Project Number: MB1-1234

# PORTFOLIO RISK MINIMIZATION USING HISTORICAL DATA

A Major Qualifying Project Report:

submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the

Degree of Bachelor of Science

by

_______________________________

Brian Duncan

Date: April 24, 2008

Approved:

_______________________________

Professor Marcel Blais

Project Advisor

ACKNOWLEDGMENTS

I would like to thank Professor Marcel Blais, Mathematical Sciences Department, Worcester Polytechnic Institute, as well as everyone who helped contribute to my success in this project.

ABSTRACT

Data from 1999 was gathered for 90 stocks in the S&P 500. The first 6 months of data was used to create a portfolio with the minimum risk while given an expected rate of return. Constraints were then added to limit short selling and limit the number of shares of certain stocks. The resulting portfolios were then tested to see if their future performance for the next 6 months would have produced a profit.

# Contents

# List of Figures

# List of Tables

# 1 Introduction

Investing in the stock market can potentially be a way to make a lot of money; however, there is a certain level of risk involved. When dealing with riskless investments such as money market accounts, investors are guaranteed a certain return without having to worry about losing their capital. On the other hand, when investing in the stock market, investors can lose money if their stocks lose value. There is a potential for a large reward, especially when investing in more risky stocks.

The stock market is ever-changing, and patterns can be difficult to discern. There may be trends, and there are many strategies to try and predict how stocks will perform, but we can never be certain. One way to decrease the risk associated with a stock portfolio is by using diversification. This means that we spread out our capital into many different stocks. Since there is a very low probability that each of the stocks will drop significantly in price at the same time, diversification gives us a mechanism to reduce the volatility of our portfolio's value.

The phrase "high-risk, high-reward" means that the riskier stocks have a higher potential reward associated with them. If we invest in stocks that fluctuate greatly, the large jumps in price can be very beneficial, but the large drops in price can also decrease our portfolio's value a great deal. Portfolio risk minimization involves taking stock price historical data and calculating past returns and their variances [1]. This will separate certain stocks which have a tendency to fluctuate greatly from day to day and those that have more consistency. If a stock's price over the last six months continued to gradually rise without any sharp changes, it might be in our best interest to invest a large percentage of our capital in this stock. Using historical data, the algorithm will be able to give us the portfolio with the minimum variance for each expected return. By varying the expected return, we construct an efficient frontier [1]. The efficient frontier is a term used for the infinite number of combinations of the mean rate of returns and corresponding minimum variances. For any mean rate of return that we choose, there is a minimum variance portfolio associated with it.

On the surface and to an inexperienced investor the stock market is a risky forum, but using Markowitz Theory [1] and minimizing our portfolio variance can give us a fairly stable return. Since we are using historical data, it is very important to gather data as recent as possible. Even then, over time that data will become less useful, and we will again have to perform this optimization routine to update our portfolio. When our main concern is to make a profit, this is an effective strategy. We can always set a high expected rate of return, and even though the portfolio variance will be high, the optimization routine will minimize it. This strategy could thus be used by both the conservative investor and the aggressive investor. In this project we implement this theory using historical data from the first six months of 1999 to find the optimal portfolio and then computing how much return we would have gotten in the next six months using a buy and hold strategy.

## 2 Background

Harry Markowitz is an economist who is best known for his work in modern portfolio theory. In 1952 he wrote a paper called "Portfolio Selection" which appeared in the Journal of Finance [7]. He later won a Nobel Prize for his contributions related to portfolio theory and his revolutionary ideas described in his 1952 work. He suggested that instead of investing primarily in low risk securities, to diversify stock portfolios and minimize the risk in those.

The returns are thought of as random variables that can follow a trend which can be estimated using historical data. It is assumed that past trends are indicative of future behavior, so past volatility can be a good representation of future volatility. The theory produces a portfolio with the minimum variance given an expected return. This is where the risk-reward relationship comes into play, and we quantify the minimal additional risk associated with a higher expected return by constructing the efficient frontier. Markowitz introduced the efficient frontier curve which represents all of the calculated optimal portfolios. Each point on the curve represents a portfolio with the minimum variance for a given expected return.

Modern portfolio theory and risk minimization has made trading in the stock market a more sophisticated investment than it was in the past. Stock portfolios can create a large profit for investors, and optimization techniques can be used to minimize risk. Procedures such as data cleaning and problem formulation are necessary for portfolio optimization to be effective.

