SJÄLVSTÄNDIGA ARBETEN I MATEMATIK

MATEMATISKA INSTITUTIONEN, STOCKHOLMS UNIVERSITET

Monte Carlo Backtesting

av

Johannes Berglind Söderqvist

2014 - No 3

MATEMATISKA INSTITUTIONEN, STOCKHOLMS UNIVERSITET, 106 91 STOCKHOLM

Monte Carlo Backtesting

Johannes Berglind Söderqvist

Självständigt arbete i matematik 15 högskolepoäng, Grundnivå

Handledare: Salla Franzén och Karl Rökaeus

2014

Abstract

This paper explores the possibility of backtesting trading strategies using Monte Carlo simulation. An illustrative example is carried out by backtesting two strategies. The main strategy is the Magic Formula, introduced by Joel Greenblatt in his book "The Little Book that Beats the Market" from 2006. A randomized version of the Magic Formula is also backtested for comparison. The strategies are then compared to the chosen equally-weighted index based on the investable universe of stocks.

The results indicate that Monte Carlo simulation could be a fruitful way to backtest strategies over a shorter time period, no longer than one year. For the longer time periods employed the backtests did not provide any informative results. The Magic Formula strategy is intended for a holding period of three years or more so no relevant conclusions about its performance could be drawn using Monte Carlo simulation. The paper also contains some suggestions for future studies.

A description of the universe of investable stocks, the full code of the matlab-program and the results from the simulation are supplied in appendices.

Monte Carlo Backtesting

Johannes Berglind Söderqvist

January 2014

Contents

1 Introduction

This study aims to explore ways to backtest portfolio strategies using Monte Carlo simulation. The first part of the paper, chapter 2, 3 and 4 will go through some basic theory regarding backtesting, Monte Carlo methods and how they could be combined. In order to provide an illustrative example the following chapters will be subjected to a simple backtest of trading strategies by means of Monte Carlo simulation.

The main idea is to estimate a probability distribution of the future portfolio value given a certain trading strategy using historical data. For the simulation in the present study the historical data to be used is a number of series of adjusted[1] daily closing prices from a universe of selected stocks. From these price series daily growth and daily return are calculated. Samples are taken from the daily growth or the daily return directly before the point of time chosen to be the starting time of the simulation. The starting time will be varied within the time frame of 2006-2011. Since the relative future to this point in time is known one can get a notion of to what degree, if any, the results of the simulation gave useful feedback for evaluating the strategy.

[1] The prices are adjusted for dividends and splits.

2 Backtesting

In order to comprehend what a backtest is, it is of importance to understand how a trading strategy can be executed on a time frame of historical data. For this purpose some basic mathematical notation will be established.

2.1 Portfolios Over Time Frames

The aim of the notation is to describe trading strategies applied to a universe of stocks. Other financial instruments are left out since they are not included in this study.

2.1.1 A Portfolio of Stocks

Let $P(t)$ be a portfolio of stocks at time t. Suppose the universe of investible assets consists of m stocks. For each j let $S_j(t)$ be the stock price of stock j at time t, with $j = 1, ..., m$. Denote by $a_j(t)$ the number of shares of stock j contained in the portfolio at time t. If no short positions are allowed in the portfolio then $a_j(t) \in \mathbf{R}^+$, otherwise $a_j(t) \in \mathbf{R}$.

The weights of the portfolio $\mathbf{w}(t) = (w_1(t), ..., w_m(t))$ for the stock holdings are given by the following relation

$$w_k(t) = a_k(t)\frac{S_k(t)}{V_{P(t)}}$$

where $V_{P(t)}$ is the value of the portfolio at time t in accordance with

$$V_{P(t)} = \sum_{k=1}^{m} a_k(t)S_k(t)$$

2.1.2 A Time Frame

A backtest is generally performed over an interval of time, or time frame. Let $\mathbf{T}$ represent a time frame and let the time frame be a sequence of consecutive days of length $\Delta t = \frac{1}{250}$ years. Set the end of the first day to $t_0 = 0$ and call the end of the last day t_n. For any two consecutive time steps in $\mathbf{T}$ let

$$t_k = \Delta t + t_{k-1}, \quad 0<k<n, \quad k, n \in N$$

so that the time frame $\mathbf{T} = [0, t_n]$ is split into n steps of length Δt. The time frame $\mathbf{T}$ can be seen as a sequence $[t_k]_0^n$.

2.1.3 Growth and Return

The daily growth of a stock is given by

$$S(t_k) = S(t_{k-1})e^{G(t_{k-1},t_k)} \Leftrightarrow G(t_{k-1}, t_k) = \ln\left(\frac{S(t_k)}{S(t_{k-1})}\right)$$

and the growth over $\mathbf{T}$

$$G(0, t_n) = \sum_{k=1}^{n} \ln\left(\frac{S(t_k)}{S(t_{k-1})}\right)$$

and the return over $\mathbf{T}$

$$R(0, t_n) = \frac{S(t_n) - S(0)}{S(0)}$$

The relation between return and growth can be concluded as

$$R(t_{k-1}, t_k) = \frac{S(t_k) - S(t_{k-1})}{S(t_{k-1})} = \frac{S(t_k)}{S(t_{k-1})} - \frac{S(t_{k-1})}{S(t_{k-1})} = e^{G(t_{k-1},t_k)} - 1$$

It might also be of interest to relate one stock to another in terms of growth or return. Covariance and the correlation can be used to satisfy this interest. The covariance of the daily growth of two stocks S_1 and S_2 over $\mathbf{T}$ is calculated as

$$Cov(G_1, G_2) = \sum_{k=1}^{n} \frac{(G_1(t_{k-1}, t_k) - \mu_1)(G_2(t_{k-1}, t_k) - \mu_2)}{n}$$

where μ_1 and μ_2 are the average daily growth over the period of time for S_1 and S_2 respectively.

If for example the daily growth of two stocks have a positive covariance it means that the daily growth of the two stocks in general have the same sign. If the covariance on the other hand is negative they tend to have daily growths of opposite signs. The same holds for the daily return of two stocks.

Usually comparisons between instruments are done by means of the correlation between them denoted as ρ. That is the covariance divided by $(\sigma_1\sigma_2)$ where σ refers to the standard deviation. Consequently the correlation between the series of daily growth for S_1 and S_2 can be written as

$$\rho_{1,2} = \frac{Cov(G_1, G_2)}{(\sigma_1\sigma_2)}$$

This yields a number $\rho \in [-1, 1]$ more convenient for comparison.

2.2 Briefly About Trading Strategies

A trading strategy gives a method for making a selection of assets from the universe, and for assigning weights to those assets at each time t. Backtesting of a trading strategy can be seen as a way of studying it in a context of historical data. In order to be fruitfully backtested a trading strategy needs to be well defined. This means that the strategy should be possible to formulate mathematically and that its performance should be quantifiable. For the strategy to be evaluated using a backtest it is also necessary to have something to compare it with. This can for instance be a relevant market-index or another benchmark considered to be relevant for the trading strategy.

Strategies can aim to pick individual stocks or instruments from the universe relying on financial analysis of their individual qualities. This approach is usually referred to as bottom-up stock picking. A top-down strategy takes for a starting point the desired exposures to different market segments from the universe. Different methods can be used selecting the instruments that maintain the desired exposure. These could involve technical analysis using larger amounts of historical data to decide on the allocation, or exposures in the portfolio strategy. The value of the instruments in the universe changes. As a consequence the portfolio strategy should also define the schedule for rebalancing the portfolio with optimal weights. To keep the allocations flexible enough to be able to maintain the exposures, management of liquidity and other possible transaction limitations might have to be taken into account.

The election of instruments based on large amounts of information drives many modern strategies to involve a high level of optimization. Because of their complexity, for someone without the proper knowledge base or without enough time at their disposal, modern strategies might be difficult to grasp. The performance of a trading strategy in a backtest can play an important role in order to attract investors. For the backtesting to be convincing it should be carried out on a variety of historical timeframes, covering relevant market conditions. A backtest might also be used in order to stress-test a strategy, taking a historical time period as a starting point and then successively modifying it with respect to relevant parameters. This subject will be discussed briefly later in this chapter.

2.3 Backtesting a Trading Strategy

A simple form of backtest illustrates expected performance under certain market conditions in terms of plain application of the strategy to historical data. This kind of backtest is going to be frequently referred to later in this paper. To make this easier a short form for it is introduced.

Plain application of a trading strategy to historical or simulated data is onwards referred to as

Application on Time Frame (ATF)

As simple as it is it reflects the basic principals of a backtest, namely to test a strategy meant for future trading on historical data. The trivial approach to performing ATF is to consider possible future market conditions then trying to find periods in history when the markets have behaved similarly to the expected future market conditions. Applying the trading strategy to the universe of historical data could give an indication of how the portfolio would behave during similar market conditions.

This simple approach disregards the fact that every actor on the market helps creating it. The back-tested strategy would in itself impact the over-all market behaviour. This ignorance might be reasonable if the volumes of the transactions employed by the strategy are relatively small. If the volumes are large one might have to modify the behaviour of assets in the universe following the expected impact of the strategy. This complicates the calculations and gives rise to interpretations and assumptions in order to assess the expected impact.

In spite of market impact considerations some aspects of ATF backtesting can be relevant. Consider for example the short term performance of strategies where the transaction volumes employed by the strategy are large. Suppose that historical data suggests that the market generally reacts to a certain kind of market event with some delay. The ATF could be used to study the short-term performance of a strategy, before the over-all market reacts to the event.

Another issue with historical back-testing is the difficulty in getting correct historical data. Not only might there be a lack of accessible information, the available data might be wrong. Moreover, the real world contains massive amounts of information that is not stored in databases but still slightly affects the market. A historical data universe is thus by no means a complete representation of that market over the historical time frame in question. Predictions based on conclusions drawn from historical data might thus be expected to have the same lack of correspondence to reality. Relating to back-testing it seems reasonable and maybe sound to keep in mind that information grows old on both sides of the present.

Asset managers can resolve the data issue by purchasing backtesting services from external data suppliers. Such data providers use more sophisticated methods than ATF. Relevant benchmarks can for instance be constructed from the key factors that affect a particular portfolio. Extensive stress-testing of portfolios can be performed, not rarely employing stochastic simulation with

estimated parameters calculated from historical data. Such stress-tests focus on the probability that certain more or less extreme events would occur.

