Zurich University
of Applied Sciences

School of
Engineering
IDP Institute of Data Analysis
and Process Design

Project work at Engineering and Management

Allocation tool for asset management from Mobiliar

Author	Pascal Simon Bühler Philipp Rieser
Main Supervisor	Prof. Dr. Marc Wildi
Industrial Partner	Mobiliar
Date	19.12.2020

Zurich University
of Applied Sciences

School of
Engineering

DECLARATION OF ORIGINALITY

Project Work at the School of Engineering

By submitting this project work, the undersigned student confirm that this work is his/her own work and was written without the help of a third party. (Group works: the performance of the other group members are not considered as third party).

The student declares that all sources in the text (including Internet pages) and appendices have been correctly disclosed. This means that there has been no plagiarism, i.e. no sections of the project work have been partially or wholly taken from other texts and represented as the student's own work or included without being correctly referenced.

Any misconduct will be dealt with according to paragraphs 39 and 40 of the General Academic Regulations for Bachelor's and Master's Degree courses at the Zurich University of Applied Sciences (Rahmenprüfungsordnung ZHAW (RPO)) and subject to the provisions for disciplinary action stipulated in the University regulations.

City, Date: Signature:

....

.....

.....

The original signed and dated document (no copies) must be included after the title sheet in the ZHAW version of all project works submitted.

Zurich University
of Applied Sciences

Contents

Abstract

In this project, 4 time-series are investigated with quantitative methods. Different technical approaches are used to try to outperform the established and often used buy-and-hold strategy. In the beginning, the theory is explained superficially. In search of the best parameters, specific optimizations are calculated according to three approaches and the results are discussed. Many of the models found lead to the conclusion that buy-and-hold is the more profitable strategy for these time series. Only with the rather simple autoregressive processes, we could find models which are able to outperform buy-and-hold in performance.

1. Introduction

The main purpose of trading is buying and selling stocks, bonds, or other financial instruments with increasing returns of investments in mind while maintaining relatively low risk. With the help of a trading strategy, an investor can try to improve his performance. One can simply divide the strategies into passive and active. The praised and well established passive strategy buy-and-hold takes no short price movements into account. Positioning and trading based on these short price movements are considered active trading.

This paper applies time-series analysis to these short price movements to create active trading strategies. The objective of these developed strategies is to outperform the buy-and-hold strategy.

1.1. Data Analysis

The dataset which will be analyzed in this paper contains 4 tradeable indexes, a visualization of the data is shown below in figure 1.

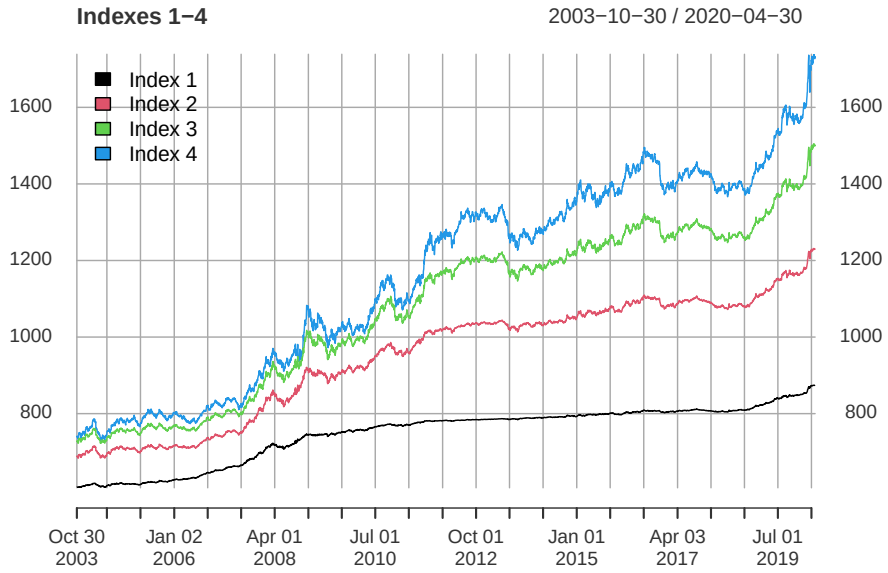

Figure 1: Visualization of the 4 indexes

Each time-series has 4306 observations and starts from October 2003 to April 2020. In all indexes is an upward drift observable, during the time period of the great recession (2008) is a slight bump visible. Also later in 2013 and 2016 are small break-ins evident. More interesting is the up and down behavior at the end of the series during the Covid19 pandemic.

In addition, to the indexes, the dataset contains 8 different interest rates of treasury bonds which will not be used for further analysis. However, a few key-values of the interest rates are shown in the following table 1 anyway.

Table 1: Summary of the 8 interest rates.

	Maturity	Mean	Volatility	Min.	Max.
Interest 1	3M	4.09	4.95	-0.28	16.27
Interest 2	6M	4.47	5.05	0.01	16.73
Interest 3	1Y	4.64	4.22	0.20	10.51
Interest 4	2Y	5.42	4.58	0.49	16.58
Interest 5	3Y	6.16	4.33	0.75	16.51
Interest 6	5Y	7.41	3.82	1.06	16.44
Interest 7	7Y	9.50	3.31	1.71	16.63
Interest 8	10Y	11.61	2.93	3.13	17.47

A typical characteristic of interest rates is shown in the given data. A bond with longer maturities is often associated with higher returns compared to those with shorter maturities. An investor which invests in short-term treasury bonds will have his gain earlier but will be confronted with a lower return.

A more in-depth analysis of the four indexes in the given dataset will follow in section 3.1.

1.2. Ojective of this paper

The objective of this paper is to trade these 4 indexes with an active trading strategy. The main objective is trying to outperform the passive buy-and-hold strategy. Methods such as the Moving-Average-Filter or the ARMA-GARCH-Model provide signals for either long or short the position to maximize the return of investments in these indexes.

The performance of these strategies is built upon various parameters and conditions. The lengths of the filters applied to a Moving-Average may result in different solutions. Models could perform differently for any given length of the in-sample or out-of-sample scope. The necessity of including a historical crisis in the starting-sample can decide if a model performs better or worse than another. The correct validation of model parameters could have a significant impact on the forecasts.

In addition to all criteria and conditions, strategies can be further customized by creating differently weighted portfolios, estimated predicted volatility can be used to modulate the position size to mitigate the risk. For the sake of completeness, we address this only superficially in the paper but do not elaborate further on it. Estimated predicted volatility can be used to modulate the position size to mitigate the risk.

Challenging will be finding the most optimal model in this wide field of conditions and parameters. The buy-and-hold strategy will be used as a benchmark to be compared with the developed active trading strategies. Computing and comparing the Sharpe ratios of each model can serve as an indicator to rely on for better or worse models.

2. Theory

It is assumed that the reader of this paper already has a basic knowledge of the mathematical principles of time-series analysis. Therefore, this section will only briefly describe the mathematical models and processes. This theory part is justified by the presentation of the different models and goes beyond the applied models in methodology.

2.1. Time-Series

Almost anything with a data point to a given timestamp can be named a time-series. The monthly gross domestic product, the weekly US gasoline production, or daily stock price movements. In this paper the focus lies in the analysis of financial time-series. Due to trades often only take place during the week, there are gaps in the time-series on the weekends, an exception would be the trading of cryptocurrencies like Bitcoin which are also tradeable at the weekends.

A series of data points with more or less equidistant time points t with the sample length of T, is called a time-series $x_t, t = 1, ..., T$ [1]. The analysis of a time-series x_t involves creating a reasonable model that can be utilized to perform forecast predictions.

2.1.1. Stationarity

In order to fit a suitable model with a given time-series x_t, the assumptions of stationarity must be met. In this practical application, only the following weak-stationarity properties are required.

$$E[x_t] = \mu \tag{1}$$

$$Var(x_t) = \sigma_x^2 \tag{2}$$

$$Cov(x_t, x_{t-k}) = R(k) \tag{3}$$

Many financial time-series are subject to shift, trends or changing volatility. In figure 2 are the stock prices of Alphabet Inc Class A (Google) visualized. This time-series shows a clear upwards drift and towards the end the volatility increases.

Figure 2: Visualization of the adjusted prices of the Alphabet Inc Class A Stock.

To improve the violated properties the first difference can be applied and additionally a logarithmic transformation can be performed [2]. The log-returns transformation can only be performed to strict positive data.

$$\text{LogReturn} = \log(x_t) - \log(x_{t-1})$$

The result is the so-called log-returns.

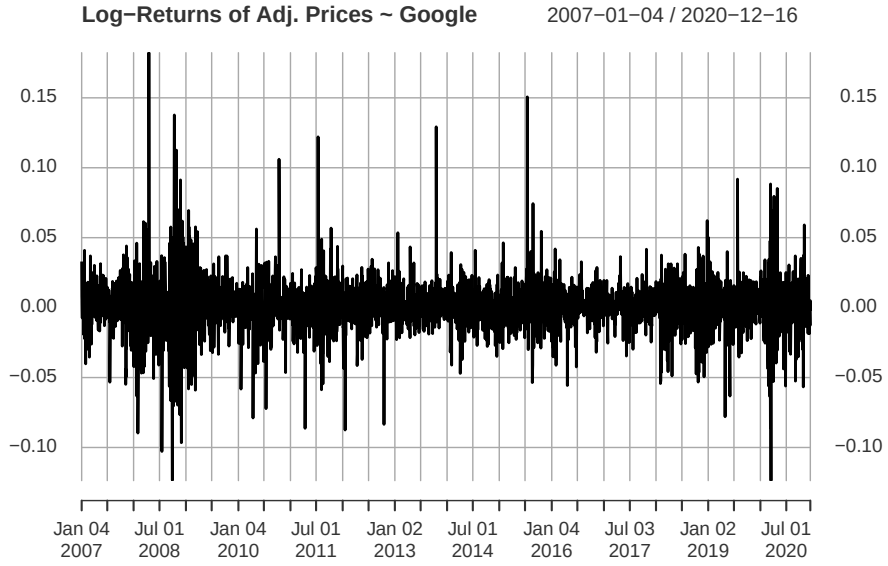

Figure 3: Visualization of the Log-Returns

Applying the transformation to the data causes the drift to disappear, but the series still contains stronger and weaker volatile phases. This effect often occurs in non-stationary financial data and is called volatility cluster. This special property is used for the modeling of forecast models, which will be discussed in chapter 2.2.

In the following examples, we will only work with a section of the time-series, as it often does not make sense to look far into the past. The further a value lies in the past, the smaller its influence on a future value will be.

2.1.2. Autocorrelation

The autocorrelation function (ACF) reveals how the correlation between any two data points of the time-series changes as their separation changes [3]. More precisely, ACF measures the dependence between x_t and $x_{t\pm k}$ at lag k. The partial autocorrelation (PACF) measures the dependency between x_t and x_{t-k} at lag k [1]. For stationary time-series, ACF can be used to identify the model order of a MA-process, PACF for AR-processes.

In the following figure 4 are ACF and PACF of the non-stationary adjusted Google stock visualized. Both graphics show the typical pattern of a non-stationary time-series. The plot above shows the dependence structure of the time-series. This means that it takes a long time until the series changes. Often a large value is followed by another large value, which indicates a strong trend. This property of the series can be seen in figure 2 as the long upward drift. The plot below indicates a significant partial autocorrelation at lag $k = 1$.

In the following section 2.2. the characteristics of the autocorrelation function can be used for the verification of ARIMA and ARCH-processes.

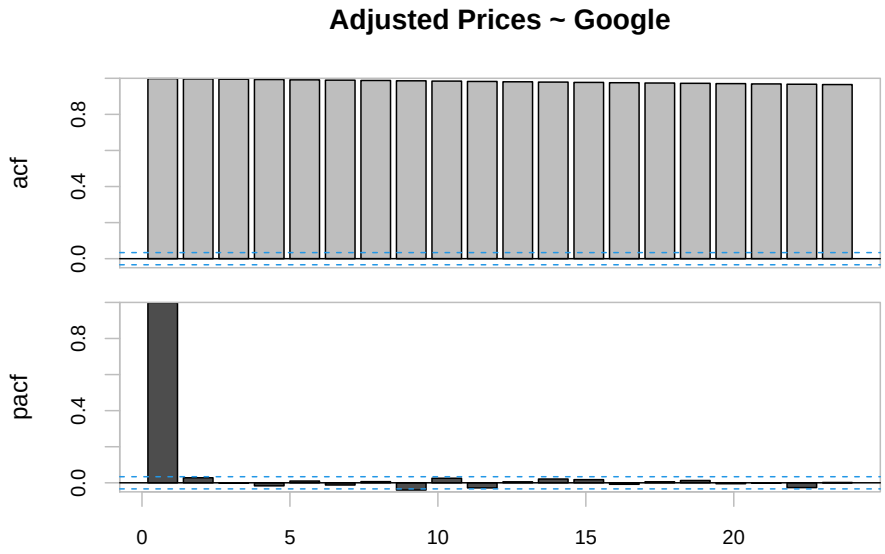

Figure 4: ACF and PACF of the Adjusted Prices of Google.

2.2. Models

The following processes are used to determine certain properties and characteristics of a time-series so that they are transformed into a model. The goal is to fit the time-series as well as possible in order to create reliable forecasts.

2.2.1. ARIMA

An ARIMA(p,d,q) process is defined as follows.

$$x_t = c + a_1 x_{t-1} + ... + a_p x_{t-p} + \epsilon_t + b_1 \epsilon_{t-1} + ... + b_q \epsilon_{t-q} \tag{4}$$

- p and q are the AR- and MA-model orders
- a and b are the AR- and MA-model parameters
- d is the differential parameter
- ϵ_t is a white noise sequence
- x_t is the given data $x_1, ..., x_T$

The mean of an ARIMA-process can be computed as:

$$\mu = \frac{c}{1 - a_1 - ... - a_p}$$

ARIMA processes can be divided into 4 different models. Choosing a model that best represents the time-series is a difficult task. The goal is to find the best possible model with as few parameters as possible.

The previously introduced ACF and PACF can help to determine the orders of simple models. Provided that the time-series is stationary, the model orders can be determined directly. For an AR(p)-process (ARIMA(p,0,0)), the ACF plot will gradually decrease and simultaneously the PACF should have a sharp drop after p significant lags. For an MA(q)-process (ARIMA(0,0,q)) the opposite is true, the ACF should show a sharp drop after a certain q number of lags while PACF should show a gradual decreasing trend. If both ACF and PACF show a gradual decreasing pattern, then the ARIMA(p,0,q)-process should be considered for modeling [4]. If the time-series is not stationary, differentiation can be considered (ARIMA(p,d,q)).

The application of an analysis method to Google prices finds an ARIMA(*1*,*1*,*0*) as the optimal model. This makes sense if you look back at figure 4. Long dependency structures in the ACF plot indicating an AR(p) process and at the same time after lag=1=p the PACF has a strong drop. The differential operator d=1 transforms the non-stationary series into a stationary one.

To convince yourself of the quality of the model, you can use the Ljung-Box statistics shown in figure 5. For the lags where the p-values are above the 5% line, the forecasts are reliable. Here the values from the 6 lag are below the significant line, but since one only wants to make short-term forecasts (for example 1 day into the future), the lag is sufficient up to 5. If you want to forecast further than 5 days into the future, you might have to adjust the ARIMA model.

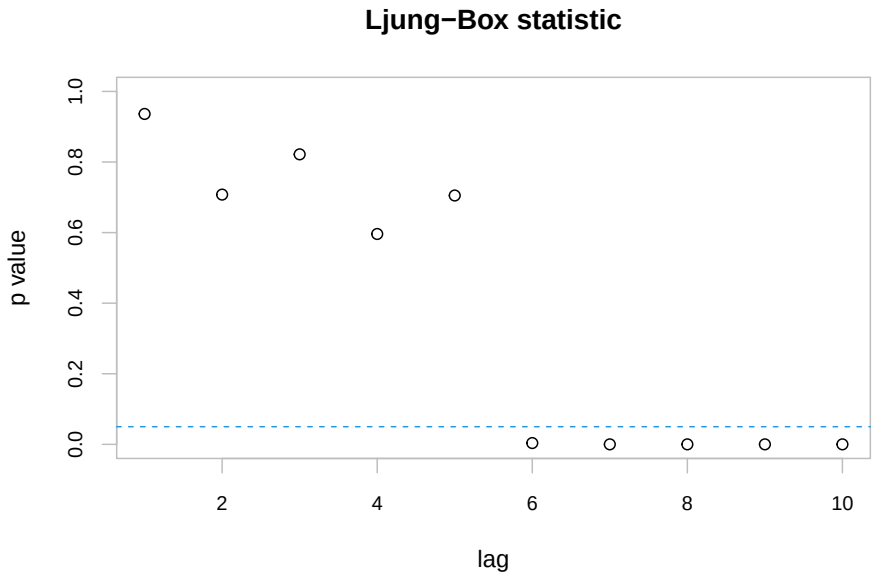

Figure 5: Ljung-Box statistic of Google log-returns.

In figure 6 you can see the prediction of the model. The whole representation is shifted by one day so that one can compare the model with a true value. The green dot is the actual value of the time-series. The red dots indicate the upper and lower 95% interval limits respectively. These indicate that a future value will be within this band. The blue dot is the point forecast predicted by the model.

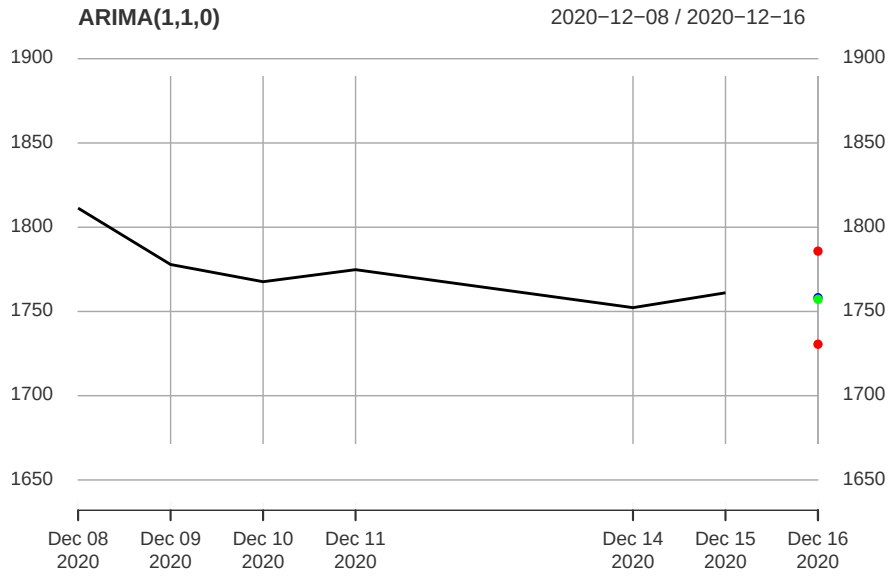

Figure 6: ARIMA-Forecast.

2.2.2. GARCH

The volatility clustering mentioned in section 2.1.1 can be handled with a generalized auto-regressive conditional heteroscedastic process.

$$\begin{aligned} \epsilon_t &= \log(x_t) - \log(x_{t-1}) \\ \epsilon_t &= \sigma_t u_t \\ \sigma_t^2 &= c\sigma^2 + \sum_{j=1}^{n} \alpha_j \sigma_{t-j}^2 + \sum_{k=1}^{m} \beta_k \epsilon_{t-k}^2 \end{aligned} \tag{5}$$

with

- x_t is the original data (often non-stationary)
- ϵ_t is the stationary log-return
- u_t is independent and identically distributed (iid) and a standardized random variable
- σ^2 is the unconditional variance of the process ϵ_t.
- σ_t^2 is the conditional variance of the process ϵ_t.

With a GARCH(n,m)-process it is possible to model the volaclusters of a time-series. The GARCH(1,1) model has become widely used in financial time-series modeling and is implemented in most statistics and econometric software packages. Those models are favored over other stochastic volatility models by many economists due to their relatively simple implementation [5].

Table 2: Coefficients GARCH(1,1).

	Estimate	Std. Error	p-Value
ω	1.088166e-05	1.840814e-06	3.393531e-09
α_1	7.329788e-02	1.073124e-02	8.472112e-12
β_1	8.953285e-01	1.373016e-02	0.000000e+00

For an optimal model some conditions must be fulfilled. Suppose you want to model the Google time-series with a GARCH(*1,1*).

In table 2 the estimated coefficients of the process can be seen. The p-values are all lower than 0.05 and thus indicate that they are essential for the model. (Note: $\omega = c\sigma^2$)

The following parameter restrictions are also examined:

$$c + \sum_{j=1}^{n} \alpha_j + \sum_{k=1}^{m} \beta_k = 1 \tag{6}$$

with

$$c > 0, \alpha_k \geq 0, j = 1, ..., n, \beta_k \geq 0, 1, ..., m$$

To satisfy formula 6, c needs to be determined from ω. First calculate the unconditional variance.

$$\sigma^2 = \frac{\omega}{1 - \alpha_1 - \beta_1}$$

Calculate c with:

$$c = \frac{\omega}{\sigma^2}$$

and then check for the restriction in 6.

For the coefficients of GARCH(*1,1*) the restrictions are fulfilled. You can see that $c = 1 - \alpha_1 - \beta_1$. So this restriction can be determined easily with:

$$\sum_{j=1}^{n} \alpha_j + \sum_{k=1}^{m} \beta_k < 1 \tag{7}$$

If the parameter restrictions are not fulfilled, complications may arise, the forecast of the conditional variance $\hat{\sigma}_t^2$ may diverge to the unconditional variance σ^2 of the process.

Furthermore, the Ljung-Box statistics are important for the standardized residuals. Looking back at formula 5, standardized residuals u_t are proportional to the conditional volatilities σ_t, which should lead to the log-returns ϵ_t. The conditional volatilities map the volacluster in the time-series. To achieve the best possible model, one does not want to find these volacluster effects in the standardized residuals, but only in the conditional volatilities. The Ljung-Box statistics check this property. In figure 7, Ljung-Box statistics of the $\hat{u}_t$ and the $\hat{u}_t^2$ are shown.

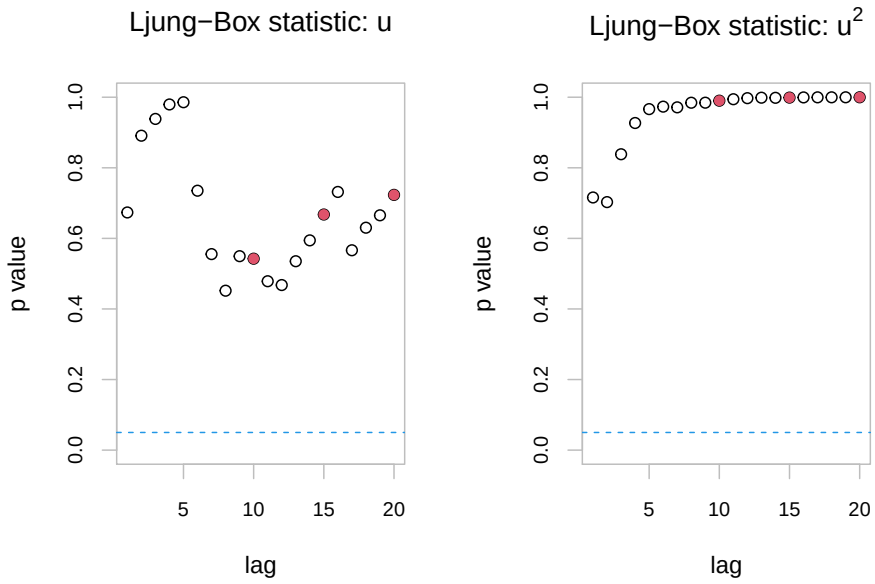

Figure 7: Ljung-Box statistic of the standardized residuals.

