# 4. Research Methodology

The research involves modelling how an average individual’s trading strategy would perform in the stock markets and how its return compares to the passive investing alternative. The passive strategy will be used as a benchmark and will be the equivalent of buying and holding the respective stock.

To achieve this, a Python script, that aims to ‘backtest’ relatively simple trading strategies, assumed to be representative of average investors, is developed. ‘Backtesting’ is a method by which it can be determined how well a strategy would have performed if it had been deployed during the timeframe taken into consideration. It also aims to infer the strategy’s future performance following the logic that if it has performed well in the past, it is likely that it will also perform well in the future, and inversely for poor performance. The logic of the ‘backtest’ is to deploy an artificial trading agent at a time point on the historic data and record its performance over the desired period. The agent is instructed with a specific trading strategy, has access to all information that has become available prior to his point in time and no knowledge of events that will happen after his point in time. The aim of this method is to simulate actual trading and record its performance. In practice, this involves iterating trough historic price data sequentially and tracking how this supposed artificial agent would perform.

Overall, the script has 3 tasks to accomplish: collect historic price data for a set of company stocks, run the trading strategies on the data, and compile the results in an interpretable manner.

## 4.1. Collecting data

Since the quantitative analysis is to be ran on a big sample of data, meaning a period spanning multiple years and involving multiple stock prices, data collection is also to be automated. Yahoo Finance has a free-to-use API that allows automated requests of the financial data they provide. Moreover, ‘yahoo\_fin’ is an open-source Python library developed by Andrew Treadway that facilitates accessing and working with Yahoo Finance’s API. The ‘stock\_info’ module is used, with its handle renamed to ‘si’ to download the historic price data:

import yahoo\_fin.stock\_info as si

#...

df = si.get\_data(ticker, start\_date=beginning) # download data into Pandas dataframe  
self.data = df['close'] # trim data to only use close price, indexed by date  
self.name = df['ticker'].iloc[0] # set name attribute from the data ticker

A daily period is used, and only close prices are considered. This is done under the assumption that the average trader will most likely not take multiple relevant decisions during a single day. What this means for the algorithm is that the strategy has only one decision point per day.

|  |  |
| --- | --- |
| Index (Date) | Close |
| **2013-01-02 00:00:00** | 19.60821 |
| **2013-01-03 00:00:00** | 19.36071 |
| **2013-01-04 00:00:00** | 18.82143 |
| **2013-01-07 00:00:00** | 18.71071 |
| **2013-01-08 00:00:00** | 18.76107 |
| **2013-01-09 00:00:00** | 18.46786 |
| **2013-01-10 00:00:00** | 18.69679 |

Table 1- Example of price data for AAPL (Source: Yahoo Finance)

Above is an example of the form of the data that results after download and processing. The prices are for Apple’s stock in January 2013.

## 4.2. Testing trading strategies on the data

The main idea behind the strategies which are implemented is using moving averages to gauge the trend of the price and open positions when the trend reverses. The formula for the moving average, referred to in financial applications as SMA, is the following:

Where is the number of periods (data-points) of the moving average and is the period for which we are calculating the SMA. The strategy that uses this indicator is commonly known as a SMA crossover strategy. Its name stems from the fact that the execution of the strategy is relatively intuitive when looking at a price chart. It uses two SMA indicators, one of a larger period and one of a lesser period, of which I will refer to from now on as the ‘fast SMA’ and the ‘slow SMA’. The idea behind this is that the slow SMA tracks the larger trend, while the fast SMA tracks the trend for a narrower timeframe. Trend reversals are interpreted as a crossover of the two SMAs, which, graphically, is exactly the intersection of the two indicators. The strategy opens and closes trades when trend reversals happen. For example, if at the start the stock price is in a downtrend, when a trend reversal happens, i.e., the downtrend turns into an uptrend, a long (buy) position is opened. When the uptrend ends and a new reversal is identified, the long position is closed, and a short (sell) position is opened.

