Profitable Algorithmic Trading Strategies in Mean-Reverting Markets

Vittoria Volta
Department of Computer Science, University College London (UCL)

***Abstract*— This paper is concerned with implementing and evaluating popular mean-reversion trading strategies. The underlying market is modeled like a sinusoidal function, consistently switching between two states: the uptrend (bull market) and down trend (bear market). The set of trading rules examined includes Moving Averages (Simple and Exponentially Weighted), Moving Average Cross-Over, the Auto-regressive model (AR), Moving Average Convergence/Divergence (MACD), Relative Strength Index (RSI), MA 200/MA 20. The objective is to identify three best performing trading strategies and evaluate their performance through popular performance indicators and statistical hypothesis testing.**

I. INTRODUCTION

Trading strategies can be classified into three categories: (i) Buy-and-Hold (B&H), (ii) Contra-trend, and (iii) Trend-following. The Buy-and-Hold strategy is desirable when the average stock return is higher than the risk-free interest rate [1]. This passive strategy generally yields lower transaction fees and higher long-term capital gains tax benefit compared to active short-term strategies. However, this trading rule calls for holding onto stocks regardless of price signals or news, and therefore, there is no limit to possible losses.

The contra-trend strategies focuses on taking advantages of mean reversion type of market behaviours. These strategies perform well when markets tend to swing around its average, or "mean", effectively producing outsize bullish (overbought market) and bearish (oversold market) moves that are corrected by a move in the opposite direction.

Trend-following strategies try to capture market trends. Contrary to the approach of oversold/overbought markets, trend-following strategies are done under the assumption that market trends are worth riding along, and that market strength and momentum are not a sign of pending correction as in mean reversion strategies. A trend follower purchases shares when prices advance to a certain level and closes the position at the first sign of upcoming bear market.

Any trading strategy comes with its own sets of advantages and disadvantages. This paper is concerned with mean-reversion trading strategies which tend to be very profitable in oscillating markets. The artificial price time series is generated by a sinusoidal function with Gaussian white noise and the investment horizon problem is finite.

Popular self-financing long-only trading strategies are implemented using the given scenario. The results show that the three best performing strategies are the Simple Moving Average (SMA), the Exponentially Weighted Moving Average (EWMA) and the Auto-regressive model (AR). The

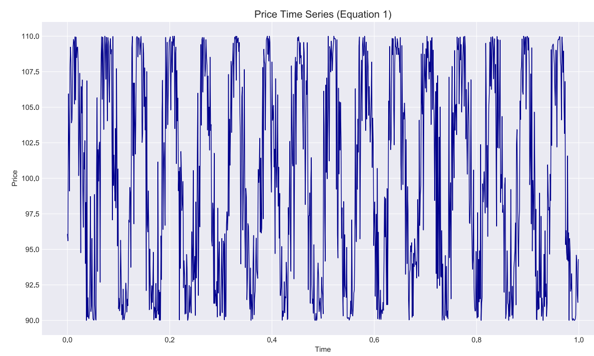

Fig. 1: Price Time Series. This figure shows the artificial price time series used in this study. It is generated through the equation described in (1).

profitability of these strategies is then discussed in terms of returns, risk/reward analysis, downside risk and multiple hypothesis testing.

The structure of this paper goes as follows. Section 2 briefly presents the problem formulation and the basic methodology regarding Trading strategies, Performance Evaluation and Multiple Testing. Section 3 shows results of the implemented methodology and comments. Section 4 and 5 conclude.

II. DATASET AND METHODOLOGY

Problem Formulation

The artificial price time series used in this study is generated by a random number generator with a seed equal to the student number 19122143. The time series satisfies the following equation:

$$p_t = p_0(1 + A * \sin(wt + 0.5\eta_t)) \tag{1}$$

where the time t ranges from from 0 to 1 in 1000 steps, the initial price $p(0) = 100$, $A = 0.1$, $w = 100$ and $\eta(t)$ is a sequence of independent and identically distributed (i.i.d.) random variables drawn from a Gaussian distribution with zero mean and unit standard deviation. A realisation of the process is shown in **Figure 1**. The sinusoidal function models a stock market with outsize bullish and bearish moves, indicating overbought and oversold levels. The

Additional Trading Strategies	
Strategy	Trading rule
SMA/EWMA Cross-Over	• Buy when the 30 days MA is below the 90 days MA • Sell when the 30 days MA is above the 90 days MA
MACD(12,26,9)	• Buy when the MACD line (difference between 12 days EWMA and 26 days EWMA) is above the signal (9 days EWMA) • Sell when the MACD line (difference between 12 days EWMA and 26 days EWMA) is below the signal (9 days EWMA)
RSI	• Buy when the RSI (ratio of up-moves to down-moves over 14 days) is below the lower threshold (30) • Sell when the RSI (ratio of up-moves to down-moves over 14 days) is above the upper threshold (70)
MA200/MA20	• Buy when the price is below MA20 and above MA200 • Sell when the price is above MA20 and above MA 200

Fig. 2: Additional Trading Strategies. This table briefly summarise the additional trading rule implemented in this study.

oscillatory behavior around an average, or ”mean”, suggests that mean-reversion strategies might be very profitable.