The plot shows reliable statistics for forecasts from the Google-GARCH(*1,1*) model. For all lags up to 20, the p-values are above the 5% line and thus hypothesis tests are discarded. However, if for a given lag=k the p-values would fall below the 5% line, then forecasts would only be reliable up to a forecast horizon k.

If one wants to improve the standardized residuals $\hat{u}_t$ (if autocorrelations exists in the $\hat{u}_t$), an ARMA part would have to be added to the existing model (see 2.2.3). This can again be optimized with different model orders. If you want to improve the squared standardized residuals $\hat{u}_t^2$ (if volaclustering exists within the $\hat{u}_t^2$), then you should modify the GARCH model order.

Now an optimal model has been found and a forecast can be made. Since a GARCH(n,m) process is a white noise sequence the expected value $E[\epsilon_{T+h}|\epsilon_T, ..., \epsilon_1] = \mu = 0$ can be assumed (if the mean value in the fit object was also estimated and is significant, then the expected value is the estimated μ).

Calculating the forecast variance is a recursive process. With increasing model order the calculation becomes more and more difficult. For this work the rather simple calculation for a GARCH(*1,1*) model is sufficient. One receives:

$$\hat{\sigma}_{T+h}^2 = \omega + (\alpha_1 + \beta_1)^2 \hat{\sigma}_{T+h-1}^2$$

If the parameter restriction from formula 7 is true, the forecast variance converges with the increasing forecast horizon to the unconditional variance of the process.

$$\hat{\sigma}_{T+h}^2 = \frac{\omega}{1 - \alpha_1 - \beta_1} = \sigma^2$$

The 95% forecast interval is calculated as followed:

$$E[\epsilon_{T+h}|\epsilon_T, ..., \epsilon_1] \pm 1.96\sqrt{\hat{\sigma}^2_{T+h}}$$

Figure 8 shows the GARCH(*1,1*) forecast for 20 days. The blue line represents the expected value of the time-series. Reds are the two 95%-interval limits and green shows the actual values of the series.

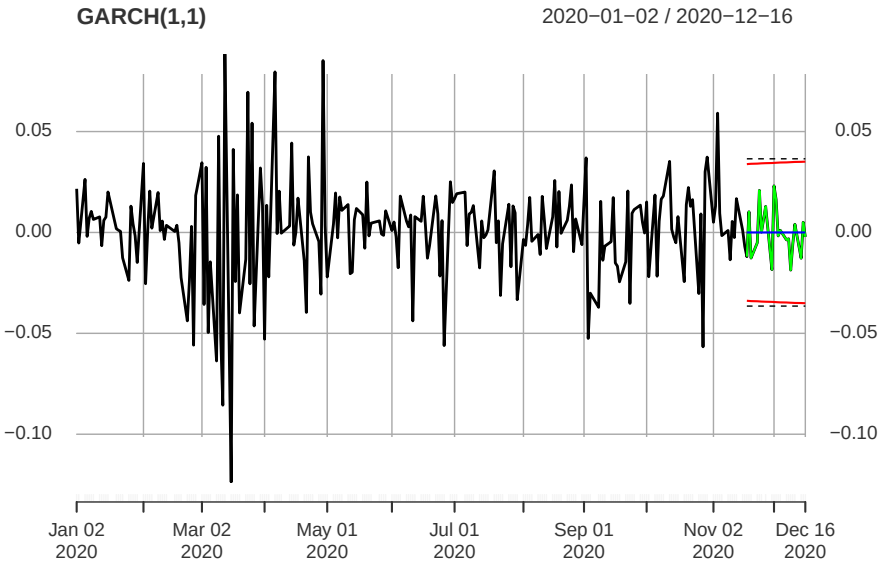

Figure 8: GARCH-Forecast.

You can see how the red lines are approaching the black dotted lines ($0 \pm 1.96 * \sqrt{\sigma^2}$) as the forecast horizon increases. This means that the forecast variance converges to the unconditional variance and thus there is another proof for the satisfaction of the parameter restrictions.

2.2.3. ARMA-GARCH

If you want to adopt a GARCH model and you discover non-vanishing autocorrelations in the standardized residuals $\hat{u}_t$, the model can be extended with an ARMA part to counteract the autocorrelations.

$$y_t = \mu + a_1 y_{t-1} + ... + a_p y_{t-p} + \epsilon_t + b_1 \epsilon_{t-1} + ... + b_q \epsilon_{t-q} \quad (8)$$

$$\epsilon_t = \sigma_t u_t$$

$$\sigma_t^2 = c\sigma^2 + \sum_{j=1}^{n} \alpha_j \sigma_{t-j}^2 + \sum_{k=1}^{m} \beta_k \epsilon_{t-k}^2 \quad (9)$$

This model contains now 4 different model orders: p,q,n and m. Experience shows that for financial time-series, a model order of n=m=1 and p,$q \leq 1$ is often sufficient to fit the data to an ARMA-GARCH model.

The equation 8 is called the *mean-equation* and describes the conditional mean of the process, which is used to obtain optimal point forecast. The equation 9 defines the variance of the process and is called *variance-equation*, which is used for the forecast error variance [6].

2.2.4. M-GARCH

The last model that is considered as a forecast model in this thesis, is the so-called MGARCH model. Highly volatile phases indicate downturns. Market participants tend to oversell during these downturns. Overselling leads to inflated volumes and this in turn leads to inflated volatilities. This model can be helpful since it emphasizes recession and crisis dynamics.

$$\begin{aligned} r_t &= \log(x_t) - \log(x_{t-1}) \\ r_t &= \mu + e\sigma_t + \epsilon_t \\ \epsilon_t &= \sigma_t u_t \end{aligned} \tag{10}$$

where

- x_t is the original data (typically non-stationary)
- r_t are the log-returns (stationary)
- μ is the long-term drift
- ϵ_t is a volacluster process (GARCH)
- e is a constant (a parameter to be estimated), $e > 0$ implies a larger expected return. $e < 0$ would imply a smaller expected return. If $e = 0$ then the MGARCH-effect vanishes [7].

For this model, the in-sample conditional standard deviations (volatilities) from any GARCH process are determined and the out-of-sample conditional standard deviation for obtaining forecasts of the future returns is then calculated by regression.

$$\hat{r}_{t+1} = \hat{\mu} + \hat{e}\hat{\sigma}_{t+1}$$

2.3. Moving Average Filters

Moving-Average-Filters are basically used to identify trends and smooth out price fluctuations. As a commonly used tool, Moving-Average-Filters are very simple in their usage, historical data from a timeframe L gets summarized and divided by the length of the filter (L). Depending on the length of the filter, the MA gets shifted, longer filters have a higher shift then shorter ones, we visualize this behavior in 9. Many different indicators are built upon the Moving-Average principle, mostly they are used in combinations of different lengths to create signals. In the following section, we introduce some of the most popular indicators based on the MA principle.

The actual challenge in using Moving-Average-Filters, is to figure out which length of the filter brings the most useful information.

2.3.1. Equally-weighted Moving Average or SMA

SMA stands for Simple Moving Average, depending on the length of the filter(L), L observations since the last noted observation will be considered. The observations are getting summarized and divided by the filterlength L. As stated in the name, all past observations are weighted equally. For every timestep, a new observation is considered and the last one eliminated [8]. SMA's are very easily customized by changeing the length of the filter.

EqMA

$$y_t = \frac{1}{L} \sum_{k=0}^{L-1} x_{t-k} \tag{11}$$

- L = filterlength
- x = original series price e.g.

2.3.1.1. Momentum

Momentum is an indicator wether a market is bullish or bearish, it measures the "speed" of the trend direction in the market. For a timespan k the last price p_k, k timesteps ago is subtracted from the last price. This is equivalent to applying and EqMA on the returns of a series.

Momentum

$$y_t = p_t - p_{t-K} \tag{12}$$

- p_t = Prices of the series
- K = Lag

2.3.2. Exponentially-weighted Moving Average

Since not all observations have the same influence on the future value, we can apply a weight to past observations. One method will be Exponentially-weighted Moving average. So we choose an optimal parameter to give past observations weights decreasing by α^k. In comparison to the SMA 2.3.1. [9], an EMA from the same length L, reacts faster than to price changes.

A skillful trader chooses an optimal α to increase the performance of the measurement. Weights could also be given individually by adding a weight vector to the filter.

EMA

$$y_t = \frac{1}{\sum_{k=0}^{m} \alpha^k} \sum_{k=0}^{m} \alpha^k x_{t-k} \tag{13}$$

- m = Filterlength
- α = Parameter to weigh the observations

2.3.3. Moving Average Crossings

Moving-Average-Crossings are basically just different MA's with different lengths applied to a time-series. The points the filters then cross, will be used as a trading signal to go long, short or hold. The "death" and "golden" cross are very popular trading patterns [9]. If a shorter MA crosses the longer MA from above, its called a "golden cross" it is an indicator that the price will rise in the future and can be used to create the buy signal. In contrast stands the "death cross" vise versa, a shorter MA (popular L = 50) crosses a longer MA (popular L = 200) from below, signalizing that further losses are in store.

MA Crossing

$$y_t = \frac{1}{L_1} \sum_{k=0}^{L_1-1} x_{t-k} - \frac{1}{L_2} \sum_{k=0}^{L_2-1} x_{t-k} \tag{14}$$

- L_1 = Filterlength 1
- L_2 = Filterlength 2
- $0 < \alpha < 1$ = Parameter to weigh the observations

An example of MA crossings with 2 SMAs of different length is visualized in figure 9. The prices are in green while the blue line represents a 50 day SMA an the red line a 250 (1 year) SMA. The crossing points of those two SMAs could now be used as trading signals. In this example these crossings would not perform very well, therefore as mentioned earlier, finding the right length of the filter depends on each timeseries, their behavior and the preferences of the trader.

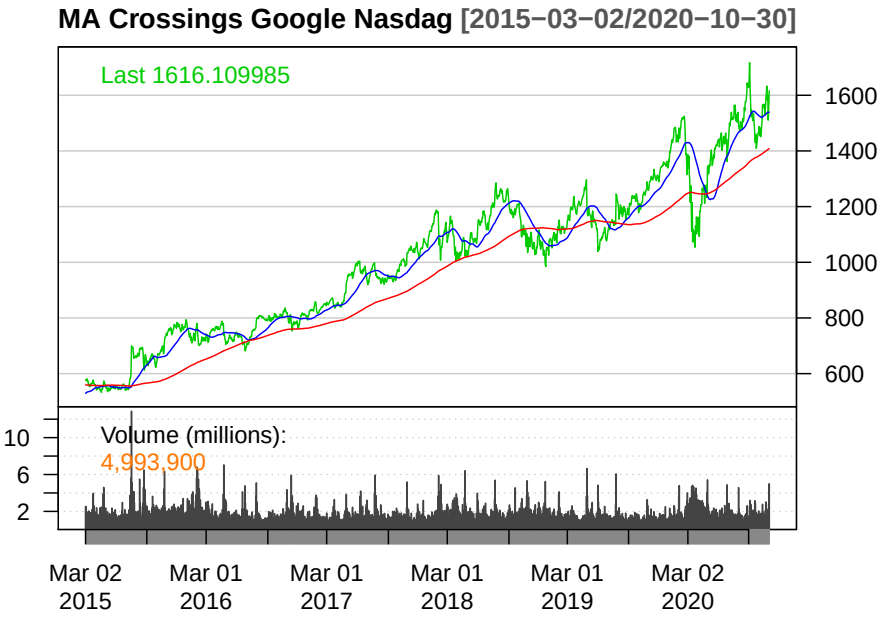

Figure 9: Moving Average Crossing

2.4. Relative Strength Index

The Relative strength index is a tool to measure momentum, the value indicates if positions are overvalued or undervalued. The scale goes from 0 to 100. [10].

$$U_t = \begin{cases} 1, \text{if } x_t \geq x_{t-1} \\ 0, \text{otherwise} \end{cases} \quad D_t = \begin{cases} 1, \text{if } x_t < x_{t-1} \\ 0, \text{otherwise} \end{cases} \tag{15}$$

- X_t =Original timeseries

We then apply SMA or EMA of length N to U_t and U_t which converts them in $up_t(N)$ and $down_t(N)$. The RSI is now computet by:

$$RSI_t(N) = 100 \frac{up_t(N)}{up_t(N) + down_t(N)} \tag{16}$$

The original developer J. Welles Wilder Jr. proposed a length of N =14 in its work from 1978 [11]. Traditional tradings signals based on the RSI are the upper 70 or lower 30 limit to buy or sell. Usually when the RSI is going over 70 it suppose that the asset is overvalued and in contrast when its under 30 then its undervalued.

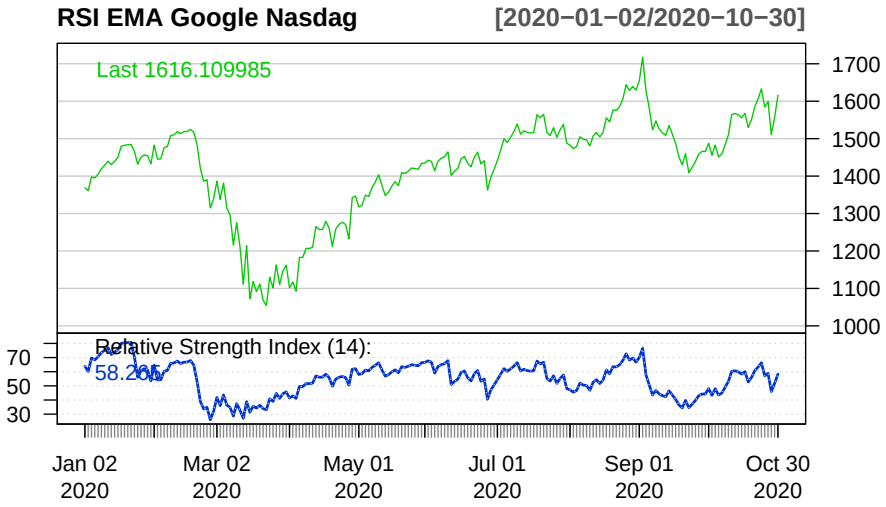

Figure 10: RSI

2.5. Moving Average Convergence Divergence

The MACD is also a commonly used filter. The basic principle is to subtract a longer EMA with length L as in section 2.3.3 from a shorter EMA from length S then smooth the result with another EMA with length R. As a result we can use the crossing of the 2 generated curves for trading. As an alternative we could use SMA's instead of EMA's

$$\mathrm{m}acd_t = \frac{1}{\sum_{k=0}^{t-1}\alpha^k}\sum_{k=0}^{t-1}\alpha^k x-t-k - \frac{1}{\sum_{k=0}^{t-1}\beta^k}\sum_{k=0}^{t-1}\beta^k x-t-k \tag{17}$$

$$MACDsignal_t = \frac{1}{\sum_{k=0}^{t-1}\gamma^k}\sum_{k=0}^{t-1}\gamma^k x-t-k \tag{18}$$

- x_t = prices or log prices
- S = length of the $short_1$ EMA usually 12
- L = length of the $long_1$ EMA usually 26
- R = length of the "double smoothing ema" usually 9
- $\alpha = 1-\frac{1}{S}$, $\beta = 1-\frac{1}{L}$, $\gamma = 1-\frac{1}{R}$

$_1$ short and long in the meaning s<l, not buy sell

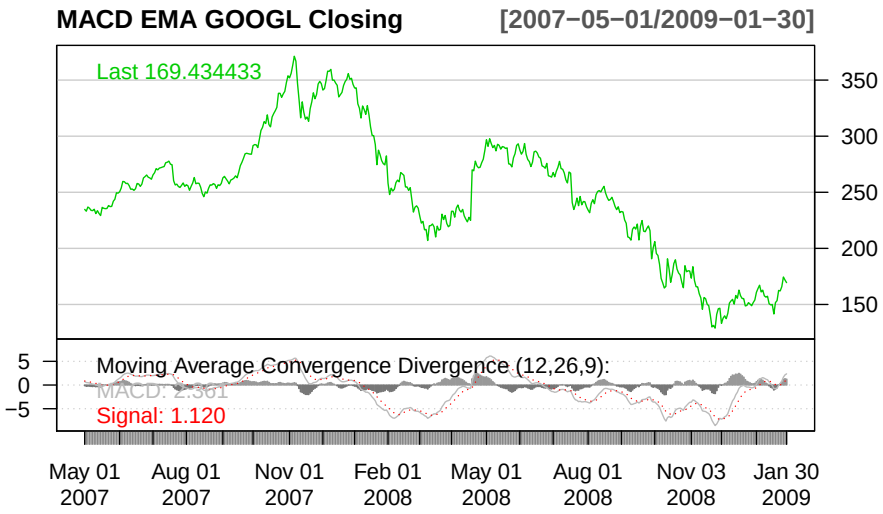

Figure 11: MACD

2.6. Bollinger bands

Bollinger bands are an analysis tool founded by John Bollinger. It contains a MA and an upper and lower band. The bands are defined by adding a constant K times a standard deviation σ_t to the MA for the upper, and subtracting it for the lower band.

$$\mathrm{U_t} = MA_t + K\sigma, L_t = MA_t - K\sigma \tag{19}$$

The variance from Bollingers theory is calculated by:

$$\sigma_\mathrm{t}^2 = \frac{1}{N}\sum_{k=0}^{N-1}(x_{t-k} - MA_t)^2 \tag{20}$$

The calculated σ_t could be problematic because it is derived from the original series and increases with the level, it is non-stationary. Therefore an other method to calculate the standard deviation could be used. As done in section 2.2.2. σ_t could be provided by a GARCH, which would handle the increasing volatility.

- N = usually the filterlength and the length considered for σ are the same
- K = Constant usually equals 2
- σ_p = standard deviation of the series
- U_t = upper band
- L_t = lower band

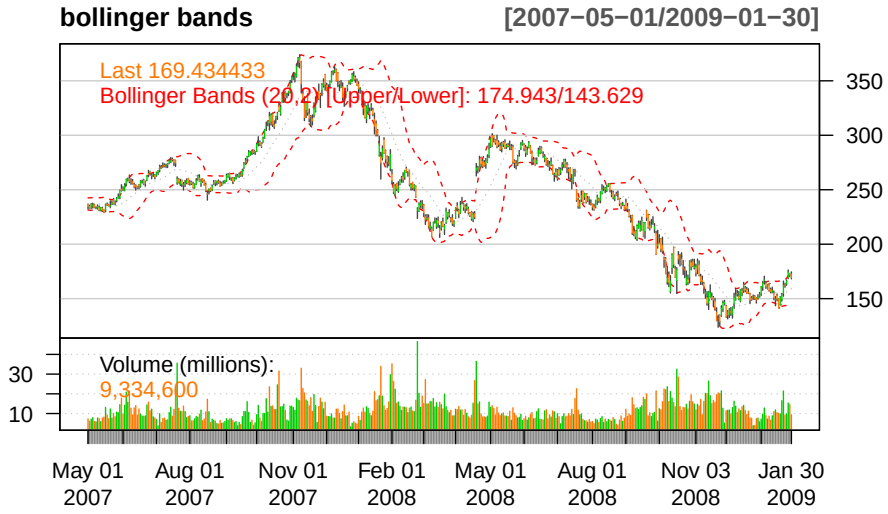

Figure 12: Bollinger Bands

2.7. Performance Indicators

2.7.1. Sharpe Ratio

Sharpe ratio is a very powerful and widely used ratio to measure performance. It describes return per risk.

$$\text{SharpeRatio} = \frac{R_p - R_f}{\sigma} \tag{21}$$

- R_p = Return of Portfolio
- R_f = Risk free Rate, mostly treasury bonds
- σ_p = standard deviation of portfolios excess return (risk)

For comparing different series often the Sharpe ratio is annualized with $\sqrt{250}$ and the R_f is considered 0.

2.7.2. Drawdown

Drawdown describes the maximum cumulated loss an investor is going to face if he is not leaving a falling market at the turning point from rise to fall of an index.

$$\text{MDD} = \frac{\text{bottomvalue} - \text{peakvalue}}{\text{peakvalue}} \tag{22}$$

Maximum drawdown is used as an indicator for risk or can be used as an optimization criterion[12].

2.8. Carry

Carry trades are trading strategies where usually money is borrowed at a lower interest rate, than the investment is giving in return. the risk of this strategy is based in the currency risk.

3. Methodology

In this section, different models are created and compared with the buy-and-hold strategy. You start with pure data analysis and then work your way from simple models to more and more complex ones.

3.1. Data Analysis

As mentioned in section 1.1 we are now going to analyze the data further to gain as much information as possible just by using some simple tools and comparisons.

3.1.1. Correlation

One could nearly tell just by looking at the indexes how strong they're correlated. The correlation matrix in table 3 confirms the assumption, the correlation is nearly 1 for every index to each other.

Table 3: Correlations oft the four indexes

	Index 1	Index 2	Index 3	Index 4
Index 1	1.0000000	0.9899111	0.9788826	0.9672956
Index 2	0.9899111	1.0000000	0.9975499	0.9921171
Index 3	0.9788826	0.9975499	1.0000000	0.9983460
Index 4	0.9672956	0.9921171	0.9983460	1.0000000

3.1.2. Transformation, Volatility and Clusters

Applying the natural logarithm to the series is an approach to cancel out increasing volatility 13. The strong upward drift is still visible, the original series has more than doubled to its original price over the whole timespan (index 4) 1.

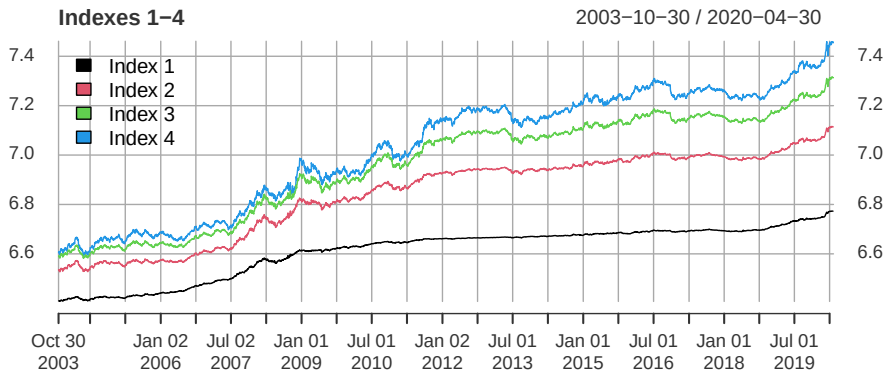

Figure 13: Visualization log indizes

By taking the returns of the transformed series we can visualize volatility clusters as seen in figure 14. The first value of the series is eliminated because of the differences. Clearly visible are the high spikes in the times of the financial crisis 2007-2009. Also at the end of the series, the impact of Covid19 in march 2020 is remarkable.

Table 4: Sharpe Ratio and Volatility of the 4 indexes (log-returns).

	Index 1	Index 2	Index 3	Index 4
Volatility	0.0008	0.0020	0.0029	0.0039
Sharpe-Ratio	1.6710	1.0601	0.9143	0.7922

As we can see in the table the volatility of index 4 is as much as 5 times higher than the volatility of index 1 and the Sharpe ratio behaves inverse to the risk.