## 3 Data

Data gathering and cleaning is a vital component of portfolio optimization. We started with a pool of 90 stocks, where each has its own tick data file. Each line of the tick data represents one trade for that particular stock. There was three years of data in total, and we focused on 1999. The first six months of data was used for calculating the optimal portfolios, and the second six months was used to test the future performance of our portfolios.

Our first step involved extracting the useful data from the information gathered. Table 1 shows a sample of the tick data for stock AA. Column 1 is the stock symbol. Columns 3 and 7 were used as an ID number. In column 2, the date was given in Julian format, where the value represents the number of days since January 1, 1900. Column 4 is the time of day. For stock AA the first trade occured on 1/4/99 at 9:31 AM. Column 5 is the number of shares and column 6 is the price of the stock for that particular trade. The number of shares is not important to us, but the prices will be used to find the mean price of each day.

Table 1: Tick Data

| Stock Symbol | Date | | Time | Shares | Price | |
|---|---|---|---|---|---|---|
| AA | 36164 | 4912 | 93100 | 13400 | 74.56 | 2130 |
| AA | 36164 | 4912 | 93200 | 1200 | 74.5 | 2130 |
| AA | 36164 | 4912 | 93200 | 500 | 74.5 | 2130 |

We set our timeframe to be one year, from 1/4/99 to 1/3/00. We chose to eliminate trades occuring at the beginning and end of each day when computing the average daily price. Many traders in the market often face the additional constraint of opening and closing large positions at the beginning and end of the trading day, respectively. This causes dramatic fluctuations in stock prices that are typically not indicative of the true market value of the stock. We set a timeframe from 10 AM to 3 PM for each day to calculate the average daily stock price. After calculating a stock price for each day, we calculated the standard deviation of the daily prices for each stock to help find errors in the data.

If the standard deviation of the daily prices was very large, this often indicated that there could be an error in the data. Sometimes we found that some of the columns were permuted. If the shares column and the price column were switched, the daily price could be off by several orders of magnitude, which would set off a red flag. These cases were easy to identify and fix. Sometimes the prices in some trades seemed nonsensical were extremely large. In this case, we needed to delete some trades and even some entire days. We did, however, have to be careful that we did not falsely identify an error in the data when there actually was a stock split.

When a company performs a stock split, they decrease the price of the stock while increasing the number of shares. Some common stock splits are 2-1, 3-1, and 3-2. For example, in the case of a 2-1 stock split, the price is cut in half while the number of shares is doubled. A stockholder keeps the same amount of market capital (shares times price). A company may want to have a stock split to decrease the price and make their stock look more attractive to a consumer. In our situation, if there is a sudden large decrease in price, this may indicate a stock split. Figure 1 shows a 2-1 stock split for Alcoa. The price steadily fluctuates around \$60 and then suddenly drops to \$30. Matlab was used to identify the splits, and new files were created where the prices after the split were adjusted to eliminate these jumps in price. For example, in the case of AA, we multiplied all the prices after the stock split by 2.

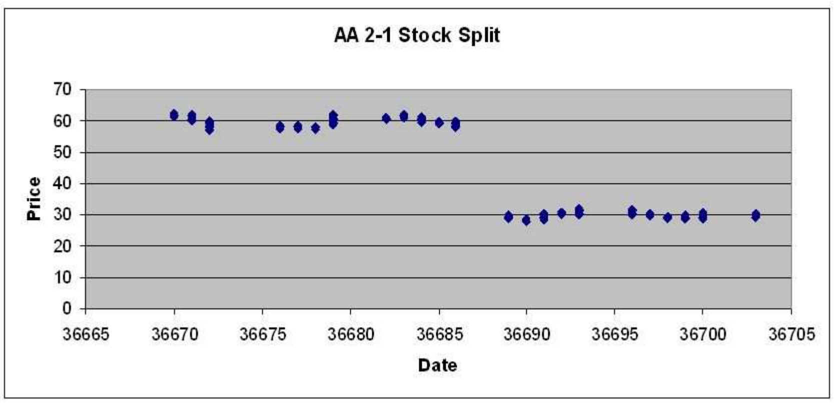


Figure 1: AA 2-1 Stock Split

After fixing the splits and errors, we needed to eliminate five more stocks. We needed 126 days of data for the optimization and 126 days of data for the portfolio testing. However, five stocks had days that needed to be deleted because of errors in the data so they could not be used in the optimization. We were thus left with 85 stocks.