To test and evaluate the trading strategies, the following problem setting is used:

- The investor has an initial capital of 1000£.
- The investment horizon covers 1000-time steps.
- The investor can only invest in the price time series defined in Equation 1.
- No transaction costs.
- No short selling (the investor can only sell what he/she currently owns). In other words, the volume V is non-negative for all time steps.
- No borrowing is also considered, the cash C is always equal or greater than zero.
- When the investor opens a position, he/she goes ”all in” - allocates all the resources for the purchase
- Strategies are self-financing, meaning that the total value of the strategies at each time step is:

$$TV_t = C_t + p_t V_t = C_{t+1} + p_t V_{t+1} \tag{2}$$

Trading Strategies

This study focuses on implementing different trading strategies that potentially fall into the category of mean-reversion trading rules. The strategies considered are the Simple (SMA) and Exponentially Weighted (EWMA) Moving Average, Auto-regressive Model (AR), Simple and Exponential Moving Average Cross-Over, Moving Average Convergence/Divergence (MACD), Relative Strength Index (RSI) and MA200/MA20.

Interestingly, the results show that relatively simple and easy-to-implement strategies, such as the Simple and Exponential Moving Averages and the AR model, are the best performing strategies for the given type of market behaviour.

These three trading strategies are described in the following paragraphs. Details on the other trading strategies are also briefly summarised in **Figure 2**.

The *Simple Moving Average* (SMA) is the most widely used of all technical indicators and it is computed by:

$$SMA_n = \frac{1}{n} \sum_{t=k-n+1}^{k} P_t \tag{3}$$

where n is the number of periods included in the average and P_t is the asset price at time t. Unlike the Exponentially Weighted Moving Average (EWMA) which weight observations in each period based on their relative position in the average, each price in the SMA is equally weighted. This can make the SMA erratic as oldest prices are discarded and most recent prices are added each period [2]. However, the impact of this effect can be controlled by varying the length of the time window.

This study uses a window size of 30 which appears to be a good compromise between smoothness and sensitivity to short-term price changes [3].

Given the way the SMA is calculated using historical prices, this moving average does not predict future market movements but rather lags the current price. In a bull market, the SMA will be below the rising price line, whereas in a bear market, the SMA will be above the falling price line. When the current price changes direction, the direction of the crossing determine the trading rule:

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

The *Exponentially Weighted Moving Average* (EWMA) has the form:

$$EMA_t = EMA_{t-1} + \lambda(P_t - EMA_{t-1}) \tag{4}$$

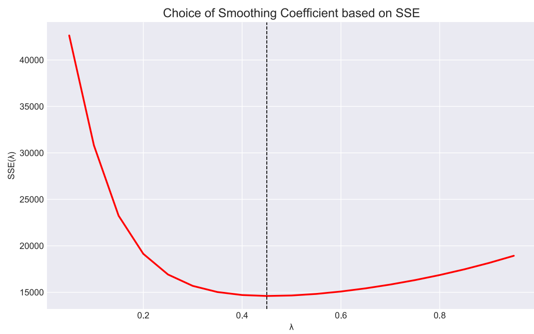

Fig. 3: Smoothing Coefficient. Plot of λ versus SSE(λ) for the choice of the optimal smoothed coefficient.

where EMA_{t-1} is the value of the Exponential Moving Average in the previous period and λ is the scaled smoothing coefficient at time t. A simple moving average is used as the previous period's EMA in the first calculation.

The parameter λ ($0 \leq \lambda \leq 1$) determines how much the process mean changes. A small λ assigns more weight to the previous observations and less weight to the current observation (and vice versa for large λ) [4]. The choice of optimal smoothing coefficient is usually done via simulation.

In this paper, the coefficient λ is chosen to minimise the sum of squared one-step-ahead forecast errors:

$$SSE(\lambda) = \sum_{t=1}^{n} e_{t-1}^2 \tag{5}$$

where $t = 1, 2, ...n$ denotes each time step and the one-step-ahead forecast errors are computed by:

$$e_{t-1} = P_t - EMAt - 1 \tag{6}$$

It is important to note that, although the EMA is the preferred average among traders, it does not necessarily mean that it is always more profitable than the SMA. Sometimes the EMA will react quickly, causing a trader to get out of a trade on a market hiccup, while the slower-moving SMA keeps the person in the trade, offering a bigger profit after the hiccup is finished.