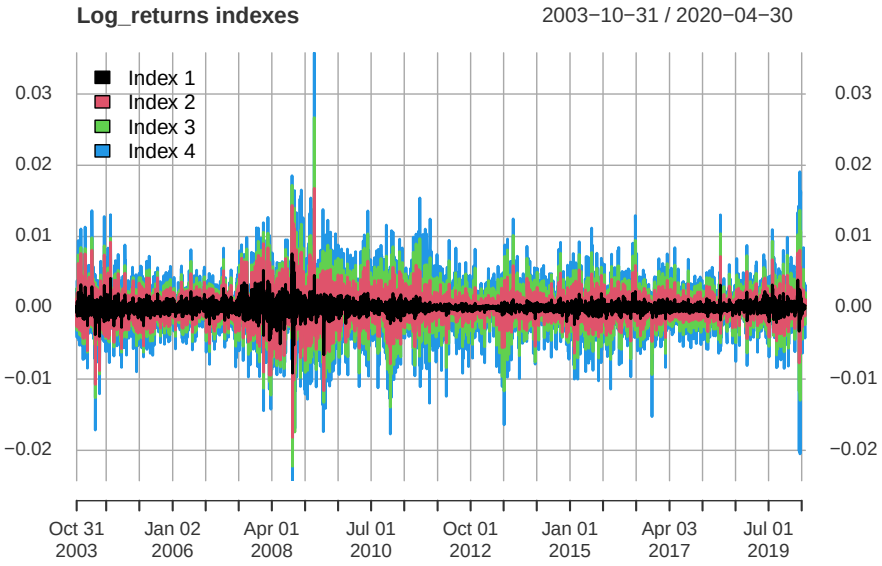

Figure 14: log-returns

3.1.2.1. Autocorrelation of log-returns

By computing the ACF of the squared log-returns we see that the volatility cluster has very long dependency structures.

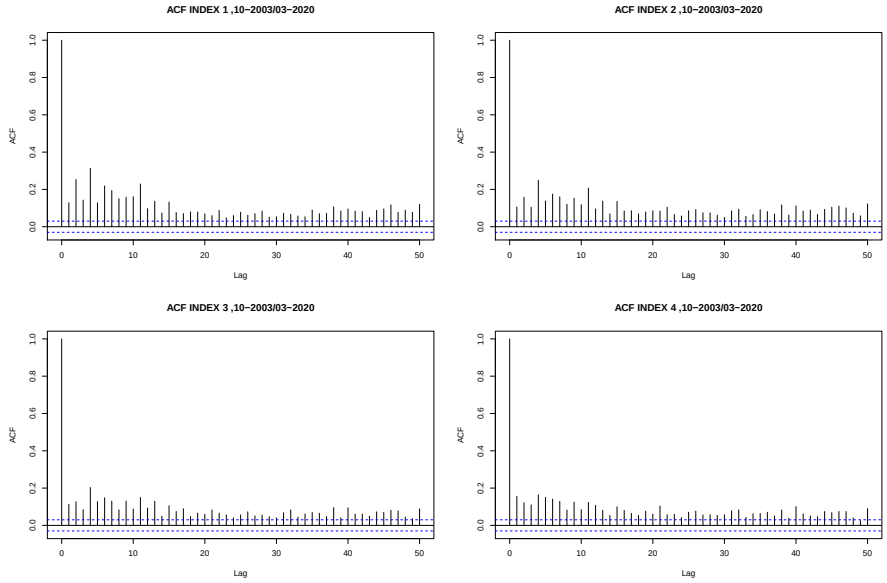

Figure 15: ACF log-returns

3.2. Trading

In section 2 we have learned different indicators and models for time-series-analysis. Some of these models and indicators are now used to trade the indexes we have introduced in the previous section.

To do so we need to create trading signals based on the models and indicators. For example, we are using the MA Crossings, as mentioned 2.3.3. the points where the two MAs cross, are now used to create a trading signal. when the longer MA comes from below to the crossing we are going long the asset and if it approaches the point from above we are shorting the position. Technically we apply a 1 to a vector at each crossing, where we intend to buy and apply a -1 at the points we want to sell.

3.2.1. Buy-and-Hold Performance

As mentioned earlier the goal of this work is trying to outperform the buy-and-hold strategy. Because the series all have a strong upward trend, this task is very tricky. Buy-and-hold has very low trading costs because the underlying is just bought once. According to swissquote [13] costs for asset trades over 50k, are 190 USD per trade [1], so these costs should also be taken into consideration for choosing the strategy. In this work, we have mostly excluded these costs and only concentrate on the trades themselves.

[1] notice: This fee is only for private investors, conditions may differ for institutions.

For the following models a good comparability should be achieved. To ensure this, all models are created with the same out-of-sample range, with the start date 2019-01-01.

3.2.1.1. Portfolios

By focussing on trading one could build a portfolio for the 4 indexes. One approach would be the equally weighted portfolio with weights for every index of $\frac{1}{4}$. As we have seen in figure 14 the volatility of the indexes strongly differs, meaning that index 4 would have the most impact, nearly 50 %, on the portfolio by weighing it equally. To cancel this effect, it may be useful to size the position with the inverse volatility, so the indexes have the weights seen in the table on the right side.

$$\text{procentual share } \sigma_k = \frac{\sigma_k}{\sum_{k=1}^{4} \sigma_k} \tag{23}$$

Table 5: Shares

	Weight 1	Weight 2
Index 1	3.70980	70.246503
Index 2	16.15568	16.130577
Index 3	31.55567	8.258436
Index 4	48.57885	5.364485

[1] Weight 1: Equally weighted volatility in %

[2] Weight 2: Weighted with inverse volatility

3.3. AR trading

As with the theory data set, these 4 indexes are non-stationary series. For the AR models, the stationary log-returns of the time-series are used. First we look at index 1, the series with the lowest volatility but with the highest sharpe ratio (highest return with lowest risk). To find the best possible AR model, different AR(p) models are fitted with different model orders p for different in-sample ranges. The parameters p and start date are so called hyper parameters, parameters which are set before modelling. For each model the out-of-sample sharpe is calculated. The execution of this calculation leads to the solution shown in table 6.

Table 6: Solution of the AR-Model calculation.

StartDate	AR-Sharpe	p	StartDate	AR-Sharpe	p
2003-01-01	3.296	1	2011-01-01	2.004	3
2004-01-01	3.037	1	2012-01-01	1.852	3
2005-01-01	3.146	1	2013-01-01	2.033	4
2006-01-01	2.957	1	2014-01-01	1.954	3
2007-01-01	2.722	1	2015-01-01	1.932	5
2008-01-01	1.867	1	2016-01-01	3.449	2
2009-01-01	2.473	1	2017-01-01	2.163	4
2010-01-01	2.651	1	2018-01-01	3.765	2

With these calculations, the optimal model is an AR(2) with the start date 2018-01-01. Note that the index 1 series has a very flat rising trend and a rising upward trend towards the end of 2018, so only a short in-sample is needed for this optimal model. It is interesting that the long in-sample ranges (2003 to 2007) also lead to relatively good performances. This probably has something to do with the fact that the trend at the beginning of the series has a higher influence on the models than the middle part.

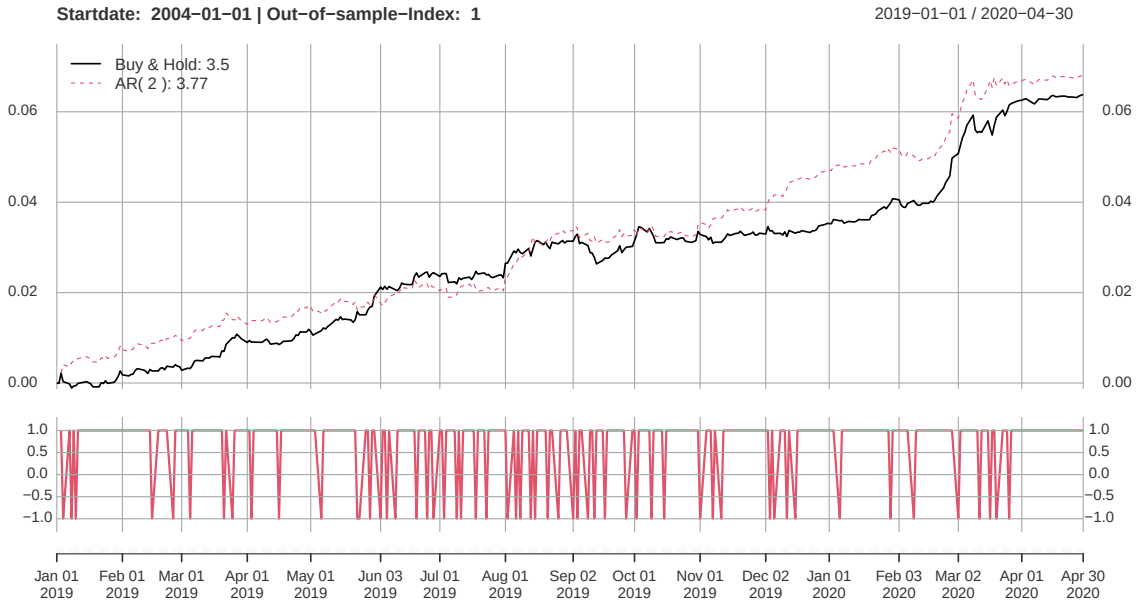

Figure 16: Visualization of the optimal AR(2)-Model.

The upper part of figure 16 shows the optimal model compared to buy-and-hold. The AR(2) has a higher sharpe than buy-and-hold. Regarding performance, it should be mentioned that the sharpe optimization of the algorithm is based on daily trades. The signals for a trade are newly generated with each daily forecast and could lead to high trading costs (positive signals = long, negative signals = short).

The lower part of figure 16 shows the trading signals visually. If one were to trade exactly according to the AR(2) forecasts, one would make a trade at each vertical line, a total of 111 trades would have to be made during the out-of-sample period. If you want to include trading costs in the performance, an AR model with daily trades would have to be much better than this one to outperform buy-and-hold.

But what influence do the trading costs really have? If you assume a fictitious investment amount of 1 million USD and you have to pay 190 USD for each trade, as mentioned in section 3.2.1. Invest this amount at the beginning of the out-of-sample period. With the buy-and-hold strategy, a return of USD 63832 is achieved by the end of the out-of-sample range. Without trading costs the AR(2) would yield USD 68130, with trading costs only USD 47040. To get at least the same return on the investment, the trading costs should not exceed USD 39. Note that all the parameters have to be taken in consideration for real life trading such as: trading costs, realizable trading volume, restrictions at the exchange, governmental restrictions, so on and so forth.

Thus it can be said that for index 1, the buy-and-hold strategy cannot be outperformed by an AR(2) model if trading costs are taken into account. Nevertheless, one would like to apply this procedure also to the other indexes, possibly the generated models perform better for the other time-series.

Table 7: Optimal AR-Models.

	StartDate	AR-Sharpe	p	B&H-Sharpe	Trades
Index 1	2018-01-01	3.765	2	3.505	111
Index 2	2004-01-01	3.100	1	2.476	19
Index 3	2005-01-01	2.528	1	2.155	21
Index 4	2003-01-01	2.218	1	1.945	9

Table 7 shows the solutions of all 4 indexes. The algorithm finds an optimal AR model with model order 1 for all 3 remaining time-series. In contrast to index 1, it is noticeable that a much longer in-sample range is required for the optimal model. This also makes sense because the indexes show an increasing trend during the complete in-sample range. Index 1 is less volatile and has a less strong trend compared to the others, so a strong increase in the trend (spring 2018) has a greater effect on model performance. The other 3 series are more volatile, have larger peaks and therefore the model performance is less influenced by single events (like the increasing trend in spring 2018). The number of trades is much smaller for all models than for the first one. The out-of-sample performances are better than the corresponding buy-and-hold performances for all AR models.

Because of the trading costs, it has been seen with index 1 that the AR model, despite the better performance, generates a lower return than buy-and-hold. But what about the other indexes. You choose the same investment amount and trading costs as before and you get the following table 8.

Table 8: Out-of-Sample Returns.

	Buy & Hold	AR without tradingcost	AR with tradingcost
Index 1	63832	68130	47040
Index 2	106055	131723	128113
Index 3	143291	167284	163294
Index 4	184657	209858	208148

Despite the trading costs, daily trading with the indexes 2-4 generates a higher return than the passive buy-and-hold strategy.

Figure 17 shows the out-of-sample performance of the second index. Right at the beginning of the series, the AR model signals a trade (one goes short and immediately long again). This trade immediately leads to an increase in performance which already outperforms the passive strategy. Further signaled trades will only occur again later. During the Covid19 crisis (spring 2020), one can see several close consecutive trades. The volatility increases sharply during this period, the AR model catches this event well and remains positive.

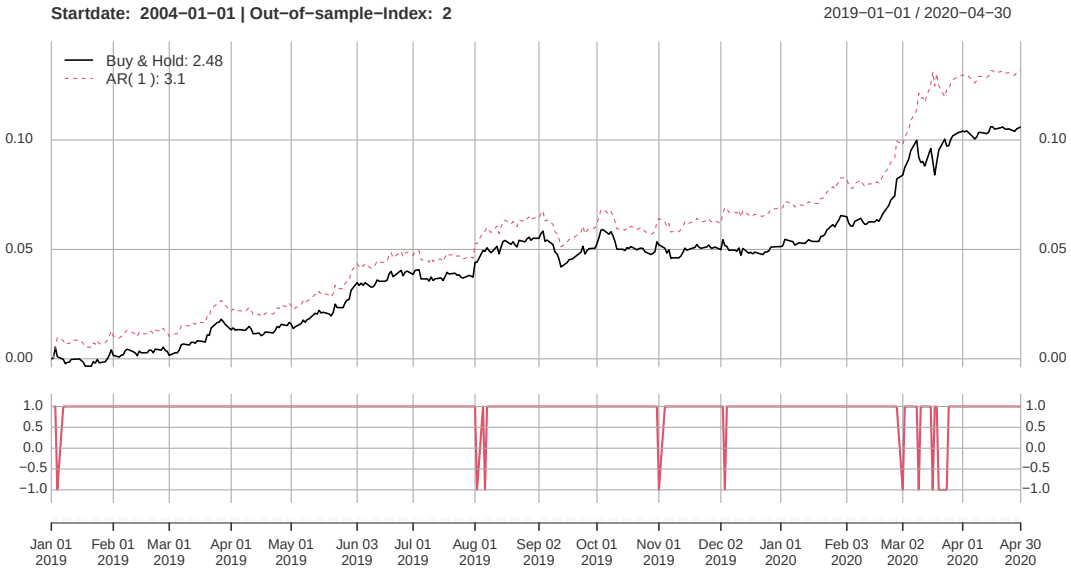

Figure 17: Visualization of the optimal Index 2 AR(1)-Model.

The optimal AR model for the third index shows almost the same behavior as index 2 (as seen in figure 18), the only difference being an additional trade in spring 2019.

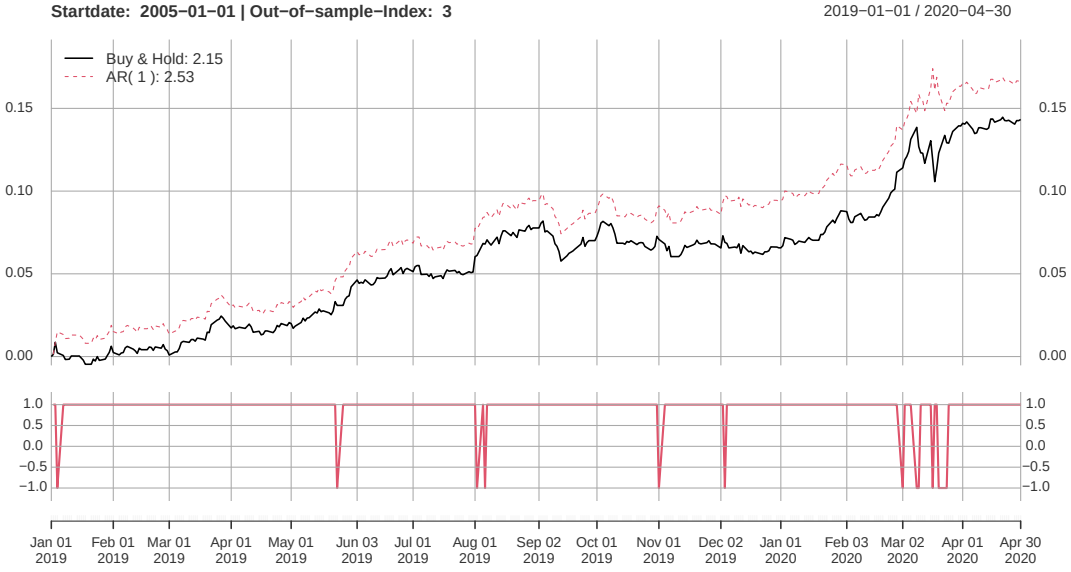

Figure 18: Visualization of the optimal Index 3 AR(1)-Model..

The fourth index, which is shown in the lower figure 18, is interesting again. The optimal AR model predicts almost the same behavior as buy-and-hold for almost the entire duration. Only during the highly volatile phase of the Covid19 crisis are the same trades listed as for index 2 and 3.

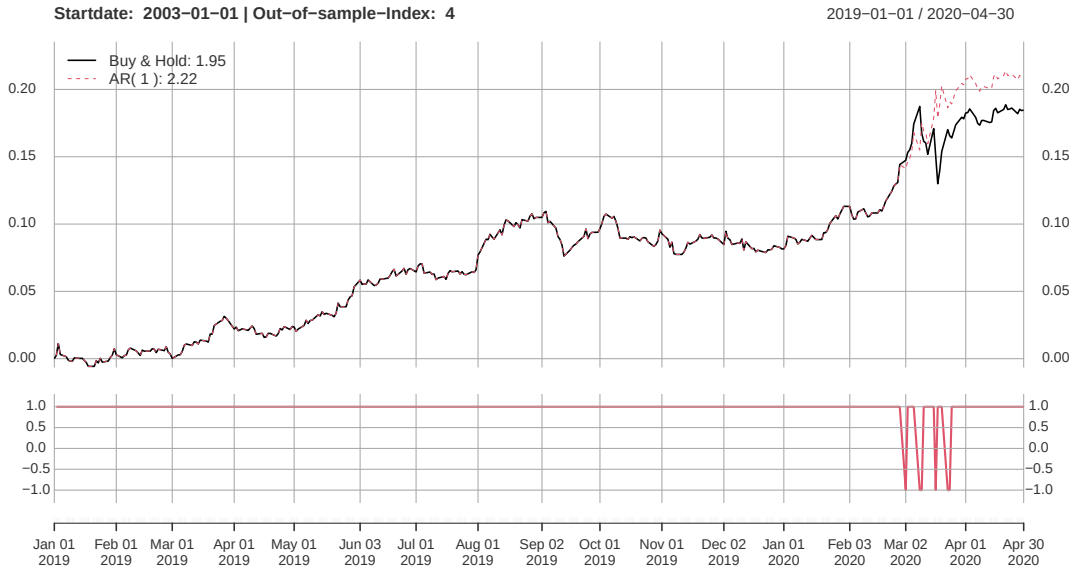

Figure 19: Visualization of the optimal Index 4 AR(1)-Model.

So it can be said that with very simple autoregressive models, a tool has been found with which it is possible to outperform the popular buy-and-hold strategy. Consider that this is only the case in this example with defined conditions. Depending on how high the trading costs and the amount of investment are, the returns can vary greatly. In addition to the buy-and-hold strategy, where you only buy the financial instrument and then hold it, you have to trade effectively with an active trading strategy.

3.4. Single Simple Moving Average Trading

Next thing to look at is the optimization with a single simple moving average. For AR models, the start date is a hyperparameter, as is the model order p. For SMA models there is no model order but a filter length L. Limit the filter length here to $5 < L < 500$, since the out-of-sample range is only 350 days long.

Additionally the in-sample range is varied again. For each start date the in-sample performances are calculated for each filter length in the interval $5 < L < 500$. The AR model was only limited to the performance indicator sharpe ratio. This time we are adding the drawdown and MSE indicators. Thus, for each start date, for each filter length L, the three different indicators are calculated and an optimal model can be choosen.

3.4.1. Optimization

The following figure 20 visually represents the complete SMA optimization of the first index. The two performance indicators sharpe (red) and MSE (green) correspond to the left y-axis. The drawdown (blue) was scaled to a visually appealing size for comparison and corresponds to the right y-axis. The x-axis shows the respective in-sample start dates, which calculates the respective performance indicators Sharpe, drawdown and MSE from filter lengths 5 to 500. The bold dots in the respective sections mark the optimal values of the respective indicator.

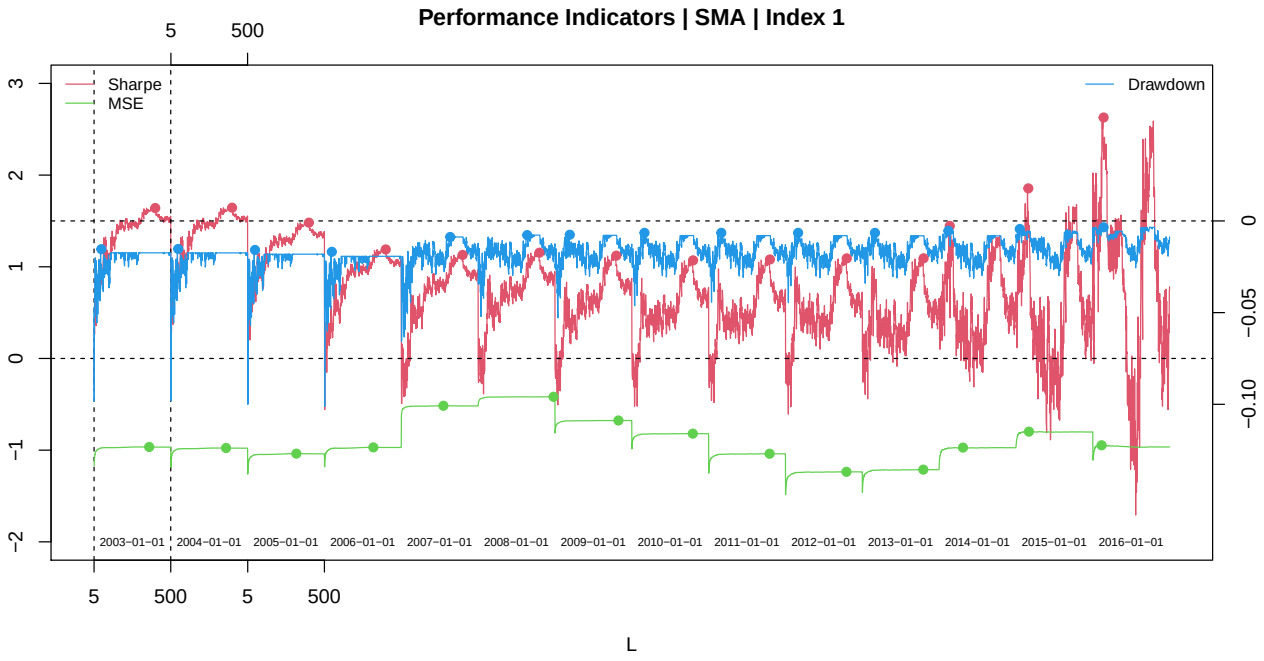

Figure 20: Optimization Index 1.