The conversion of the strategy into an algorithm is relatively simple, as it uses only simple conditions for the execution of a trade. The algorithm first checks for the trend in the previous period relative to its point in iteration. The only two possible states are a downtrend, defined by the fast SMA being of lower value than the slow SMA, and an uptrend, with the fast SMA being of higher value than the slow SMA. In the case where the two SMAs are equal a tertiary condition is activated, and the algorithm doesn’t emit a trade signal. After the trend of the previous period is identified, it checks whether the current period presents a trade reversal, that is, whether the two SMA indicators have crossed over since the last period.

if market.sma(period - 1, fastSMA) < market.sma(period - 1, slowSMA):

# if downtrend  
 if market.sma(period, fastSMA) > market.sma(period, slowSMA):

# if downtrend reversal  
 return "closelong"  
 else:  
 return 'no posittion'

elif market.sma(period - 1, fastSMA) > market.sma(period - 1, slowSMA):

# if uptrend  
 if market.sma(period, fastSMA) < market.sma(period, slowSMA):

# if uptrend reversal  
 return "closeshort"  
 else:  
 return 'no posittion'  
else: # condition for catching exceptions (e.g. fastSMA=slowSMA)  
 return 'no posittion'

For every strategy execution iteration (one strategy tested on one stock’s historic price data), the algorithm keeps track of every trade executed and its performance (return). Additionally, it tracks if there is currently an active trade existing at its point in iteration, for the purpose of correctly calculating the returns of the trades it executes. The ‘trade signals’ it emits are a sequence of two actions. The first one will always be ‘close’, which signals that the previously opened trade needs to be closed. This is due to the fact that the SMA crossover strategy, and trading strategies in general, won’t signal having both a short and a long position concurrently. The second action will be either ‘long’, for initiating a long position, or ‘short’, for initiating a short position.

The strategy implemented has some particularities:

1. The strategy ‘activates’ after it has enough trading days prior to its point in iteration to calculate the slow SMA. For example, if we set the slow SMA to a SMA10 (simple moving average of 10 periods), the algorithm will begin looking for trades after period 10 (after 10 trading days’ worth of data)
2. Only one existing position is allowed by the algorithm. The strategy allows only one long or short position at a time, as multiple opposite positions would be incoherent with the logic of the strategy: as the focus is on trend reversals, a trade signal for a short position will close the existing long position and open the short one and vice-versa. There cannot be, for example, 2 consecutive long signals, as no trend reversal can be logically present in this case.
3. The strategy takes the form of a function, which also takes as input the period parameters for the two SMAs. This allows backtesting of multiple sets of periods for the two SMAs:

def strategySma5\_10(market, period, stratParams=[5,10]): # [5,10] are the default values of the SMAs, if no specific values are passsed  
 fastSMA = stratParams[0]  
 slowSMA = stratParams[1]

# …

data = execute(stock, strat.strategySmaCrossover, smaParams)

The algorithm, after downloading the price data, has a processing time of an average of 0.08 seconds per year of price data.

## 4.3. Data sample and set of strategy parameters

For the sample of companies, I chose two groups of tickers. The first group is the top 100 biggest and most popular companies from ‘stockanalysis.com’. This group includes companies like Apple, Google, Microsoft, Walmart, etc. The second group is formed by the 50 most popular and biggest companies which have a negative return over the last 10 years, in order to balance out the fact that the first group are all primarily winners in terms of returns. 8 companies from the criteria of the second group are also found in the first group, so the duplicates have been removed. The second group consists of companies such as Shell, Nike, Comcast, Intel, etc. Also, of importance is the fact, that all the companies selected satisfy the condition of having their stock listed publicly for at least the last 10 years. This ensures that price data is available for their respective stocks. In total, I will run the ‘backtest’ on the historic price data of 142 companies for the period 01/01/2013 – 15/06/2023.