In line with the SMA, the trading rule is:

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

An *Auto-regressive* process of order p, denoted as $AR(p)$, is a random walk of the form:

$$X_t = \phi_0 + \sum_{i=1}^{p} \phi_i X_{t-1} + \epsilon_t \tag{7}$$

where ϕ are parameters and ϵ_{t-i} is independent white noise, distributed as $\mathcal{N}(0, \sigma^2)$ random variable. In other words, this model assumes that X_t depends linearly on the p previous values. In the $p = 1$ case, the process can be rewritten as:

$$X_t = X_{t-1} + \rho(\mu - X_{t-1}) + \epsilon_t \tag{8}$$

where $\mu = \frac{\phi_0}{1-\phi_1}$ and $\rho = (1 - \phi_1)$. This highlights the mean-reverting nature of the process, as it recalls the discrete time analogue of the well-known Ornstein-Uhlenbeck process [5].

The trading rule is:

- Buy when the expected price at time t+1 is larger than the present price, $\hat{p}_{t+1} > p_t$
- Sell when the expected price at time t+1 is smaller than the present price, $\hat{p}_{t+1} < p_t$

Evaluating Trading Strategies

One of the most widely quoted figures when discussing strategy performance is the *annual return* of a strategy, an average of daily *strategy returns* that are computed as follows:

$$r_a(t) = log(\frac{TV_a(t)}{TV_a(t-1)}) \tag{9}$$

The daily log returns are now used to calculate three representative performance indicators to evaluate the proposed trading strategies.

The first comparative measure computed is the *Sharpe Ratio* which quantifies how much return can be achieved for the level of volatility endured. Mathematically:

$$SR_a = \frac{\mu_a - r_f}{\sigma_a} \tag{10}$$

where μ_a and σ_a are the expected return and risk of strategy a and r_f is the risk-free rate which is here assumed to be equal to 0.

Despite being ubiquitous as a comparative measure in quantitative finance, the Sharpe Ratio suffers from some limitations. In particular, it is poor at characterising *tail risk* [6]. It assumes that returns are normally distributed, while in practice they exhibit fat tails, indicating that extreme events are more likely to occur than a Gaussian distribution.

Facing investment choices, investors may care more about potentially excess losses in a downtrend market than excess gains in an upside market. A solution to this is the *Conditional Sharpe ratio*, which takes a more conservative view than the traditional Sharpe ratio by including the effects of higher order moments of the return distribution. The Conditional SR replaces the standard deviation in the denominator with the *Expected Shortfall* (ES), also known as Conditional Value-at-Risk, which is a popular measure of tail risk.

The Conditional Sharpe Ratio is computed by:

$$CSR_a = \frac{\mu_a}{ES_a} \tag{11}$$

where ES_a is the expected shortfall, i.e. the average of the strategy returns that are lower than the $VaR(a)$.

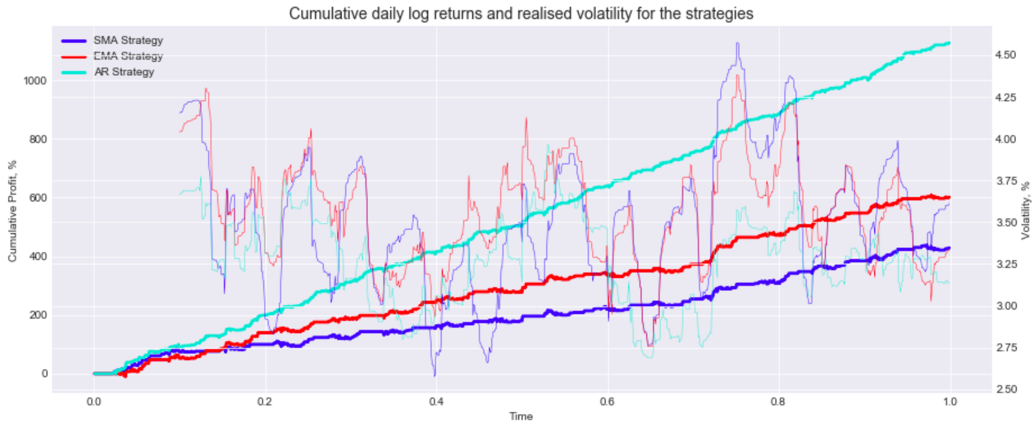

Fig. 4: Cumulative daily log returns and realised volatility. This figure shows the cumulative log returns of the three strategies together with the 100-days rolling volatility.

Another indicator of downside risk over a specified time period is the *Maximum Drawdown* (MDD), which is the largest overall peak-to-trough percentage drop in the value of a strategy.

$$MDD = \frac{MinTroughValue - MaxPeakValue}{MaxPeakValue} \quad (12)$$

where the minimum trough value of the price time series has to follow in time the maximum peak value.

Multiple testing

A good practice in evaluating the profitability of trading strategies is to perform a statistical test to see if the expected excess returns to the standard deviations are different from zero. When testing multiple strategies, false rejections of the null hypotheses are more likely to occur, i.e. a trading strategy is incorrectly "discovered" to be profitable [7].