It is immediately noticeable that in each time window the trajectories are very repetitive. If we consider a single time window (2003), we can see that the Sharpe gets better for longer filters. Towards the end of the Sharpe optimization, the amplitude increases and diverges towards the end. With the drawdown this behaves exactly the opposite. It is noticeable with the drawdown, however, that with short filters the values fluctuate very strongly, which is not the case with long filters. What is also noticeable is that especially towards the middle (from 2007) the drawdown seems to level off, but in this time window several points could point to the optimal value. This randomness is discussed in more detail in section 3.5. Already after a few iteration steps, the MSE seems to stabilize. It is difficult to find an optimum here, because practically all values are very similar.

Table 9 shows the numerically best values for the respective start date and performance indicators of the optimization. The marked values should indicate the best filter lengths in each case. What is also striking here is that the Sharpe filter lengths from 2003 to 2013 are all identical. To describe this phenomenon further, a deeper analysis would be necessary, but this is not described further here.

The optimal filter lengths for the corresponding indicators are now used to trade out-of-sample by sign in figure 21.

Table 9: SMA-Optima by performance indicators | Index 1

StartDate	Sharpe	L_Sharpe	Drawdown	L_Drawdown	MSE	L_MSE
2003-01-01	1.64	395	-0.016885	47	-0.965	356
2004-01-01	1.64	395	-0.016766	47	-0.977	356
2005-01-01	1.48	395	-0.017368	47	-1.038	314
2006-01-01	1.19	395	-0.018482	47	-0.970	315
2007-01-01	1.13	395	-0.009597	315	-0.516	271
2008-01-01	1.15	395	-0.008487	316	-0.418	486
2009-01-01	1.12	395	-0.008263	95	-0.676	409
2010-01-01	1.07	395	-0.007125	81	-0.820	393
2011-01-01	1.08	395	-0.007129	81	-1.039	393
2012-01-01	1.09	395	-0.007124	81	-1.236	393
2013-01-01	1.09	395	-0.007143	81	-1.212	393
2014-01-01	1.45	68	-0.005884	59	-0.972	153
2015-01-01	1.86	79	-0.004819	24	-0.798	83
2016-01-01	2.63	69	-0.003817	69	-0.947	56

Note that a little trickery was used for the out-of-sample calculation. If SMA is used, the initialization phase of the filters must be taken into account. For example, for an SMA with a filter length of L=50, an initialization phase of 50 time units would have to be considered. Here the in-sample ranges were also used in order to be able to represent the comparability of the filters better. In practice, reliable trading signals could only be expected after the initialization phase.

Applying the filter lengths of the best indicators to index 1, the following out-of-sample solutions are obtained 21. For the MSE (green), a very long filter L=486 was chosen as the optimal filter. This reacts very late to fluctuations in the time-series. The index 1 shows a constantly rising trend, which leads to the fact that the MSE filter does not signal any trades.

Sharpe and drawdown lead to the exact same filter length of L=69 with the same in-sample range with start date 2016-01-01. At the beginning of December 2019 a trade is signaled, which leads to a decline in performance.

With none of the found optimal filter lengths and in-sample ranges the passive buy-and-hold strategy could be outperformed.

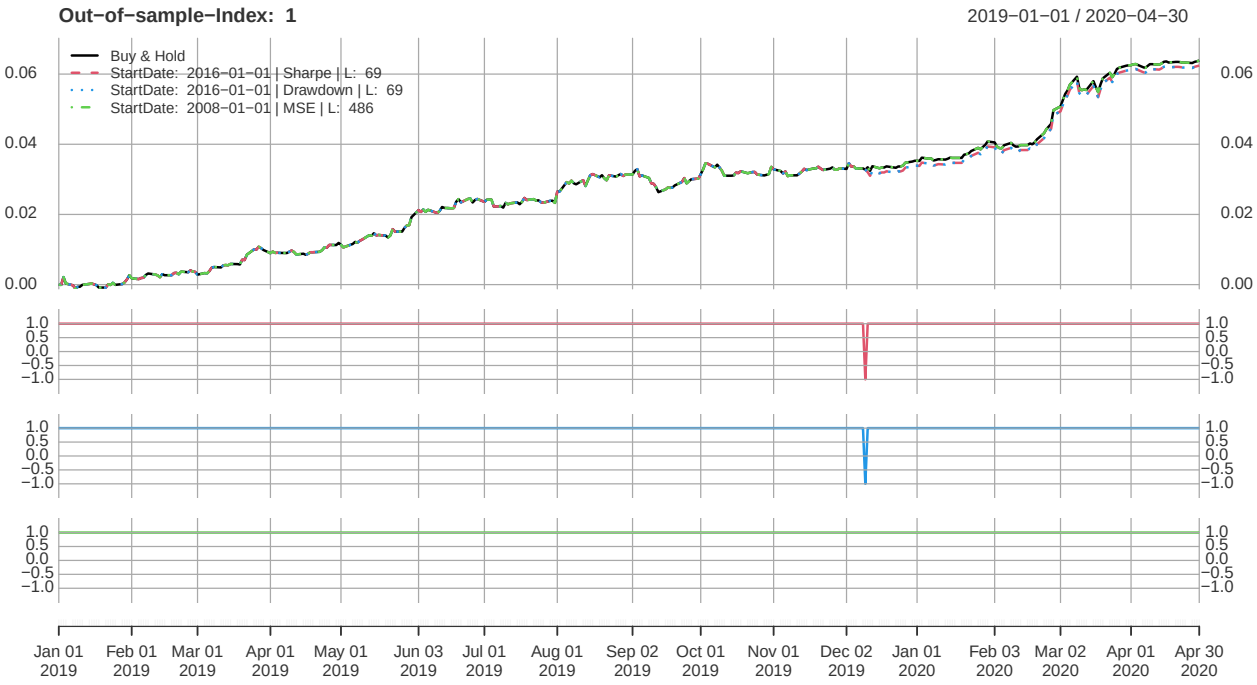

Figure 21: Applying optimal filterlengths to the Index 1.

Applying the optimization algorithm to the other 3 time-series leads to similar results, which can be seen in the appendix 6.3.4.2. It seems almost impossible to outperform the highly praised buy-and-hold strategy with an SMA.

3.4.2. Out-of-Sample Optimization

Different in-sample lengths have been used to try to find an optimal in-sample model, which should lead to satisfying solutions in the out-of-sample area. This was not possible with the methods used. Now one would like to examine additionally whether it would be at all possible to find good solutions in the out-of-sample range. Therefore we investigate the out-of-sample area directly. For the entire runtime of the time-series, filtered SMAs are generated for all filter lengths $5 < L < 500$. However, one considers only the out-of-sample range for the analysis. One limits oneself here to the performance indicator Sharpe and calculates this for each L.

Note, however, that this is only possible here because you know the real values of the out-of-sample range. If this procedure finds filter lengths that outperform buy-and-hold, these cannot be applied in reality. The filters could only be found by the out-of-sample data, which we do not know in reality. Here, we only investigate whether it is at all possible to outperform buy-and-hold.

In figure 22, one can see the results of the out-of-sample Sharpe ratios of the different filter lengths of the 4 indexes. The dashed horizontal lines reflects the buy-and-hold Sharpe of the indexes. The dots indicate Sharpe ratios that have a higher value than buy-and-hold, i.e. filter lengths that lead to an outperformance.

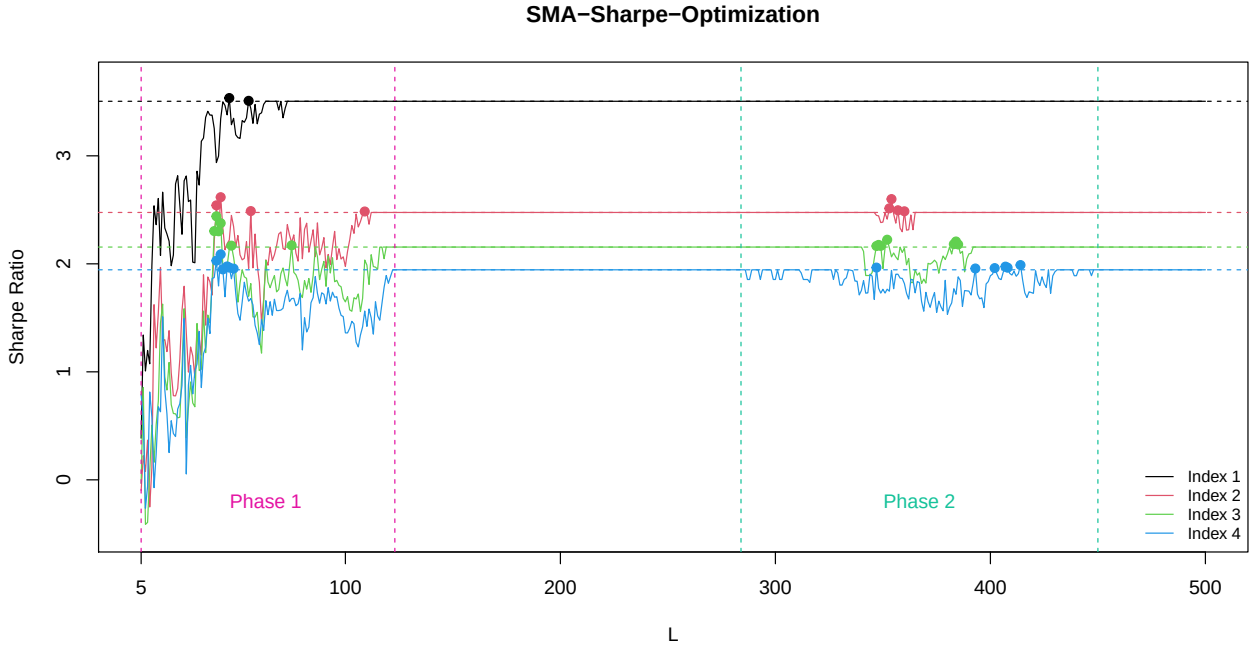

Figure 22: Out-of-sample Sharpe for each Index.

Here one can see again a typical characteristic of these time-series. In chapter 3.1.2. we compared the volatility with the Sharpe ratios, here we can see this phenomenon well. The index 1 leads to higher Sharpe ratios than for example the index 4, but is less volatile. The Sharpe ratio of index 4 shows very strong volatile phases in comparison.

Up to a filter length of about 40, the Sharpe ratios fluctuate very strongly. This is followed by a volatile phase until the filter Sharpe converges to the buy-and-hold Sharpe (from about L=120). At high filter lengths the Sharpe ratios for indexes 2,3 and 4 start to fluctuate again, the more volatile the original index is, the earlier the volatile phase starts and the longer it is. What is noticeable is that there are 2 phases which leads to good models. On the one hand you can find SMA's with short filters (phase 1) and on the other hand with long ones (phase 2). In both phases models are found, which can lead to an outperformance (they are above the dashed buy-and-hold line).

In the following table 10, the marked Sharpe ratios are shown with their corresponding filter lengths. Additionally, the trades signaled by the filtered series are listed. The first row corresponds to the buy-and-hold. The respective phases have been visualized in color.

Table 10: Best ouf-of-sample filterlengths.

Index 1			Index 2			Index 3			Index 4		
L	Trades	Sharpe	L	Trades	Sharpe	L	Trades	Sharpe	L	Trades	Sharpe
1	1	3.505	1	1	2.476	1	1	2.155	1	1	1.945
46	7	3.535	40	11	2.542	39	13	2.303	40	17	2.029
55	5	3.509	42	13	2.617	40	13	2.440	42	11	2.088
			56	9	2.488	41	11	2.301	43	11	1.949
			109	3	2.485	42	11	2.377	45	11	1.976
			353	3	2.514	47	17	2.168	46	15	1.965
			354	3	2.6	75	13	2.171	48	11	1.956
			357	3	2.496	347	4	2.165	347	4	1.965
			360	5	2.486	348	5	2.175	393	7	1.957
						349	6	2.166	402	3	1.961
						352	3	2.223	407	9	1.974
						383	3	2.182	408	5	1.964
						384	3	2.207	414	3	1.989
						385	3	2.180			

The short filters (pink) lead to more trades compared to the long filters (green). This also makes sense, short filters react faster and tend to lead to a higher number of trading signals. Some models found are also only marginally better than buy-and-hold, some would lose relevance when trading costs are taken into account.

Lastly, one would just like to investigate how the generated signals of the short and long filters differ. In the following figure 23 two models with different filter lengths of index 2 are shown as an example.

The two chosen filters are the most optimal for index 2. Both lead to a similar result. Over the entire out-of-sample period, the long filter (green) generates a single signal in mid-January 2019, which leads to an outperformance. The short filter (pink) generates a signal in March 2019 that leads to nothing. Only towards the end of 2019 to the beginning of 2020 several consecutive trading signals are generated, which finally lead to a similar result as the long filter.

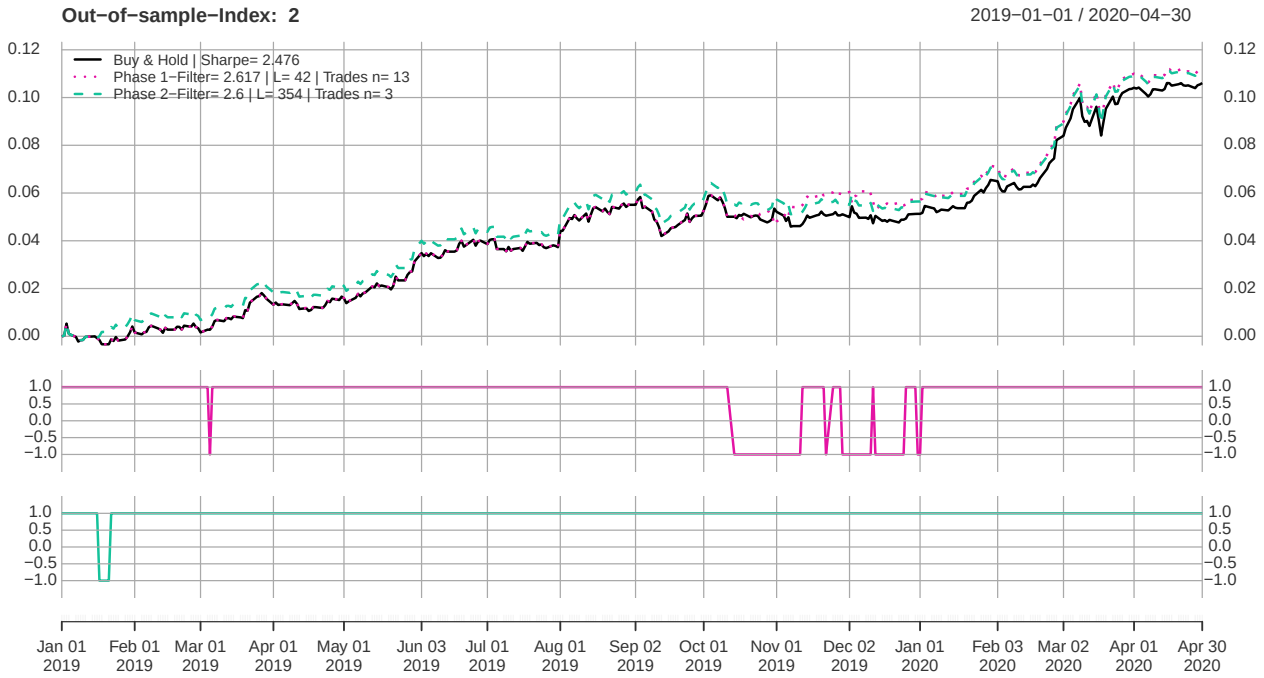

Figure 23: Two good models | Index 2

Thus, it would be proven that buy-and-hold can be outperformed with a single simple moving average. However, this could only be done by meeting some conditions. The optimal filters could only be found by knowing the out-of-sample data, which is not known in reality. If the used time-series would take a different course, the solutions with the used filters could look completely different.

3.5. Moving Average Crossings Trading

As a third approach we apply 2 SMAs on every time-series and trade them with the common Ma crossings rules introduced in 2.3.3. With an optimization we are trying to find the best filterlengths for the 2 SMAs.

3.5.1. Optimization

For the optimization the same procedure as in the previous sections gets executed.

The optimization has two levels, the "first level" or outer level is narrowing the in-sample timespan from (2003-01 / 01-2019) to (01-2018 / 01-2019). In the inner level "second level" the real optimization is calculated by finding the optimal combination of filters, regarding maximum Sharpe and maximum drawdown trading them with the common crossing trading rules.

Hyperparameter:

- Date = startdate for the in-sample timespan 2003-01-01 $<$ date $\leq$ 2018-01-01, by 1 trading year

Parameters$_2$:

- $L1$ = length of shorter SMA with $1 < L1 \leq 50$
- $L2$ = length of longer SMA with $100 < L1 \leq 250$

Figure 24 visualizes one iteration of the optimization. For the fixed L1=1, L2 is variable and the optimal sharpe is calculated. This procedure is done for every filterlength as described in $_2$ and further visualized in 25. The drawdown is also calculated as seen in plot 26. Notice that the visualization is from the starting date 2015-10-30, which is randomly chosen for explaining the optimization process.

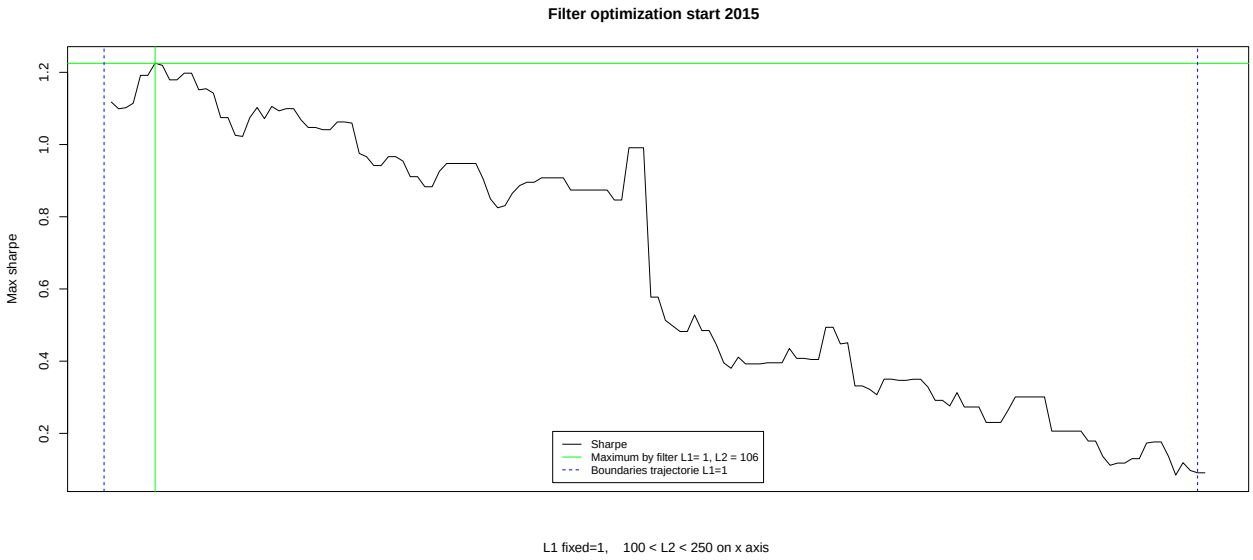

Figure 24: Optimization Sharpe L1=1 ,L2=variable

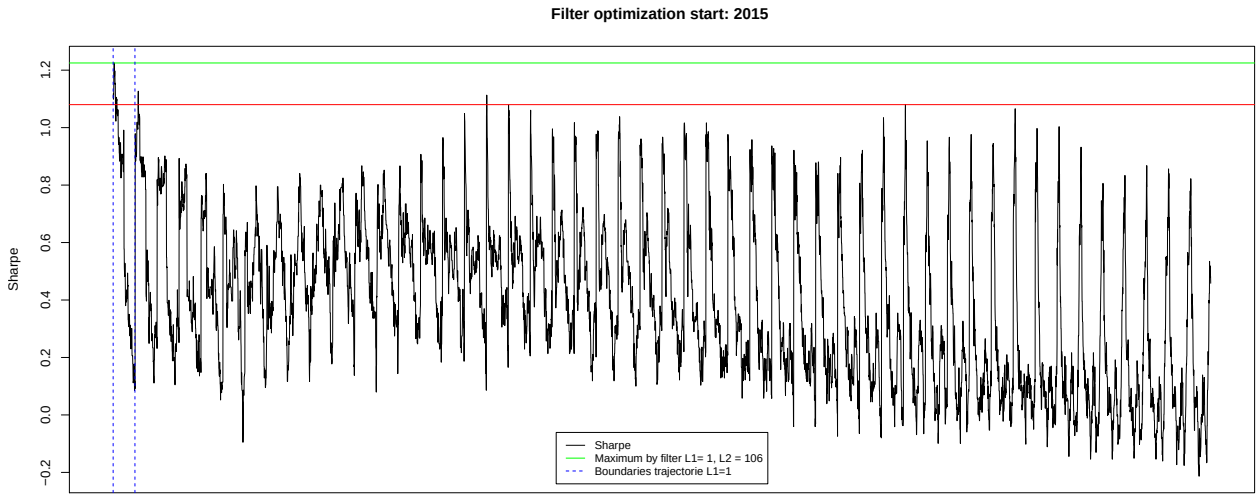

Figure 25: Sharpe index 1

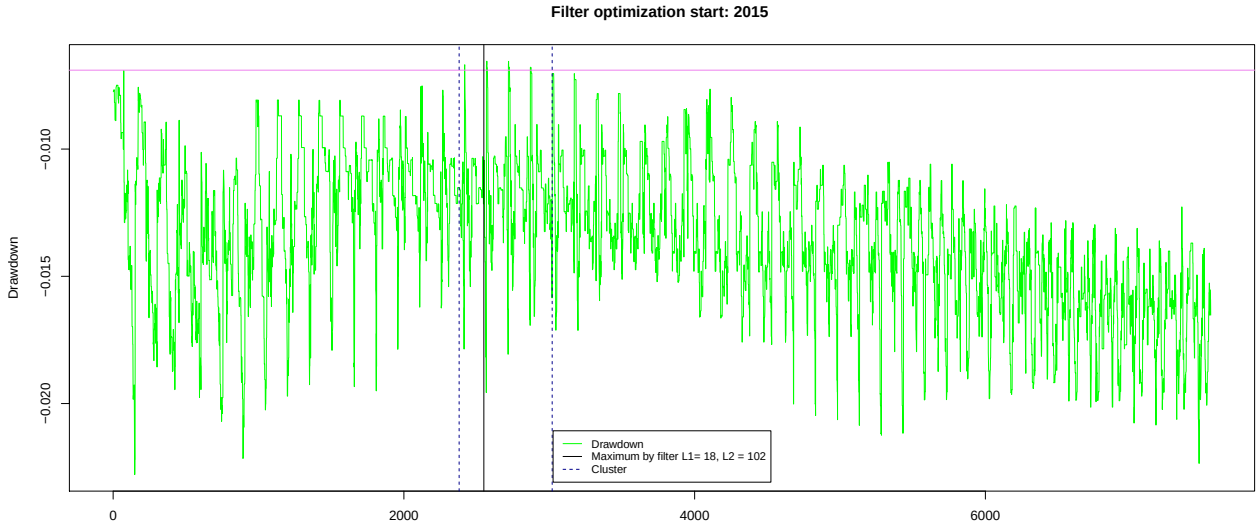

Figure 26: Drawdown index 1

The pattern we observe in 25 is the repeating optimization by fixing L1 and vary L2. The maximum Sharpe was found with L1=*1*, L2=*106*, Sharpe=*1.225*, which is also visualized in 25, where the "window" (blue dotted lines) is shown. When we look at the red line we can clearly see there are other similar high spikes in other filter lengths. The green pattern in 26 shows interesting news. At L1 = 17, 18, 19 and 20 we see a cluster(blue dotted lines) of high spikes, all very similar. The optimum in this plot is by L1=*18* and L2=*102* and a drawdown *6.55e-3*, laying in the described cluster. But when we look at the violet line drawn, we can clearly see there is another high spike at the beginning of the plot. This spike is identical with the maximum Sharpe in 25. So these results could just be random and the solution is not finite, this problem will be further discussed$_3$.