# 4 Optimization

In this optimization problem we have N stocks, a fixed expected return rate $\mu$, and portfolio weights denoted by $\Pi_i$. Our objective is to minimize the variance (risk) of the portfolio returns. We use $R_i$ as the mean return rate for each stock. Generally, the higher the expected return rate is, the higher the risk is going to be. The problem formulation will be split into two parts. The first will be the "unconstrained problem", where we require only that the portfolio weights sum to 1 and that the expected return is $\mu$. The second part will be the constrained problem, where lower and upper bounds will be imposed on the portfolio weights for each stock. In both cases, short selling is allowed, and thus $\Pi_i \in (-\infty, \infty)$.

## 4.1 Unconstrained Problem

### 4.1.1 Problem Formulation

We formulate a standard risk minimization problem with the given price and return data. The goal is to minimize the portfolio variance, $\sigma_\Pi^2$, while satisfying two initial constraints. The first constraint assures that the portfolio will achieve the expected return rate, $\mu$. The second constraint says that the portfolio weights add up to one, so 100% of our capital is invested into stock.

$$\min \sigma_\Pi^2 = \sum_{i=1}^{N} \sum_{j=1}^{N} \Pi_i \Pi_j \sigma_{ij} \tag{1}$$

s.t.

$$\sum_{i=1}^{N} \Pi_i R_i = \mu \tag{2}$$

$$\sum_{i=1}^{N} \Pi_i = 1 \tag{3}$$

We can then use Lagrange Multipliers $\lambda_1$ and $\lambda_2$ to make a Lagrangian function [1]. This method is a form of nonlinear optimization which enables us to identify the minimum portfolio. The new equation becomes

$$L = \sum_{i=1}^{N} \sum_{j=1}^{N} \Pi_i \Pi_j \sigma_{ij} - \lambda_1 (\sum_{i=1}^{N} \Pi_i R_i - \mu) - \lambda_2 (\sum_{i=1}^{N} \Pi_i - 1) \tag{4}$$

Now we need to differentiate the Lagrangian and set the derivatives equal to zero [1]. Since we only have two constraints, the general form is

$$\nabla f = \lambda_1 \nabla g + \lambda_2 \nabla h \tag{5}$$

or

$$\nabla f - \lambda_1 \nabla g - \lambda_2 \nabla h = 0 \tag{6}$$

where $f$ is our original objective function and $g$ and $h$ are the constraints. Now to differentiate we need to factor out a $\Pi_k$,

$$L = \sum_{i=1}^{N}(\Pi_i \sum_{j=1}^{N} \Pi_j \sigma_{ij}) - \lambda_1(\sum_{i=1}^{N} \Pi_i R_i - \mu) - \lambda_2(\sum_{i=1}^{N} \Pi_i - 1)$$

$$= \sum_{\substack{i=1\\i\neq k}}^{N}(\Pi_i(\sum_{\substack{j=1\\j\neq k}}^{N} \Pi_j \sigma_{ij} + \Pi_k \sigma_{ik})) + \Pi_k(\sum_{\substack{j=1\\j\neq k}}^{N} \Pi_j \sigma_{kj} + \Pi_k \sigma_{kk})$$

$$-\lambda_1(\sum_{\substack{i=1\\i\neq k}}^{N} \Pi_i R_i + \Pi_k R_k - \mu) - \lambda_2(\sum_{\substack{i=1\\i\neq k}}^{N} \Pi_i + \Pi_k - 1)$$

After making this adjustment to the equation, the next step is to differentiate with respect to $\Pi_k$,

$$\tfrac{\partial L}{\partial \Pi_k} = \sum_{\substack{i=1\\i\neq k}}^{N} \Pi_i \sigma_{ik} + \sum_{\substack{j=1\\j\neq k}}^{N} \Pi_j \sigma_{kj} + 2\Pi_k \sigma_{kk} - \lambda_1 R_k - \lambda_2$$

$$= 2\sum_{i=1}^{N} \Pi_i \sigma_{ik} - \lambda_1 R_k - \lambda_2$$

If we repeat this process for all k from 1 to N and set the derivative equal to 0, we get

$$\frac{\partial L}{\partial \Pi_i} = 2\sum_{j=1}^{N} \Pi_j \sigma_{ij} - \lambda_1 R_i - \lambda_2 = 0, \qquad i = 1, ..., N \tag{7}$$