This study will explore the possibility to backtest strategies by means of stochastic simulation. The idea is, roughly, to calculate a probability distribution customized for a particular portfolio strategy and simulate possible outcomes of the strategy on this distribution. Stochastic simulation, commonly referred to as Monte Carlo simulation will be the theme of the next chapter.

3 Monte Carlo

Methods referred to as 'Monte Carlo' are used for many different purposes, usually when no other methods or models provide satisfactory performance in terms of accuracy or speed. What Monte Carlo methods have in common is the use of randomly generated numbers[2]. It is for example quite easy to approximate π by marking dots on uniformly random places on a square with side length $1\,m$. Draw a circle in the square with radius $\frac{1}{2}\,m$. The square has the area $1\,m^2$ and the circle $\frac{\pi}{4}\,m^2$. As the number of dots approaches infinity the number of dots inside the circle n and the total number of dots inside the square N will satisfy the following relation

$$\frac{n}{N} = \frac{\pi}{4}/1 \Leftrightarrow \pi = \frac{4n}{N}$$

The trustworthiness of Monte Carlo methods can be derived from the Strong Law of Large Numbers. Let X be a stochastic variable and let x_k, $k = 1, ..., n$, be a sample of X. Then the Strong Law of Large Numbers state that the average value of $x_1, ...x_n$ will converge allmost surely (meaning with probability 1) to the expected value of X as $n \to \infty$. Thus

$$\mu = \lim_{n\to\infty} \mu_n$$

where

$$\mu_n = \frac{1}{n}\sum_{k=1}^{n} x_k.$$

The variance of X can further be estimated as

$$\sigma_n^2 = \frac{1}{n}\sum_{k=1}^{n}(x_k - \mu_n)^2$$

To find an estimate of the expectation of a random variable X with sample space $U \subseteq \mathbf{R}$ it is thus a reasonable approach to draw a large number of outcomes of X and calculate μ_n. Relating this to the π-example above, imagine a division of the square into a grid of $k<n$ smaller squares. The expected value of a uniformly distributed random variable is $\frac{a+b}{2}$, with the sample space $[a, b]$. Specify where to mark each dot on the square using two random variables X and Y. Then the expected hit point in each square will converge almost surely to a point in the middle of it. Let $k, n \to \infty$ and the expected hit points cover the larger square uniformly so that the number of dots in subareas are subjected to the same relation as the subareas.

[2] An important part of any Monte Carlo calculation is to generate random numbers, or more precisely pseudorandom numbers. They can be generated in many different ways which is not the focus of this present study. For more information about this subject, see for example "Markovprocesser" -Rydén, Lindgren

Aiming to draw that large number of outcomes of X implies the venture of finding a probability density function that suits X. The probability distribution could be roughly estimated as the number of outcomes from a large sample, $\hat{X}$ that falls within $u_k \subset U$, associated with the k^{th} event divided by the total number of outcomes in $\hat{X}$. Such a distribution might be a bit cumbersome to deal with mathematically.

Another way of estimating a probability distribution for X is to fit a known probability distribution to a sample from a large number of observed outcomes of X and then assume the corresponding probability density function to hold for the actual sample space.

3.1 Dependency Between Random Variables

In many cases the outcomes of some stochastic variables $X_1, ..., X_m$ depend on each other and as a consequence a probability for one of them depend on the probabilities for the others. A probability distribution for such a collection of stochastic variables is usually referred to as an m-dimensional multivariate stochastic probability distribution. Such a distribution gives the probability for a certain set of outcomes $x_1, ..., x_m$ to occur. In the case of a normally distributed multivariate random variable of dimension m the dependency between $X_1, ..., X_m$ is captured in the covariance matrix. The elements in it are specified by

$$\sigma_{ij}^2 = Cov(X_i, X_j), \quad i, j = 1, ..., m$$

where the covariance can be estimated from a sample of n outcomes of the multivariate stochastic variable $\mathbf{X} = X_1, ..., X_m$.

$$Cov(\hat{X}_i, \hat{X}_j) = \sum_{k=1}^{n} \frac{(x_i(k) - \mu_i)(x_j(k) - \mu_j)}{n}$$

The covariance matrix of a univariate stochastic variable is thus its variance.

To generate random numbers from $\mathbf{X} = X_1, ..., X_m$ one generates m-dimensional random vectors from a multivariate distribution that is seen fit for $\mathbf{X}$ analogus to the univariate case.

4 Monte Carlo Backtesting

4.1 Assumption of Normally Distributed Data

The stockprice can be written as $S(t) = S(0)e^{G(0,t)}$ or $S(t) = S(0)(1 + R(0,t))$. Because of that the simulation of it could essentially be about simulating the growth $G(0,t)$ or the return $R(0,t)$ of the stock. Choosing to simulate with underlying growth by means of Monte Carlo simulation might in some cases be unnecessary complicated. Let us consider a simulation of the stockprice $S(t)$ over the time frame $\mathbf{T}$. Denote the daily growth of $S(t)$ as $G(t_{k-1}, t_k) = \ln \frac{S(t_k)}{S(t_{k-1})}$. Moreover, assume $G(t_{k-1}, t_k)$ to be a stochastic variable $\sim N(\mu, \sigma)$. Let $\{G_{(t_{k-1},t_k)} : k = 1, 2, ..., n\}$ be a stochastic process over $\mathbf{T}$. Realizing the process a large number of times will for each time t_k yield a sample space of all the realizations at t_k which can be seen as an estimate of the probability distribution of the growth of $S(t)$ from the first until the k^{th} day of the time frame.

Just looking at the sum as a stochastic variable will however give the same distribution as the simulation will approach. This follows from the fact that The sum of normally distributed stochastic variables is normally distributed. The sum is then normally distributed with the corresponding sum of expected values and variance as its respectively expected value and variance. Since

$$G(0, t_k) = \ln\left(\frac{S(t_1)}{S(t_0)}\right) + ... + \ln\left(\frac{S(t_k)}{S(t_{k-1})}\right)$$

consequently $G(0, t_k) \sim N(k\mu, \sqrt{k\sigma^2})$ and the probability distribution of $S(t_k)$ is given by $S(0)e^{G(0,t_k)}$.

Simulate by representing daily return with stochastic variables for $R(t_k, t_{k+1})$ does not provide the possibility of adding daily random variables together in the same way. In order to do so, the variables would have had to be defined as $\hat{R}(t_k, t_{k+1}) = 1 + R(t_k, t_{k+1})$. Assuming also that this variable is normally distributed it is possible to assess the price $S(t_k)$ by

$$\ln(S(t_k)) = \ln(S(0)) + \ln \hat{R}(t_0, t_1) + ... + \ln \hat{R}(t_k, t_{k+1}) = \ln(S(0)) + G(0, t_k)$$

If underlying daily growth is used the growth over a period of days is also normally distributed. This does not hold for daily return. It is quite common that the assumption of normally distributed growth does not completely correspond to reality as real distributions tend to have thicker tails, even though the assumption is frequently used in finance (Hull -2012, ch. 21.7). Two examples of this are shown below. In these cases it might therefore be reasonable to consider the fit of another distribution to the underlying data.

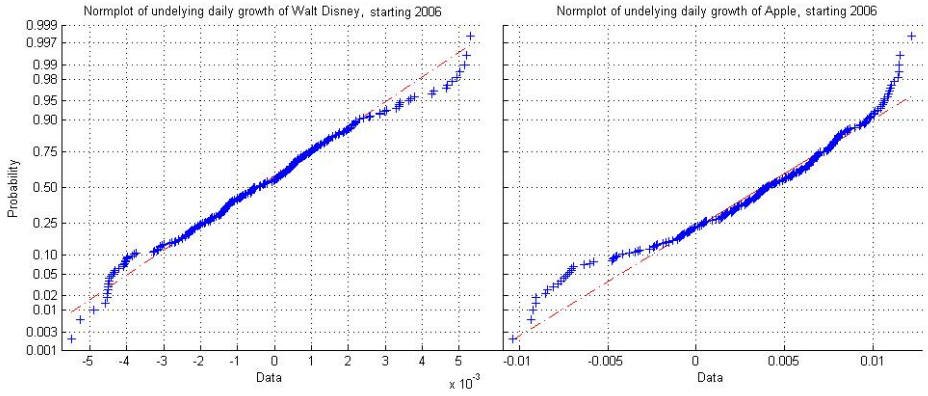

Figure 1: Observations of daily growths from Walt Disney and Apple respectively plottet agains a normal probability distribution. If they agree the dots should be clustered over the diagonal line. As can be seen there are clear deviations from that line.

4.2 Backtesting by Means of Monte Carlo

What really can be estimated from backtesting is at most expected properties of a stategy under certain market conditions and maybe the likeliness of future growths or rates of return. With this in mind the definite character of the result from an ATF-backtest leaves some to be desired in terms of nuances. Monte Carlo-backtesting over a number of time frames provides an estimated probability distribution for the performance of a portfolio strategy. This could make for a refined comparison with other strategies.

Assume we want to know how a portfolio would perform in the market conditions of the previous year. Let the universe representing the market consist of m stocks. A backtest for this purpose could be performed starting by electing a time frame $\mathbf{T}$, covering n days, that has the desired properties. Using data from the time preceding $\mathbf{T}$, estimate the random variables needed to represent the daily growth or daily return of the stocks in the universe.

Suppose we want to backtest a portfolio strategy $P(t_k), \quad k = 1, ..., n$ using a Monte Carlo simulation. It could be performed in accordance with the following steps:

1. Collect historical data for the stocks in the universe and calculate the daily growths or returns. Using these, estimate the expected value and covariance matrix for the daily growth or return of the m stocks in the universe.

2. Choose a multivariate probability distribution with dimension m and the expected values and covariance matrix from the previous step.

3. Generate sufficiently many samples from the distribution to represent the daily growths or returns of the stocks in the universe over $\mathbf{T}$.

4. Calculate the prices of the instruments using the generated random numbers. Iterate steps 2 and 3 to simulate N representations of the universe.

5. Calculate an ATF of the strategy on each of the N simulated representations of the universe over time $\mathbf{T}$.

6. Gather the outcomes of the ATFs at time t_k as a sample space and calculate an estimated probability distribution for the portfolio value $V_{P(t_k)}$.

This distribution can then be compared to the ATF of the straegy over the real historical data.

5 Example: Simulation

In order to explore whether relevant information can be provided by a Monte Carlo Simulation backtest, a simple example is going to be carried out. Several strategies will be backtested and compared in order to provide a more general idea of backtesting by means of Monte Carlo simulation.

