jerzy Pawlowski (jpawlowski@machinetrader.io (mailto:jpawlowski@machinetrader.io))

01/21/2024

The Moving Average Price

The moving average prices can be calculated for a streaming time series of prices p_i.

The Simple Moving Average (SMA) price p_i^{SMA} is the average of the past prices p_{i-j} over a look-back time interval n:

$$p_i^{SMA} = \frac{1}{n}\sum_{j=1}^{n} p_{i-j}$$

The drawback of the SMA is that it applies the same weight to the recent and past prices. Its calculation also requires maintaining a buffer (queue) of past prices.

The Exponential Moving Average (EMA) price p_i^{EMA} is calculated using a decay factor λ and the recursive formula:

$$p_i^{EMA} = \lambda p_{i-1}^{EMA} + (1-\lambda)p_{i-1}$$

The advantage of the EMA is that it applies a greater weight to more recent prices than to past ones. Its calculation is also faster because there's less computation, and it doesn't require maintaining a buffer (queue) of past prices.

The SMA and EMA values are not the same, but they are related to each other, and they move in sympathy with each other.

If the decay factor λ is closer to one then the EMA adjusts slowly to new prices, corresponding to a longer look-back time interval. If λ is closer to zero then the EMA adjusts quickly to new prices, corresponding to a shorter look-back time interval.

The decay factor λ determines the memory, i.e. how quickly the effect of past data fades over time. If the decay factor λ is closer to one then the effect of past data fades slowly over time. If λ is closer to zero then the effect of past data fades quickly over time.

The Moving Average Variance of Returns

The moving average variance of returns σ_i^2 measures the dispersion of the streaming returns around the moving average return. It can be calculated for a streaming time series of prices p_i.

The dollar return r_i is the difference between the current price p_i minus the previous price p_{i-1}:

$$r_i = p_i - p_{i-1}$$

The Simple Moving Average (SMA) return r_i^{SMA} is equal to the average of the past returns:

$$r_i^{SMA} = \frac{1}{n}\sum_{j=1}^{n} r_{i-j}$$

The Simple Moving Average (SMA) variance σ_i^{2SMA} is then given by the sum:

$$\sigma_i^{2SMA} = \frac{1}{n-1}\sum_{j=1}^{n}(r_{i-j} - r_{i-1}^{SMA})^2$$

Note that the average return r_{i-1}^{SMA} is from the previous time step, to better capture the unexpected return in the current time step.

The Exponential Moving Average (EMA) variance σ_i^{2EMA} is calculated using the two recursive formulas:

$$\sigma_i^{2EMA} = \lambda\sigma_{i-1}^{2EMA} + (1-\lambda)(r_i - r_{i-1}^{EMA})^2$$

$$r_i^{EMA} = \lambda r_{i-1}^{EMA} + (1-\lambda)r_{i-1}$$

The SMA and EMA variances are not the same, but they are related to each other, and they move in sympathy with each other.

If the decay factor λ is closer to one then the EMA variance adjusts slowly to new prices, corresponding to a longer look-back time interval. If λ is closer to zero then the EMA variance adjusts quickly to new prices, corresponding to a shorter look-back time interval.

The Moving Average Variance of Prices

The moving average variance of prices σ_i^2 measures the dispersion of the streaming prices around the moving average price. It can be calculated for a streaming time series of prices p_i.

The Simple Moving Average (SMA) price p_i^{SMA} is equal to the average of the past prices:

$$p_i^{SMA} = \frac{1}{n}\sum_{j=1}^{n} p_{i-j}$$

The Simple Moving Average (SMA) variance σ_i^{2SMA} is then given by the sum:

$$\sigma_i^{2SMA} = \frac{1}{n-1}\sum_{j=1}^{n}(p_{i-j} - p_{i-1}^{SMA})^2$$

Note that the average price p_{i-1}^{SMA} is from the previous time step, to better capture the unexpected price in the current time step.