To limit such occurrences, multiple testing methods such as *Bonferroni* and *Holm* are used to control the *family-wise error rate* (FWER) (or Type I error), i.e. the probability of making at least one false discovery, and to adjust the p-values of the tests accordingly.

The uncorrected p-values are first obtained by transforming the Sharpe ratios of $M = 3$ strategies into t-ratios and performing M t-tests. The t-ratio is defined as follows:

$$t - ratio = SR * \sqrt{T} \quad (13)$$

where SR is the Sharpe ratio and T is the sample size. The p-values are then ordered in ascending order, $p(1) \leq p(2) \leq ... \leq p(M)$ and adjusted through both Bonferroni's and Holm's method.

The Bonferroni's method inflates the original p-values by the number of tests M:

$$p_{(i)}^{Bonferroni} = min[Mp_{(i)}, 1], i = 1, ..., M \quad (14)$$

The Holm's method adjusts each p-value by:

$$p_{(i)}^{Holm} = min[\max_{j \leq i}(M - j + 1)p_{(i)}, 1], i = 1, ..., M \quad (15)$$

To evaluate whether the Sharpe ratios of the strategies are significantly different from zero, the adjusted p-values are compared with a confidence level of 5%. A statistically significant result is given by p-values lower than this threshold.

III. RESULTS AND DISCUSSION

Figure 4 shows the cumulative daily log returns of the strategies together with the realised volatility of the three strategies, SMA, EMA and AR model. All three strategies have performed extremely well, with an average annual return of 1.52 (SMA), 2.89 (EMA) and 13.88 (AR) expressed in growth rates. These high total returns suggest that the price time series generated using the equation (1) is relatively simple to predict, especially for an auto-regressive model. The 100-days rolling volatility fluctuates within the range 2.75% to 4.5%, however, the AR strategy appears to be slightly more stable than the moving averages. The annual volatility is equal to 0.51 for the AR strategy and equal to 0.56 for both SMA and EMA strategy.

It is straightforward to observe that both the SMA and EMA timeseries are much less noisy than the original price time series (**Figure 5**). They help visualise the trend by filtering out the "noise" from the random price fluctuations. However, this comes at a cost: the SMA, in particular, lags the original price time series, which means that changes in the trend are only seen with a delay (lag) of L days. The EMA partially reduces the lag by assigning more weights on recent observations. According to **Figure 3**, the optimal smoothing coefficient is $\lambda = 0.4$, which yields the minimum sum of squared (one- step-ahead) prediction errors.

Conversely, the AR predictions appear to accurately fit the original price time series, including the irrelevant information or randomness of the time series. This model

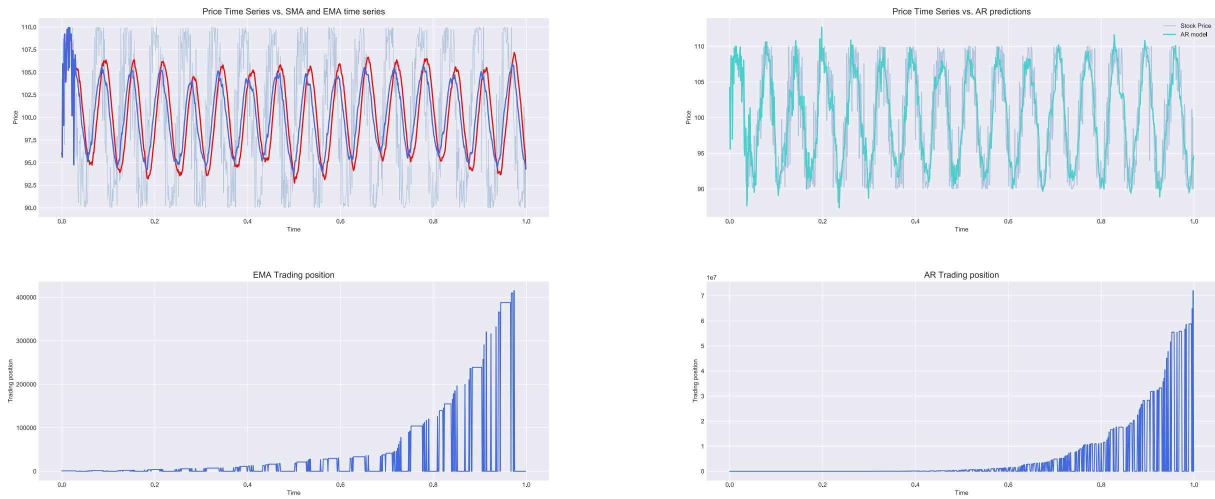

Fig. 5: Price time series vs SMA, EWMA and AR predictions (above), trading positions (below). The above plots show the original price time series and the SMA, EWMA (top left) and AR (top right) predictions. Below, the trading positions for EWMA (bottom left) and AR (bottom right) strategies.