This gives us N equations, but we have N+2 unknowns ($\Pi_i$, $\lambda_1$, and $\lambda_2$). Therefore, we need two more equations to be able to solve this optimization problem. These two equations come from the initial constraints [1]. So, our first-order conditions are

$$2\sum_{j=1}^{N}\Pi_j\sigma_{ij} - \lambda_1 R_i - \lambda_2 = 0, \qquad i = 1, ..., N \tag{8}$$

$$\sum_{i=1}^{N}\Pi_i R_i = \mu \tag{9}$$

$$\sum_{i=1}^{N}\Pi_i = 1 \tag{10}$$

$$\lambda_1 \geq 0 \tag{11}$$

$$\lambda_2 \geq 0 \tag{12}$$

or

$$2(\Pi_1\sigma_{1\,1} + \Pi_2\sigma_{1\,2} + ... + \Pi_N\sigma_{1\,N}) - \lambda_1 R_1 - \lambda_2 = 0$$
$$2(\Pi_1\sigma_{2\,1} + \Pi_2\sigma_{2\,2} + ... + \Pi_N\sigma_{2\,N}) - \lambda_1 R_2 - \lambda_2 = 0$$
$$\vdots$$
$$2(\Pi_1\sigma_{N\,1} + \Pi_2\sigma_{N\,2} + ... + \Pi_N\sigma_{N\,N}) - \lambda_1 R_N - \lambda_2 = 0$$

$$\Pi_1 R_1 + \Pi_2 R_2 + ... + \Pi_N R_N = \mu$$

$$\Pi_1 + \Pi_2 + ... + \Pi_N = 1$$

$$\lambda_1 \geq 0$$

$\lambda_2 \geq 0$

To solve this system numerically we convert it into matrix form. First we will identify all of the known variables and unknown variables.

$$\sigma = \begin{pmatrix} \sigma_{1\,1} & \sigma_{1\,2} & \dots & \sigma_{1\,N} \\ \sigma_{2\,1} & \sigma_{2\,2} & \dots & \sigma_{2\,N} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{N\,1} & \sigma_{N\,2} & \dots & \sigma_{N\,N} \end{pmatrix}$$

$$\underline{\Pi} = \begin{pmatrix} \Pi_1 \\ \vdots \\ \Pi_N \end{pmatrix}$$

$$\underline{R} = \begin{pmatrix} R_1 \\ \vdots \\ R_N \end{pmatrix}$$

$$\underline{\lambda} = \begin{pmatrix} \lambda_1 \\ \lambda_2 \end{pmatrix}$$

To create the one-line equations in matrix form we will need some additional matrices. These will be used to adjust the dimensions of the $R_i$'s. So, our vectorized first-order equations are

$$2\sigma\underline{\Pi} - R^{*}R^{**}\underline{\lambda} = \underline{0} \tag{13}$$

$$\underline{\Pi}^T\underline{R} = \mu \tag{14}$$

$$\underline{\Pi}^T\underline{1} = 1 \tag{15}$$

Now we combine the three equations into $A\underline{x} = \underline{b}$ form. This will isolate $\underline{x}$ as the vector of all N+2 unknowns. Since A is invertible, the equation becomes $\underline{x} = A^{-1}\underline{b}$, which can be solved in Matlab. After some rearranging of terms, the equations become

$$\begin{pmatrix} 2\sigma & -\underline{R} & -\underline{1} \\ \underline{R}^T & 0 & 0 \\ \underline{1}^T & 0 & 0 \end{pmatrix} \begin{pmatrix} \underline{\Pi} \\ \lambda_1 \\ \lambda_2 \end{pmatrix} = \begin{pmatrix} \underline{0} \\ \mu \\ 1 \end{pmatrix}$$

or

$$\begin{pmatrix} 2\sigma_{1\,1} & 2\sigma_{1\,2} & \dots & 2\sigma_{1\,N} & -R_1 & -1 \\ 2\sigma_{2\,1} & 2\sigma_{2\,2} & \dots & 2\sigma_{2\,N} & -R_2 & -1 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 2\sigma_{N\,1} & 2\sigma_{N\,2} & \dots & 2\sigma_{N\,N} & -R_N & -1 \\ R_1 & R_2 & \dots & R_N & 0 & 0 \\ 1 & 1 & \dots & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} \Pi_1 \\ \Pi_2 \\ \vdots \\ \Pi_N \\ \lambda_1 \\ \lambda_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ \mu \\ 1 \end{pmatrix}$$