5.1 The Universe

The universe of stocks for this simulation was chosen from companies with a large market capitalization. This was chosen partly for reasons of data access, as large companies in general provide more historic data records. Another reason for this was that the simulations here are not taking into account the impact on the market caused by the strategies . Large companies would likely require a larger volume invested for a strategy to have a notable impact on the universe. The S&P100-index was taken as a starting point and the universe was then modified to make the simulations possible in practice. For more details on which stocks are in the universe see Appendix 6.

5.2 The strategies

Three trading strategies will be defined and backtested using Monte Carlo simulation: The Magic Formula, The Random Formula and the Universe Formula. They are all traded in the same volumes. One fifth of a dollar is invested each ten week period until one dollar is invested. The full investment is conluded after the last fifth is invested in the portfolio. the difference between the strategies is determined by the way the stock selection is performed.

The strategies are chosen to fullfil the following criterias as far as possible

- They should both be well defined and possible to formulate in terms of code.
- They should intuitively have different expected performances in order for reasonable and clearly different hypothesises to be formulated.
- They should be relatively uncomplicated to fit into the time frame of the present study.

The universe from where the strategies selects instruments to buy and sell is chosen by

- Diversity of components in order to reflect a market that can be considered relevant for the chosen strategies.
- Data accessibility.
- Reasonable number of components considering the limited number of working hours at the disposal of the present study.

5.2.1 The Magic Formula

The first strategy to be tested is the rather straight-forward portfolio strategy called "The Magic Formula" formulated by Joel Greenblatt in his book "The Little Book that Beats the Market" (2006). In this book Greenblatt argues that MF beats the market on a long enough time line. By this he means that the portfolios holdings need to be bought and held for at least 3 to 5 years in order to do what he claims can be expected from it, that it beats the market.

The strategy gives a method to select stocks from the top performing companies judging by two key numbers. The first one, Return on Capital, intended to reflect the efficiency of the activities performed by the company. The second one, Earnings yield, is intended to reflect the most undervalued company. Return on capital relates the earnings of the company to the resources it employs in order to make them. Earnings Yield relates the earnings to the price that has to be paid in order to own a share of them. The result of this selection aims to be a collection of under valued stocks from companies that use their available resources efficiently.

Since these two key numbers just vaguely reflect all the relevant characteristics of the actual performance and value of a company, the formula is only expected to work on average, calling for a few different stocks to be held at the same time, and that they regularly are replaced with new ones. Joel Greenblatt provides in his book step by step instructions. These assume that one uses the online screener at www.magicformulainvesting.com which is not to be used in this study[3]. The interpretation of the strategy used in this study is the following:

1. Categorize the stocks in the universe according to their performance relative to the key numbers.
2. Choose to buy the top five performers for a fifth of the money intended for the portfolio.
3. Iterate the two preceding steps every 50 days ($\frac{1}{5}$ of a year consisting of 250 trading days) until all the money intended for the portfolio is employed, that is, for one year.
4. Sell each stock after holding it one year and buy a new one from the top five performers in the universe to replace it.
5. Repeat the last step until the portfolio has been held held for at least two or three years.

Another difference from the original formula is that because of limited data access EBITDA has been used instead of EBIT. This means that depreziations

[3]Limited access to the underlying data used by this screener made it natural to construct a screener specifically for this study. See Appendix for the code.

and amortizations are not considered in this study as they would be in the original Magic Formula.

The Magic Formula is well defined, as long as there is access to the information needed. Regarding the stocks as shares of companies it is reasonable to relate the prospects of the company to affect the value of the shares. Because of this, a strategy that takes into account more information about the company in question in order to evaluate which stocks to buy might reasonably be expected, just like Joel Greenblatt suggests, to perform better than a strategy disregarding such information.

5.2.2 The Random Formula

The Random Formula is a version of the Magic Formula that in order to facilitate comparison has been defined exactly like the interpretation of The Magic Formula, made for this study, but disregarding the key number information.

5.2.3 The Universe Formula

The Universe Formula can be seen as the Monte Carlo simulation of the chosen index. This strategy is traded with the same volume as the two former ones in order to be comparable. It buys equally weighted of all stocks and thus represent a simulated universe.

5.3 Assumptions and Decisions

Each simulation is to be carried out using the expected values and the covariance matrix of underlying samples of daily growths or daily returns fitted to a multivariate normal distribution.

They are going to be carried out for underlying daily growth and daily returns respectively, with two different sample sizes, the full preceding year and the preceding half year. Sample spaces are going to be generated with 1000 repetitions and collected for analysis after simulating one, three and five years from the starting time.

The key numbers for the Magic Formula were accessible to a varying degree from different companies. In the best cases they were accessible on quarterly basis, and in the worst, not at all, or just for a few years during the time period subjected for the study. In order to make a somewhat fair interpretation the companies for which no key number data were available, for 20 of the cmpanies that is, were taken out of the universe. For the rest the average of each key number for the underlying sample for each simulation were used. Another possible way to deal with this would have been to randomly generate them as well taking into account their covariance, but since some of the key numbers only were provided for one or two years of the time period, the sample covariation

could not be expected to realistically reflect the covariance of the underlying. A backside of choosing the average is that the keynumbers being constant makes for the Magic Formula to pick the 'best stocks' first, then picking successively 'worse stocks' until the first picks are sold and so once again available to be bought.

5.4 The Construction

In this section the program written and used for the simulations is going to be described. Throughout this section Appendix 2 will be useful, it contains the full code of the program with comments.

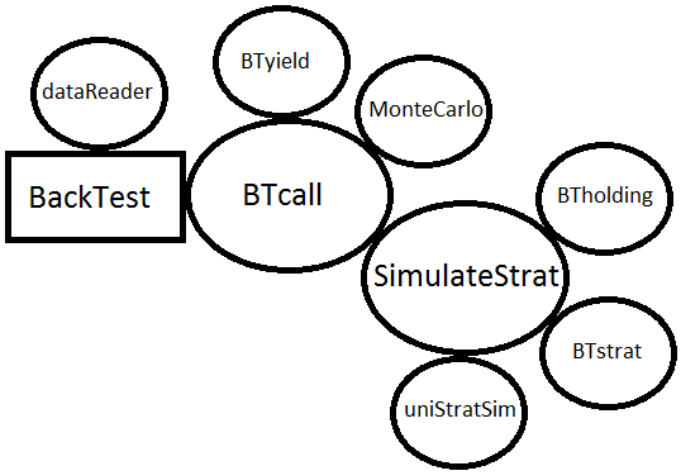

Figure 2: An illustration of the program structure

5.4.1 The Main File

In the main file 'BackTest' starting date, the number of repetitions and sample size is set manually. Running the file it calls the functions 'dataReader' and 'BTcall'. These do what their names imply. 'dataReader' reads the stock price histories for the companies from files in the directory, following this 'BTcall' is called for to call all the other parts of the program.

5.4.2 The Functions

From 'BTcall' several functions are called for starting with BTyield. This one processes the data originating from 'dataReader' and calculates daily growth

and daily return according to the formulas described in section 2.1.3 and estimates the expected values from these.

Next, the covariance matrix is calculated for the daily growth and the daily return respectively. Then the function 'MonteCarlo' is called for in order to generate the simulated universes.

The last function called for by 'BTcall' is 'simulateStrats'. This function compiles, partly by means of other functions, the portfolios of the strategies from the simulated universes. The other major functions it uses that has been written for this study are, 'BTholding', 'BTstrat' and 'uniStratSim'. It starts by reading the key numbers for the Magic Formula using the function 'orgKeyNr'. 'orgKeyNr' reads pre organized key number data from another folder. 'simulateStrats' calculates both the average of the key numbers as well as generates random numbers to represent them. For the simulations however, only the former was used. They are used calling the function 'BTholding' that selects the stocks to be contained in the portfolios over time. These are selected as column indexes, since the different stocks are organized as columns in 'the universe matrix'.

'BTholding' is using a built in sort function in matlab in a few steps to categorize the stocks for the Magic Formula in order to make stock selections[4]. The Random Formula has stocks selected in much the same way except for the selection being based on chance. The stock selections that are returned to 'simulateStrats' containing stocks going in and out of the portfolio each ten week period of the simulation. Using the function 'BTstrat' a sample path is created for each of the simulated universes. In order to simulate the Universe Formula a modified version of 'BTstrat' is used, namely 'uniStratSim'. It was created to make it have the same traded volume as the two strategies it is to be compared to.

From these 1000 sample paths of each trading strategy expected value and standard deviation were calculated after one, three and five years.

[4] An explicit explanation of how this is done can be found in the comments of the code of 'BTholding' in Appendix 2, section 9.7, row 36

6 Analysis

Throughout the analysis the figures in Appendix 1 will come in handy. A comparison between the results using daily growths or daily returns as underlying data for the simulations did not suggest any significant difference between the two. Using half a year of daily growths or daily returns generally resulted in worse predictions of the simulation. The analysis will focus on the results based on underlying data of one year of preceding daily growths, which will be the case in the upcoming discussion if nothing else is mentioned. For natural reasons the simulations can only be compared with the actual development[5] one year into the future starting in January of 2011 and only three years for the ones starting in January of 2009.

6.1 General Trends

The development one year after the starting point of the backtest in January of 2006, 2009 and 2011 seem to reveal some gains with using monte carlo methods for backtesting compared to the ATF. An ATF starting 2006 and 2011 falls well within the sample spaces of the simulations but not for the one starting in January of 2009 (See figures in section 8.2.1 and 8.2.3). This is probably due to the high volatility of the market during 2008 and 2009.

For 2006 the result seems quite intuitive. More precisely the simulations starting in January of 2006 has the expected daily growth taken from the year of 2005. Looking at the development of the universe over 2005 and 2006 they are rather alike. In fact just assuming the ATF for 2005 of the Universe Formula to forecast the development of 2006 would have been a pretty acurate prediction, giving 1.1156 dollars. Even though it is actually closer to the ATF of 2006 than the expected value of the simulation (Which still is well within two standard deviations from the ATF), it still doesn't provide any notion of the variation span of possible outcomes.

Looking at the results of the simulations starting in January of 2011 the standard deviations of all the strategies are greater than the ones for the simulations starting in January of 2006 (See the columns 'std' in the tables of section 8.2.1 and 8.2.3). Again, looking at the development of the universe for the years 2010 and 2011 they also look rather alike in terms of volatility. An ATF over 2010 would however correspond poorly to 2011 in terms of portfolio value for the different strategies. The Simulation on the other hand, based on data from 2010 gives a fairly just prediction.