The last thing to decide on is what period parameters I will use for the SMA crossover strategy. This means choosing of what periods the slow and the fast SMA will be. However, since, by implementation, I have the option of easily testing multiple combinations of parameters, I will choose to do exactly that. I will input the following parameters: 1, 2, 3, 5, 8, 10, 15, 20, 50. The SMAs will take on these parameters trough the method of combinatorics, with the constraint that the fast SMA needs to be of a lower period than the slow SMA. For example, the first combinations of fast SMA period – slow SMA period will be the following: 1-2, 1-3, 1-5, …, 2-3, 2-5, 2-8, etc.

fast = [1, 2, 3, 5, 8, 10, 15, 20] # parameters allowed for the fast SMA  
slow = [2, 3, 5, 8, 10, 15, 20, 50] # parameters allowed for the slow SMA

for sma1 in fast:  
 for sma2 in slow: # SMA parameters loop  
 if sma2 <= sma1: continue  
 smaParams = [sma1, sma2]  
  
 number = 0  
 for stock in stocks: # companies loop  
 data = execute(stock, strat.strategySmaCrossover, smaParams)

The algorithm sequentially tests each strategy variation for each stock in the tickers list it receives as input. In our specific case it results in 36 possible strategy variations each tested for 142 historic prices of stocks, for a total of 5112 results.

## 4.4. Compiling the results

The output from every strategy execution iteration also includes the total return generated by the strategy and the number of trades it has executed. The rest of the individual trades data is not used in the compilation of the results, as I am more interested in the general performance trend across the strategies, which will be calculated based on their total returns. More specifically, the ranking system will be based on the difference between the total result ( of the strategy and the absolute value of the respective company’s return during the period also chosen for testing the strategy (last ten years, in our case).

The reasoning behind this is that I want to test the effectiveness of the strategy, unrelated to the stock’s performance on which it is run on. In other words, the ‘score’ on which I will base the ranking, is the excess return generated by the strategy. This means that if a company’s stock over a ten-year period has a total return of 200%, and the strategy executed on it has a total return of only 150%, the strategy has resulted in deficit return of -50%. On the other hand, if the company has a total return of -10% and the strategy of 100%, the strategy has resulted in excess return of 90%. The absolute value of the company return is used because the strategy, being allowed both long and short positions, should be able to profit both from losing and winning companies.

Of important note is accounting for the cost of each trade taken. The cost per trade is implemented when calculating the return of the trade as follows:

To be able to calculate the total return of an executed strategy based on varying costs, the following formula is used:

This allows ex-post analysis of the total result for multiple cases of cost, without needing to rerun the entire script.

Finally, the result of each strategy execution is grouped by the parameters of the two SMAs. This allows comparison of the performance of the strategy variations, each having a sample of 142 sets of ten-year stock data. Additionally, the analysis is done for 5 cases of cost per trade: 1%, 0.75%, 0.5%, 0.25%, and 0%.

# 5. Results

An example of the executing a SMA5-SMA10 crossover strategy on Apple’s stock price from March to April of 2013 can be seen graphically below:

A picture containing diagram, line, map, plot

Description automatically generated

Figure 6- SMA5-SMA10 crossover strategy executed on Apple's stock price (Source: own research)

The green arrows indicate closing the previous short position and opening a long one and the red arrows indicated closing the previous long position and opening a short one. The data for the respective trades outputted by the algorithm:

|  |  |  |  |
| --- | --- | --- | --- |
| **Period** | **Date** | **Trade state** | **Trade return** |
| 47 | 12/03/2013 | existing position | 0.00% |
| 48 | 13/03/2013 | closelong | 5.88% |
| 49 | 14/03/2013 | existing position | 0.00% |
| … | | | |
| 59 | 28/03/2013 | existing position | 0.00% |
| 60 | 01/04/2013 | closeshort | -0.87% |
| 61 | 02/04/2013 | existing position | 0.00% |
| … | | | |
| 68 | 11/04/2013 | existing position | 0.00% |
| 69 | 12/04/2013 | closelong | -0.79% |
| 70 | 15/04/2013 | existing position | 0.00% |
| 71 | 16/04/2013 | existing position | 0.00% |
| 72 | 17/04/2013 | closeshort | -7.28% |

Table 2- Example of trade data (Source: own research)