Before we perform the optimization and test for future performance we need to make sure this matrix is positive definite. That way we will know it is invertible and has a unique solution. Since we need the matrix to be symmetric, and $\sigma_{x\,y} = \sigma_{y\,x}$, an equal form of the optimization equation is

$$\begin{pmatrix} 2\sigma_{1\,1} & 2\sigma_{1\,2} & \dots & 2\sigma_{1\,N} & R_1 & 1 \\ 2\sigma_{1\,2} & 2\sigma_{2\,2} & \dots & 2\sigma_{2\,N} & R_2 & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 2\sigma_{1\,N} & 2\sigma_{2\,N} & \dots & 2\sigma_{N\,N} & R_N & 1 \\ R_1 & R_2 & \dots & R_N & 0 & 0 \\ 1 & 1 & \dots & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} \Pi_1 \\ \Pi_2 \\ \vdots \\ \Pi_N \\ -\lambda_1 \\ -\lambda_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ \mu \\ 1 \end{pmatrix}$$

So, the matrix we need to test for positive-definiteness is

$$A = \begin{pmatrix} 2\sigma_{1\,1} & 2\sigma_{1\,2} & ... & 2\sigma_{1\,N} & R_1 & 1 \\ 2\sigma_{1\,2} & 2\sigma_{2\,2} & ... & 2\sigma_{2\,N} & R_2 & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 2\sigma_{1\,N} & 2\sigma_{2\,N} & ... & 2\sigma_{N\,N} & R_N & 1 \\ R_1 & R_2 & ... & R_N & 0 & 0 \\ 1 & 1 & ... & 1 & 0 & 0 \end{pmatrix}$$

There is a theorem that says if all the principal minors of a square symmetric matrix are positive, then the matrix is positive definite [3]. After calculating all of the principal minors in Matlab and confirming they are all positive, we can say that A is postive definite, is invertible, and has a unique solution.

### 4.1.2 Results

The first step when we gathered the results was to find the minimum variance portfolio, which is the portfolio with the smallest possible variance [1]. This gives us the safest portfolio. The corresponding expected return rate is very low, but so is the variance. In order to calculate the proportions in the minimum variance portfolio, we set a range of expected return rates and calculate the corresponding minimum variances. We then needed to find the point at which the variances went from decreasing to increasing. We first set an extremely low expected return and calculated the portfolio variance. Then we increased the expected return by a very small amount, and repeated the process. The variances gradually decrease as the expected returns increase, but there is a point where the variances also start to increase. After sorting through 10,000 trials, we narrowed the search down to the values shown in Table 2.

Table 2: Minimum Variance Portfolios

| Expected Return Rate | Portfolio Variance |
|---|---|
| 5.0000000e-004 | 1.3665251e-005 |
| 5.1000000e-004 | 1.3664817e-005 |
| 5.2000000e-004 | 1.3664476e-005 |
| 5.3000000e-004 | 1.3664229e-005 |
| 5.4000000e-004 | 1.3664075e-005 |
| 5.5000000e-004 | 1.3664015e-005 |
| 5.6000000e-004 | 1.3664048e-005 |
| 5.7000000e-004 | 1.3664175e-005 |
| 5.8000000e-004 | 1.3664395e-005 |
| 5.9000000e-004 | 1.3664709e-005 |
| 6.0000000e-004 | 1.3665116e-005 |

We see that the minimum variance occurs when the expected return rate is 5.5000000e-004. This is equal to a return of 0.055%, which is a very small return and would never logically be a goal for a stockholder; however, this data can be used to form the efficient frontier, as shown in Figure 2. The efficient frontier is a curve where each point represents an optimal portfolio, that is, a portfolio with minimized variance for the given expected return. The leftmost point on the curve represents the minimum variance portfolio.

We ignore the bottom half of the curve, since if we set a certain amount of risk, we will always prefer the portfolio with the higher expected return. It is clear that if the expected return rises, so does the portfolio variance.