For the simulation starting in January of 2009 the distributions for all the strategies deviate from the actual development during the time period (See figures in section 8.2.2). It is clear that the simulations only carry information

[5]The 'actual development' is represented by an ATF, that is, the strategies applied to the actual historical price development over the time frame forecasted by the simulation.

about the history before they started. The simulation starts at a radical trend shift. Thus, using historical data yields a negative expected value of growth for most of the stocks in the universe while the actual development changed to a positive growth for most of the stocks. This simulation method seems unable to comprise major trendshifts over time. This will later be discussed as a prospect for future studies. Looking three or five years further the simulations fall quite far off from the result of the ATF for all starting points and strategies (See figures in section 8.3 and 8.4).

The table below shows the differences between the expected values of the distributions for the portfolio values and the corresponding ATF after one, three and five years. The difference is expressed in absolute terms of standard deviations of the portfolio distributions. UF1 stands for the Universe Formula after 1 year, RF3 stands for the Random Formula after 3 years and MF5 stands for the Magic Formula after five years and so on.

Start	UF1	RF1	MF1	UF3	RF3	MF3	UF5	RF5	MF5
Jan. 2006	1.7292	0.10633	1.8507	12.557	11.606	9.809	9.8724	5.0715	6.5676
Jan. 2009	16.146	14.727	13.568	31.939	35.381	29.632	NaN	NaN	NaN
Jan. 2011	1.8368	1.7801	0.079756	NaN	NaN	NaN	NaN	NaN	NaN

Figure 3: `The differences between the expected values of the distributions for the portfolio values and the corresponding ATF. They are expressed in terms of standard deviations of the portfolio value distributions.`

As can be seen in the table, an outcome equal to the ATF is at least five standard deviations away from the expected portfolio value in all cases on a time horizon of three or five years. The probability of an outcome deviating five standard deviations or more from the sample mean is 0.00005733%. Consequently this particular backtesting method by means of Monte Carlo simulation does not seem to give any informative distribution on a time horizon of three or five years. The seeming inabillity of the simulation method to comprise major trendshifts over time might contribute to this, as the number of trend shifts over time can be expected to increase over a larger time span.

6.2 The Strategies

The Magic Formula is a long-term trading strategy, meant to be maintained over at least three to five years. Because of that a comparison of the strategies over shorter time frame such as one year could not give any relevant conclusion. It would be better to compare the strategies after three or five years. Since this particular simulation method does not seem to provide informative distributions after such periods of time it would be fruitless to point out a 'best strategy'[6].

[6] "Best" in this connection means the strategy that performs best compared to the Universe Formula, chosen to reflect the universe.

6.3 Thoughts and Comments

I believe there is a more simple and maybe better way of performing the simulations given the assumption of the underlying daily growths being normally distributed. It is to add the distributions of the underlying daily growths for the different stocks held in the portfolio when rebalancing it, creating a portfolio specific distribution. From this distribution it would be possible to simulate the daily growth until it is time to rebalance the holdings again. The reason that I chose to build the program and the simulation the way I did is that I aimed to use the real distribution of the underlying instead of fitting the normal distribution to it. Gathering usable data took more of my time than I had expected and I simply hadn't enough time to implement the second distribution. Hopefully simulating the way I simulated in this study might be a step stone for a future study, relating to my code and implementing a better distribution for the underlying.

7 Conclusions and Ideas

This section is divided in two sections. The first sums up the present study. The second one is about ideas for endeavouring further into related subjects.

7.1 Conclusion from the study

This study indicates that backtesting through simulation of a trading strategy by means of Monte Carlo might give relevant information about it derived from historical developments. Backtesting this way generates a clear distribution of different possible outcomes of the portfolio. The method employed by this study does not seem to give a fair representation of possible over all trendshifts over time of the market. The results of the study also suggests this form of Monte Carlo backtesting to be more suitable for short term trading strategies. The number of trend shifts over time can be expected to decrease over a shorter time span. Consequently a short term trading strategy might be less sensitive to the kind of events that these simulations seem unable to comprise.

7.2 Prospects for Future Studies

The normal distribution does not provide a perfect fit for the underlying samples of daily growths. While the normal distribution makes for easier calculation it does not provide a fair simulation given the underlying data provided by reality. Prospects for future studies would be to investigate other distributions that might provide a better fit. What has caught my interest through out this study is the prospect of using the actual distribution of the underlying daily growths

or returns. Here follows an explanation of a way to randomly draw a sample of one days daily growth for a collection of stocks from the distribution given by an underlying sample of historical daily growth.

Given the historical daily growth $G_l(t_k)$, $l = 1, 2, ..., m$, $k = 1, ..., n$ for a collection of m stocks over a specified historical time frame $\mathbf{T}$ of n days. Let $\mathbf{G}(t_k)$ be interpreted as an m-dimensional vector with the daily growths of each stock at day t_k so that in $\mathbf{G}(t_k)$, the l^{th} element is $G_l(t_k)$, the growth of stock l at day t_k.

1. Create an interval $D \subset \mathbf{R}$ between the single largest and the single smallest daily growth of all the stocks and all the days in $\mathbf{T}$. Divide D into p subintervals of equal length $\Delta_j \subset D$, $j = 1, ..., p$ so that $\Delta_1 \cup ... \cup \Delta_p = D$. p can be seen as the resolution or precision of the emerging interpretation of the underlying distribution of historical daily growths $\mathbf{G}(t_k)$, $k = 1, ..., n$.

2. Let $\mathbf{I}$ be the list of all possible combinations of m elements from $\{1, ..., p\} \subset \mathbf{N}$, assigning each combination an index $i = 1, ..., p^m$ (as p^m is the number of all possible combinations of the elements in $\{1, ..., p\}$). This way $\mathbf{I}(i)$ is a unique combination of m elements from $\{1, ..., p\} \subset \mathbf{N}$ that in code could be formulated as an m-dimensional vector with the l^{th} element $j \in \{1, ..., p\} \subset \mathbf{N}$ that correspond to the subinterval Δ_j.

3. Let $N(i) \in \mathbf{N}$ be the number of vectors $\mathbf{G}(t_k)$, $k = 1, ..., n$ such that their elements fall within the subintervals Δ_j, $j \in \{1, ..., p\} \subset \mathbf{N}$ corresponding to the elements in the vector $\mathbf{I}(i)$. This way $\frac{N(i)}{n}$ can be seen as the probability for the daily growth of the m stocks to simultanously fall into the corresponding combination of m subintervals given by the elements of $\mathbf{I}(i)$.

4. Draw a number i from $\{1, ..., p^m\} \subset \mathbf{N}$ using the probability distribution given by $\frac{N(i)}{n}$ over $i = 1, ..., p^m$. Generate numbers g_l, $l = 1, ..., m$ from a univariate distribution on the subinterval corresponding to the l^{th} element in $\mathbf{I}(i)$. The vector $\mathbf{g} = [g_1 \quad ... \quad g_m]$ could be used as a randomly drawn sample of daily growth for the m stocks.

This way random numbers are drawn from the underlying distribution. To some extent it also preserves the market trends from within the underlying collection of instruments, at least on a daily basis. A backside of using such a distribution for generating sample paths is that the possible values are limited to the ones that have appeared in the sampled history. The future might of course deviate from the historical span of variation.

No matter the underlying distribution, possible market trends over time are not considered by the above simulation methods. Fluctuations might occur in a simulation with many consecutive days of growth deviating from average all up or all down. An event of many consecutive days of extreme deviation from

average growths is however very unlikely. Economic crisis, bubbles that burst or occur do not seem to have their proper chance to appear in these simulations. Looking at plotted time series of stocks the extreme daily events seemingly appear in clusters. With this in mind some mechanism to deal with trends over time could probably provide a simulation better reflecting reality.

One could for example consider resampling from the simulated history so that a downward trend rebalances the probability distribution for the coming days. This could be thought of as a way of representing trends over time on the market. This method might not reflect the actual behaviour of trends on the market. Despite that, many repeated simulations could provide a more realistic sample space since the extreme deviations from the expected average path based on historical data might be expected to increase.

I had thoughts, inspired by my mentors, of implementing the rebalancing mechanism in my simulations. In that case I would have performed the simulations in steps, say 20 days at the time. After 20 days, calculating the expected value and the covariance based on the past 250 days. The first effect of this, that comes into my mind is that the ratio of 'real' data in the underlying will decrease as the simulation advances. On the other hand the most recent 'real' data will be reused more times the closer to the starting point it is. Had I had much more time for this study I would probably have implemented it. The reason that I did not put a higher priority on it was because on a long time frame, I don't believe such simulated trends would carry information about the behaviour of the underlying. The gain, I believe, would be a greater spread of the resulting distributions, but these would be based more on randomness than on the underlying historical data.

Another prospect of future studies is to do a study similar to this one but with short term strategies. On a short time horizon the problem of major trendshifts might be less prominent and so the Monte Carlo backtesting method described here could be more useful. In section 2.3 an ATF is suggested to be useful to study the short-term performance of a strategy, before the over-all market reacts to an event provoked by the strategy. For such a study historical data would have to suggest that the market generally reacts to that market event with some delay that can be somewhat specified. Such a situation might be interesting to study using the methods employed in the simulations of this study.

8 Appendix 1: Results

The following results are based on daily growths from the year preceding the starting time of the simulation.

8.1 The development of the stock universe

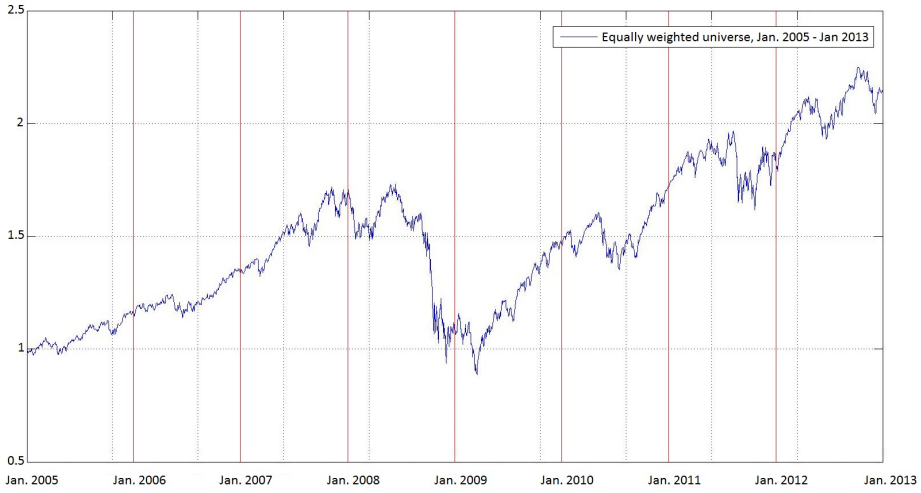

Figure 4: The development of the stock universe

8.2 After One Year

8.2.1 starting in January of 2006

The three plots show the sample spaces generated by the simulations after one year for each strategy. The table below is giving the corresponding figures of expected value and the standard deviation of the distributions as well as the actual outcome of the different portfolios represented by an ATF.