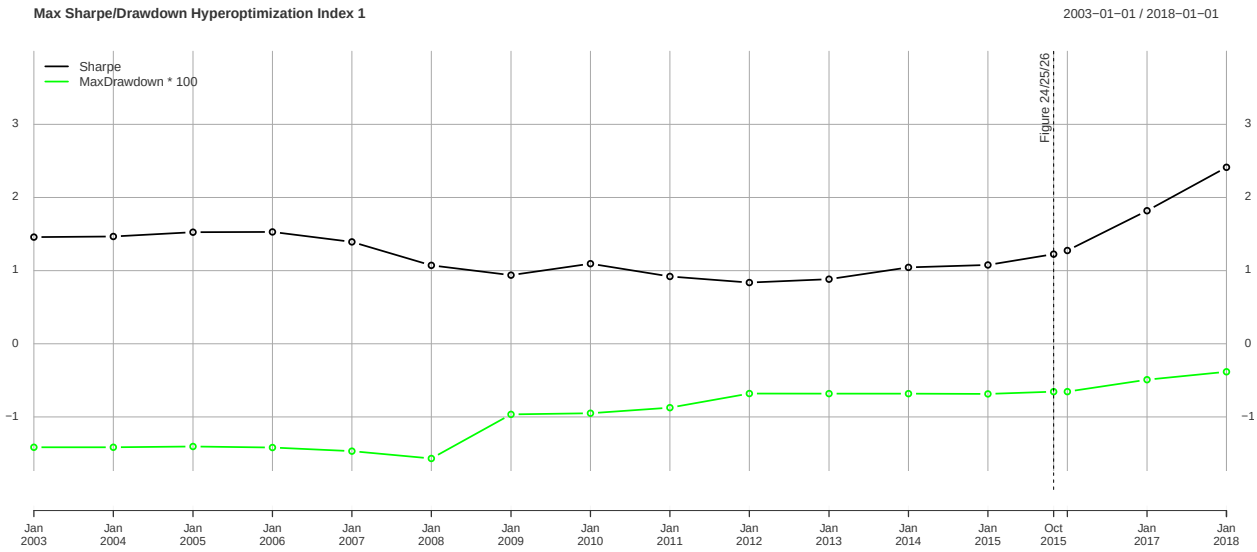

Figure 27: Hyperoptimization Index 1

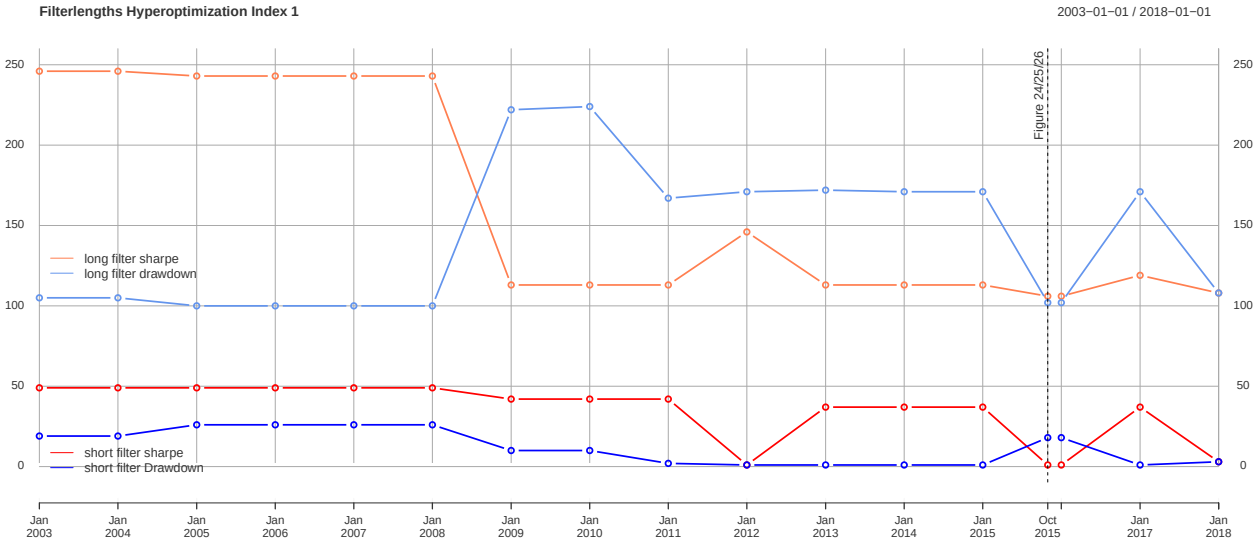

Figure 28: Hyperoptimization Index 1

By plotting the Sharpe ratio / max drawdown$_1$ in 27 and the optimal filterlenghts L1/L2 in 28, for every in-sample timestep as described at the beginning of 3.5.1, the hyperoptimization is visualized. The code is linked in the attachement 6.3.5. The results from the trajectories in 25 and 26 are showed at the black dotted line. Because of the problem we have mentioned in $_3$ the best result could also be just random. Therefore the ten best results for each in-sample timesteps are calculated and visualized in 29 and 30 to get an idea about the accuracy of these filters and analyze them a bit further.

$_1$ Note: maxdrawdown is scaled by the factor 100 to visualize it in the same plot.

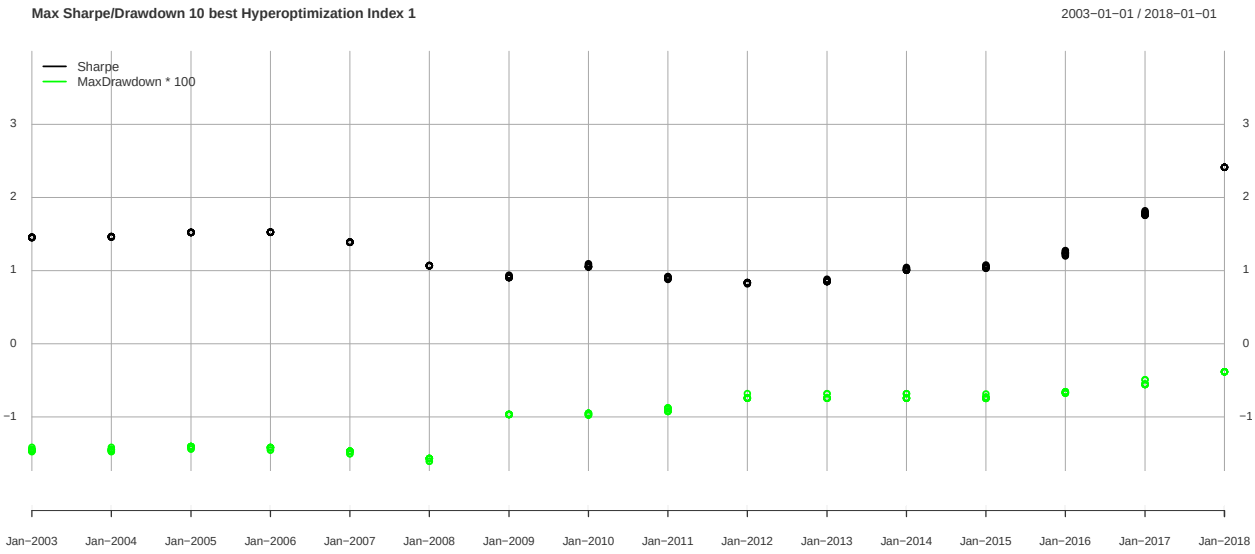

Figure 29: Hyperoptimization 10 best Index 1

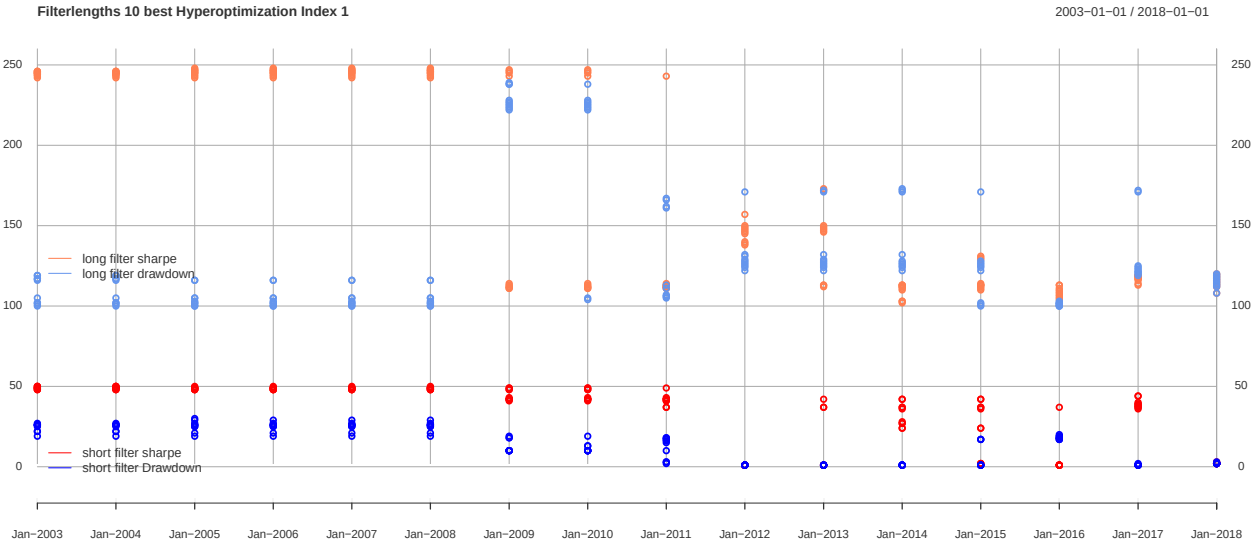

Figure 30: Hyperoptimization 10 best Index 1

Whilst the ten best Sharpe ratios / drawdowns in 29 are always very close together, the filterplot 29 shows a completely different and interesting image. We can observe differences, specially in the long filters in 2009, 2010, 2011 the result nearly diverge 150 units. Also in the timespan 2011 to 2018 the filters fluctuate strongly. What further catches the eye, are the long- and the shortfilter sharpes from 2003 to 2011. They are all concentrated at their upper limits of 50 and 250, which indicates that they actually have to be longer than the restriction. The same behaviour is observed for the short drawdown from 2011 to 2018 where the observations are concentrated around length 1. The plot would surely look different if the restrictions of the filterlengths would be greater. Also the steps of the in-sample timespan are always one year. Changing the restriction would have a massive impact on the optimization time and would most likely deliver different results.

Curiosity leads us to recreate the optimization for $1 < L1 \leq 1000$ and $1 < L2 \leq 1000$ which calculates 16 million Sharpes. Now L2 can be smaller than L1 while the trading rule stays the same. The numeric results can be found in 18.

In figure 31 the Sharpes of the 10 best models look all very similar in each step except for 2009 and 2012

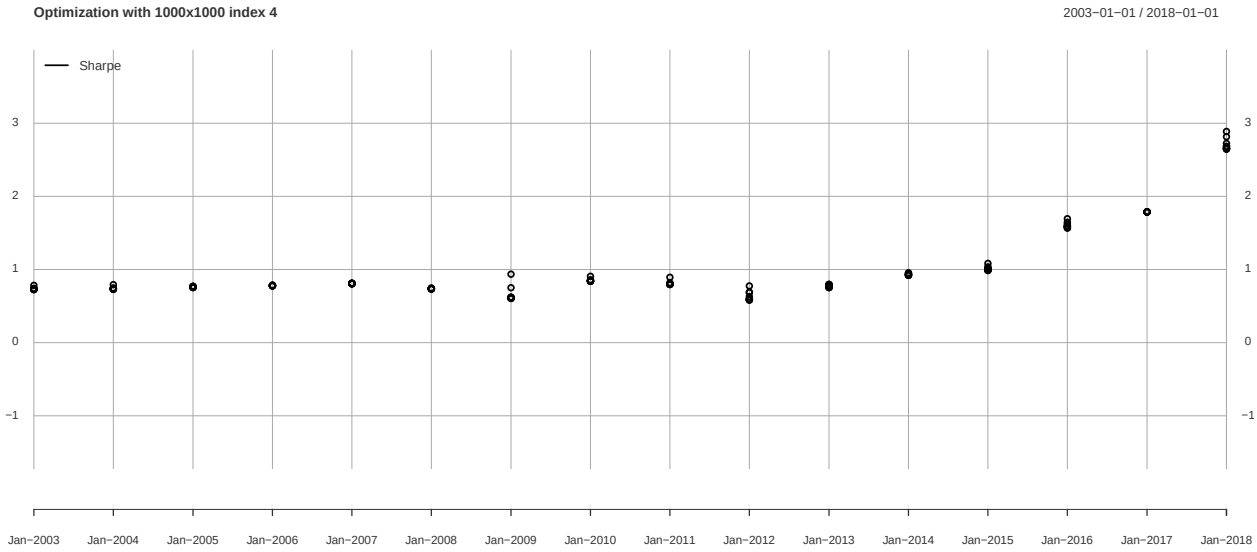

Figure 31: Hyperoptimization 1000 x 1000 Index 4

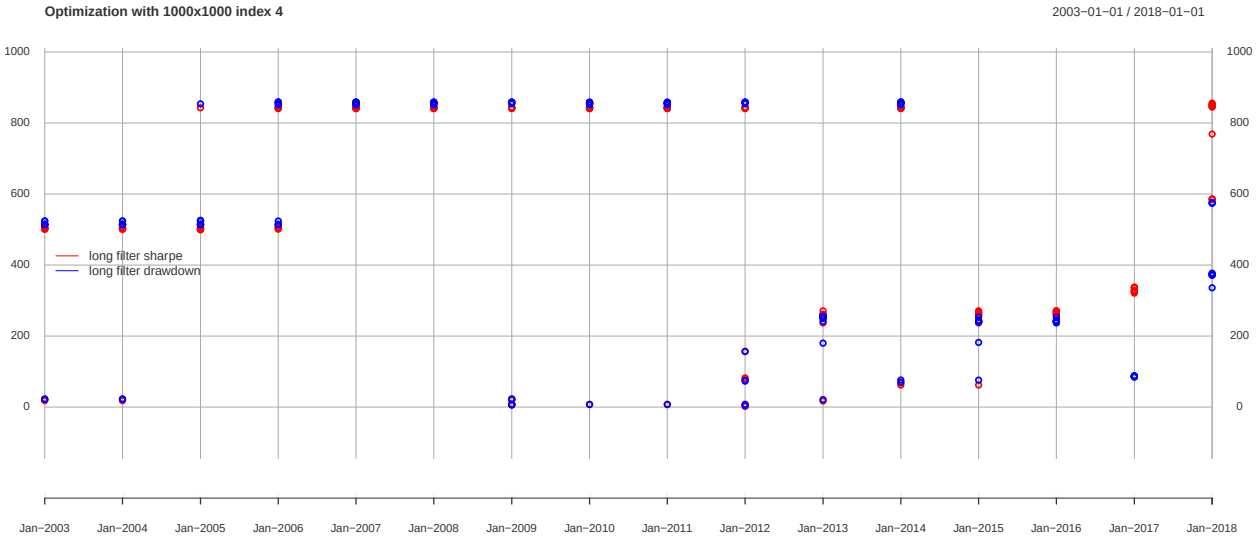

Figure 32: Hyperoptimization 10 best Index 4

they vary a little bit. Until 2015 they are all slightly under 1. Very important is the fact that a filterlength of 100 means 4 years which is a very long time. The results narrowing Jan-2019 are close to or directly buy-and-hold, therefore the sharpe rises to nearly 3. Interesting are also the result in figure 32. From 2002 to 2011 the long and the short filter are always very close together. This plot could lead to a more interesting analysis. Due the time and length limitation this optimization is closed here and not further discussed.

In good knowledge we are taking the risk of maybe not finding the right filterlength and use the result of the optimizations for trading the series in the next section.

3.5.2 Trading

As noticed the Visualizations in 3.5.1 are only done for index 1, the optimal values and their fluctuation are shown in the tables 14,15,16,17 and 41,42,43,44,45,46.

Based on the optimal filterlenghts calculated in the last section, the SMAs are now applied to the "out-of-sample" span from 2019-01-01 to 2020-04-01. Keeping in mind that the Moving Averages need the same initialization time as the length of filter itself, only the optimal filterlengths from years smaller than 2017 are considered. Index 1 brings us the optimal maximum Sharpe *1.275* at end of the series *2016* with filterlengths *L1 = 1 / L2=106* applying this filters to the put opf sample leads us to buy-and-hold no trades are executed. Even considering the filters from the earlier timesspans *2006-01-01 sharpe = 1.529 L1= 49 L2= 243* brings no other result than buy-and-hold. Trying the maximum drawdown filters leads to the same result.

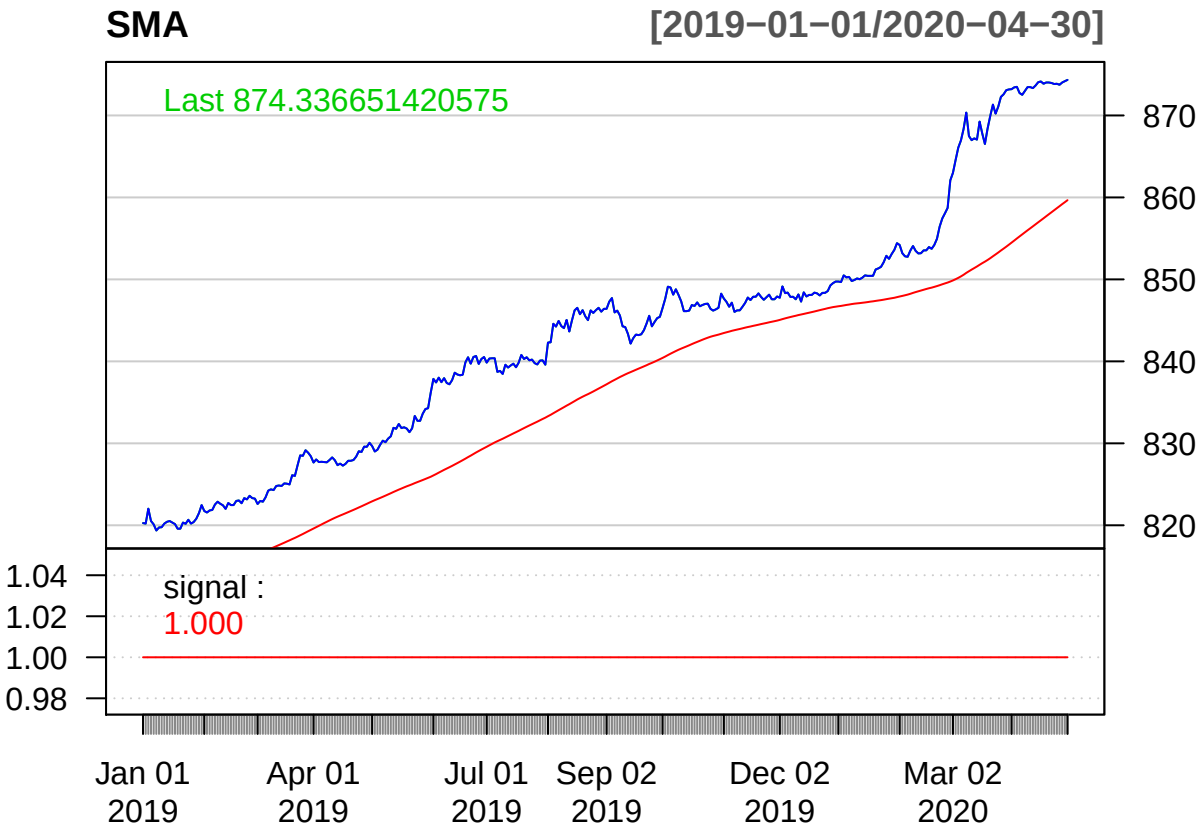

Figure 33: Macrossing index 1 L1 = 1 L2 = 106

Regarding the other indexes, the results look all very similiar. None of the Moving Average Crossing are able to outperform buy-and-hold. For visualizing we use only the fourth index with *2016-01-01* Sharpe=0.998, L1=12, L2=134. For this index we have leastways two trades 34. Following the longer filter (red) from the left side, it crosses the blue line (L1) in the beginning of 2020, that is where a selling is executed. Shortly after the buy signals comes with the red line crossing the blue from above. With this signals we get a Sharpe of *1.761906*, in comparison to the buy-and-hold Sharpe index 4 *1.963966*.

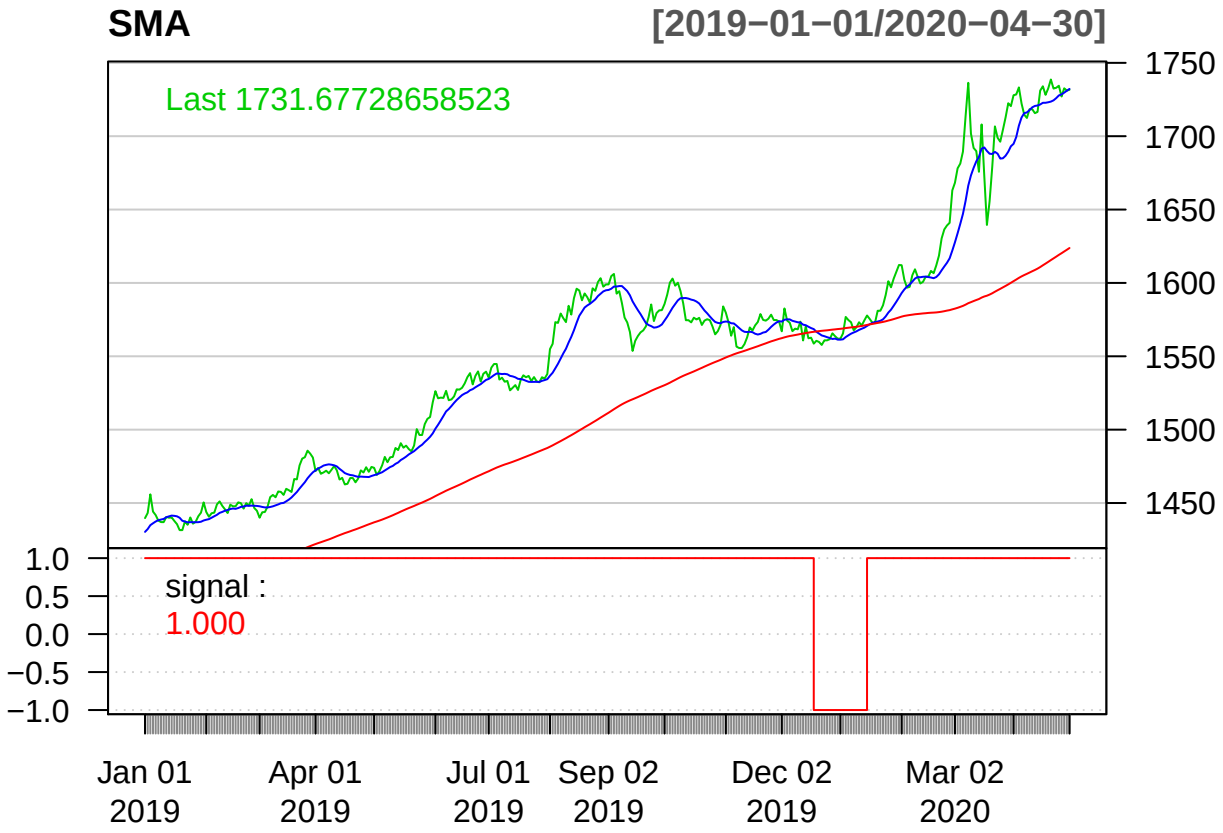

Figure 34: Macrossing index 4 L1 = 12 L2 = 134

Also neither of the results from 32 are clearly satisfying.

