Chart information
Bar state
Math
Time & date
Time zone
Session
Technical analysis
Moving averages
Moving averages overview
Simple average (SMA)
Exponential average (EMA)
Weighted average (WMA)
Hull average (HMA)
Relative average (RMA)
EMA versus RMA
Symmetrical average (SWMA)
Arnaud Legoux (ALMA)
Volume-Weighted (VWMA)
Volume-Weighted (VWAP)
Select average with code
Example indicators
5) VISUAL OUTPUT
Colours
Bar colour
Background colour
Plots

Calculate the Relative Moving Average (RMA) in TradingView's Pine Script

SUN THIS LESSON
Intro
Default pattern
Quick examples
Features of `ta.rma()`
Calculation process
Example script
Summary

In Pine Script, we calculate the Relative Moving Average (RMA) with the `ta.rma()` function [1].

A Relative Moving Average adds more weight to recent data (and gives less importance to older data). This makes the RMA similar to the Exponential Moving Average, although it's somewhat slower to respond than an EMA is.

Default pattern

The function has this definition [1]:

```
ta.rma(source, length)
```

- `source` is the series of numerical values to process. It's the (price) data we calculate the RMA on.
- `length` is an integer with the moving average length in bars. This is the lookback period over which Pine Script calculates the average.

`ta.rma()` returns the Relative Moving Average as a floating-point value [1].

Quick examples

So to get the RMA we give `ta.rma()` two things: the data to analyse and the length to calculate on. Let's see how that goes in practice.

A typical task is to get the moving average of the bar's close prices. To get a Relative Moving Average from that data, we run the function on the `close` variable. For example:

```
// Get the 12-bar relative moving average of close prices
closeAverage = ta.rma(close, 12)

plot(closeAverage)
```

Besides the bar's prices, a Relative Moving Average can calculate on any series with numbers. Say we want the relative average of the volume. For that we have `ta.rma()` process the `volume` variable:

```
// Calculate the 20-bar RMA of volume
volumeAverage = ta.rma(volume, 20)

plot(volume, style=plot.style_histogram)
plot(volumeAverage)
```

The `ta.rma()` function isn't limited to only built-in variables. We can, in fact, run the function on any numerical expression that has a value on each bar. If we, for instance, want the average 5-bar difference in volume, we do:

```
// Get the 15-bar average of the 5-bar change in volume
volChangeAvg = ta.rma(volume - volume[5], 15)

plot(volChangeAvg)
```

Another option is to calculate the average on data returned by another function. Say we want to smooth the Commodity Channel Index (CCI). To do that we first calculate the CCI with the `ta.cci()` function. Then have `ta.rma()` process the data that `ta.cci()` returned:

```
// Get the 10-bar average of the 12-bar Commodity Channel Index
ciValue = ta.cci(close, 12)
rmaValue = ta.rma(ciValue, 10)

plot(ciValue)
plot(rmaValue, color=color.orange)
```

If our script doesn't need the CCI as a separate value, we can also place the `ta.cci()` function directly inside `ta.rma()`. That creates more compact code:

```
// Get the 10-bar average of the 12-bar Commodity Channel Index
rmaCciValue = ta.rma(ta.cci(close, 12), 10)

plot(rmaCciValue, color=color.orange)
```

Features of `ta.rma()`

- The Relative Moving Average isn't a well-known moving average. But TradingView uses this average with two popular indicators: the Relative Strength Index (RSI) and Average True Range (ATR) [1] [2].
- The weighting factors that the Relative Moving Average uses decrease exponentially. That way recent bars have the highest weight, while earlier bars get smaller weights the older they are.
- The `ta.rma()` function can cause script repainting. To calculate the RMA, Pine Script uses data from the previous bar. That makes the average sensitive to which bar the script starts its calculations on. Because TradingView chart's don't have a fixed date range, the script's first bar isn't always the same. (However, on charts with thousands of bars this effect is incredibly small.)