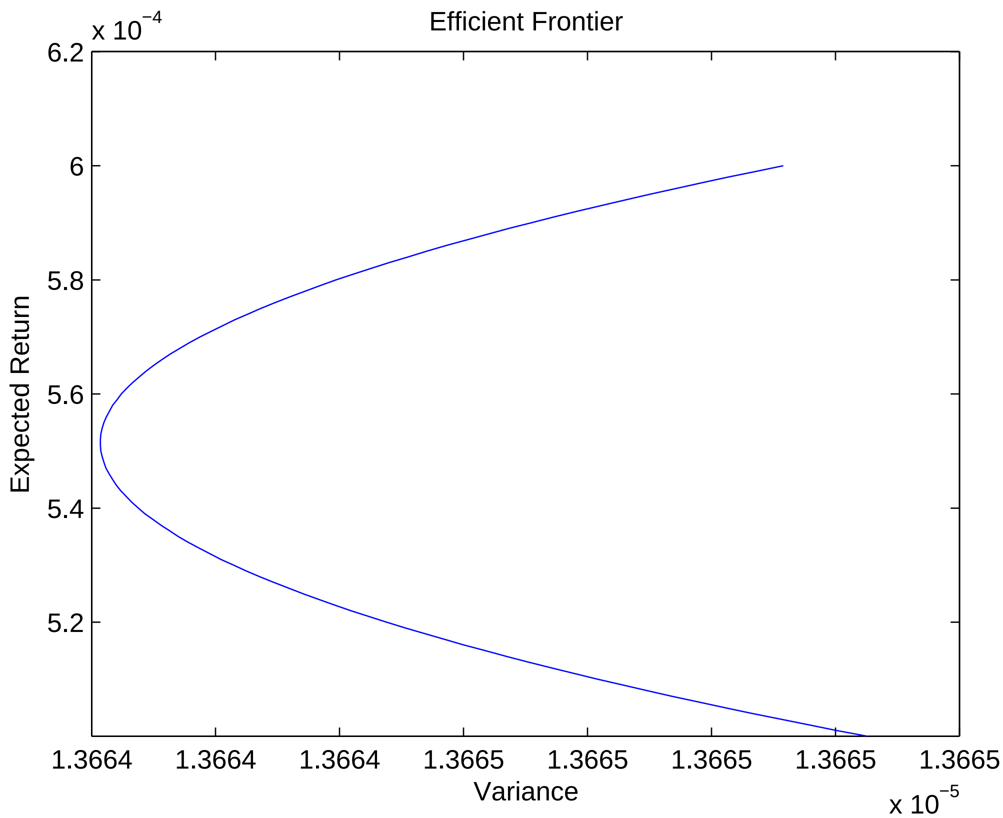


Figure 2: Efficient Frontier Optimal

If we expand the range of expected returns to include more reasonable values, the variances also rise, but not significantly. Figure 3 shows the efficient frontier when the expected return ranges up to 50%.