As endresult buy-and-hold stays better as the combinations of MA Crossigs, at least with the parameter restrictions we implied. The intuition, and also the common used filters L1=50 / L2=250 equals buy-and-hold. This strategy may not be so bad, why leaving a trend market instead of profiting from it.

4. Conclusion

Only the simple AR models, of the methods we use, lead to the outperforming of the buy-and-hold strategy under parameter restrictions. In contrast to the Moving Average Filter, the AR methods can handle short-term, strong volatility fluctuations (e.g. Covid 19) well, which is why the filter methods remain long in strongly growing phases. The optimizations of the moving average filters did not lead to better results despite the detailed and very exciting applications. All methods used are very interesting in themselves and would be worthwhile to pursue further. For example, the out-of-sample time could be adjusted somewhat to better test the models mentioned. It would also be interesting to look at the period after April 2020 until today to see if the Ar models would have maintained the performance.

In the theory section, we have presented many, but by no means all, models. Since the choice of models is huge and the application of them almost infinite, it would be an extremely exciting work to outperform buy-and-hold with further methods. Furthermore, a portfolio could have been created from the different approaches and weightings, which did not make it into the thesis, as it would go beyond the scope of this work.

This work serves as the basis for a more advanced bachelor thesis. While the focus here was on the analysis of the data and the application of rather simpler models, more complex approaches from the field of machine learning or artificial intelligence can be applied in a further step.

5. References

[1] M. Wildi, *Econometrics 1: Time series analysis*. Winterthur: ZHAW, 2017, p. 221.

[2] M. Wildi, *Econometrics 3: Conditional heteroscedasticity models, part 1*. ZHAW, 2020, p. 56.

[3] A. H. A. L. Kholkin D. A. Kiselev, *Encyclopedia of materials: Science and technology (second edition)*. Elsevier, 2001, p. 10388.

[4] M. Masum, "Time series analysis: Identifying ar and ma using acf and pacf plots." https://towardsdatascience.com/identifying-ar-and-ma-terms-using-acf-and-pacf-plots-in-time-series-forecasting-ccb9fd073db8 (accessed Nov. 16, 2020).

[5] B. Williams, *GARCH(1,1) models*. Ruprecht-Karls-Universität Heidelberg, 2011, p. 42.

[6] M. Wildi, *An introduction to conditional volatility models*. Winterthur: ZHAW, 2020, p. 32.

[7] M. Wildi, *Econometrics 3: Conditional heteroscedasticity models, part 3*. ZHAW, 2020, p. 53.

[8] A. Hayes, "Simple moving average." https://www.investopedia.com/terms/s/sma.asp (accessed Sep. 22, 2020).

[9] A. Hayes, "Exponential moving average." https://www.investopedia.com/terms/e/ema.asp (accessed Sep. 10, 2020).

[10] M. Wildi, *Econometrics 2: Trading indicators*. ZHAW, 2020, p. 78.

[11] J. Fernando, "Relative strength index (rsi)." https://www.investopedia.com/terms/r/rsi.asp (accessed Nov. 30, 2020).

[12] J. Gordon, "Relative strength index (rsi)." https://www.investopedia.com/terms/m/maximum-drawdown-mdd.asp#:~:text=What%20Is%20a%20Maximum%20Drawdown,over%20a%20specified%20time%20period. (accessed Mar. 24, 2020).

[13] swissquote, "Trading fees." https://library.swissquote.com/shared-images/brochure-trading-pricing-bank-de?_ga=2.183413157.222455671.1606673517-1327743316.1606673517 (accessed Nov. 29, 2020).

6. Attachment

This project work is created with R-4.0.2 , RStudio Version 1.4.904 and RMarkdown in collaborative working via Git / Github

```
# libraries
library(xts)
library(PerformanceAnalytics)
library(kableExtra)
library(dplyr)
library(quantmod)
library(forecast)
library(fGarch)
```

6.3.3. AR

6.3.3.1. Optimization code

```
# AR-Optim####
performante_ar <- function(xin, xout) {
    # AR-Model Order p -> 1:8 Find best in-sample ar_obj Compute
    # out-of-of sample daily trading returns Find best Sharpe
    best <- 0
    for (j in 1:8) {
        arma_obj <- arima(xin, order = c(j, 0, 0))
        mat_out <- cbind(xout)
        for (k in 1:(j)) {
            mat_out <- cbind(mat_out, lag(xout, k = k))
        }
        ar_pred <- arma_obj$coef[length(arma_obj$coef)] + as.matrix(mat_out[,
            2:ncol(mat_out)]) %*% arma_obj$coef[1:(length(arma_obj$coef) -
            1)]

        ret_arima <- na.exclude(sign(ar_pred) * xout)

        sharpe_ar <- as.double(sqrt(250) * mean(ret_arima, na.rm = T)/sqrt(var(ret_arima,
            na.rm = T)))

        if (sharpe_ar > best) {
            best <- as.data.frame(sharpe_ar)
            rownames(best) <- paste(j)
        }
    }
    return(list(sharpe_ar = best, ar_p = rownames(best)))
}
perfplot_ar <- function(optim_ar_obj, lr_series, inx, insamp = "2019-01-01") {
    # Max Optim-Element
    max_obj <- optim_ar_obj[which.max(optim_ar_obj[, 2]), ]


    # Startdate
    start_date <- optim_ar_obj[which.max(optim_ar_obj[, 2]),
        ][1]

    xall <- lr_series[paste(start_date, "/", sep = "")][, inx]
```

```
    xin <- xall[paste("/", insamp, sep = "")]
    xout <- xall[paste(insamp, "/", sep = "")]


    # sharpe
    sharpe_bnh <- as.double(sqrt(250) * mean(xout)/sqrt(var(xout)))
    sharpe_ar <- as.numeric(optim_ar_obj[which.max(optim_ar_obj[,
        2]), ][2])

    # Model Order AR(p)
    p <- as.numeric(optim_ar_obj[which.max(optim_ar_obj[, 2]),
        ][3])

    # AR-Fit, in-sample
    arma_obj <- arima(xin, order = c(p, 0, 0))

    # AR-Pred, out-of-sample
    mat_out <- cbind(xout)
    for (k in 1:(p)) {
        mat_out <- cbind(mat_out, lag(xout, k = k))
    }
    ar_pred <- arma_obj$coef[length(arma_obj$coef)] + as.matrix(mat_out[,
        2:ncol(mat_out)]) %*% arma_obj$coef[1:(length(arma_obj$coef) -
        1)]

    # AR-Returns
    ret_arima <- sign(ar_pred) * xout

    # Plot
    perf_bnh <- cumsum(xout)
    perf_ar <- cumsum(na.exclude(ret_arima))
    ymin <- min(c(min(perf_bnh), min(perf_ar)))
    ymax <- max(c(max(perf_bnh), max(perf_ar)))

    plot(perf_bnh, main = paste("Startdate: ", start_date, "| Out-of-sample-Index: ",
        inx), lwd = 1.5, ylim = c(ymin - 0.1 * abs(ymin), ymax +
        0.1 * ymax))
    lines(perf_ar, lty = 2, lwd = 1, col = "red")
    print(addLegend("topleft", legend.names = c(paste("Buy & Hold:",
        round(sharpe_bnh, 2)), paste("AR(", p, "):", round(sharpe_ar,
        2))), lty = c(1, 2), lwd = c(1.5, 1), col = c("black",
        "red"), cex = 0.8))
}
optim_ar <- function(x.lr, inx, insamp = "2019-01-01", minyear = 3,
    maxyear = 18) {
    ind.lr <- x.lr
    year_list <- NULL
    for (yx in minyear:maxyear) {
        year_list <- c(year_list, paste(2000 + yx, "-01-01",
            sep = ""))
    }

    # Create result_matrix
```

```
    res_mat <- matrix(1:(3 * length(year_list)), ncol = 3, dimnames = list(1:length(year_list),
        c("StartDate", "AR-Sharpe", "p")))


    pb <- txtProgressBar(min = 1, max = length(year_list), style = 3)

    # Loop through years
    for (i in 1:length(year_list)) {
        startdate <- year_list[i]

        ind.lr.all <- ind.lr[paste(startdate, "/", sep = "")]
        ind.lr.in <- ind.lr.all[paste("/", insamp, sep = "")]
        ind.lr.out <- ind.lr.all[paste(insamp, "/", sep = "")]

        yolo1 <- performante_ar(xin = ind.lr.in[, inx], xout = ind.lr.out[,
            inx])

        res_mat[i, 1] <- year_list[i]
        res_mat[i, 2:3] <- c(round(as.numeric(yolo1$sharpe_ar),
            3), round(as.numeric(yolo1$ar_p)))

        setTxtProgressBar(pb, i)
    }
    res_df <- as.data.frame(res_mat)
    close(pb)

    # print(head(ind.lr.all)) print(tail(ind.lr.all))
    perfplot_ar(res_df, ind.lr, inx)

    return(res_df)
}
```

6.3.4. SMA

6.3.4.1. Optimization Code

```
loop_sma <- function(x = ind.lr, inx = 1, insamp = "2019-01-01",
    minyear = 3, maxyear = 16) {

    year_list <- NULL
    for (yx in minyear:maxyear) {
        year_list <- c(year_list, paste(2000 + yx, "-01-01",
            sep = ""))
    }

    # Create result_matrix
    res_mat <- matrix(1:(7 * length(year_list)), ncol = 7, dimnames = list(1:length(year_list),
        c("StartDate", "Sharpe", "L_Sharpe", "Drawdown", "L_Drawdown",
            "MSE", "L_MSE")))

    lirino <- NULL

    pb <- txtProgressBar(min = 1, max = length(year_list), style = 3)

    # Loop through years
    for (i in 1:length(year_list)) {
        startdate <- year_list[i]


        opt_obj <- optimize_sma(x, inx, min_L = 5, max_L = 500,
            L_forecast = 1, start = startdate, insamp = "2019-01-01",
            plot_T = FALSE)
        res_mat[i, ] <- c(year_list[i], as.numeric(opt_obj$sharpe_opt),
            rownames(opt_obj$sharpe_opt), as.numeric(opt_obj$draw_opt),
            rownames(opt_obj$draw_opt), as.numeric(opt_obj$mse_opt),
            rownames(opt_obj$mse_opt))


        resulterino <- opt_obj$res_mat
        StartDate <- rep(year_list[i], dim(resulterino)[1])
        result <- as.data.frame(cbind(resulterino, StartDate))

        lirino <- rbind(lirino, result)


        setTxtProgressBar(pb, i)
    }
    res_df <- as.data.frame(res_mat)
    close(pb)

    return(list(res_df = res_df, lirino = lirino))
}
optimize_sma_opt <- function(x, inx, min_L = 5, max_L = 500,
    L_forecast = 1, start = "2015-10-30", insamp = "2019-01-01") {

    xall <- x[paste(start, "/", sep = "")][, inx]
```

```
xin <- xall[paste("/", insamp, sep = "")]
xout <- xall[paste(insamp, "/", sep = "")]

sharpe_opt <- -9e+99
draw_opt <- -9e+99
mse_opt <- -9e+99

filler <- 1
res_mat <- matrix(1:((length(min_L:max_L)) * 4), ncol = 4,
    dimnames = list(1:(length(min_L:max_L)), c("L", "Sharpe",
        "Drawdown", "MSE")))

pb <- txtProgressBar(min = min_L, max = max_L, style = 3)

for (L in min_L:max_L) {
    # Moving Average
    yhat_full <- SMA(xin, n = L)
    # Cut init
    yhat <- yhat_full[max_L:length(yhat_full)]

    # Perf (daily)
    perf <- lag(sign(yhat)) * xin

    sharpe <- sqrt(250) * mean(perf, na.rm = T)/sqrt(var(perf,
        na.rm = T))

    maxdraw <- -max(abs(Drawdowns(na.exclude(perf), geometric = F)))

    future_returns <- lag(SMA(xin, n = L_forecast), k = -L_forecast)
    MSE <- -as.double(mean((future_returns - yhat)^2, na.rm = T))/var(xin,
        na.rm = T)

    res_mat[filler, ] <- c(L, sharpe, maxdraw, MSE)

    filler <- filler + 1

    # Best
    if (sharpe > sharpe_opt) {
        sharpe <- as.data.frame(sharpe)
        sharpe_opt <- sharpe
        rownames(sharpe_opt) <- paste(L)
        colnames(sharpe_opt) <- "Sharpe"
    }

    if (maxdraw > draw_opt) {
        maxdraw <- as.data.frame(maxdraw)
        draw_opt <- maxdraw
        rownames(draw_opt) <- paste(L)
        colnames(draw_opt) <- "Drawdown"
    }

    if (MSE > mse_opt) {
        MSE <- as.data.frame(MSE)
```

```
            mse_opt <- MSE
            rownames(mse_opt) <- paste(L)
            colnames(mse_opt) <- "MSE"
        }

        setTxtProgressBar(pb, L)
    }
    close(pb)

    listerino <- list(sharpe_opt = sharpe_opt, draw_opt = draw_opt,
        mse_opt = mse_opt, res_mat = res_mat)
    return(listerino)
}

perfplot_sma_opt <- function(xall, insamp, res_obj, inx) {
    xall <- xall[, inx]

    # Optima
    sharpe_opt <- res_obj$res_df[which.max(res_obj$res_df[, 2]),
        1:3]
    drawdown_opt <- res_obj$res_df[which.max(res_obj$res_df[,
        4]), c(1, 4:5)]
    mse_opt <- res_obj$res_df[which.max(res_obj$res_df[, 6]),
        c(1, 6:7)]

    # Filterlengths
    sharpe_L <- as.numeric(sharpe_opt[3])
    drawdown_L <- as.numeric(drawdown_opt[3])
    MSE_L <- as.numeric(mse_opt[3])

    # Startdates
    sharpe_startdate <- sharpe_opt[1]

    # Trading Signal
    sharpe_signal <- sign(SMA(xall, n = sharpe_L))
    drawdown_signal <- sign(SMA(xall, n = drawdown_L))
    MSE_signal <- sign(SMA(xall, n = MSE_L))

    # Returns
    sharpe_ret <- lag(sharpe_signal) * xall
    drawdown_ret <- lag(drawdown_signal) * xall
    MSE_ret <- lag(MSE_signal) * xall

    bnh_perf <- cumsum(na.exclude(xall[paste(insamp, "/", sep = ""),
        ]))
    sharpe_perf <- cumsum(na.exclude(sharpe_ret[paste(insamp,
        "/", sep = ""), ]))
    drawdown_perf <- cumsum(na.exclude(drawdown_ret[paste(insamp,
        "/", sep = ""), ]))
    MSE_perf <- cumsum(na.exclude(MSE_ret[paste(insamp, "/",
        sep = ""), ]))

    ymin <- min(c(min(bnh_perf), min(sharpe_perf), min(drawdown_perf),
```

```
        min(MSE_perf)))
    ymax <- max(c(max(bnh_perf), max(sharpe_perf), max(drawdown_perf),
        max(MSE_perf)))


    plot(bnh_perf, main = paste("Out-of-sample-Index: ", inx),
        lwd = 2, ylim = c(ymin - 0.1 * abs(ymin), ymax + 0.1 *
            ymax))
    lines(sharpe_perf, col = "#DF536B", lty = 2, lwd = 2)
    lines(drawdown_perf, col = "#2297E6", lty = 3, lwd = 2)
    lines(MSE_perf, col = "#61D04F", lty = 4, lwd = 2)

    addLegend("topleft", legend.names = c("Buy & Hold", paste("StartDate: ",
        sharpe_opt[1], "| Sharpe | L: ", sharpe_L), paste("StartDate: ",
        drawdown_opt[1], "| Drawdown | L: ", drawdown_L), paste("StartDate: ",
        mse_opt[1], "| MSE | L: ", MSE_L)), lty = c(1, 2, 3,
        4), lwd = c(2, 2, 2, 2), col = c("#000000", "#DF536B",
        "#2297E6", "#61D04F"), cex = 0.8)

    lines(sharpe_signal, on = NA, ylim = c(-1.5, 1.5), col = "#DF536B",
        lwd = 2)
    lines(drawdown_signal, on = NA, ylim = c(-1.5, 1.5), col = "#2297E6",
        lwd = 2)
    print(lines(MSE_signal, on = NA, ylim = c(-1.5, 1.5), col = "#61D04F",
        lwd = 2))
}

loop_sma_trading <- function(x, inx, min_L = 5, max_L = 500,
    insamp = "2018-12-31", invest_amount = 1e+06, tradingcosts = 190) {


    filler <- 1
    res_mat <- matrix(1:((length(min_L:max_L)) * 4), ncol = 4,
        dimnames = list(1:(length(min_L:max_L)), c("L", "SMA-Return",
            "Trades", "Sharpe")))

    pb <- txtProgressBar(min = min_L, max = max_L, style = 3)

    for (L in min_L:max_L) {
        # SMA L<-328
        sma_full <- SMA(x[, inx], n = L)

        # Cut init
        sma_cut <- sma_full[paste(insamp, "/", sep = "")]

        # Perf (daily)
        perf <- lag(sign(sma_cut)) * x[, inx][paste(insamp, "/",
            sep = "")]
        trade_sig <- lag(sign(sma_cut))

        # Count Trades
        sma_trade_count <- trading_counter(as.numeric(na.exclude(trade_sig)))
```

```
        # Sharpe
        sharpe <- sharpe_fun(na.exclude(perf))

        sma_ret <- tail((cumsum(na.exclude(perf)) * invest_amount) -
            (sma_trade_count * tradingcosts), 1)
        bnh_ret <- cumsum(x[, inx][paste(insamp, "/", sep = "")]) *
            invest_amount


        res_mat[filler, ] <- c(L, sma_ret, sma_trade_count, sharpe)

        filler <- filler + 1

        setTxtProgressBar(pb, L)
    }
    close(pb)
    res_df <- as.data.frame(res_mat)
    return(res_df)
}

plot_sma <- function(x, inx, L, insamp = "2018-12-31") {
    sma_full <- SMA(x[, inx], n = L)

    # Cut init
    sma_cut <- sma_full[paste(insamp, "/", sep = "")]

    # Perf (daily)
    perf <- lag(sign(sma_cut)) * x[, inx][paste(insamp, "/",
        sep = "")]
    trade_sig <- lag(sign(sma_cut))

    # Trade Counter
    sma_trade_count <- trading_counter(as.numeric(na.exclude(trade_sig)))

    # Sharpe
    bnh_sharpe <- round(sharpe_fun(x[, inx]["2019-01-01/"]),
        3)
    trade_sharpe <- round(sharpe_fun(na.exclude(perf)), 3)

    bnh_perf <- cumsum(x[, inx]["2019-01-01/"])
    trade_perf <- cumsum(na.exclude(perf))

    ymin <- min(c(min(bnh_perf), min(trade_perf)))
    ymax <- max(c(max(bnh_perf), max(trade_perf)))

    plot(bnh_perf, main = paste("Out-of-sample-Index: ", inx),
        lwd = 2, ylim = c(ymin - 0.1 * abs(ymin), ymax + 0.1 *
            ymax))
    lines(trade_perf, col = "#E413A3", lty = 2, lwd = 2)


    addLegend("topleft", legend.names = c(paste("Buy & Hold | Sharpe=",
        bnh_sharpe), paste("Trading-Opt.=", trade_sharpe, "| L=",
```

```
        L, "| Trades n=", sma_trade_count)), lty = c(1, 2), lwd = c(2,
        2), col = c("#000000", "#E413A3"), cex = 0.8)

    print(lines(trade_sig, on = NA, ylim = c(-1.5, 1.5), col = "#E413A3",
        lwd = 2))
}
```

6.3.4.2. Plots & Tables

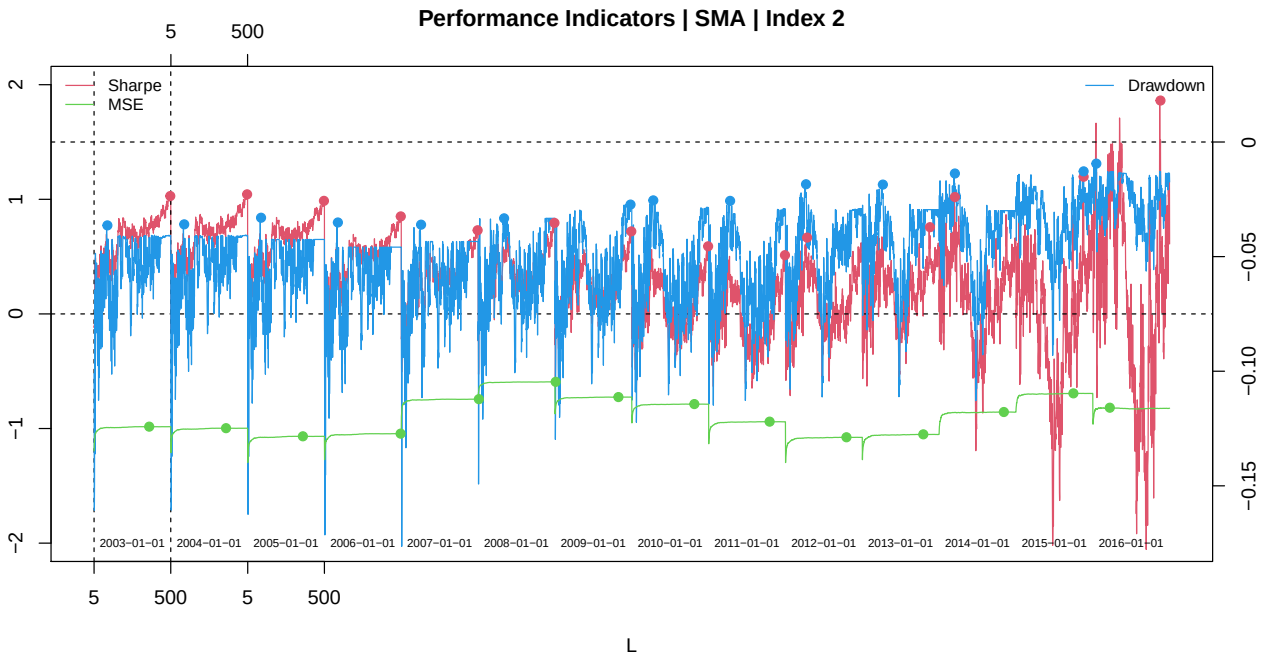

Figure 35: Optim

Table 11: SMA-Optima by performance indicators | Index 2

StartDate	Sharpe	L_Sharpe	Drawdown	L_Drawdown	MSE	L_MSE
2003-01-01	1.03	492	-0.039010	86	-0.984	356
2004-01-01	1.04	492	-0.038522	86	-0.997	356
2005-01-01	0.99	492	-0.035429	86	-1.068	356
2006-01-01	0.85	492	-0.037649	86	-1.045	491
2007-01-01	0.73	492	-0.038677	127	-0.744	500
2008-01-01	0.79	492	-0.035709	167	-0.593	500
2009-01-01	0.72	492	-0.029269	487	-0.726	409
2010-01-01	0.59	492	-0.027305	138	-0.787	403
2011-01-01	0.51	492	-0.027542	138	-0.940	393
2012-01-01	0.67	139	-0.019760	132	-1.076	393
2013-01-01	0.76	436	-0.019920	132	-1.051	393
2014-01-01	1.02	101	-0.014730	101	-0.856	417
2015-01-01	1.20	436	-0.013729	436	-0.693	370
2016-01-01	1.86	436	-0.010118	22	-0.818	109