The Exponential Moving Average (EMA) variance of prices σ_i^{2EMA} is calculated using the two recursive formulas:

$$\sigma_i^{2EMA} = \lambda\sigma_{i-1}^{2EMA} + (1-\lambda)(p_i - p_{i-1}^{EMA})^2$$

$$p_i^{EMA} = \lambda p_{i-1}^{EMA} + (1-\lambda)p_{i-1}$$

The variance of returns and of prices measure different forms of dispersion.

Take for example the extreme case when the returns are constant $r_i = r$.
Then the prices $p_i = \sum r_i$ increase in a straight line. The variance of returns is zero, but the variance of prices is not zero. So if the prices exhibit a significant trend, then the variance of prices is large, even if the variance of returns is small.

Another example is Brownian motion. The returns are normally distributed, with a constant volatility. But the variance of prices is not constant, and it increases with the factor λ, since then prices have more time to drift.

Implementation of the Moving Average Price and Volatility

The Node-RED flow named *Tech indicators* contains implementations of various technical indicators in MachineTrader, including the moving average price and volatility.

The flow named *Tech indicators* calculates the moving average prices and volatilities from live streaming stock prices for a single selected stock.

You can download this flow from the **MachineTrader-Community repository on GitHub** (https://github.com/predictivetechnologysystems/MachineTrader-Community).

You must also download the flows named *Alpaca Prices* and *Globals*. After you download the flow named *Tech indicators*, you must connect the *link-in node* to the left of the function node named *Get stock price* to the *link-out node* called *Prices out* in the flow *Alpaca Prices*.

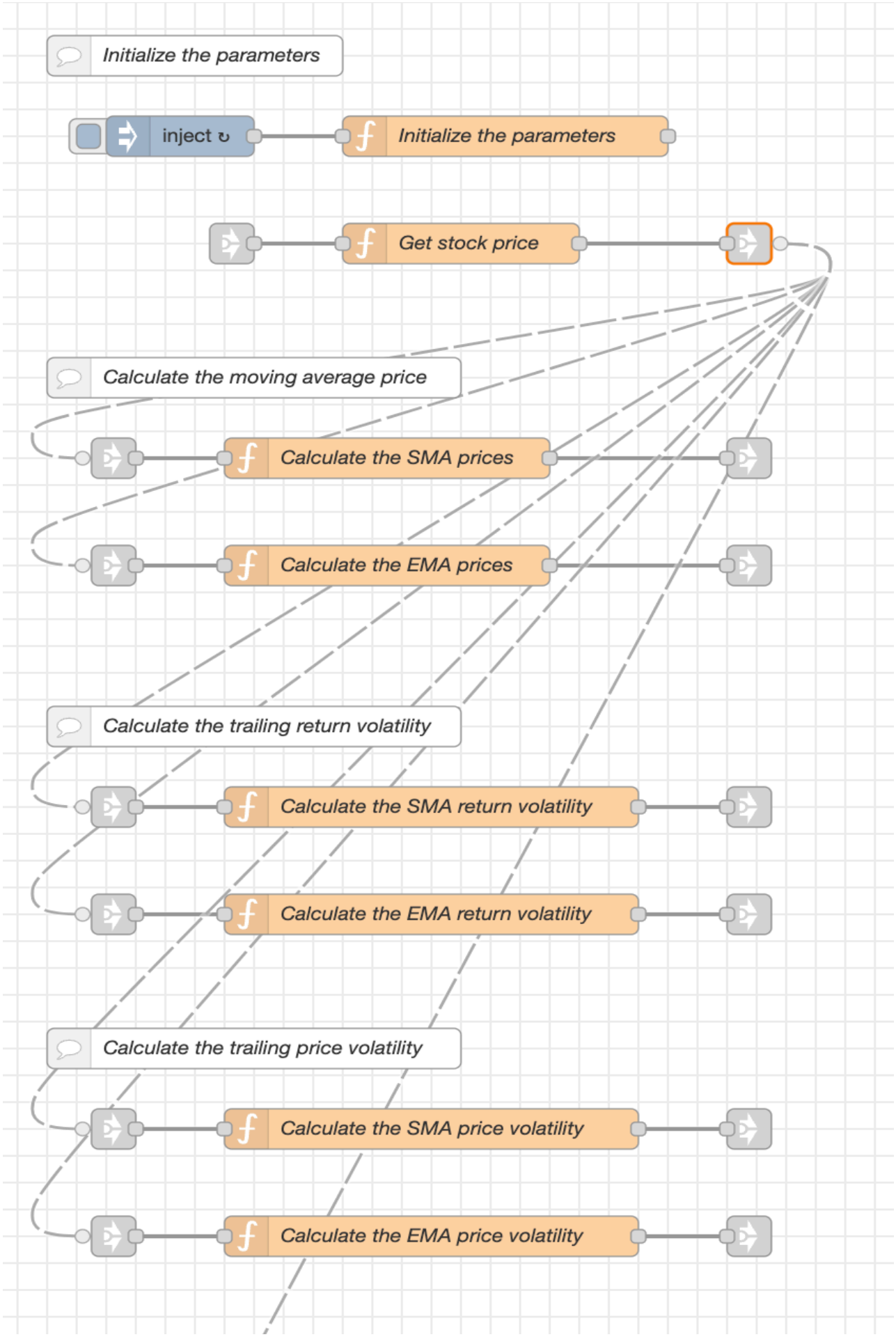
You can also watch an instructional video about **Calculating Moving Average Prices And Volatilities** (https://youtu.be/iAZMTEZeZDM) on the MachineTrader YouTube channel.