Performance Indicators							
Strategy	Ann. Return	Ann. Volatility	Ann. SR	Ann. CSR	Maximum Drawdown	Skewness	Ex. Kurtosis
SMA(30)	1.52	0.56	1.92	1.79	-3.08	0.53	1.17
EMA(30)	2.89	0.56	2.68	3.58	-2.84	0.90	0.81
AR	13.88	0.51	5.47	4.70	-4.21	1.43	1.21

Fig. 6: Performance metrics. This table reports the metrics used to evaluate the proposed trading strategies. Average returns, volatility, Sharpe ratio and Conditional Sharpe ratio are annualised.

Multiple Testing								
	In-sample				*Out-of-sample*			
Strategy	t-statistic	Uncorrected p-vals	Bonferroni	Holm	t-statistic	Uncorrected p-vals	Bonferroni	Holm
SMA(30)	3.80	1.49e-04	4.47e-04	1.49e-04	2.96	3.11e-03	9.33e-03	3.11e-03
EMA(30)	5.35	1.06e-07	3.20e-07	2.13e-07	5.58	2.94e-08	8.84e-08	5.89e-08
AR	10.88	3.70e-26	1.11e-25	1.11e-25	11.32	4.55e-28	1.36e-27	1.36e-27

Fig. 7: Multiple testing. This table reports the results from the multiple hypothesis testing and the Bonferroni's and Holm's correction methods.

is generally prone to over-fitting when many lags are added as model parameters. In this study, the initial number of lags is approximately 9 and it reaches 21 as the number of observations increases. Clearly, the model learns the detail and noise in the training data, potentially leading to inaccurate out-of-sample forecasts. However, in the present case the predictions are only one-step ahead, and therefore, the AR strategy is able to accurately predict the short-term price fluctuations using the previous time series observations.

It is important to note that, although the AR strategy gives the highest profit compared to the SMA and EMA, it does not necessarily mean that it is the one preferred by a trader. By looking at the dynamics of cash in **Figure 5**, it is evident that AR strategy is more dynamic and requires to open and close positions more frequently than the SMA and EMA strategy. Therefore, this strategy may not be suitable for a trend trader or long-term trader, who is less concerned with short-term price fluctuations.

The performance of the strategies are summarised in **Figure 6**. As suggested by the high annual returns and low annual volatility, the AR strategy exhibits an excellent annual Sharpe ratio of 5.47. The SMA and EMA strategies have lower annual return and thus smaller yet good Sharpe ratios.

The positive skewness and positive excess kurtosis suggest that the distribution of the returns of the strategies is not normal but rather asymmetric and fat tailed. Therefore, Conditional Sharpe Ratio and Maximum Drawdown which account for the downside risk of a strategy are likely to be preferred as performance measures by risk averse investors.

The Conditional Sharpe ratio is here obtained as the excess returns over the 95% ES. Interestingly, the AR and SMA strategies have the highest expected shortfall (0.60),

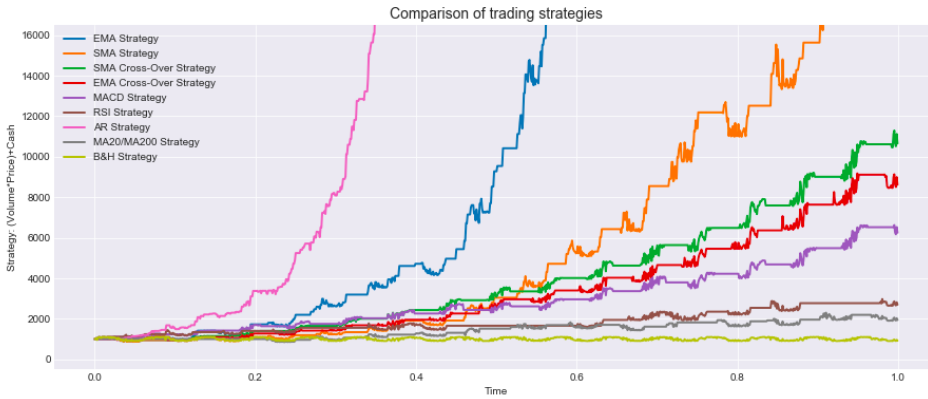

Fig. 8: Comparison of trading strategies. This figure shows the total value of the strategies implemented in this study. The B&H strategy is used as a benchmark. Clearly, the three most profitable strategies are the SMA, EWMA and AR, which notably outperform the B&H strategy.

suggesting that on bad days these strategies, on average, will perform worse than the EMA (ES = 0.42). However, in the case of the AR strategy, the relatively high expected shortfall is compensated by large annual returns, and thus, the Conditional Sharpe Ratio is still notably high.