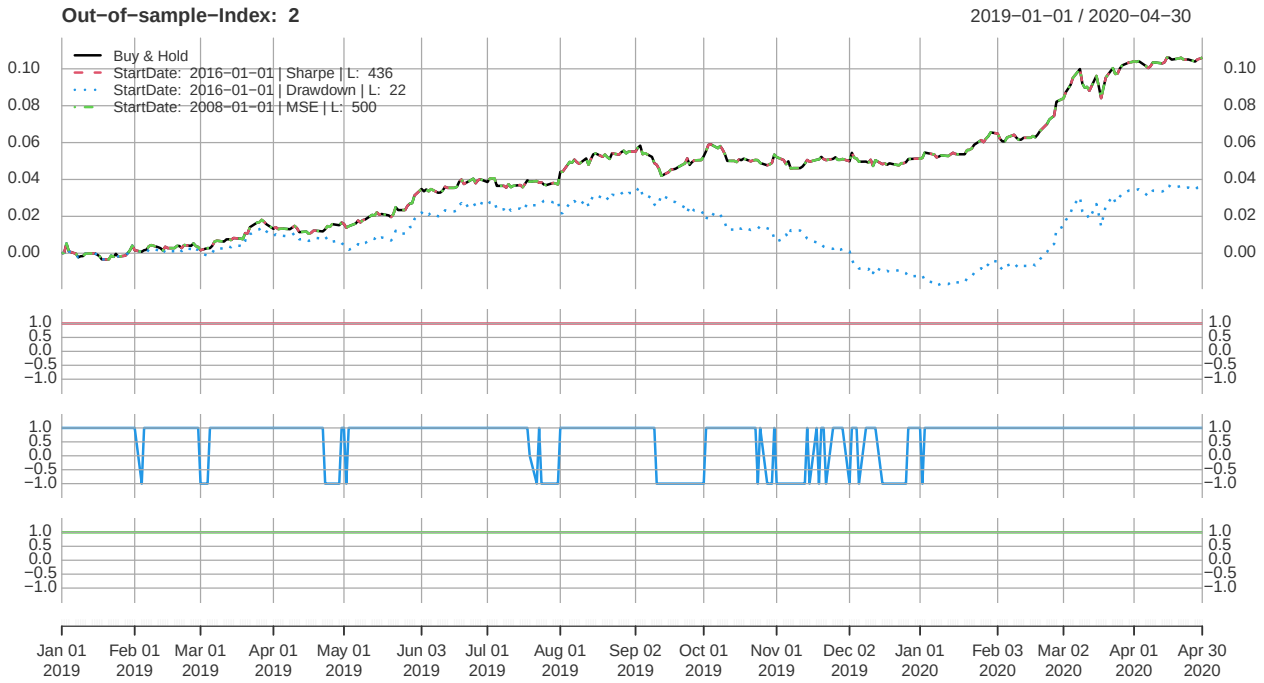

Figure 36: Best

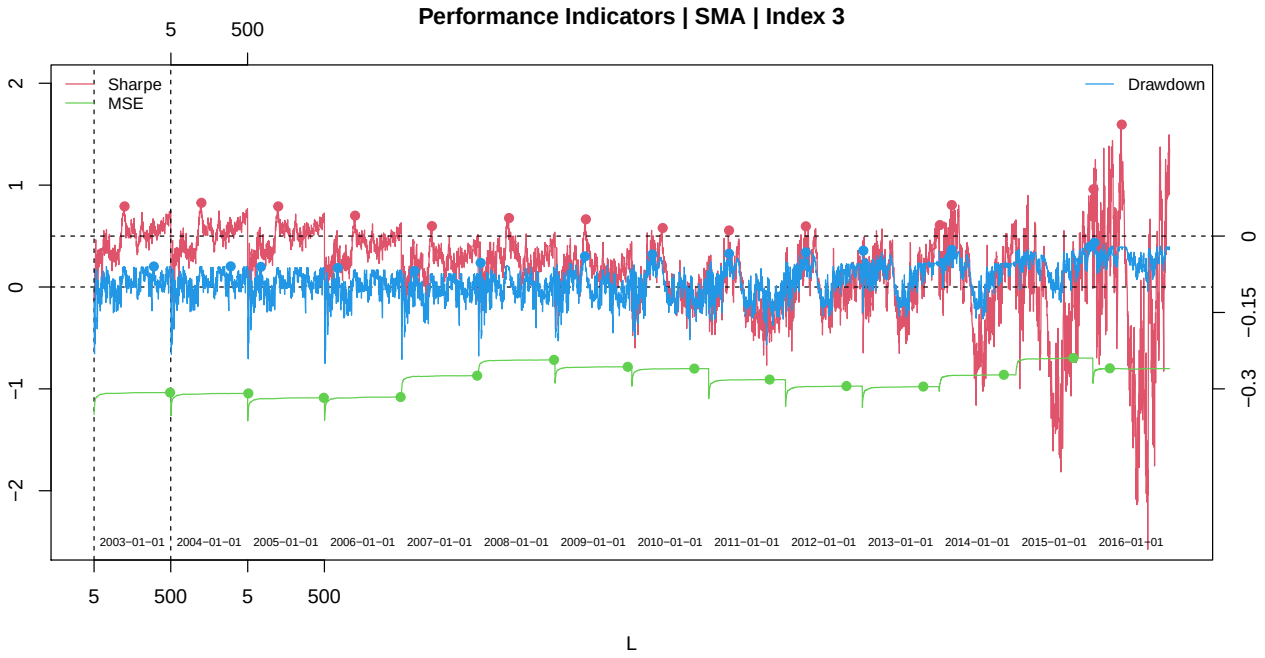

Figure 37: Optim

Table 12: SMA-Optima by performance indicators | Index 3

StartDate	Sharpe	L_Sharpe	Drawdown	L_Drawdown	MSE	L_MSE
2003-01-01	0.79	197	-0.060662	386	-1.036	491
2004-01-01	0.83	197	-0.060434	386	-1.045	500
2005-01-01	0.79	197	-0.061444	85	-1.087	491
2006-01-01	0.70	197	-0.063696	85	-1.080	491
2007-01-01	0.60	197	-0.069731	85	-0.870	489
2008-01-01	0.68	199	-0.053510	16	-0.715	489
2009-01-01	0.66	199	-0.040467	193	-0.783	469
2010-01-01	0.58	199	-0.036689	132	-0.801	403
2011-01-01	0.56	131	-0.035098	131	-0.908	393
2012-01-01	0.60	131	-0.032591	131	-0.973	393
2013-01-01	0.61	500	-0.029062	6	-0.977	393
2014-01-01	0.80	81	-0.027517	81	-0.862	417
2015-01-01	0.96	500	-0.021496	474	-0.697	370
2016-01-01	1.59	186	-0.013063	10	-0.798	109

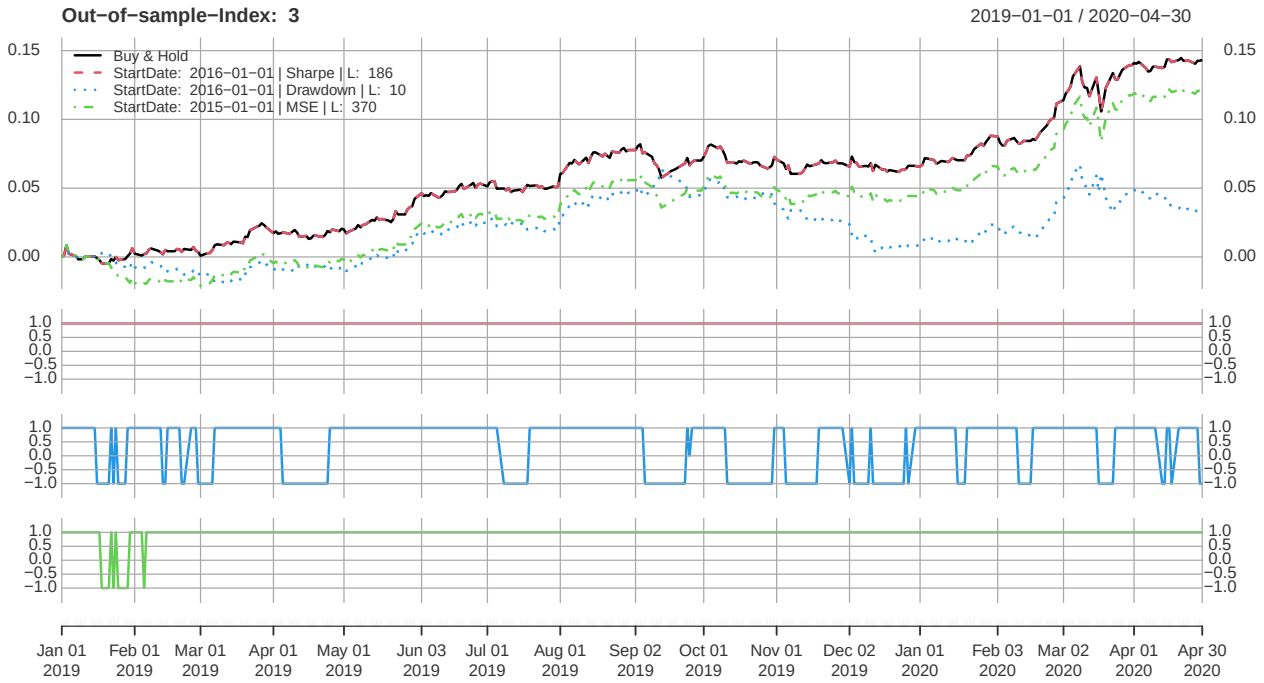

Figure 38: Best

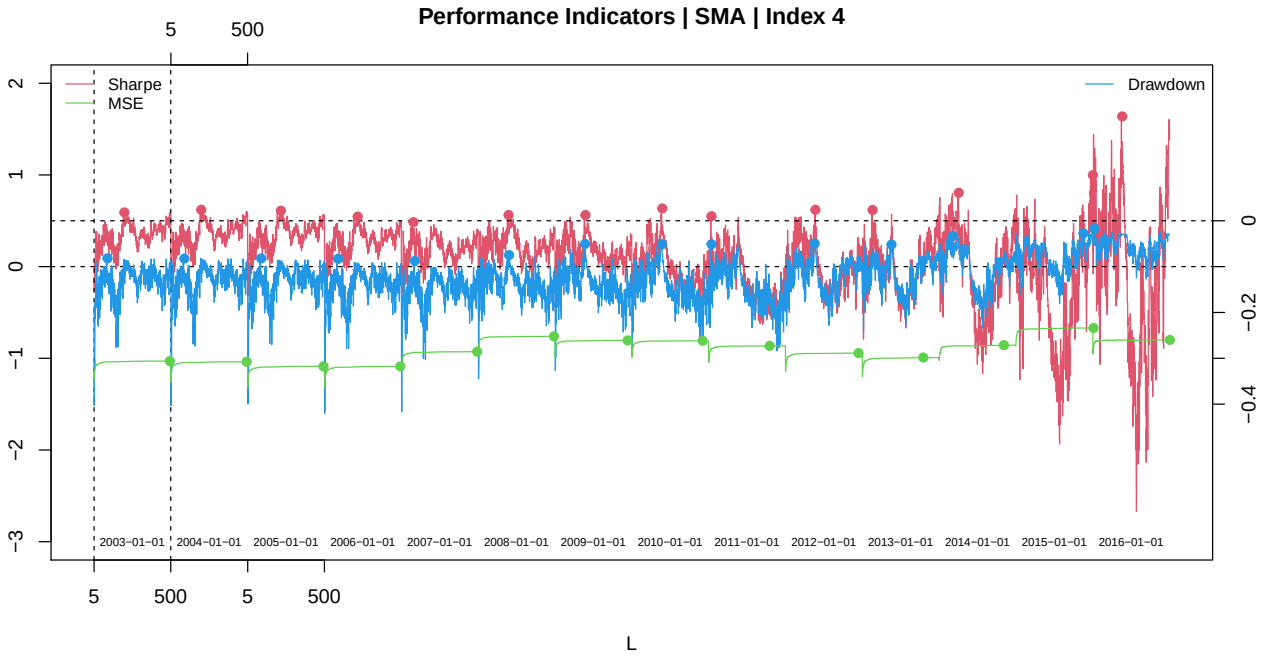

Figure 39: Optim

Table 13: SMA-Optima by performance indicators | Index 4

StartDate	Sharpe	L_Sharpe	Drawdown	L_Drawdown	MSE	L_MSE
2003-01-01	0.59	196	-0.081763	87	-1.030	489
2004-01-01	0.62	196	-0.082296	87	-1.038	489
2005-01-01	0.61	214	-0.082023	87	-1.087	489
2006-01-01	0.55	214	-0.082992	87	-1.088	489
2007-01-01	0.49	78	-0.087710	87	-0.928	489
2008-01-01	0.56	196	-0.074848	199	-0.760	489
2009-01-01	0.56	196	-0.050453	194	-0.805	469
2010-01-01	0.63	196	-0.051127	194	-0.809	457
2011-01-01	0.55	18	-0.051190	18	-0.864	393
2012-01-01	0.62	193	-0.049622	188	-0.943	471
2013-01-01	0.62	66	-0.051850	188	-0.992	393
2014-01-01	0.81	126	-0.033376	88	-0.857	417
2015-01-01	1.00	496	-0.027309	435	-0.669	500
2016-01-01	1.64	189	-0.017155	9	-0.799	497

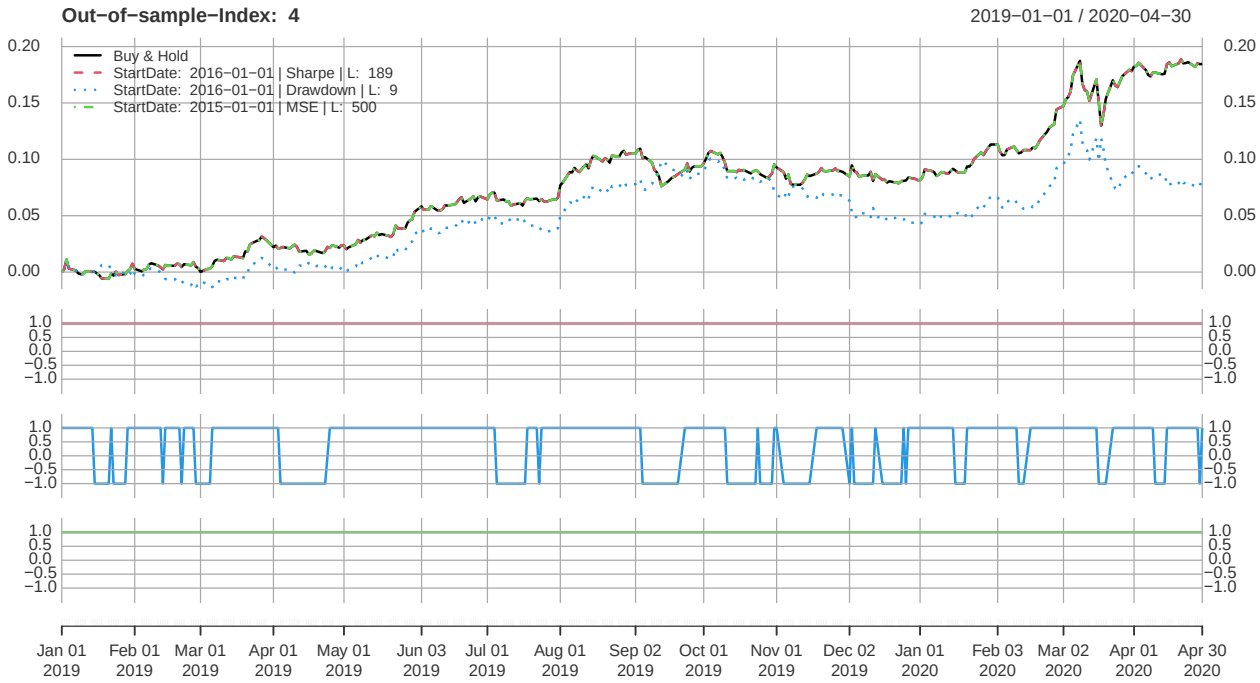

Figure 40: Best

6.3.5. MA Crossings

6.3.5.1. Optimization Code

```
loop_func_v3 <- function(data, minyear = 3, maxyear = 18, returnmax = 10) {
    # This function uses the cross_optim_easy_v2 function an
    # loops the function through the insample timespan and
    # returns a data frame

    year_list <- NULL
    for (year in minyear:maxyear) {
        year_list <- c(year_list, paste(2000 + year, "-01-01",
            sep = ""))
    }

    pb <- txtProgressBar(min = 1, max = length(year_list), style = 3)
    startdate = as.Date(paste(2000 + minyear, "-01-01", sep = ""))

    datevec <- as.data.frame(c(rep(startdate, returnmax)))
    colnames(datevec) = "startdate"
    iteration <- cross_optim_easy_v3(x = data, start = startdate,
        returnmax = returnmax)
    iteration_withdate = cbind(datevec, iteration)

    # Loop through years
    for (i in 2:length(year_list)) {
        startdate <- year_list[i]
        iteration <- cross_optim_easy_v3(x = data, start = startdate,
            returnmax = returnmax)

        datevec <- as.data.frame(c(rep(as.Date(startdate), returnmax)))
        colnames(datevec) = "startdate"

        iteration_withdate_loop = cbind(datevec, iteration)
        iteration_withdate = rbind(iteration_withdate, iteration_withdate_loop)

        setTxtProgressBar(pb, i)
    }
    close(pb)
    iteration_withdate <- xts(iteration_withdate[, -1], iteration_withdate[,
        1])
    return(iteration_withdate)
}
cross_optim_easy_v3 <- function(x, start, end = "2018-12-31",
    L1min = 1, L1max = 50, L2min = 100, L2max = 250, returnmax = 10) {
    ### This function optimizes the ma crossings with the given
    ### filterlength in the given timespan and returns a dataframe
    ### with the 'returnmax' sharpe and drawdownfilter which where
    ### found by common trading rules of ma crossings.
    horizon <- paste(start, "::", end, sep = "")

    sharpmat <- matrix(NA, L1max, L2max)  ## creating empty matrizes for sharpe and maxdrawdown
    drawmat <- matrix(NA, L1max, L2max)
```

```
    pb = txtProgressBar(min = L1min, max = L1max, style = 3)
    for (k in L1min:L1max) {

        for (j in L2min:L2max) {

            # Simple Moving Averages
            sma1 <- SMA(x, k)
            sma2 <- SMA(x, j)

            # Signals
            signal <- rep(0, length(sma1))
            signal[which(sma1 > sma2 & lag(sma1) < lag(sma2))] <- 1
            signal[which(sma1 < sma2 & lag(sma1) > lag(sma2))] <
                --1
            signal[which(sma1 > sma2)] <- 1
            signal[which(sma1 < sma2)] <- -1
            signal <- reclass(signal, sma1)

            # Trading
            trade <- Lag(signal[horizon], 1)
            return <- diff(log(x))
            ret <- return * trade
            ret <- na.exclude(ret)

            # Sharpe
            sharpmat[k, j] <- sqrt(250) * mean(ret)/sqrt(var(ret))

            # Drawdown
            drawmat[k, j] <- -max(abs(Drawdowns(ret, geometric = F)))
        }
        setTxtProgressBar(pb, k)
    }
    close(pb)

    # finding the [returmax] best sharpevalues an thei indexes
    # =l1 / l2
    sharpmax <- which(sharpmat >= sort(sharpmat, decreasing = T)[returnmax],
        arr.ind = T)
    sharpmax = head(sharpmax, returnmax)
    # finding the [returmax] best drawevalues an thei indexes =l1
    # / l2
    drawmax <- which(drawmat >= sort(drawmat, decreasing = T)[returnmax],
        arr.ind = T)
    drawmax = head(drawmax, returnmax)

    maximus = cbind(sharpmat[sharpmax], sharpmax, drawmat[drawmax],
        drawmax)  #combining the vector to return each value
    colnames(maximus) = c("sharpe", "sharpe_l1", "sharpe_l2",
        "drawdown", "drawdown_l1", "drawdown_l2")
    return(as.data.frame(maximus))
}
```

6.3.5.2. Plots & Tables

Table 14: index 1 best

	sharpe	sharpe_l1	sharpe_l2	drawdown	drawdown_l1	drawdown_l2
2003-01-01	1.458	49	246	-0.014	19	105
2004-01-01	1.467	49	246	-0.014	19	105
2005-01-01	1.526	49	243	-0.014	26	100
2006-01-01	1.529	49	243	-0.014	26	100
2007-01-01	1.393	49	243	-0.015	26	100
2008-01-01	1.072	49	243	-0.016	26	100
2009-01-01	0.938	42	113	-0.010	10	222
2010-01-01	1.095	42	113	-0.009	10	224
2011-01-01	0.920	42	113	-0.009	2	167
2012-01-01	0.837	1	146	-0.007	1	171
2013-01-01	0.883	37	113	-0.007	1	172
2014-01-01	1.044	37	113	-0.007	1	171
2015-01-01	1.078	37	113	-0.007	1	171
2016-01-01	1.275	1	106	-0.007	18	102
2017-01-01	1.819	37	119	-0.005	1	171
2018-01-01	2.413	3	108	-0.004	3	108

Table 15: index 2 best

	sharpe	sharpe_l1	sharpe_l2	drawdown	drawdown_l1	drawdown_l2
2003-01-01	0.714	35	242	-0.044	50	249
2004-01-01	0.718	35	242	-0.044	50	249
2005-01-01	0.747	39	250	-0.040	30	118
2006-01-01	0.881	32	250	-0.034	37	100
2007-01-01	0.880	32	250	-0.034	14	119
2008-01-01	0.713	32	250	-0.036	14	119
2009-01-01	0.592	38	106	-0.034	14	127
2010-01-01	0.749	35	182	-0.032	39	197
2011-01-01	0.606	35	182	-0.024	15	131
2012-01-01	0.545	35	182	-0.025	15	131
2013-01-01	0.502	35	182	-0.023	15	133
2014-01-01	0.733	38	106	-0.019	27	119
2015-01-01	0.764	16	135	-0.019	27	119
2016-01-01	1.052	17	111	-0.014	17	106
2017-01-01	1.096	17	111	-0.014	19	100
2018-01-01	1.869	24	178	-0.014	16	100