Calculation process

TradingView calculates the Relative Moving Average with this following formula [see 1]:

$$\text{RMA} = \text{alpha} * \text{source} + (1 - \text{alpha}) * \text{RMA}[1]$$

$$\text{alpha} = \frac{1}{\text{length}}$$

In terms of code, Pine Script implements the RMA as follows [1]:

```
pine_rma(source, length) =>
    alpha = 1 / length
    sum = 0.0
    sum := na(sum[1]) ?
        ta.sma(source, length) :
        alpha * source + (1 - alpha) * nz(sum[1])

plot(pine_rma(close, 15))
```

For convenience we better use the `ta.rma()` function instead of this custom function. There's also a chance that the built-in `ta.rma()` function runs more efficient due to optimisations TradingView performs behind the scenes.

Example script

Let's see how a complete script can use Relative Moving Averages (RMAs). The indicator below calculates two of those averages. They use close prices and have a length of 10 and 40 bars.

We plot both RMAs as a solid line. Then we compare the current price against the averages to determine the market's trend. Based on that outcome we give the chart's price bars a certain colour.

The indicator's full code is:

```
//@version=6
indicator(title="Relative Moving Average example", overlay=true)

// Input options
averageData = input.source(close, title="Average Data Source")

fastLength = input.int(10, title="Fast Average length")
slowLength = input.int(40, title="Slow Average length")

// Calculate relative moving averages
fastAverage = ta.rma(averageData, fastLength)
slowAverage = ta.rma(averageData, slowLength)

// Plot averages
plot(fastAverage, color=#FFCB7C, title="Fast RMA")
plot(slowAverage, color=#B19CD9, title="Slow RMA")

// Show the moving average trend with a coloured bar
barColour = if close > fastAverage and fastAverage > slowAverage
    color.lime
else if fastAverage > slowAverage
    color.green
else if close < fastAverage and fastAverage < slowAverage
    color.red
else if fastAverage < slowAverage
    color.maroon
else
    color.orange

barcolor(barColour)
```

We start with the `indicator()` function. That names the script. And `overlay` places the indicator on top of the chart's instrument.

Then we make three inputs. The first, a source input option (`input.source()`), says which data the moving averages calculate on. The other two inputs are integer inputs (`input.int()`). These set the length of the Relative Moving Averages.

Those averages we calculate next:

```
// Calculate relative moving averages
fastAverage = ta.rma(averageData, fastLength)
slowAverage = ta.rma(averageData, slowLength)
```

The first `ta.rma()` call calculates the fast average. This function processes the `averageData` input variable, which holds the instrument's close prices by default. Its length is set with `fastLength`. That input variable starts with a value of `10`.

We calculate the second Relative Moving Average in a similar way. We again call `ta.rma()` and have it process the `averageData` values. For its length we use the `slowLength` input variable, whose standard value is `40`.

After computing the averages the script continues with other tasks. First it plots both RMAs with the `plot()` function. The fast average shows in the hexadecimal colour value of `#FFCB7C` (topaz), and the slow one uses `#B19CD9` (light pastel purple). We don't set the plot type, so Pine Script makes them as regular lines by default.

Then a cascaded if statement figures out what bar colour to use. With a close above the fast average, and that fast average above the slow one, the code picks the lime (`color.lime`) colour. If only the fast average is above the slow RMA, we'll use green (`color.green`).

If the fast RMA isn't above the slow one, we check other conditions. Should the close be below the fast RMA, and that average below the slow RMA, we red (`color.red`). If only the fast average is under the slow one, we settle on maroon (`color.maroon`) instead. In case both averages have the same value we use orange (`color.orange`).

The indicator's last line of code makes the `barcolor()` function apply the chosen colour to the instrument's price bars.

On the chart, the fast and slow Relative Moving Average show as solid lines. The bar's colours tell how prices relate to those averages:

For other moving averages, see all moving averages in Pine Script.

Summary

- The `ta.rma()` function calculates the Relative Moving Average (RMA).
- That function needs two things: a series of numerical values to calculate on. And a length in number of bars.

References

1. TradingView (n.d.). *Pine Script Language Reference Manual*. Retrieved on January 3, 2022, from https://www.tradingview.com/pine-script-reference/v5/#
2. TradingView Help (n.d.). *Average True Range (ATR)*. Retrieved on January 3, 2022, from https://www.tradingview.com/support/solutions/43000501823-average-true-range-atr/#

Last updated March 19, 2022

PREVIOUS
Hull average (HMA)
NEXT
EMA versus RMA

© 2013—2024 TradingCode