With respect to the maximum drawdown - here calculated over time periods of 30 days - the AR strategy has the highest maximum potential loss over the specified time period, followed by SMA and EMA. This measure of downside risk further suggests that down movements could be volatile for an AR strategy. However, it is to be noted that the asymmetric distribution of the returns indicates that very large drawdowns occur less frequently than smaller ones.

To evaluate whether these strategies can generate and maintain "true" profits, a statistical test is performed on the Sharpe ratios. The results of the t-tests for the three strategies are shown in **Figure 7**.

From an in-sample perspective, the t-statistics are 3.80 (SMA), 5.35 (EMA) and 10.88 (AR). This means that the observed profitability is about 4, 5 and 11 standard deviations away from the null hypothesis of zero profitability.

However, the p-values of single tests may greatly overstate the true statistical performance. To avoid Type I errors and control for false discoveries, the p-values are then adjusted using Bonferroni's and Holm's method.

As shown in **Figure 7**, both of these procedures are designed to reduce FWER so that it is below the original confidence level. However, comparing the two adjustments, $p_{(i)}^{Holm} \leq p_{(i)}^{Bonferroni}$ for any i [8]. Indeed, the Bonferroni's method inflates the original p-values more than Holm's method, resulting in smaller adjusted p-values.

Overall, the adjusted p-values are lower than the 5% threshold and thus, there is enough statistical evidence to reject the null hypothesis and conclude that the Sharpe ratios of the proposed strategies are significantly different from zero.

This study has shown that relatively simple and easy-to-implement strategies such as SMA, EMA and AR model can be highly profitable. The same holds true when implementing more sophisticated strategies on the same time series, exploiting both short-term and long-term price fluctuations. **Figure 8** shows the total value of 9 strategies plotted against time. The Buy-and-Hold strategy is also reported, serving as a benchmark. All the implemented strategies appear to outperform the passive buy-and-hold strategy. However, only few of them show a profit that is comparable with the three best performing strategies.

One inherent disadvantage of SMA Cross-Over, EWMA Cross-Over and MA 200/MA 20 is that they are lagging indicators, in that they measure the change in stock prices over a period of time in the past and they tend to be late at giving signals. Often, when a fast moving average crosses over a slower one, the market will already turn back the other way and some of the gains will have been missed.

Similarly, the MACD indicator, that is calculated based on two short-term Exponentially Weighted Moving Averages, it relies heavily on cross-overs to give its signal.

With the RSI indicator, here calculated on a 14-days timeframe, the market did not reach the oversold (30) and overbought (70) levels before a shifting direction occurred. A reason might be that the given price time series shows very short-term (3 or 4 days) price oscillations, and therefore, the resulting ratio of average gains/average losses calculated over 14-days is not sufficiently unbalanced (large or small) to cross the given thresholds. A shorter period RSI could perhaps be more reactive to short-term price changes and it could succeed in identifying zones of oversold or overbought market.

IV. LIMITATIONS

- This trading exercise is based on an unrealistic stock price behaviour, with constant upwards and downwards trends. The given price time series also fails to meet the volatility clustering requirement.

- The given problem setting is also unrealistic. Investors can use short-selling to profit from stocks when they go down in value.
- Limitations in the performance measures. No single measure appears to give a reasonable view of what we can intuitively recognise as a good strategy. For instance, the max drawdown might not be enough to measure risk. A very important metric is the *drawdown duration*, which gives an idea of the time it takes to recover from drawdowns.

V. CONCLUSIONS

This paper explores different trading strategies in a mean-reverting price time series. The SMA, EWMA and AR strategies appear to outperform more complex trading rules. However, more research is needed on how these strategies could be improved to model the underlying market behaviour. For instance, it would be interesting to see whether other RSI models, such as the 2-period RSI or the Connors RSI, are able to offer higher profits than the RSI model used in this study and outperform the other strategies.

These proposed strategies could also be implemented on different price time series or improved by introducing leverage. Their performance is expected to be different from the results shown in this study.

Overall, this study illuminates on the usefulness of relatively simple strategies in generating profits and saving time, i.e. money, for investors.

REFERENCES

[1] M. Dai, Z. Yang, Q. Zhang, and Q. J. Zhu, "Optimal trend following trading rules," *Institute for Operations Research and the Management Sciences (INFORMS)*, 2020.

[2] C. L. Ellis and S. Parbery, "Is smarter better? a comparison of adaptive, and simple moving average trading strategies," *Research in International Business and Finance*, vol. 19.

[3] 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.

[4] M. B. Perry, "The exponentially weighted moving average," *Wiley Encyclopedia of Operations Research and Management Science*, 2010.

[5] B. Uyumazturk and V. Portilheiro, "Rise and fall: An autoregressive approach to pairs trading," https://pdfs.semanticscholar.org/1c62/376b50304ebfdd83773f27e334cf27ef0873.pdf, 2017.