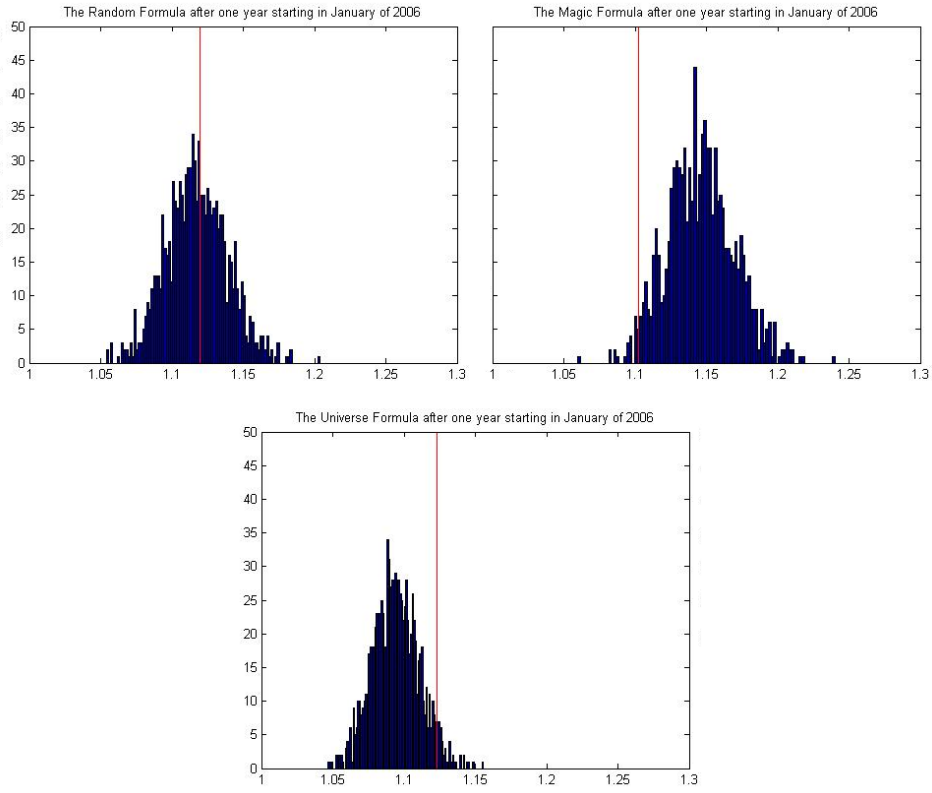

Figure 5: The three strategies after one year, starting January of 2006. The red line indicates the result of the corresponding ATF.

	ATF	mu	std
The Universe Formula	1.1231	1.0945	0.016539
The Random Formula	1.1196	1.1173	0.021631
The Magic Formula	1.1024	1.146	0.023559

8.2.2 starting in January of 2009

The three plots show the sample spaces generated by the simulations after one year for each strategy. The table below is giving the corresponding figures of expected value and the standard deviation of the distributions as well as the actual outcome of the different portfolios represented by an ATF.

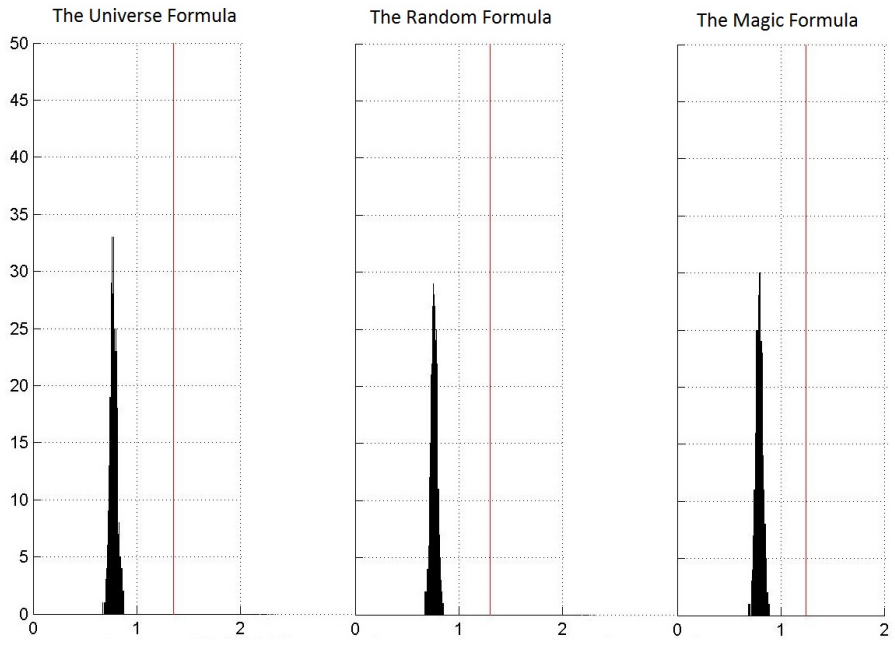

Figure 6: The three strategies after one year, starting January of 2009. The red line indicates the result of the corresponding ATF.

	ATF	mu	std
The Universe Formula	1.2957	0.7815	0.031846
The Random Formula	1.235	0.76258	0.032079
The Magic Formula	1.2598	0.79856	0.033995

8.2.3 starting in January of 2011

The three plots show the sample spaces generated by the simulations after one year for each strategy. The table below is giving the corresponding figures of expected value and the standard deviation of the distributions as well as the actual outcome of the different portfolios represented by an ATF.

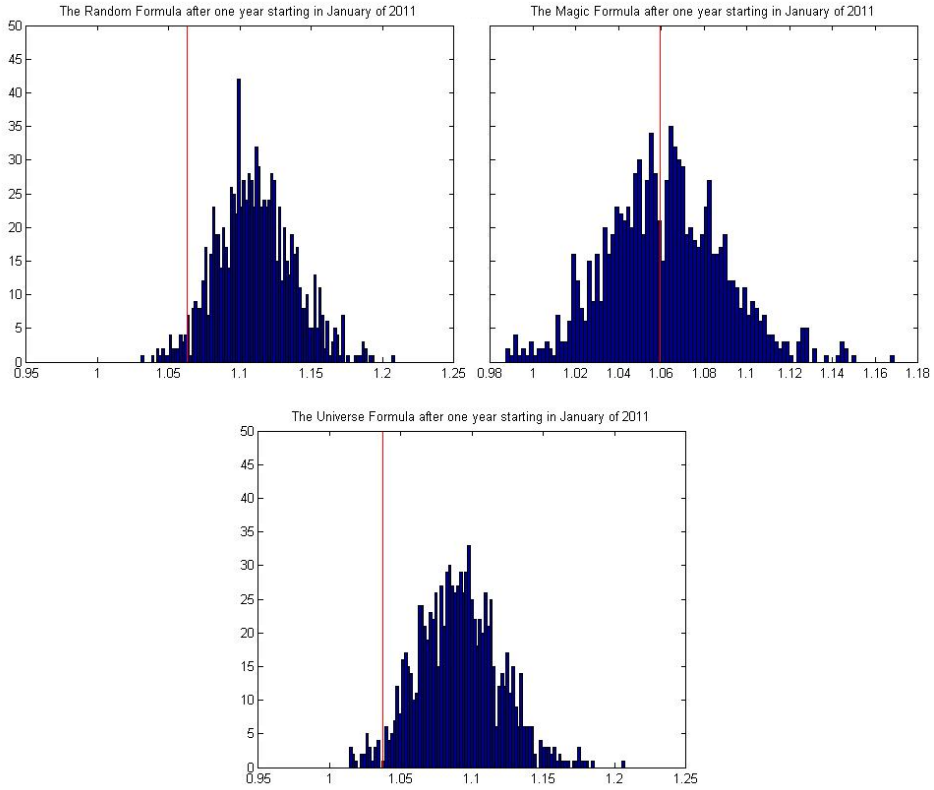

Figure 7: The three strategies after one year, starting January of 2011. The red line indicates the result of the corresponding ATF.

	ATF	mu	std
The Universe Formula	1.0374	1.0907	0.029018
The Random Formula	1.0626	1.1104	0.026852
The Magic Formula	1.0594	1.0616	0.027584

8.3 After Three Years

	ATF	mu	std
The Universe Formula	0.86791	1.7603	0.071066
The Random Formula	0.88117	1.5648	0.058903
The Magic Formula	0.91357	1.7183	0.08204

Figure 8: Starting January of 2006

	ATF	mu	std
The Universe Formula	1.5713	0.41545	0.036189
The Random Formula	1.5468	0.33407	0.034276
The Magic Formula	1.5665	0.40667	0.039141

Figure 9: Starting January of 2009

8.4 After Five Years

	ATF	mu	std
The Universe Formula	1.3969	3.9968	0.26335
The Random Formula	1.4911	1.9814	0.096678
The Magic Formula	1.5006	2.8597	0.20694