Let's take a look at the flow named *Tech indicators*.

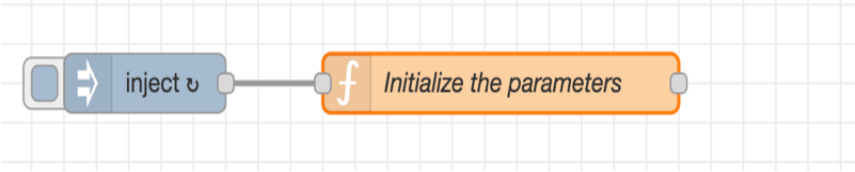
The function node named *Initialize the parameters* creates flow variables needed for the calculations. It should be run once before the other nodes are run, by pressing the *inject* node on the left.

The multiple stock prices stream in from the flow *Alpaca Prices* into the node named *Get stock price*, through the small *link-in node* to its left. The function node *Get stock price* selects the prices only for the ticker in the flow variable *symboln*. The single stock prices then stream out through the small *link-out node* to its right, to the nodes which calculate the moving averages.

The dashed lines are the connectors for passing the live streaming stock prices to the function nodes.



The function node named *Initialize the parameters* creates flow variables needed for the calculations:



It should be run once before the other nodes are run, by pressing the *inject* node on the left. The *inject* node is on a timer, to run on weekdays at 9:30 AM, so that all the flow variables are initialized before the start of stock trading.

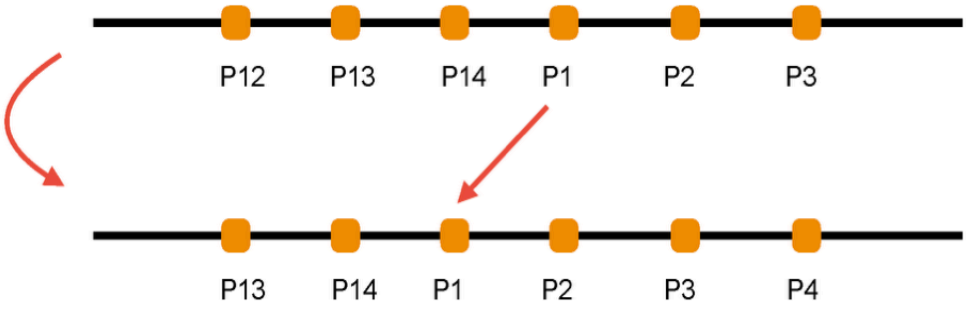
The function node named *Initialize the parameters* creates the following flow variables:

- symboln = the ticker string of the selected stock.
- lambdaf = the lambda decay factor which multiplies past estimates.
- volf = the volatility floor.
- pricec = the current stock price.
- pricep = the previous stock price, initially set to NaN.
- pricema = the moving average price, initially set to NaN.
- lookb = the look-back interval, i.e. the number of elements in the data queue.
- priceq = the data queue (buffer) for storing the recent stock prices.
- endpq = the position of the end of the price queue.
- retq = the data queue (buffer) for storing the recent stock returns.
- endrq = the position of the end of the returns queue.

The data queue is used for calculating the simple moving averages of prices and volatilities.

The data queue is an array (vector) for storing the recent stock prices or returns. The data queue only stores the most recent number of elements equal to *lookb*.
The data queue is a first in last out queue. Data is added one element at a time and it's stored in the order it's received. New data is written to the end of the queue, and the *endq* variable is shifted to the next to last element.
The element *P14* at the end of the queue is replaced with new data and becomes the element *P1* at the front of the queue.
The element *P13* becomes the end of the queue *P14*. The data elements in the queue are just relabeled - not copied. This avoids copying the other data elements in the queue, which speeds up the code.

Below is an illustration of how an element is added to the data queue:

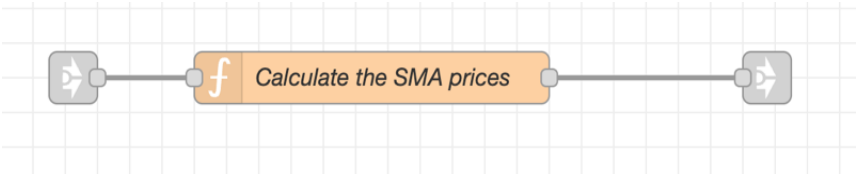


The function node named *Get stock price* obtains the streaming prices for the ticker *symboln* from the flow named *Alpaca Prices*.

The prices for all the stocks stream from the flow *Alpaca Prices* into the small *link-in node* on the left. The function node *Get stock price* then extracts the prices only for the ticker *symboln*, and it passes them to the small *link-out node* on the right. The streaming stock prices from the *link-out node* can then be passed to the other nodes and strategy flows.

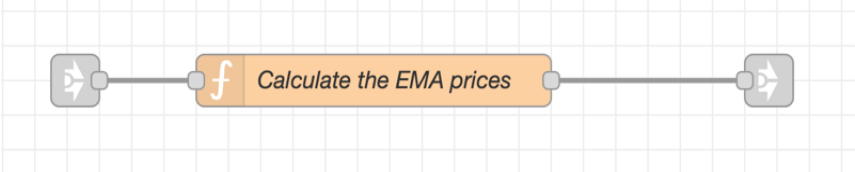
For the live prices to stream, it's necessary to download and install the flows named *Alpaca Prices* and *Globals*. Then you must connect the *link-in node* to the left of the function node named *Get stock price* to the *link-out node* called *Prices out* in the flow *Alpaca Prices*.

The function node named *Calculate the SMA prices* calculates the simple moving average prices.



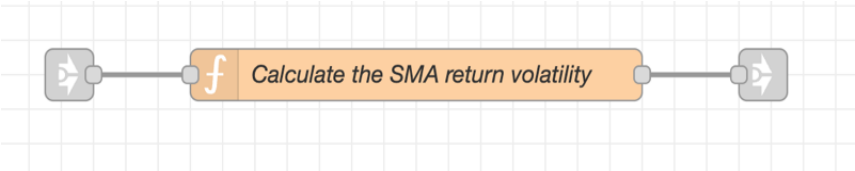
The stock prices flow from the node *Get stock price* into the small *link-in node* on the left. The function node *Calculate the SMA prices* then calculates the simple moving average prices, and it passes them to the small *link-out node* on the right. The streaming SMA prices from the *link-out node* can then be passed to the other nodes and strategy flows.