Compiling the data, we get the results in the following form:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Fast-slow params.** | **Stock** | **Strat. return** | **No. of trades** | **Stock’s return** | **1% cost**  **test** | **0.75% cost test** | **0.5% cost test** | **0.25% cost test** | **0% cost**  **test** |
| 1-2 | AAPL | -201.48% | 1338 | 834.86% | -1036.34% | -701.84% | -367.34% | -32.84% | 301.66% |
| 1-2 | MSFT | -389.58% | 1346 | 1110.31% | -1499.90% | -1163.40% | -826.90% | -490.40% | -153.90% |
| 1-2 | GOOGL | -324.02% | 1298 | 584.16% | -908.19% | -583.69% | -259.19% | 65.31% | 389.81% |

After the data is grouped by the parameters of the SMA crossover strategy and averaged we get the following results, for 1% cost scenario:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Fast SMA** | **1% cost scenario, average results by parameter group** | | | | | | |
| **Slow SMA** | **1** | **2** | **3** | **5** | **8** | **10** | **15** | **20** |
| 2 | -707% |  |  |  |  |  |  |  |
| 3 | -493% | -473% |  |  |  |  |  |  |
| 5 | -378% | -298% | -309% |  |  |  |  |  |
| 8 | -324% | -256% | -245% | -239% |  |  |  |  |
| 10 | -310% | -247% | -230% | -214% | -226% |  |  |  |
| 15 | -304% | -244% | -223% | -207% | -204% | -209% |  |  |
| 20 | -299% | -244% | -226% | -214% | -213% | -216% | -217% |  |
| 50 | -324% | -292% | -280% | -271% | -267% | -266% | -272% | -270% |

Table 3- 1% cost scenario, average of parameter groups (Source: own research)

As it can be seen, with 1% cost per trade, no group can even come close to 0% excess return, meaning they are all in deficit. The most frequently trading strategy is the 1-2, with an average number of trades of 1312, is the worst performing by far. This can be explained simply by the fact that its performance takes a hit of 1312% just from costs alone. The ‘slower’ (less frequently trading), or in other words, the more long-term, the strategy is, the better it seems to perform under this cost scenario. The most short-term strategies are at the top-left of the table, while the most long-term are at the bottom-right. The difference in number of trades executed is also very noticeable:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Fast SMA** | **Average number of trades by parameter group** | | | | | | |
| **Slow SMA** | **1** | **2** | **3** | **5** | **8** | **10** | **15** | **20** |
| **2** | 1313 |  |  |  |  |  |  |  |
| **3** | 964 | 969 |  |  |  |  |  |  |
| **5** | 694 | 575 | 576 |  |  |  |  |  |
| **8** | 523 | 407 | 367 | 369 |  |  |  |  |
| **10** | 459 | 350 | 309 | 283 | 353 |  |  |  |
| **15** | 366 | 272 | 233 | 201 | 189 | 200 |  |  |
| **20** | 308 | 229 | 195 | 164 | 144 | 143 | 163 |  |
| **50** | 189 | 138 | 115 | 92 | 76 | 70 | 62 | 59 |

Table 4- Average number of trades by parameter group (Source: own research)

The correlation between the number of trades executed and the performance of the strategy variations in the 1% cost scenario is -0.86, a very strong inverse correlation. We can observe the trend of short-term strategies under forming long-term strategies in the 1% cost scenario graphically:

Figure 7- 1% cost scenario, average results by parameter group (Source: own research)

For the 0.75% cost scenario, the situation is relatively similar, although a bit more optimistic:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Fast SMA** | **0.75% cost scenario, average results by parameter group** | | | | | | |
| **Slow SMA** | **1** | **2** | **3** | **5** | **8** | **10** | **15** | **20** |
| 2 | -378% |  |  |  |  |  |  |  |
| 3 | -252% | -231% |  |  |  |  |  |  |
| 5 | -204% | -155% | -164% |  |  |  |  |  |
| 8 | -193% | -155% | -153% | -147% |  |  |  |  |
| 10 | -195% | -159% | -153% | -143% | -137% |  |  |  |
| 15 | -213% | -176% | -165% | -157% | -157% | -159% |  |  |
| 20 | -221% | -186% | -177% | -173% | -177% | -181% | -176% |  |
| 50 | -277% | -258% | -251% | -248% | -248% | -249% | -256% | -255% |