Figure 10: Starting January of 2006

9 Appendix 2: The Code

9.1 BackTest

```
%               Monte Carlo Backtesting
%
%This is the main file for a for the Monte Carlo backtesting.
%It is devided in two sections, the first one gathers data,
%the second one performs the backtest.
%
%The result of running both section are two csv files in the
%folder 'sample spaces'. They each contain sample spaces for
%the strategies, based on daily returns and daily growths
%respectively. Sample spaces are provided for ten half year
%steps from the starting date. It also contains the ATF result
%at each half year step.
%
%                    Section 1.


%Format of the dates in the time series data that is to be loaded.
format='yyyy mm dd';

%Calls the function dataReader() which returns a price series for
%each stock. These are organized as column vectors in a matrix
%'data' with the first column containing dates.
%The earliest date possible to start a simulation from, and the
%last for which data is available are also returned.
old=cd('data');
[data, minStart, maxStop]=dataReader(format);
cd(old);

%Prints the accessible timeframe that allows for estimations to
%be made from the year before.
Tidsintervall=[minStart ' till ' maxStop]



%%
%                          Section 2.

%The desired number of repetitions of the simulation
rep=1000;

%Set the starting year between 2006 and 2012.
setYear=2006;
%To start from 2013 see comments below.

%The sample size, 1 => 1/2 year, 2 => 1 year
sampleSize=1;


yearIndex=setYear2005;%This number is used to set the starting day
%the data of the key numbers for the Magic Formula.
Y={'2006', '2007', '2008', '2009', '2010', '2011', '2012'};
```

```
52 Y=Y{yearIndex};
53
54 fileName={'Half a', 'One'};
55 nrOfDays=[125 250];
56
57 tit=fileName{sampleSize};
58 smplSize=nrOfDays(sampleSize);
59
60 %Sets the desired distribution to be fit to the daily growth
61 %or daily returns.
62 Dist='norm';
63
64 %Specifies the day when the simulation should start and when it
65 %should end end and converts it to a serial number. It should be
66 %set to the first trading day of a year.
67 begin=datenum([Y ' 01 03'],format);
68 keyNrStart=(yearIndex 1)*250;%This is used to set the key number
69 %data for the magic formula to the right starting date.
70
71 %                      2013
72
73 %For a simulation based on data during 2012, use 20121228 and
74 %uncomment the next two lines:
75 % Y='2013';
76 % keyNrStart=1750;
77
78 %
79
80 %The following code looks for the specified beginning dates
81 %and determines wether it is missing from the time series,
82 %a message is returned if a new date needs to be set above.
83 initial=sum(data(:,1)==ones(size(data(:,1)))*begin);
84
85 if initial==0
86     date='Prices are missing for that starting date.'
87     break;
88 else
89     date='ok'
90 end
91
92 %Converts 'begin' to a row index in the data matrix.
93 begin=find(data(:,1)==begin,1,'first');
94
95
96 %To perform the simulation the function BTcall() is used.
97 BTcall(data, Dist, rep, begin, smplSize, keyNrStart, tit, Y);
```

9.2 dataReader

```
1
2  % The input argument 'format' is the specific format of
3  % the dates in the csv files in the directory.
4
5  % dataReader() returns the data located in the directory
6  % reformatted into a single matrix 'data'. In 'data' the
7  % daily stock prices are organized columnwise for the
8  % different stocks. The first column in 'data' contains
9  % the dates of the price series for the stocks.
10 % dataReader also returns the time span over which simulations
11 % are possible in the variables 'minStart' and 'maxStop'.
12
13
14 function [data, minStart, maxStop]=dataReader(format)
15
16 %All the csv files in the current directory, supposedly the
17 %stock price time series are saved in to 'files'.
18 files=dir('*.csv');
19
20 %These stocks, that I have not complete key number data for,
21 %are excluded.
22 files([5 6 13 15 17 48 51 53 55 63 75 78 80 81 82 85 86 93 97])='';
23
24 %'inst' get the length of the vector 'files' assigned to it,
25 %that is the number of stock in the universe.
26 inst=length(files);
27
28 %Two empty row vector with inst number of elements are
29 %createt to later contain the first and last value for
30 %each stock.
31 startNr=zeros(size(files));
32 slutNr=zeros(size(files));
33
34 %The following for loop specifies the first and last date
35 %that has a price assigned to it and puts them in the
36 %vectors startNr and slutNr.
37 for i=1:inst
38     kurs=importdata(files(i).name);
39     slutNr(i)=datenum(kurs.textdata(2,1), format);
40     startNr(i)=datenum(kurs.textdata(end,1),format);
41 end
42
43 %Finds the latest start date and the first ending date
44 %and creates a time series of serial date numbers for
45 %all the days in between. In this series there are as
46 %well weekends and other trade free days.
47 slut=min(slutNr);
48 start=max(startNr);
49 tid=(start:slut)';
50
51 %Creates a matrix later to be filled, each column
52 %with the daily adjusted closing prices of one stock
53 %in the universe.
54 univers=zeros(length(tid),inst+1);
```

```
55 univers(:,1)=tid;
56
57 %For each stock in the univers, the loop matches
58 %each date in 'tid' to a value of the daily adjusted
59 %closing prices(that happens to be the 6th one
60 %from the current source)
61 for i=1:inst
62     post=importdata(files(i).name);
63     fileDat=post.textdata(2:end,1);
64     file=post.data(:,6);
65     for j=1:length(fileDat)
66         nr=datenum(fileDat(j),format);
67         day=find(univers(:,1)==nr, 1, 'first');
68         univers(day,i+1)=file(j);
69     end
70 end
71
72 %The time series now has rows containing zeros,
73 %since there are trade free days, these row vectors
74 %are eliminated through the next line of code.
75 univers(any(univers==0,2),:)=[];
76
77 %The universe ie returned as 'data' and the first
78 %column is still a time series of serial date numbers.
79 data=univers;
80
81 %270=one months lag for the calculation of daily growth
82 %+ one year of preceeding underlying data sample=20+250=270
83 minStart=datestr(univers(270,1));
84 maxStop=datestr(univers(end,1));
85
86 end
```

9.3 BTcall

```
% BTcall() preforms the simulations by means of other functions.
% All the input arguments to the functions are therefor past on
% as input arguments to other functions.


function []=BTcall(data, Dist, rep, begin, smplSize, keyNrStart, ...
    tit, Y)


%BTyield basically prepares information for MonteCarlo.m
%that will produce the simulation a few lines down. It uses
%the histrorical prices of the stocks in the parameter 'data',
%the chosen distribution in 'Dist' and sthe starting date
%'begin'. The starting date is relevant since the expectance
%of growth is calculated from the one year history prior to
%the simulation to come. 'G' and 'R' are matrixes containing
%the calculated daily growth and return respectively. 'drift' and
%'ExpR' are vectors with the expected values for the growth
%and the return respectively.
[G, drift, R, ExpR]=BTyield(data, Dist, begin, smplSize, Y);

%The covariance matrixes are calculated using
%the built in function cov()
Qg=cov(G);
Qr=cov(R);


% This function provides the number of simulated
%universes requested in BackTest.m using the covariance
%matrixes 'Qg' and 'Qr', the calculated expected values,
%the number of repetitions and the beginning date for
%the simulation at hand.
[simuleringG, simuleringR]=MonteCarlo(Qg, drift, rep, Qr, ExpR);


%simulateStrats() creates the paths of the strategies, which
%together
titleG=[Y ' ' tit ' year of preceeding daily growths'];
titleR=[Y ' ' tit ' year of preceeding daily returns'];

figure('Name', titleG)
simulateStrats(data, simuleringG, smplSize, keyNrStart, begin, ...
    rep, titleG);
figure('Name', titleR)
simulateStrats(data, simuleringR, smplSize, keyNrStart, begin, ...
    rep, titleR);



end
```

9.4 BTyield

```
%The main idea with this function is to produce the
%vector 'drift' with the assessed drift for each of
%the stocks in the universe and also the vector 'ExpR',
%which contains the expected daily return for each stock.
function [growth, DRIFT, R, ExpR]=BTyield(dat, Dist, begin, ...
    smplSize, Y)

%the first column in the universe, that is, the serial date numbers,
%are saved in 'indexDat'. After that a new matrix is created
%being the original universe without the columnvector of serial
%date numbers. This separation is made for practical reasons.
indexDat=dat(:,1);
dat=dat(:,2:end);

%The number of columns in the matrix of the stock prices is
%the same as the number of stocks, why, 'inst', short for
%instruments is the number of instruments in the universe.
%n=20 is the number of days that form the basis of the assumed
%daily growth. To avoid discrepencies due to daily fluctuations
%that do not represent the general changes on the market, an
%average is taken from the n days before each day and the
%growth used for further calculations is thus a more stable
%one than the original.(Reference [8])
inst=length(dat(1,:));
n=20;

%An empty cell array of inst elements is created to contain the
%distribution of growth for each stock in the universe.
PDG=cell(1,inst);


%Two vectors are created to contain the volatility and the
%expected value of growth respectively.
vol=ones(size(PDG));
mu=vol;

%Daily growth is calculated as the logaritmic difference
%between consecutive days in the historical prices.
G=log(dat(2:end,:)) log(dat(1:end 1,:));

%Selecting daily growth from a time period of one year
%plus 20 days to provide data for the first 20 days
%of that year when calculating the average daily growths.
sample=begin (smplSize+n):begin;

Gsim=G(sample,:);

%Anorher vector is created to contain the daily
%growth averages that are to be used for calculating
%the estimations of the parameters.
Gstat=zeros(smplSize,length(dat(1,:)));

%calculate the moving average for one year.
for i=n+1:smplSize+n
```

```
54      Gstat(i n,:)=(1/n)*sum(Gsim(i n:i,:));
55  end
56
57  %Fits a distribution to each series of growths and assigns
58  %a distribution from which parameters are estimated.
59  for i=1:inst
60      PDG{i}=fitdist(Gstat(:,i),Dist);
61      vol(i)=sqrt(PDG{i}.sigma);
62      mu(i)=PDG{i}.mu;
63  end
64
65
66  %Estimates expected value directly from data, this is not
67  %in use at the moment.
68  EG=ones(1,length(inst));
69  for i=1:length(EG)
70      EG(i)=sum(Gstat(:,i))/length(Gstat(:,i));
71  end
72
73
74  volatilitet=vol;
75  DRIFT=mu;
76  growth=Gstat;
77  PDg=PDG;
78  data=[indexDat dat];
79
80
81  %Daily return is now calculated from the data.
82  Ret=(dat(2:end,:) dat(1:end1,:))./dat(1:end1,:);
83
84  %Selecting daily return from a time period of one year
85  %plus 20 days to provide data for the first 20 days
86  %of that year when calculating the average daily return.
87
88  Rsim=Ret(sample,:);
89
90  %Anorher vector is created to contain the daily
91  %return averages that are to be used for calculating
92  %the estimations of the parameters.
93  Rstat=zeros(smplSize,length(Ret(1,:)));
94
95  %calculate the moving average for one year.
96  for i=n+1:smplSize+n
97      Rstat(i n,:)=(1/n)*sum(Rsim(i n:i,:));
98  end
99
100 PDR=cell(1,inst);
101 vol=ones(size(PDR));
102 mu=vol;
103
104 %Fits a distribution to each series of growths and assigns
105 %a distribution from which parameters are estimated.
106 for i=1:inst
107     PDR{i}=fitdist(Rstat(:,i),Dist);
108     vol(i)=sqrt(PDR{i}.sigma);
109     mu(i)=PDR{i}.mu;
110 end
```

```
111
112 Rvol=vol;
113 ExpR=mu;
114 PDr=PDR;
115 R=Rstat;
116
117 %Saves the returns and growths in the folder 'underlying data'.
118 older=cd('underlying data');
119 old=cd(Y);
120
121 csvwrite('growth.csv', growth);
122 csvwrite('return.csv', R);
123
124 cd(old);
125 cd(older);
126
127
128
129 end
```