The function node named *Calculate the EMA prices* calculates the exponential moving average prices.



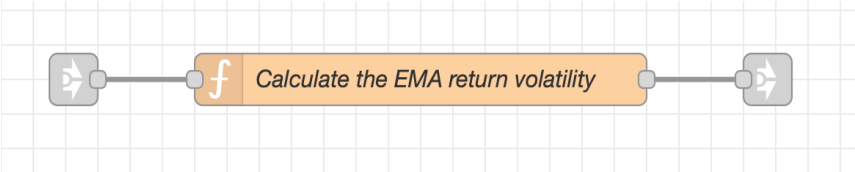
The stock prices flow from the node *Get stock price* into the small *link-in node* on the left. The function node *Calculate the EMA prices* then calculates the exponential moving average prices, and it passes them to the small *link-out node* on the right. The streaming EMA prices from the *link-out node* can then be passed to the other nodes and strategy flows.

The function node named *Calculate the SMA return volatility* calculates the simple moving average volatility of the stock returns.



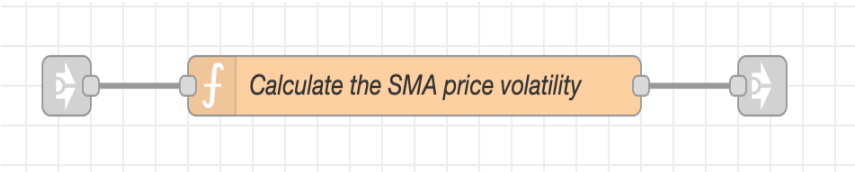
The stock prices flow from the node *Get stock price* into the small *link-in node* on the left. The function node *Calculate the SMA return volatility* then calculates the SMA return volatility, and it passes it to the small *link-out node* on the right. The streaming SMA return volatility from the *link-out node* can then be passed to the other nodes and strategy flows.

The function node named *Calculate the EMA return volatility* calculates the exponential moving average volatility of the stock returns.



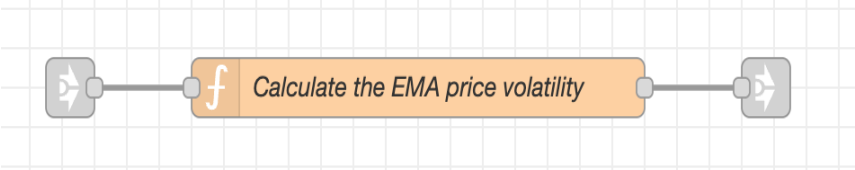
The stock prices flow from the node *Get stock price* into the small *link-in node* on the left. The function node *Calculate the EMA return volatility* then calculates the EMA return volatility, and it passes it to the small *link-out node* on the right. The streaming EMA return volatility from the *link-out node* can then be passed to the other nodes and strategy flows.

The function node named *Calculate the SMA price volatility* calculates the simple moving average volatility of the stock prices.



The stock prices flow from the node *Get stock price* into the small *link-in node* on the left. The function node *Calculate the SMA price volatility* then calculates the SMA price volatility, and it passes it to the small *link-out node* on the right. The streaming SMA price volatility from the *link-out node* can then be passed to the other nodes and strategy flows.

The function node named *Calculate the EMA price volatility* calculates the exponential moving average volatility of the stock prices.



The stock prices flow from the node *Get stock price* into the small *link-in node* on the left. The function node *Calculate the EMA price volatility* then calculates the EMA price volatility, and it passes it to the small *link-out node* on the right. The streaming EMA price volatility from the *link-out node* can then be passed to the other nodes and strategy flows.