Algorithmic Trading and Strategies

Sachin Napate
Dr. D. Y. Patil Vidyapeeth
3 PUBLICATIONS **0** CITATIONS
SEE PROFILE

Mukul Thakur
1 PUBLICATION **0** CITATIONS
SEE PROFILE

Algorithmic Trading and Strategies

Dr. Sachin Napate, Mukul Thakur
Dr. D.Y. Patil B-School

Abstract: - The underlying market is modeled like sinusoidal function, consistently switching between two states: the uptrend (bull market) and down trend (bear market). In this research paper we will discuss about Algorithmic Trading and trading strategies with Quantopian platform, to create intelligent trading algorithms as well as back testing them to see how they would perform on historical data. The set of trading algorithms example includes strategies and how would it be helpful for all the others and how we can utilize these strategies in real live trading to make profit and most important to understand the market data.

Keywords: SMA, EWMA/EMA, Cross-over, Time-series, Backtesting, Quantopian.

I Introduction

1. **<u>Algorithmic Trading</u>: -**
   Algorithmic trading (also called automated trading, black-box trading, or algo-trading) uses a computer program that follows a defined set of instructions (an algorithm) to place a trade. The trade, in theory, can generate profits at a speed and frequency that is impossible for a human trader.
   The defined sets of instructions are based on timing, price, quantity, or any mathematical model. Apart from profit opportunities for the trader, algo-trading renders markets more liquid and trading more systematic by ruling out the impact of human emotions on trading activities.

2. **<u>Benefits of Algorithmic Trading</u>: -**

   - Trades are executed at the best possible prices.
   - Trade order placement is instant and accurate (there is a high chance of execution at the desired levels).
   - Trades are timed correctly and instantly to avoid significant price changes.
   - Reduced transaction costs.
   - Simultaneous automated checks on multiple market conditions.
   - Reduced risk of manual errors when placing trades.
   - Algo-trading can be backtested using available historical and real-time data to see if it is a viable trading strategy.
   - Reduced the possibility of mistakes by human traders based on emotional and psychological factors.

Most algo-trading today is **High Frequency Trading (HFT),** which attempts to capitalize on placing a large number of orders at rapid speeds across multiple markets and multiple decision parameters based on preprogrammed instructions.

Algorithmic trading provides a more systematic approach to active trading than methods based on trader intuition or instinct

II Literature Review

In this literature review we will discuss an overview of some of the studies on the performance of the trading strategies. The basic principle in technical analysis involves identifying trends and using them to generate forecast signals. Trading rules based on historical data such as the moving average strategy has been a widely discussed topic in the field of finance since the introduction of the concept of efficient capital markets.
In an early widely discusses study Brock et al. (1992) presents compelling evidence of the simple moving average strategy outperforming the market by using data samples from the Dow Jones index. The study claims that stock returns are predictable and suggested two competing explanations (1) market inefficiency in which prices takes swings from their fundamental values and (2) markets are efficient and the predictable variation can be explained by time-varying equilibrium returns. The forecast abilities of simple trading rules documented in the study have later been both been supported and questioned by several studies. The study finds that the results in Brock et al appear to be robust against data-snooping, but the superior performance of the moving strategy was not significant after performing out-of-sample tests. Similar findings were found by Bauer and Dahlquist (2001) by measuring monthly, quarterly, and annual market-timing strategies for six major U.S. asset classes and LeBaron (1999) by implementing several robustness checks re-examining the data from the Dow Jones index.
Naved and Srivastava (2015) investigated the profitability of moving averages trading strategy in the Indian stock market. The study involved the testing of five versions of moving averages: simple, triangular, exponential, variable, and weighted, the performance was checked with three trading rules: Direction of the moving average, price and moving average crossover and crossover of two moving averages with different periods on stocks from the Indian S&P, CNX, Nifty 50 index. The study finds that moving average was profitable in all three trading rules, but short term look-back period generated the best result and simple moving average performed better than the other versions on moving average.
While several studies prior to Brock et al. (1992) finds that technical analysis is ineffective, in the recent decades the strategy has been increasingly popular (Park and Irwin (2007)) despite several studies presenting mixed finding son its true effectiveness and performance. In this brief overview, we find studies demonstrating strong predictive power and high profitability of simple averages trading strategy while other studies question its true performance. We find in previous studies that the performance of the moving average strategy may be dependent on several factors, such as in which market and time period it is applied in and level of transaction cost. In conclusion, the current literature is inconclusive as to the performance of simple averages as a trading strategy. Therefore, there is a need to continue with investigations in this

area in order to build upon the current knowledge and consequently determine the performance of technical analysis in stock market trading.

III Objectives

1. To discuss the SMA and EWMA/EMA cross-over trading strategies in time series data.
2. Understanding trading strategies in quantopian environment which appears to be a good compromise between smoothness of data and analyze the stock price.