[6] L. M. Halls-Moore, "Successful algorithmic trading," 2015.

[7] C. R. Harvey and Y. Liu, "Backtesting," *SSRN*, 2013.

[8] C. R. Harvey and Y. Liu, "Evaluating trading strategies," *SSRN*, 2019.

PRICE TIME SERIES

```
np.random.seed (19122143)
```

```
T = 1
t_steps = 1000
dt = T/t_steps
time = np.arange(0,T,dt)

p0 = 100
A = 0.1
w = 100

P = pd.DataFrame(columns = ["Price"])

for t in time:
    price = p0*(1+A*np.sin(w*t + 0.5*np.random.randn()))
    P.loc[len(P)] = price

S = P["Price"].values
```

STRATEGIES

1) Simple Moving Average (SMA)

```
time_window = 30
```

```
cumsum = [0]

sma = np.zeros(np.shape(S))
sma_w = np.zeros(np.shape(S))
sma_cash = np.zeros(np.shape(S))

sma_cash[0] = 1000

for i, x in enumerate(S[:-1], 0):
    cumsum.append(cumsum[i] + x)
    sma[i] = x
    if i >= time_window:
        moving_ave = (cumsum[i] - cumsum[i-time_window])/(time_window)
        sma[i] = moving_ave

    if sma[i] == x:
        sma_w[i+1] = sma_w[i]
        sma_cash[i+1] = sma_cash[i]

    if sma[i] > x:
        sma_w[i+1] = sma_cash[i]/x  + sma_w[i]
        sma_cash[i+1] = 0

    if sma[i] < x:
        sma_cash[i+1] = sma_w[i]*x + sma_cash[i]
        sma_w[i+1] = 0

sma[i+1] = S[len(S)-1]
sma_strategy = [a*b for a,b in zip(sma_w,S)] + sma_cash
```

```
totret_sma = (sma_strategy[-1] - sma_strategy[0])/sma_strategy[0]
```

```
# log returns, cumulative and relative
ret_sma_strategy = np.log(sma_strategy[1:]/sma_strategy[:-1])
cum_ret_sma_strategy = ret_sma_strategy.cumsum()
cum_ret_sma_strategy_relative = np.exp(cum_ret_sma_strategy) - 1
```

2) Exponentially Weighted Moving Average (EWMA)

```
time_window = 30
```

```
cumsum = [0]

ema = np.zeros(np.shape(S))
ema_w = np.zeros(np.shape(S))
ema_cash = np.zeros(np.shape(S))

ema_cash[0] = 1000

for i, x in enumerate(S[:-1], 0):
    cumsum.append(cumsum[i] + x)
    ema[i] = x
    if i == time_window:
        moving_ave = (cumsum[i] - cumsum[i-time_window])/(time_window)
        ema[i] = moving_ave
    if i > time_window:
        multiplier = 2.0/(1+time_window)
        ex_moving_ave = (x - ema[i-1])*multiplier + ema[i-1]
        ema[i] = ex_moving_ave

    if ema[i] == x:
        ema_w[i+1] = ema_w[i]
        ema_cash[i+1] = ema_cash[i]

    if ema[i] > x:
        ema_w[i+1] = ema_cash[i]/x  + ema_w[i]
        ema_cash[i+1] = 0

    if ema[i] < x:
        ema_cash[i+1] = ema_w[i]*x + ema_cash[i]
        ema_w[i+1] = 0

ema[i+1] = S[len(S)-1]
ema_strategy = [a*b for a,b in zip(ema_w,S)]+ ema_cash
```

```
totret_ema = (ema_strategy[-1] - ema_strategy[0])/ema_strategy[0]
```

```
# log returns, cumulative and relative
ret_ema_strategy = np.log(ema_strategy[1:]/ema_strategy[:-1])
cum_ret_ema_strategy = ret_ema_strategy.cumsum()
cum_ret_ema_strategy_relative = np.exp(cum_ret_ema_strategy) - 1
```

3) AR model

```
time_window = 30
```

```
cumsum = [0]

ar_prediction = np.zeros(np.shape(S))

ar_w = np.zeros(np.shape(S))
ar_cash = np.zeros(np.shape(S))

ar_cash[0] = 1000

for i, x in enumerate(S[:-1], 0):
    cumsum.append(cumsum[i] + x)
    ar_prediction[i] = x
    if i >= time_window:
        X = S[0:i]
        train = X
        # train autoregression
        model = AR(train)
        model_fit = model.fit()
        predictions = model_fit.predict(start=len(train), end=len(train), dynamic=False)
        ar_prediction[i] = predictions[0]

    if ar_prediction[i] == x: # we compare the price today with the our predictions coming from arima model
        ar_w[i+1] = ar_w[i]
        ar_cash[i+1] = ar_cash[i]

    if ar_prediction[i] > x:
        ar_w[i+1] = ar_cash[i]/x  + ar_w[i]
        ar_cash[i+1] = 0

    if ar_prediction[i] < x:
        ar_cash[i+1] = ar_w[i]*x + ar_cash[i]
        ar_w[i+1] = 0

ar_prediction[i+1] = S[len(S)-1]

ar_strategy = [a*b for a,b in zip(ar_w,S)]+ ar_cash
```

```
totret_ar = (ar_strategy[-1] - ar_strategy[0])/ar_strategy[0]
```

```
# log returns, cumulative and relative
ret_ar_strategy = np.log(ar_strategy[1:]/ar_strategy[:-1])
cum_ret_ar_strategy = ret_ar_strategy.cumsum()
cum_ret_ar_strategy_relative = np.exp(cum_ret_ar_strategy) - 1
```