9.5 MonteCarlo

```
1
2  %MonteCarlo() returns 'simuleringG' and 'simuleringR', cell arrays
3  %containing number of simulations of the universe.
4  %Each simulation is a matrix of the same size as 'data'.
5  %For this purpose it uses the covariance matrix 'Q',
6  %'drift' and 'ExpR' that are the expected values,
7  %the number of simulations 'rep'.
8
9
10 function [simuleringG, simuleringR]=MonteCarlo(Qg, drift, rep, ...
       Qr, ExpR)
11
12
13 %The length of the simulation span is set to five years,
14 %saved in the variable 'tid'. The number of instruments
15 %in the universe is saved in the variable inst.
16 tid=1250;
17 inst=length(drift);
18
19 %A container is created to gather the simulations before returning
20 %them in 'simulering'
21 sim=cell(1,rep);
22
23 for i=1:rep
24     %A built in function is called for to return a matrix with
25     %random elements of the multivariate normal distribution with
26     %covariance matrix 'Qg' and the expected value vector 'drift'.
27     %The size of the returned matrix is 'tid' times the dimension
28     %of Qg, which is the same as 'inst'.
29
30     delSim=mvnrnd(drift, Qg, tid);
31
32     %The values of the first row (day) is set to be one, that is
33     %for the stocks to be comparable. The prices are thereafter
34     %calculated using 'delSim'.
35
36     delSim(1,:)=ones(1,inst);
37     for j=2:tid
38         delSim(j,:)=delSim(j-1,:).*exp(delSim(j,:));
39     end
40     sim{i}=delSim;
41 end
42 simuleringG=sim;
43
44 %
45
46 %Simulating the stock yield by means of daily return. It is ...
       performed
47 %much like the above.
48 for i=1:rep
49
50     delSim=mvnrnd(ExpR, Qr, tid);
51
52     delSim=delSim+ones(size(delSim));
```

```
53
54      delSim(1,:)=ones(1,inst);
55      for j=2:tid
56          delSim(j,:)=delSim(j-1,:).*delSim(j,:);
57      end
58      sim{i}=delSim;
59  end
60
61  simuleringR=sim;
62
63  end
```

9.6 simulateStrats

```
% simulateStrats() is called for to perform the simulations,
% partly by means of other functions. The input arguments are

% 'data' which is the matrix of the underlying price series
% used here to perofrm the ATFs,

% 'simulering' which is an array of 'rep' simulated universes,

% 'begin' which is the starting date of the simulation in the
% form of a row index and

% 'tit' which is a string to be used for the title in the
% plot made at the end of the program.

% 'smplSize' is a variable containing the number of days used
% as sample for the simulation.

% 'keyNrStart' is a row index for the key number data
% corresponding to the date given implicitly by 'begin'.


function []=simulateStrats(data, simulering, smplSize, ...
    keyNrStart, begin, rep, tit)

%This code line is used to make the simulated universes and the key
%number data agree datewise.
initial=500+keyNrStart;


%The key number data are calculated by the function orgKeyNr()
old=cd('MF Key Numbers');

[RoC, EY]=orgKeyNr();

cd(old);

%Calculating average over the sample period
averageRoC=(1/smplSize)*sum(RoC(initial smplSize:initial,:));
averageEY=(1/smplSize)*sum(EY(initial smplSize:initial,:));

%Calculating key numbers as the average for the sample period
avRoC=bsxfun(@times, ones(size(RoC)), averageRoC);
avEY=bsxfun(@times, ones(size(EY)), averageEY);

%               In case of simulated key numbers

%Calculating the covariance matrixes
QRoC=cov(RoC(250:500,:));
QEY=cov(EY(250:500,:));

%Generates the key numbers from a multivariate normal distribution
rndRoC=mvnrnd(averageRoC, QRoC, 250);
rndEY=mvnrnd(averageEY, QEY, 250);

```

```
54
55  %
56
57
58
59
60  %BTholding is called for with time series of yearly
61  %Return on Capital, 'RoC', and Earnings yield for each stock.
62  %Stock selections are returned in terms of column indeces.
63  %These indeces indicates five stocks that go in to the portfolio
64  %and five stocks that goes out of it for every ten week period.
65  [magi, slump]=BTholding(avRoC, avEY);
66
67  %With BTstrat() specific simulations are generated from the
68  %universe through the stock selections generated above.
69  %'MF' and 'RF' are each cell arrays with one cell
70  %for each repetition 'rep', now implied by the number of
71  %cells in 'simulering'.
72  MF=BTstrat(simulering, magi);
73  RF=BTstrat(simulering, slump);
74
75
76  %  Simulating the equally weighted univers portfolio strategy
77
78
79
80  %For the strategies to be comparable the function uniStratSim() ...
        is used.
81  Uni=uniStratSim(simulering);
82
83
84
85
86  %The date column is taken off the data matrix, it is added again ...
        below.
87  dat={data(begin:end, 2:end)};
88
89  %ATF is performed on the index
90  UniAtf=uniStratSim(dat);
91
92  %ATF is performed on the Magic Formla
93  MfAtf=BTstrat(dat, magi);
94
95
96  %ATF is performed on the Random Formula
97  RfAtf=BTstrat(dat, slump);
98
99
100 %PLOTTING THE RESULTS
101 toPlot={MF, RF, Uni, MfAtf, RfAtf, UniAtf};
102 atTime={125, 250, 375, 500, 625, 750, 875, 1000, 1125, 1250};
103
104 sampleSpaces(toPlot, atTime, rep, tit)
105
106 clf
107 subplot(2,3,1)
108 axis([1 1250  0.5 8])
```

```
title('The Magic Formula over five years with one invested dollar')
grid minor
hold on
plot(MF(1:1250,:))
hold on
plot(MfAtf, 'LineWidth',0.75)
legend('ATF')

subplot(2,3,4)
axis([0 8 0 100])
title('The sample space after five years for the Magic Formula')
hold on
hist(MF(1250,:),100)

subplot(2,3,2)
axis([1 1250  0.5 8])
title('The Random Formula over five years with one invested dollar')
grid minor
hold on
plot(RF(1:1250,:))
hold on
plot(RfAtf, 'LineWidth',0.75)
legend('ATF')

subplot(2,3,5)
axis([0 8 0 60])
title('The sample space after five years for the Random Formula')
hold on
hist(RF(1250,:),100)
hold on

subplot(2,3,3)
axis([1 1250  0.5 8])
hold on
title('The Universe Formula over five years with one invested ...
    dollar')
hold on
plot(Uni(1:1250,:))
hold on
grid minor
hold on
plot(UniAtf, 'LineWidth',0.75)
legend('ATF')

subplot(2,3,6)
title('The sample space after five years for the Universe Formula')
axis([0 8 0 60])
hold on
hist(Uni(1250,:),100)
hold on
end
```

9.7 BTholding

```
%This function generates the selection of stocks, in terms of
%indexes over time, that are to be contained in the portfolios.
%Two selection 1 x 2 cell arrays are returned, each containing
%matrixes for stocks to be sold from the portfolio at the
%beginning of each ten week period and one for the stocks
%that are to be bought into the portfolio at the same time.
%The arrays are named 'magi' for the Magic Formula, and 'slump'
%for its random counterpart. It is only 'magi' that requires input
%to the function and this input is in form of w x inst where w
%is at least the number of ten week periods that covers the whole
%simulation time frame, and inst is the number of stocks
%in the universe.

function [magi, slump]=BTholding(RoA, PE)

% n is set to be the shortest length of the column vectors
%of the input matrixes. After that both the matrixes are
%set to have the same size.
n=min(length(RoA(:,1)),length(PE(:,1)));
RoA=RoA(1:n,:);
PE=PE(1:n,:);

%The number of stocks is saved into 'inst'.
inst=length(RoA(1,:));

%The number of ten week periods is determined and saved into 'tenW'.
tenW=length(RoA(:,1));

%A matrix is created for the purpose of containing the buy matrix
%for the two output arrays.
addStock=zeros(5,tenW);

%This for loop categorizes the stocks by means of the input and
%creates the buy matrix for the Magic Formula.
for i=1:tenW
    %sort() arranges two vectors, the first one with the input
    %vector elements ordered descendingly and the second one with
    %the former index of each value in the first vector.
    %Since the stocks are identified by their column index
    %RoAi and PEi identifies which stock is first, second
    %and third best etc acording to the P/E ratio or
    %Return on Asset.
    [RoAy, RoAi]=sort(RoA(i,:),'descend');
    [PEy, PEi]=sort(PE(i,:),'descend');
    %Now sorting the identification vectors in ascending order
    %'j' and 'k' are memory vectors keeping the hierarchy of each
    %vector from the sorting regarding P/E ratio and RoA while
    %sorted in the original order of the stock universe.
    [y, j]=sort(RoAi,'ascend');
    [x, k]=sort(PEi,'ascend');
    %Adding j and k makes for a vector that produces a vector with
    %the best stock index last.
    [result, stockIn]=sort(j+k,'descend');
    %The last five stock indexes in the vector 'stockIn' are saved
```

```
55      %into the i:th column of the 'addStock' matrix.
56      addStock(:,i)=stockIn(end 4 :end)';%stockIn(1:5)';
57      %'RoA' and 'PE' are modified so that the first
58      %sort() function in the next four lap will place the selected
59      %five stocks in the bottom for the categorization by assigning
60      %them the value 0. Since the stocks are held for one year
61      %and the same stock should not be bought more than once
62      %during the same year.
63      RoA(i:i+4,addStock(:,i))=zeros(5,5);
64      PE(i:i+4,addStock(:,i))=zeros(5,5);
65  end
66
67  %Creates the matrix containing indexes for the stocks to be sold
68  %at each ten week period
69  magi=cell(1,2);
70  magi{1}=addStock;
71  takeStock=zeros(size(addStock));
72  takeStock(:,6:end)=addStock(:,1:end 5);
73  magi{2}=takeStock;
74
75
76
77  %'weave' is the matrix containing indexes that are
78  %elected for the Random Formula during the comming for loop.
79  weave=zeros(5,tenW);
80
81  %A matrix is created to keep track of which stocks that are
82  %not currently in the portfolio.
83  port=ones(tenW,inst);
84
85  for i=1:tenW
86  %   Checks which stocks and how many that are not in the portfolio
87      nonZ=find(port(i,:));
88      nrNonZ=length(nonZ);
89
90      stockInd=zeros(5,1);
91      nonZZ=nonZ;
92      for j=1:5
93  %       Stocks are selected randomly from the ones that are not
94  %       allready in the portfolio.
95          stockInd(j)=unidrnd(nrNonZ,1,1);
96          weave(j,i)=nonZZ(stockInd(j));
97          nonZZ(stockInd(j))='';
98          nrNonZ=nrNonZ 1;
99      end
100     port(i:i+5,weave(:,i)')=zeros(6,5);
101 end
102
103 % Creates an array for the two matrixes that are to contain
104 % the indeces of the stocks that are going in and out of the
105 % portfolio at each ten week step.
106 slump=cell(1,2);
107
108 % Creates the matrix containing the stocks going in.
109 slump{1}=weave;
110 spit=zeros(size(weave));
111 spit(:,6:end)=weave(:,1:end 5);
```

```
112
113 % Creates the matrix containing the stocks going out.
114 slump{2}=spit;
115
116 end
```