Table 16: index 3 best

	sharpe	sharpe_l1	sharpe_l2	drawdown	drawdown_l1	drawdown_l2
2003-01-01	0.567	18	188	-0.055	22	120
2004-01-01	0.571	18	188	-0.055	22	120
2005-01-01	0.554	18	188	-0.055	6	105
2006-01-01	0.660	28	102	-0.050	37	106
2007-01-01	0.651	31	102	-0.051	37	106
2008-01-01	0.640	31	102	-0.053	37	106
2009-01-01	0.501	28	102	-0.055	5	139
2010-01-01	0.603	18	188	-0.054	5	139
2011-01-01	0.572	24	104	-0.041	14	178
2012-01-01	0.417	16	185	-0.038	18	172
2013-01-01	0.416	18	115	-0.038	18	172
2014-01-01	0.691	17	118	-0.028	20	121
2015-01-01	0.768	17	118	-0.029	20	121
2016-01-01	0.990	16	111	-0.021	14	102
2017-01-01	0.929	18	102	-0.020	19	101
2018-01-01	1.414	16	111	-0.020	14	100

Table 17: index 4 best

	sharpe	sharpe_l1	sharpe_l2	drawdown	drawdown_l1	drawdown_l2
2003-01-01	0.511	7	116	-0.078	15	129
2004-01-01	0.514	7	116	-0.078	15	129
2005-01-01	0.563	7	116	-0.078	9	154
2006-01-01	0.635	12	103	-0.076	9	154
2007-01-01	0.621	11	101	-0.078	9	154
2008-01-01	0.610	11	101	-0.081	11	100
2009-01-01	0.490	12	103	-0.083	15	130
2010-01-01	0.624	13	148	-0.062	6	115
2011-01-01	0.696	13	146	-0.049	4	198
2012-01-01	0.569	13	146	-0.051	14	166
2013-01-01	0.711	13	146	-0.042	13	146
2014-01-01	0.782	15	136	-0.038	15	136
2015-01-01	0.731	15	136	-0.040	21	106
2016-01-01	0.998	12	134	-0.025	19	101
2017-01-01	0.861	11	103	-0.026	12	100
2018-01-01	1.379	11	103	-0.026	10	100

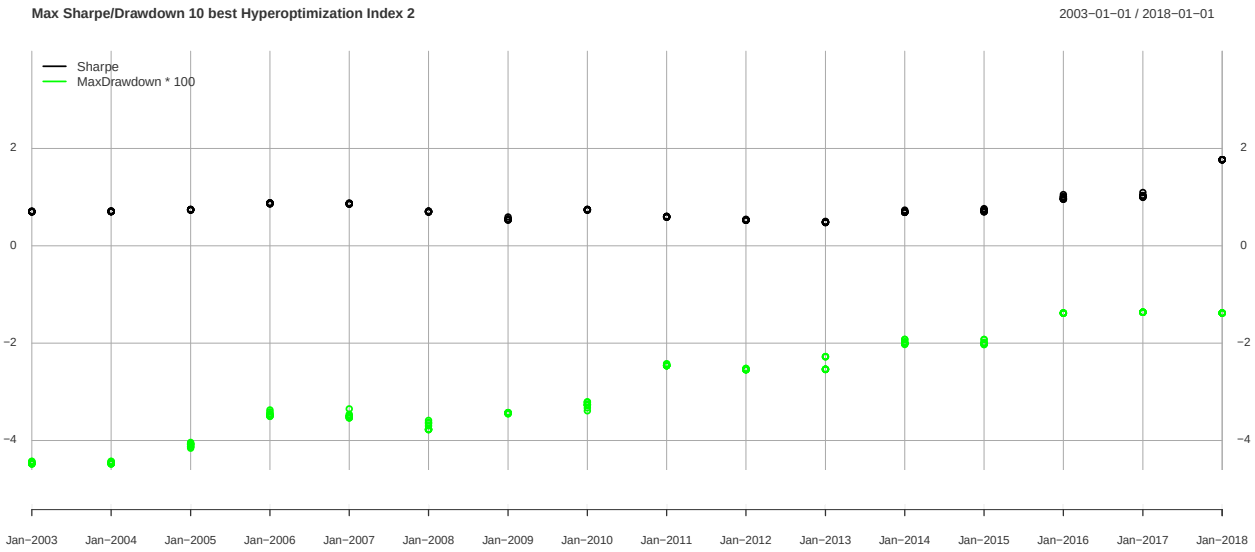

Figure 41: Hyperoptimization 10 best Index 2

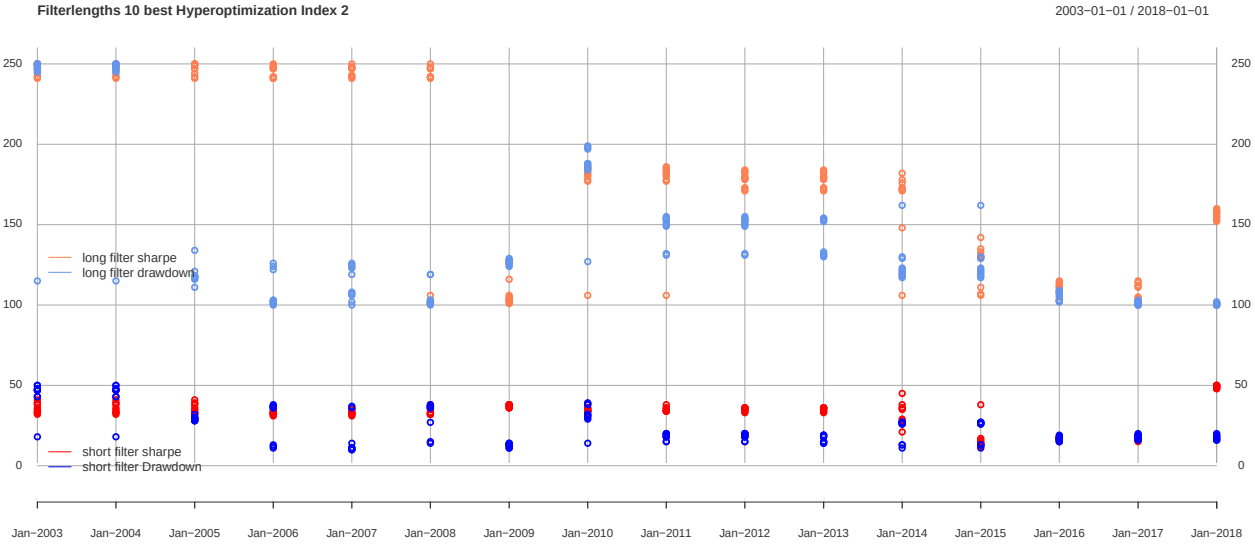

Figure 42: Hyperoptimization 10 best Index 2

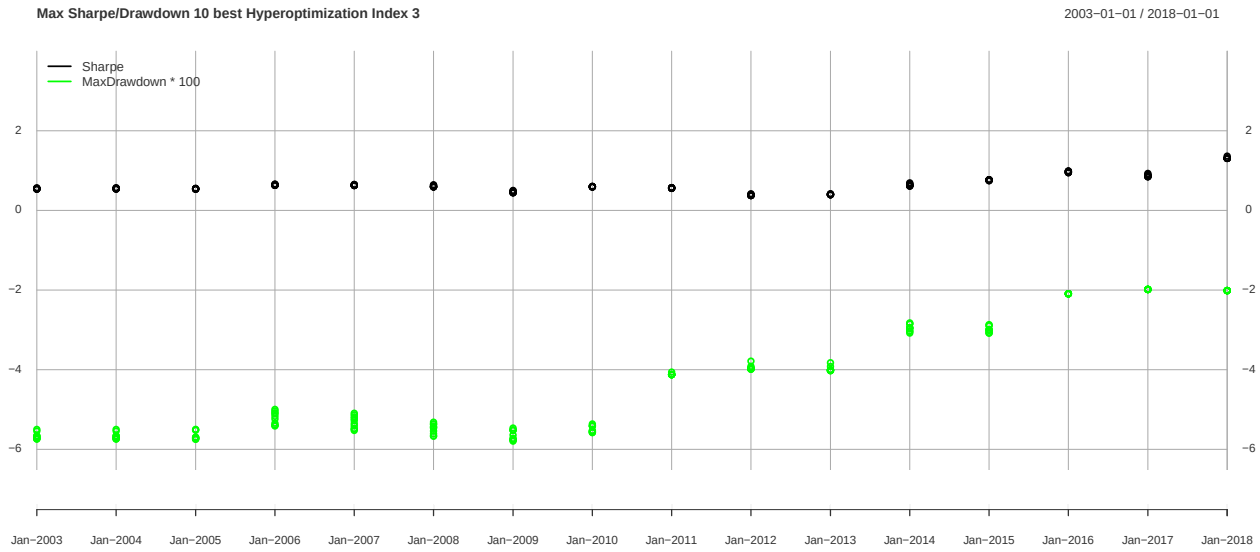

Figure 43: Hyperoptimization 10 best Index 3

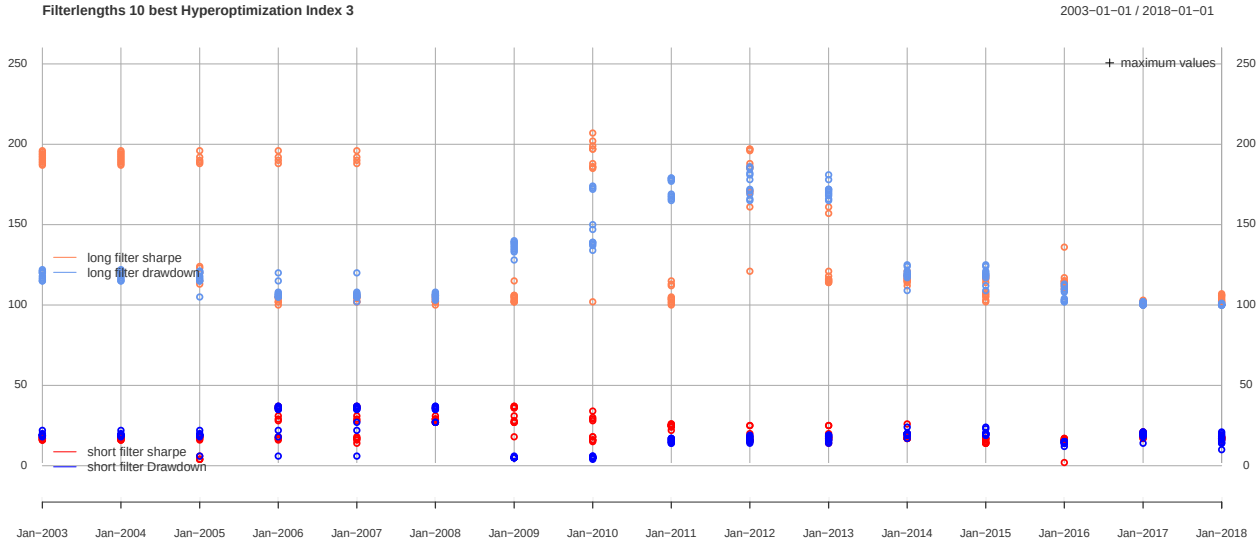

Figure 44: Hyperoptimization 10 best Index 3

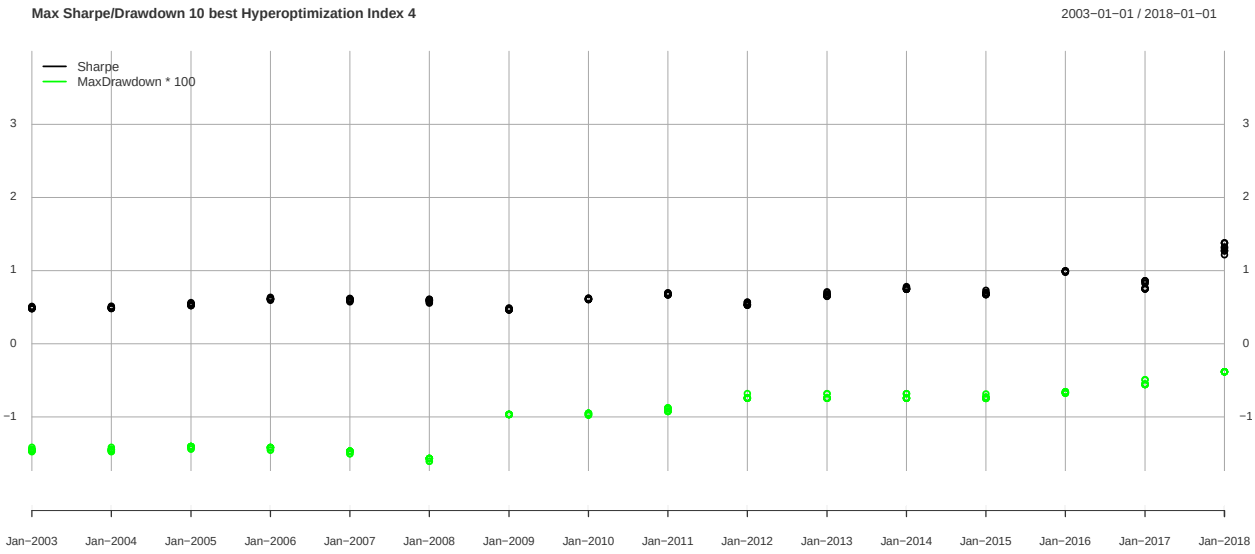

Figure 45: Hyperoptimization 10 best Index 4

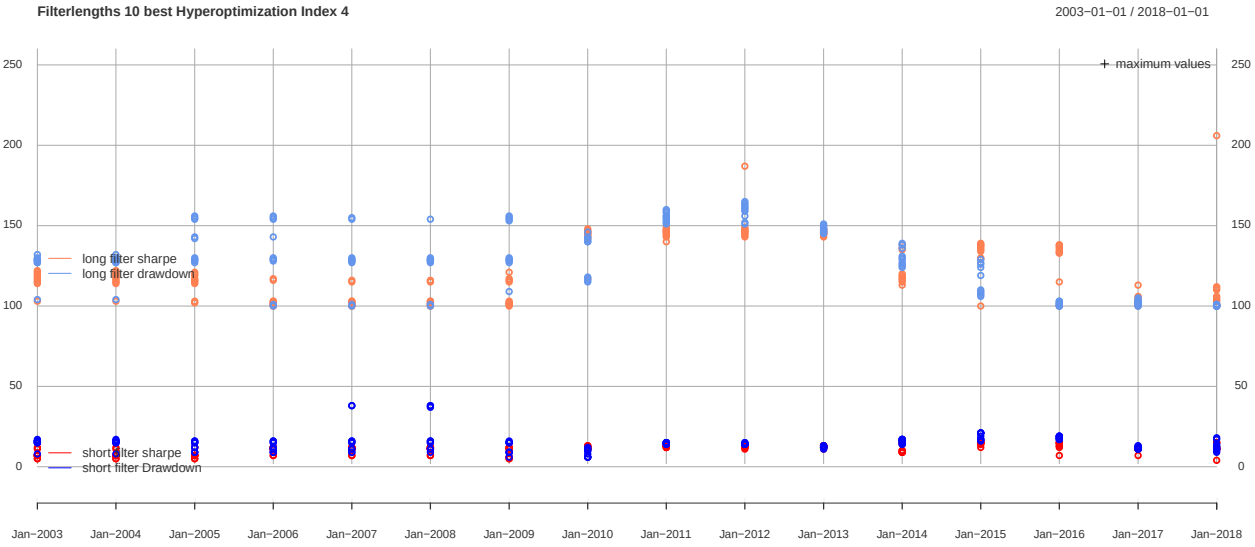

Figure 46: Hyperoptimization 10 best Index 4

Table 18: index 4 10 best

	sharpe	sharpe_l1	sharpe_l2
X2003.01.01	0.7832364	21	22
X2003.01.01.1	0.7331710	18	23
X2003.01.01.2	0.7335254	507	513
X2003.01.01.3	0.7271866	502	514
X2003.01.01.4	0.7278421	504	514
X2003.01.01.5	0.7409739	505	514
X2003.01.01.6	0.7248579	501	515
X2003.01.01.7	0.7359293	503	515
X2003.01.01.8	0.7394346	501	524
X2003.01.01.9	0.7274717	502	524
X2004.01.01	0.7932739	21	22
X2004.01.01.1	0.7376909	18	23
X2004.01.01.2	0.7377403	507	513
X2004.01.01.3	0.7313649	502	514
X2004.01.01.4	0.7320242	504	514
X2004.01.01.5	0.7452318	505	514
X2004.01.01.6	0.7290228	501	515
X2004.01.01.7	0.7401581	503	515
X2004.01.01.8	0.7436836	501	524
X2004.01.01.9	0.7316516	502	524
X2005.01.01	0.7638120	507	513
X2005.01.01.1	0.7572102	502	514
X2005.01.01.2	0.7578930	504	514
X2005.01.01.3	0.7715694	505	514
X2005.01.01.4	0.7547850	501	515
X2005.01.01.5	0.7663156	503	515
X2005.01.01.6	0.7699662	501	524
X2005.01.01.7	0.7575071	502	524
X2005.01.01.8	0.7546504	499	526
X2005.01.01.9	0.7541511	843	854
X2006.01.01	0.7782930	507	513
X2006.01.01.1	0.7873044	505	514
X2006.01.01.2	0.7833092	503	515
X2006.01.01.3	0.7795234	501	524
X2006.01.01.4	0.7782690	846	851
X2006.01.01.5	0.7825972	843	854
X2006.01.01.6	0.7778744	841	858
X2006.01.01.7	0.7789489	841	859
X2006.01.01.8	0.7763608	842	859
X2006.01.01.9	0.7764900	843	859

Table 19: index 4 10 best

	sharpe	sharpe_l1	sharpe_l2
X2007.01.01	0.8100201	846	851
X2007.01.01.1	0.8052617	846	853
X2007.01.01.2	0.8145258	843	854
X2007.01.01.3	0.8036027	841	857
X2007.01.01.4	0.8054720	844	857
X2007.01.01.5	0.8096094	841	858
X2007.01.01.6	0.8074541	843	858
X2007.01.01.7	0.8107279	841	859
X2007.01.01.8	0.8080337	842	859
X2007.01.01.9	0.8081681	843	859
X2008.01.01	0.7356758	846	851
X2008.01.01.1	0.7477013	843	854
X2008.01.01.2	0.7341643	842	855
X2008.01.01.3	0.7341643	842	856
X2008.01.01.4	0.7341643	841	857
X2008.01.01.5	0.7361727	844	857
X2008.01.01.6	0.7394919	841	858
X2008.01.01.7	0.7371762	843	858
X2008.01.01.8	0.7399265	841	859
X2008.01.01.9	0.7370318	842	859
X2009.01.01	0.6200652	7	6
X2009.01.01.1	0.6181580	8	6
X2009.01.01.2	0.9360669	8	7
X2009.01.01.3	0.7511134	21	22
X2009.01.01.4	0.6211622	20	23
X2009.01.01.5	0.6173982	843	854
X2009.01.01.6	0.6078073	841	858
X2009.01.01.7	0.6051020	843	858
X2009.01.01.8	0.6083151	841	859
X2009.01.01.9	0.6051020	843	859
X2010.01.01	0.9072085	8	7
X2010.01.01.1	0.8423911	846	851
X2010.01.01.2	0.8593225	843	854
X2010.01.01.3	0.8402631	842	855
X2010.01.01.4	0.8402631	842	856
X2010.01.01.5	0.8402631	841	857
X2010.01.01.6	0.8430908	844	857
X2010.01.01.7	0.8477639	841	858
X2010.01.01.8	0.8445037	843	858
X2010.01.01.9	0.8483759	841	859

Table 20: index 4 10 best

	sharpe	sharpe_l1	sharpe_l2
X2011.01.01	0.8936544	8	7
X2011.01.01.1	0.7983280	846	851
X2011.01.01.2	0.8182340	843	854
X2011.01.01.3	0.7958261	842	855
X2011.01.01.4	0.7958261	842	856
X2011.01.01.5	0.7958261	841	857
X2011.01.01.6	0.7991505	844	857
X2011.01.01.7	0.8046447	841	858
X2011.01.01.8	0.8008116	843	858
X2011.01.01.9	0.8053641	841	859
X2012.01.01	0.6903122	2	4
X2012.01.01.1	0.7753777	8	7
X2012.01.01.2	0.5932644	82	73
X2012.01.01.3	0.6855402	80	75
X2012.01.01.4	0.6285551	156	157
X2012.01.01.5	0.6043955	843	854
X2012.01.01.6	0.5878216	841	858
X2012.01.01.7	0.5831466	843	858
X2012.01.01.8	0.5886990	841	859
X2012.01.01.9	0.5831466	843	859
X2013.01.01	0.7538800	17	21
X2013.01.01.1	0.7937625	237	180
X2013.01.01.2	0.7566394	271	241
X2013.01.01.3	0.7821553	259	250
X2013.01.01.4	0.7548670	258	251
X2013.01.01.5	0.7705623	259	251
X2013.01.01.6	0.7968154	259	254
X2013.01.01.7	0.7524896	261	255
X2013.01.01.8	0.7832750	260	256
X2013.01.01.9	0.7959845	261	256
X2014.01.01	0.9476422	68	70
X2014.01.01.1	0.9403540	62	76
X2014.01.01.2	0.9201888	846	851
X2014.01.01.3	0.9552773	843	854
X2014.01.01.4	0.9216385	844	857
X2014.01.01.5	0.9313225	841	858
X2014.01.01.6	0.9245663	843	858
X2014.01.01.7	0.9325905	841	859
X2014.01.01.8	0.9241449	842	859
X2014.01.01.9	0.9245663	843	859

Table 21: index 4 10 best

	sharpe	sharpe_l1	sharpe_l2
X2015.01.01	1.0056340	62	76
X2015.01.01.1	0.9885516	237	182
X2015.01.01.2	0.9926590	266	240
X2015.01.01.3	1.0250787	268	240
X2015.01.01.4	0.9862364	271	240
X2015.01.01.5	0.9897647	264	241
X2015.01.01.6	1.0020151	265	241
X2015.01.01.7	1.0355885	261	244
X2015.01.01.8	1.0833421	262	244
X2015.01.01.9	0.9938176	259	254
X2016.01.01	1.5924966	264	237
X2016.01.01.1	1.6450740	268	240
X2016.01.01.2	1.5680311	270	240
X2016.01.01.3	1.5867347	271	240
X2016.01.01.4	1.5737134	265	241
X2016.01.01.5	1.5676278	271	241
X2016.01.01.6	1.5962519	261	243
X2016.01.01.7	1.6228009	261	244
X2016.01.01.8	1.6945367	262	244
X2016.01.01.9	1.5981183	259	254
X2017.01.01	1.7869970	336	85
X2017.01.01.1	1.7869970	337	85
X2017.01.01.2	1.7869970	338	85
X2017.01.01.3	1.7869970	328	86
X2017.01.01.4	1.7869970	329	86
X2017.01.01.5	1.7869970	324	87
X2017.01.01.6	1.7869970	325	87
X2017.01.01.7	1.7869970	326	87
X2017.01.01.8	1.7869970	327	87
X2017.01.01.9	1.7869970	321	88
X2018.01.01	2.6845743	769	336
X2018.01.01.1	2.8875217	845	371
X2018.01.01.2	2.6798980	846	372
X2018.01.01.3	2.6598773	849	373
X2018.01.01.4	2.6543246	851	374
X2018.01.01.5	2.7243716	853	375
X2018.01.01.6	2.8148887	856	377
X2018.01.01.7	2.6479815	586	574
X2018.01.01.8	2.6479815	586	575
X2018.01.01.9	2.6479815	585	576