Table 5- 0.75% cost scenario, average results by parameter group (Source: own research)

It can be observed now that the best performing strategies are now situated in the middle of the short-term-long-term spectrum:

Figure 8- 0.75% cost scenario, average results by parameter group (Source: own research)

This is due to the fact, which will become very clear when observing the low-cost scenarios, that short-term strategies are indeed much better performers than long-term ones, as they can react to the market changes quicker and more efficiently. However, they are also much more susceptible to the impact of high costs, due to their relatively larger number of trades executed.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Fast SMA** | **0.5% cost scenario, average results by parameter group** | | | | | | |
| **Slow SMA** | **1** | **2** | **3** | **5** | **8** | **10** | **15** | **20** |
| 2 | -50% |  |  |  |  |  |  |  |
| 3 | -10% | 11% |  |  |  |  |  |  |
| 5 | -31% | -11% | -20% |  |  |  |  |  |
| 8 | -63% | -53% | -62% | -55% |  |  |  |  |
| 10 | -80% | -72% | -75% | -72% | -49% |  |  |  |
| 15 | -121% | -108% | -107% | -107% | -110% | -109% |  |  |
| 20 | -144% | -129% | -128% | -132% | -141% | -145% | -136% |  |
| 50 | -229% | -223% | -222% | -225% | -229% | -231% | -240% | -240% |

Table 6- 0.5% cost scenario, average results by parameter group (Source: own research)

At 0.5%, the first average positive result can be observed. The 2-3 strategy resulted in an average of 11% excess return. After lowering the cost, it can be observed that the best performing strategies shift towards the short-term end of the spectrum, meaning they are able to take advantage of quick reaction to market changes.

Figure 9- 0.5% cost scenario, average results by parameter group (Source: own research)

Referring to the number of trades executed by strategy parameters table, the correlation between the number of trades executed and the return of the strategy is 0.75, a strong correlation. The ‘slowest’ strategies are under-performing by a massive margin in this scenario, and this trend will only become more exaggerated as the cost is lowered to 0%.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Fast SMA** | **0.25% cost scenario, average results by parameter group** | | | | | | |
| **Slow SMA** | **1** | **2** | **3** | **5** | **8** | **10** | **15** | **20** |
| 2 | 278% |  |  |  |  |  |  |  |
| 3 | 231% | 254% |  |  |  |  |  |  |
| 5 | 143% | 133% | 124% |  |  |  |  |  |
| 8 | 68% | 49% | 30% | 37% |  |  |  |  |
| 10 | 34% | 16% | 2% | -1% | 39% |  |  |  |
| 15 | -30% | -40% | -49% | -57% | -62% | -59% |  |  |
| 20 | -67% | -72% | -79% | -91% | -105% | -109% | -95% |  |
| 50 | -182% | -189% | -193% | -202% | -210% | -213% | -225% | -226% |

Table 7- 0.25% cost scenario, average results by parameter group (Source: own research)

At 0.25% cost, the 1-2 boasts an impressive 278% excess return, or 14.2% annualized return for the 10 years the strategy ran. The ‘slower’ the strategy at this cost level, the worse it performs. This means that the 0.25% cost per trade level is approximately the point above which the strategies are inefficient and under which it makes sense to actively trade.

Figure 10- 0.25% cost scenario, average results by parameter group (Source: own research)

Finally, at 0% cost, the situation is mostly green for the SMA crossover strategy:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Fast SMA** | **0% cost scenario, average results by parameter group** | | | | | | |
| **Slow SMA** | **1** | **2** | **3** | **5** | **8** | **10** | **15** | **20** |
| 2 | 606% |  |  |  |  |  |  |  |
| 3 | 472% | 496% |  |  |  |  |  |  |
| 5 | 316% | 276% | 268% |  |  |  |  |  |
| 8 | 199% | 150% | 122% | 129% |  |  |  |  |
| 10 | 149% | 103% | 79% | 70% | 127% |  |  |  |
| 15 | 61% | 28% | 10% | -7% | -15% | -9% |  |  |
| 20 | 10% | -15% | -30% | -50% | -69% | -74% | -54% |  |
| 50 | -135% | -154% | -165% | -179% | -191% | -196% | -209% | -211% |

Table 8- 0% cost scenario, average results by parameter group (Source: own research)

The 1-2 strategy is the definite best performer, with 606% total return or 21.5% annualized return.