IV Evaluating Trading Strategies

Trading Strategies: -

A trading strategy is the method of buying and selling in markets that is based on predefined rules used to make trading decisions.

A trading strategy includes a well-considered investing and trading plan that specifies investing objectives, risk tolerance, time horizon and tax implications. Ideas and best practices need to be researched and adopted then adhered to. Planning for trading includes developing methods that include buying or selling stocks, bonds, ETFs or other investments and may extend to more complex trades such as options or futures. Placing trades means working with a broker or broker dealer and identifying and managing trading costs including spreads, commissions and fees. Once executed, trading positions are monitored and managed, including adjusting or closing them as needed. Risk and return are measured as well as portfolio impacts of trades. The longer term tax results of trading are a major factor and may encompass capital gains or tax-loss harvesting strategies to offset gains with losses.

There are many types of trading strategies, but they are based largely on either technical or fundamentals. The common thread is that both rely on quantifiable information that can be back tested for accuracy. Any strategy for algorithmic trading requires an identified opportunity that is profitable in terms of improved earnings or cost reduction. In this paper, the study focuses on evaluating few trading strategies and trading algorithms for traders to implement it while simultaneously getting the idea to investigate the profitability of moving average trading strategies in the stock market and reinvesting debt to gain greater return on our investments. The strategies considered are **Simple Moving Average (SMA) and Exponentially Weighted Moving Average (EWMA).** For this research, we used a Quantopian platform (Algorithm IDE), which is a Python development environment designed to help for coding trading strategies using the Algorithm API.

1. Simple Moving Average (SMA): -

A simple moving average (SMA) calculates the average of a selected range of prices, usually closing prices, by the number of periods in that range. It is the most widely used of all technical indicators and it is computed by;

$$SMA = \frac{A1 + A2 + \cdots + An}{n}$$

Where '*n*' is the number of period included in the average and '*An*' is the price of an asset at period n. The SMA is calculated using historical prices, this moving average doesn't predict future market movements but rather lags the current price. The trading rules are:

- Buy when the current price is below the moving average.
- Sell when the current price is above the moving average.

Often daily financial data can be a bit noisy, what we can do is use the rolling mean method (often called simple moving average) to get more signal about the general trend of the data. In this study, the data usually provide a window of a set time period and then use that to calculate the average. The following **Fig: 1** shows the example of SMA-200 (moving average of 200 days) for AAPL open price.

```
apple['SMA-200'] = apple['Open'].rolling(window=200).mean()
```

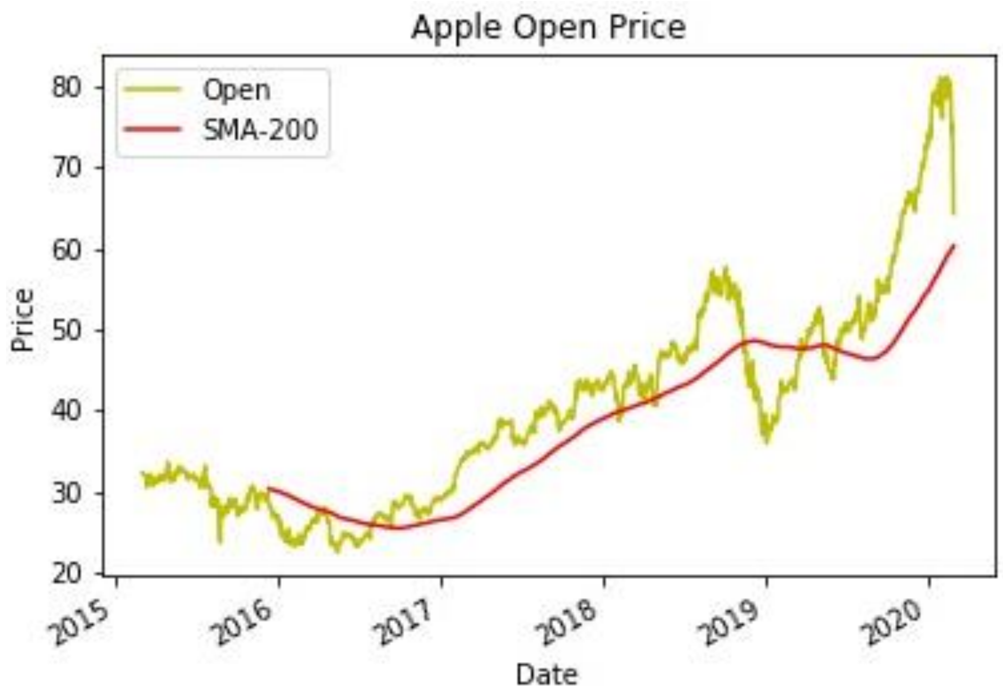

Fig: 1

2. <u>**Exponential Weighted Moving Average**</u>**: -**

Exponentially weighted moving average or Exponentially moving average will allow us to reduce the lag effect from simple moving average (SMA) and it will put more weight on values that occurred more recently (by applying more weight to the more recent values, thus the name). So as the values get closer to the present time then apply more weight to them when calculating the average. Therefore, the amount of weight applied to the most recent values will depend on the actual parameters used in the EWMA or EMA and the number of periods given in a window size. So the general formula of EMA is:

$$y_t = \frac{\sum_{i=0}^{t} w_i \, x_{t-i}}{\sum_{i=0}^{t} w_i}$$

Where x_t is the input value, w_i is the applied weight and y_t is the output. Now the question arise is how it can change from i=0 to t and how do we define the w_i

This depends on the adjust parameters provided to the ewm() method in pandas,
<u>Case 1:</u> when put adjust parameter is True which is in default then weighted average are calculated using weights as follows:

$$y_t = \frac{x_t + (1-\propto)x_{t-1} + (1-\propto)^2 x_{t-2} + \cdots + (1-\propto)^t x_0}{1 + (1-\propto) + (1-\propto)^2 + \cdots + (1-\propto)^t}$$

So we have y_t as the actual output value on the left hand side and on the right hand side (denominator) is the sum of all the weight. But in the numerator there is decrease in weight as you move further and further into the time series because we will have $(1-\propto)$ and $\propto$ needs to between 0 and 1. This means if we keep squaring the fraction to power of some other higher number which eventually gives the smaller value and we are providing smaller weights for the data as it gets older and older. Therefore, that very first data point has a smallest weight attached to it and the recent data point has the most weight attached to it, essentially 1.

<u>Case 2</u>: when put adjust parameter is False specified then moving averages are calculated as follows:

$y_0 = x_0$, means that first input value is the first output value

$y_t = (1-\propto)y_{t-1} + \propto x_t$, which is equivalent to using weights:

$$w_i = \begin{cases} \propto (1-\propto)^i & if\ i < t \\ (1-\propto)^i & if\ i = t \end{cases}$$

When adjust=False we have $y_0 = x_0$ and the last representation from above we have $y_t = (1-\propto)y_{t-1} + \propto x_t$, therefore there is an assumption that x_0 is not an ordinary value but rather an exponentially weighted moment of the infinite series up to that point.
While ∝ should be between 0 and 1 and it's impossible to pass an ∝ directly so this whole things comes down to these three conditions:

$$\propto = \begin{cases} \dfrac{2}{s+1}, & for\ span\ s \geq 1 \\ \dfrac{1}{1+c}, & for\ center\ of\ mass\ c\ \geq 0 \\ 1 - e^{\frac{log0.5}{h}}, & for\ half-life\ h > 0 \end{cases}$$

- Span corresponds to what is commonly called an “N-day EW moving average”.
- Center of mass has a more physical interpretation and can be thought of in terms of span: c=$(s-1)/2$ and inversely proportion to span.
- Half-life is the period of time for the exponential weight to reduce to one half.
- Or ∝ specifies the smoothing factor directly.

Typically when doing EMA the best way to do is with span. In **Fig: 2** shows the example of EMA-90 (moving average of 90 days) for apple open price:

```
apple['EMA-90'] = apple['Open'].ewm(span=90).mean()
```

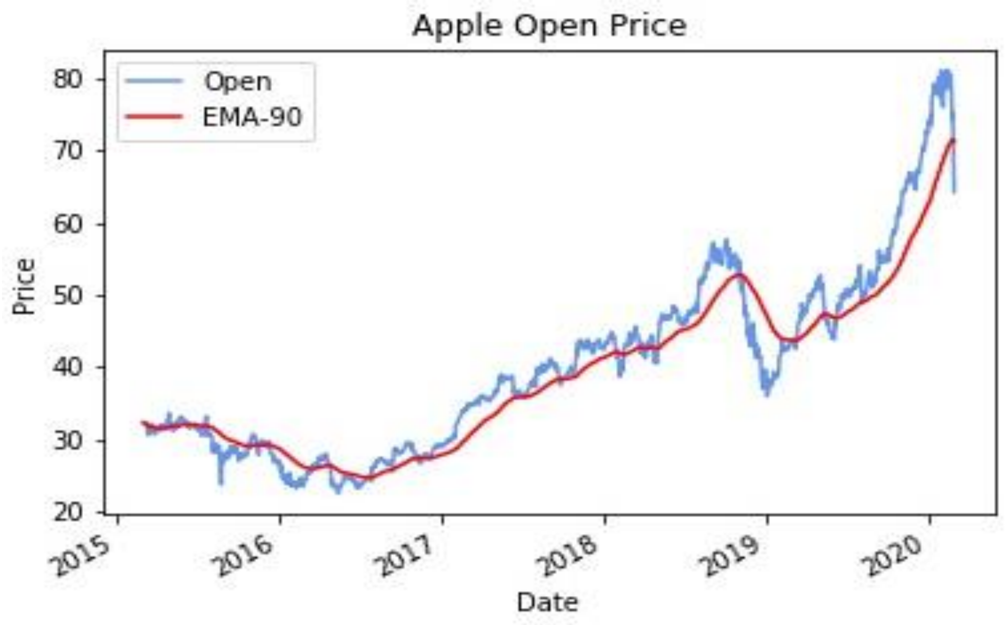

Fig 2:

V Coding for Backtesting

1. **SMA Cross Over: -** In this strategy we use Quantopian environment for coding this out and in this case the strategies are as follows:
   - Buy when 30 days MA is below the 90 days MA
   - Sell when 30 days MA is above the 90 days MA

```python
def initialize(context):
    context.apple = sid(24)

    schedule_function(check_moving_avg, date_rules.every_day(),
time_rules.market_close(minutes=30))

def check_moving_avg(context,data):
    current_price = data.current(context.apple, 'close')

    closing_price_1 = data.history(context.apple, 'close', 30, '1d')
    closing_price_2 = data.history(context.apple, 'close', 90, '1d')

    avg_1 = closing_price_1.mean()
    avg_2 = closing_price_2.mean()
    ma_1 = avg_1 < avg_2
    ma_2 = avg_1 > avg_2

    if ma_1 < ma_2:
        order_target_percent(context.apple, 1.0)
        print('Buying')

    elif ma_1 > ma_2:
        order_target_percent(context.apple, -1.0)
        print('Shorting')
    else:
        pass


    record(MA30=avg_1, MA90=avg_2, closing_price=current_price)
```

2. **Exponential Weighted Moving Average (EWMA) Or (EMA) Cross Over: -** In this we use the same cases as SMA cross over.

```python
import talib

def initialize(context):
    context.microsoft = sid(5061)
    context.invested = False

    schedule_function(handle_data, date_rules.every_day(),
time_rules.market_close(minutes=30))

def handle_data(context, data):
    price_history = data.history(context.microsoft, 'price', 90, '1d')

    ema_short = talib.EMA(price_history, timeperiod=30)
    ema_long = talib.EMA(price_history, timeperiod=90)
    ema_1 = ema_short[-1]
    ema_2 = ema_long[-1]

    if ema_2 > ema_1 and not context.invested:
        order_target_percent(context.microsoft, 1.0)
        context.invested = True

    elif ema_2 < ema_1 and context.invested and
context.portfolio.positions[context.microsoft].amount > 100:
        order_target_percent(context.microsoft, -1.0)
        context.invested = False


    record(EMA30=ema_1, EMA90=ema_2)
```

VI Recommendations

We just showed how to calculate the SMA and EMA based on some window/span. However, basic SMA has some "weaknesses".

- Smaller windows will lead to more noise, rather than signal
- It will always lag by the size of the window
- It will never reach to full peak or valley of the data due to the averaging.

- Does not really inform you about possible future behaviour, all it really does is describe trends in your data.
- Extreme historical values can skew your SMA significantly

To help fix some of these issues, we used a EWMA (Exponentially-weighted moving average) to better understand the market data and analyzed it more clearly.

VII Conclusion

The algorithmic trading is the mixture of core statistical methods and information technology. In the absence of either core statistical methods or information technology, such program of trading is not possible and cannot be executed. This paper explores two different strategies in a time series data. The SMA and EWMA cross over trading strategies are based on a company's data as an example. However, the data on this paper is provided for informational purposes only and does not constitute an offer to sell, a solicitation to buy, or a recommendation or endorsement for any security or strategy, nor does it constitute an offer to provide investment advisory services correspond to this paper. These proposed strategies could also be implemented on different price time series or improved by introducing leverage. Overall, this study explained the usefulness of relatively simple strategies in generating profits and analyzes market data for investors.

VIII Limitations

1. Limitation of data.
2. More time spend on computer screen.
3. Loss of human control.
4. Not all strategies can be automated.
5. Need to know the programming process.

IX References

[1] M. Souza1, D. Ramos, M. Pena, V. Sobreiro, and H. Kimura, "Examination of the profitability of technical analysis based on moving average strategies in brics," Financial Innovation, vol. 4:3, 2018.
[2] https://therobusttrader.com/pros-and-cons-of-algo-trading/
[3] https://www.investopedia.com/terms/t/trading-strategy.asp

[4] https://www.investopedia.com/articles/active-trading/101014/basics-algorithmic-trading-concepts-and-examples.asp
[5] Testing the Performance of Simple Moving Average with the Extension of Short Selling by Geirmund Glendrange and Sondre Tveiten