EVALUATION OF STRATEGIES

```
sma_sr = np.sqrt(252)*np.mean(ret_sma_strategy)/np.std(ret_sma_strategy)
ema_sr = np.sqrt(252)*np.mean(ret_ema_strategy)/np.std(ret_ema_strategy)
ar_sr = np.sqrt(252)*np.mean(ret_ar_strategy)/np.std(ret_ar_strategy)
```

```
windows_maxdrawdown = np.arange(0,1000,30)
```

```
def drawdown(prices):
    prevmaxi = 0
    prevmini = 0
    maxi = 0

    for i in range(len(prices))[1:]:
        if prices[i] >= prices[maxi]:
            maxi = i
        else:
      # You can only determine the largest drawdown on a downward price!
            if (prices[maxi] - prices[i]) > (prices[prevmaxi] - prices[prevmini]):
                prevmaxi = maxi
                prevmini = i
    return (prices[prevmaxi], prices[prevmini])
```

```
drdowns = []
for w in range(1, len(windows_maxdrawdown)):
    md = drawdown(ret_sma_strategy[round(windows_maxdrawdown[w-1]):round(windows_maxdrawdown[w])])
    if md[0] != md[1]:
        drdowns.append((md[1] - md[0])/md[0])
print(min(drdowns))
```

```
drdowns = []
for w in range(1, len(windows_maxdrawdown)):
    md = drawdown(ret_ema_strategy[round(windows_maxdrawdown[w-1]):round(windows_maxdrawdown[w])])
    if md[0] != md[1]:
        drdowns.append((md[1] - md[0])/md[0])
print(min(drdowns))
```

```
drdowns = []
for w in range(1, len(windows_maxdrawdown)):
    md = drawdown(ret_ar_strategy[round(windows_maxdrawdown[w-1]):round(windows_maxdrawdown[w])])
    if md[0] != md[1]:
        drdowns.append((md[1] - md[0])/md[0])
print(min(drdowns))
```

```
alpha = 0.95
sma_var = abs(np.quantile(ret_sma_strategy,1-alpha))
sma_es = abs(np.mean(ret_sma_strategy[ret_sma_strategy < sma_var]))
sma_cvar = np.mean(ret_sma_strategy)/sma_es
```

```
alpha = 0.95
ema_var = abs(np.quantile(ret_ema_strategy,1-alpha))
ema_es = abs(np.mean(ret_ema_strategy[ret_ema_strategy < ema_var]))
ema_cvar = np.mean(ret_ema_strategy)/ema_es
```

```
alpha = 0.95
ar_var = abs(np.quantile(ret_ar_strategy,1-alpha))
ar_es = abs(np.mean(ret_ar_strategy[ret_ar_strategy < ar_var]))
ar_cvar = np.mean(ret_ar_strategy)/ema_es
```

MULTIPLE TESTING

```
import statsmodels
sma_sr = sma_sr/np.sqrt(252)
ema_sr = ema_sr/np.sqrt(252)
ar_sr = ar_sr/np.sqrt(252)
```

```
muH = 0
T = len(ret_sma_strategy)
t_ratioSMA = (sma_sr)*np.sqrt(T)
pvalSMA = stats.t.sf(np.abs(t_ratioSMA), T-1)*2

T = len(ret_ema_strategy)
t_ratioEMA = (ema_sr)*np.sqrt(T)
pvalEMA = stats.t.sf(np.abs(t_ratioEMA), T-1)*2

T = len(ret_ar_strategy)
t_ratioAR = (ar_sr)*np.sqrt(T)
pvalAR = stats.t.sf(np.abs(t_ratioAR), T-1)*2
```

```
pvals = np.array([pvalSMA,pvalEMA, pvalAR])
```

```
multipleTestBonf = statsmodels.stats.multitest.multipletests(pvals, alpha=0.05, method='bonferroni', is_sorted=Fals
multipleTestBonf = statsmodels.stats.multitest.multipletests(pvals, alpha=0.05, method='holm', is_sorted=False, ret
```