Figure 11- 0% cost scenario, average results by parameter group (Source: own research)

The correlation between the number of trades executed and the return of the strategy is 0.96, almost perfect direct correlation. It is therefore conclusive that at 0% cost, the ‘fastest’, or most short-term strategy will be the best performer.

Finally, we can analyse how the correlation between number of trades executed, an indicator of where on the ‘fast’- ‘slow’ spectrum the strategy is situated, and its performance:

Figure 12- Correlation between number of trades executed and performance by cost level (Source: own research)

We can conclude that somewhere between 0.75% and 0.5% cost is the point where the correlation inverses. In other words, at 0.75% cost per trade and higher, generally, the more long-term the strategy is the better it will perform. At 0.5% cost per trade and lower, generally, the more short-term the strategy is the better it will perform. More detailed researched around the 0.5%- 0.75% cost interval is needed to determine exactly the point of 0 correlation. However, while interesting, knowing the exact point isn’t of much practical use.

# 6. Conclusions

Research was conducted in the form of quantitative analysis on how variations of a SMA crossover strategy, representative of an average investor’s strategy, would perform. The method used for testing the strategies was a ‘backtest’ ran on the last ten years of stock prices of 142 popular companies, including both positive and negative ten-year return companies. 36 sets of parameters were tested for the SMA crossover strategy for each of the 142 sets of stock data, resulting in a total of 5112 results. The results were then adjusted to account for five different scenarios of cost per trade: 1%, 0.75%, 0.5%, 0.25%, and 0%. The results were grouped and averaged by strategy parameters. The result of the strategy variation, measured by calculating the excess return generated, was obtained by subtracting the absolute value of the respective stock’s return and total costs from the return of the strategy. The results were then analyzed to observe whether excess return was present and what impact active trading costs had on excess return.

For the 1% and 0.75% cost per trade scenarios, all strategy variations were in deficit return, meaning they returned a lower value than the absolute value of the stock’s return on which each respective strategy was run. For the 1% cost scenario, the best-performing strategies were the most long-term ones, which executed the least number of trades. The correlation between the number of trades executed by the strategy and the strategy’s performance for the 1% cost scenario was -0.86. For the 0.75% cost scenario, the best performing strategies shifted moderately towards the short-term end of the spectrum, being situated in the middle between the most short-term and the most long-term strategies. The 0.5% cost scenario featured the first strategy resulting in excess return. While still almost all strategies being in deficit return, the correlation between the number of trades executed by the strategy and the strategy’s return inversed, having a value of 0.75. This is conclusive of the fact that at 0.5% cost per trade and lower, generally, the more short-term the strategy is the better it will perform and, vice-versa, at 0.75% cost per trade and higher, generally, the more long-term the strategy is the better it will perform. Still, without researching other fundamental types of trading strategies, this conclusion cannot be validly extended past the case of the SMA crossover strategy. The 0.25% and 0% cost scenarios resulted in many strategies achieving excess return, the correlation between number of trades executed and performance being 0.93, respectively 0.97. The best strategy in the 0% cost scenario was the 1-2, averaging at 1312 trades executed and 21.5% excess annualized return.

Concluding, the impact of 1% extra costs has a tremendous negative influence on the performance of trading strategies, especially strategies that execute trades frequently. The interval 0.25%-0.5% cost per trade is the maximum cost that allows the best of the trading strategies tested to achieve excess return.

The limitations of the analysis are that only one type of fundamental trading strategy was tested, which is solely based on the evolution of price trends, and that a relatively simple strategy was tested, although with a multitude of parameters (variations). Also, ideally the sample of stock prices of the ‘backtest’ would be larger.

Further analyses can be conducted involving the performance of more complex, better, and fundamentally different strategies and the impact that varying levels of costs have on them.