9.8 BTstrat

```
%This function constructs the paths of the portfolios
%in the simulated universes. It takes the simulated universes
%and the array containing buy and sell matrixes and then
%produces a matrix in which each column is a outcome
%of the portfolio.

function simStrategi=BTstrat(dat, choice)

%'addStock' gets assigned the buy matrix and 'takeStock'
%the sell matrix.
addStock=choice{1};
takeStock=choice{2};

%The number of repetitions is the same as the length of
%the cell array 'dat'.
rep=length(dat);

%The number of days of the simulation, the number of stocks
%in the universe and useful matrixes for the construction
%of the portfolio simulation are declared.
days=length(dat{1}(:,1));
uniInst=length(dat{1}(1,:));
eternity=3750;%15 years in days
spacious=zeros(eternity,uniInst);
delStrat=zeros(eternity,rep);

%This loop creates 'rep' number of paths for the strategy
%generated from the 'rep' number of simulated universes
%respectively.
for k=1:rep
    %choose the k:th universe and fill it up with zeros
    %so that it covers 15 years, this is done so that it
    %will be no problem stepping through it in steps of
    %50 days.
    data=dat{k};
    portfolio=zeros(eternity,uniInst);
    spacious(1:days,1:uniInst)=data;
    data=spacious;
    tenW=ceil(days/50);
    %This loop assigns stocks for the portfolio for each
    %ten week period.
    for i=1:tenW
        if takeStock(:,i)==zeros(5,1)

            %The number of different stocks to ultimately
            %be held in the portfolio is 25. Therefor the
            %amount invested (1 $) is divided by 25 for
            %each stock bought until 25 stocks are held.
            nrOfS=(1/25)./data((i 1)*50+1,addStock(:,i));
        else
            %If there are 25 different stocks in the
            %portfolio the amount used to buy new stock
            %is a fifth of the amount recieved for
            %the once sold.
```

```
55              nrOfS=(sum(portfolio((i 1)*50,takeStock(:,i)))/5)./data((i 1)*50+1,addStock(:,i));
56          end
57          %The simulated price series of the newly bought
58          %stocks for a ten week period is assigned to
59          %'addData'.
60          addData=data((i 1)*50+1:(i 1)*50+250,addStock(:,i));
61          portfolio((i 1)*50+1:(i 1)*50+250,addStock(:,i))=[nrOfS(1)*addData(:,1) ...
                nrOfS(2)*addData(:,2) nrOfS(3)*addData(:,3) ...
                nrOfS(4)*addData(:,4) nrOfS(5)*addData(:,5)];
62      end
63      delStrat(:,k)=(portfolio*ones(uniInst,1))';
64  end
65
66  %Takes away unnecessary zeros
67  lastDay=find(delStrat(:,1),1,'last');
68  delStrat=delStrat(1:lastDay,:);
69
70  simStrategi=delStrat;
71  end
```

9.9 uniStratSim

```
%uniStratSim() is a version of BTstrat() with 'rep'=1 in
%which all the stocks are bought in steps of 1/5 to end up
%with one invested dollar after one year.

function indexSim=uniStratSim(simulering)


indexSim=zeros(length(simulering{1}(:,1)), length(simulering));


for i=1:length(simulering)
    dat=simulering{i};
    %
    %A break is defined in case the ATF starting from a late
    %year can not be carried out.
    if length(dat(:,1))<201
        indexSim=zeros(size(simulering{1}));
        break
    end
    %
     nrOfS1=((1/5)/length(dat(1,:)))./dat(1,:);
      nrOfS2=nrOfS1+((1/5)/length(dat(1,:)))./dat(50,:);
       nrOfS3=nrOfS2+((1/5)/length(dat(1,:)))./dat(100,:);
        nrOfS4=nrOfS3+((1/5)/length(dat(1,:)))./dat(150,:);
         nrOfS5=nrOfS4+((1/5)/length(dat(1,:)))./dat(200,:);

    indexPort1=bsxfun(@times, dat(1:50,:), nrOfS1);
    indexSim(1:50,i)=indexPort1*ones(length(dat(1,:)),1);

    %'bsxfun(@times, X, Y) multiplies the column vectors X(i)
    %with the scalars Y(i)
    indexPort2=bsxfun(@times, dat(51:100,:), nrOfS2);
    indexSim(51:100,i)=indexPort2*ones(length(dat(1,:)),1);

    indexPort3=bsxfun(@times, dat(101:150,:), nrOfS3);
    indexSim(101:150,i)=indexPort3*ones(length(dat(1,:)),1);

    indexPort4=bsxfun(@times, dat(151:200,:), nrOfS4);
    indexSim(151:200,i)=indexPort4*ones(length(dat(1,:)),1);

    indexPort5=bsxfun(@times, dat(201:end,:), nrOfS5);
    indexSim(201:end,i)=indexPort5*ones(length(dat(1,:)),1);
end
```

10 Appendix 3: The Universe

Starting out the universe was intended to consist of the stocks in the SP100 index with equal weights. The universe was intended to be unchanged over the time for the simulations so full price history for the stocks was required. For different reasons the accessability for a few of the stocks in the S&P100(at the end of June 2012). These were replaced with stocks from the same industry segment, not necessarily from the same part of the world, but with compareable market capitalization. All the data was downloaded from Yahoo Finance. The companies who's stocks became the resulting universe were:

Apple Inc, Abbott Laboratories, Accenture plc, American Electric Power Co, American International Group Inc., The Allstate Corporation, Amgen Inc., Amazon.com Inc., Apache Corp., Anadarko Petroleum Corporation, American Express Company, The Boeing Company, Bank of America Corporation, Baxter International Inc., The Bank of New York Mellon Corporation, Bristol-Myers Squibb Company, Berkshire Hathaway Inc., British American Tobacco plc, Citigroup Inc., Caterpillar Inc., Colgate-Palmolive Co., Comcast Corporation,Capital One Financial Corp., ConocoPhillips, Costco Wholesale Corporation, Ciscao Systems Inc., CVS Caremark Corporation, Chevron Corporation, E. I. du Pont Nemours and Company, Dell Inc., The Walt Disney Company, Dish Network Corp, The Dow Chemical company, Devon Energy Corporation, eBay Inc., EMC Corporation, Emerson Electric Co., Exelon Corporation, Ford Motor Co., Freeport-McMoRan Copper & Gold Inc., FedEx Corporation, General Dynamics Corp., General Electric Company, Gilead Sciences Inc., Google Inc., The Goldman Sachs Group Inc., Halliburton Company, The Home Depot Inc., Honeywell International Inc., Hewlett-Packard Company, HSBC Holdings plc, International Business Machines Corporation,Intel Corporation, Johnson & Johnson, JPMorgan Chase & Co., The Coca-Cola Company, Eli Lilly Company, Lockheed Martin Corporation, Lowe's Companies Inc., McDonald's Corp., Mondelez International Inc., Medtronic Inc., MetLife Inc., 3m Company, Altria Group Inc., Monsanto Company, Merck & Co. Inc., Morgan Stanley, Microsoft Corporation, Nike Inc., National Oilwell Varco Inc., Norfolk Southern Corporation, Oracle Corporation, Occidental Petroleum Corporation, Pepsico Inc., Pfizer Inc., The Procter & Gamble Company, QUALCOMM Incorporated, Raytheon Co., Starbucks Corporation, Schlumberger Limited, Southern Company, Simon Property Group Inc., AT&T Inc., Target Corp., Toyota Motor Corporation, The Travelers Companies Inc., Time Warner Inc., Texas Instruments Inc., UnitedHealth Group Incorporated, Union Pacific Corporation, United Parcel Service Inc., U.S. Bancorp, United Technologies Corp., Walgreen Co., Wells Fargo & Company, Williams Companies Inc., Wall-Mart Stores Inc., Verizon Communications Inc., Exxon Mobil Corporation

Because of lack of access to complete key number data for the purposes of the Magic Formula these were taken out of the universe:

American International Group Inc., The Allstate Corporation, Bank of America Corporation, The Bank of New York Mellon Corporation, Berkshire Hathaway Inc., The Home Depot Inc., HSBC Holdings plc, Intel Corporation, JP-Morgan Chase & Co., MetLife Inc., Pepsico Inc., QUALCOMM Incorporated, Starbucks Corporation.

11 References

1. Campbell, S. (2005) A Review of Backtesting and Backtesting Procedures
2. Staum, J. (2009) Monte Carlo Computation in Finance
3. Greenblatt, J. (2006) The Little Book That Beats the Market
4. Hull, J. (2012) Options, Futures and Other Derivatives
5. Glasserman, P. (2003) Monte Carlo Methods in Financial Engineering
6. Rydén, T., Lindgren, G. (1996) Markovprocesser
7. Höglund, T. (2008) Mathematical Asset Management
8. Interview with Salla Franzén

The Code

The code was inspired by the introduction section of [2] and the last chapter of [1]. The strategies were of course implemented in the code following [3]. The calculations of daily growth and daily return followed the guidance of [7] and [8].

Chapter 2: Backtesting

This chapter used information from [8] and [7], with 2.1 using [7] and 2.2-3 using [8].

Chapter 3: Monte Carlo

This chapter used information from the indroduction sections of [3] and [5] and was inspired by Appendix A in [6] as well as using information from wolframalpha.com for section 3.1.