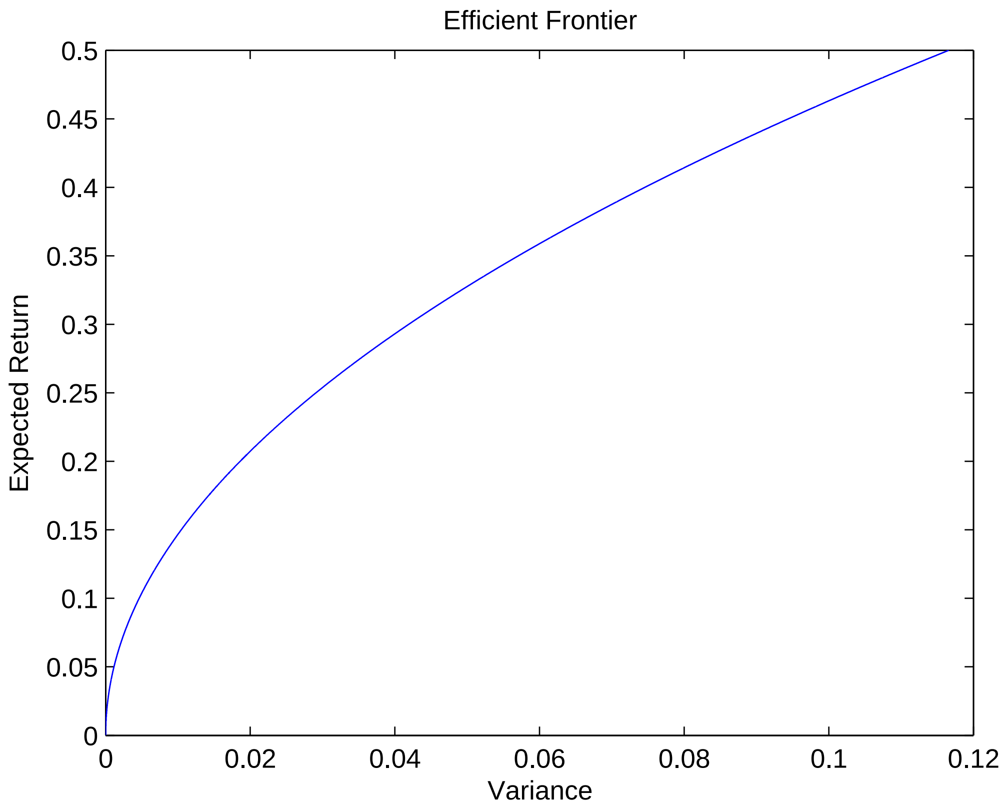


Figure 3: Efficient Frontier

Since the minimum variance portfolio gave an unreasonably low expected return, we need to calculate some optimal portfolios with some realistic expected returns. We chose to use 1%, 5%, 10%, 25%, and 50%. Tables 3, 4, and 5 show the optimal portfolios for the given expected returns. Each of the stocks holds a proportion of the portfolio. If the proportion is negative, that means the optimization suggests short selling and we would sell shares of these stocks that we don't own, while hoping the stocks' prices go down so we can buy them back at a lower price.

For the 5% expected return optimal portfolio, all 85 stocks held a non-zero proportion of the portfolio. 43 stocks were held long, and 42 stocks were held short. The stock with the proportion closest to zero was GM (General Motors) at 0.006. The stock with the greatest long proportion was JNJ (Johnson & Johnson) at 2.989. The stock with the greatest short proportion was AEP (American Electric Power) at -3.977. All of these proportions are shown in tables 3, 4, and 5.

Since we now have the optimal portfolios based on historical data of the first six months of 1999, we now need to test their future performances for the next six months to see if they actually would have made a profit. We set a starting captial of $10,000 for each of the portfolios. We then used the computed optimal portfolios to set the proportions of that $10,000 into the various stocks. A buy and hold strategy for six months was tested. For the minimum variance portfolio, where the expected return was 0.055%, the actual return was -12.88%, as shown by figures 4 and 5. Even though we would have lost money with this portfolio, our expected rate of return was so close to 0 that this result is not a surprise. The daily value and return stayed fairly steady except for one day, where an abnormal jump in price for one stock occured.

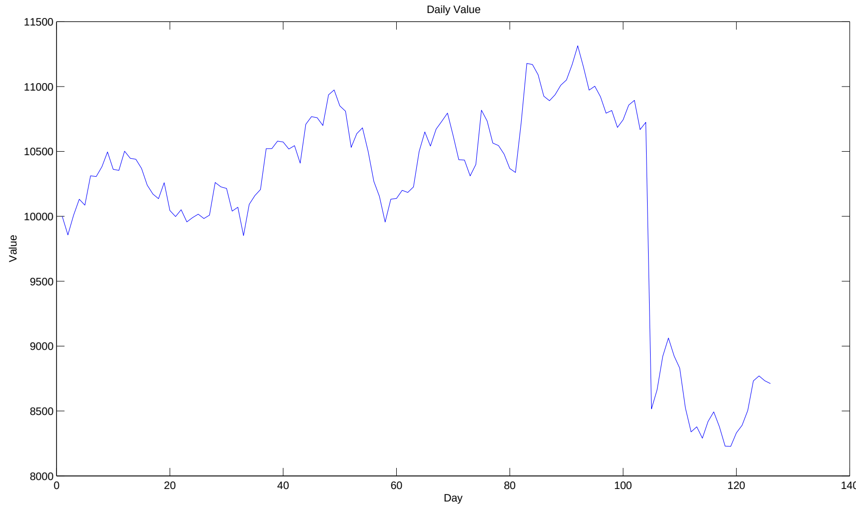


Figure 4: Daily Value for Portfolio with Minimum Variance

After testing each of the six portfolios for the next six months of data, it was shown that each one made a profit significantly larger than the expected return indicated. Figure 7 shows the progress and the gradual increase in value over time for each of the six portfolios. Figure 6 is a zoomed in look at the portfolios of 0.055%, 1%, and 5%. Figure 9 shows the daily returns for each of the portfolios, while figure 8 shows the three smallest expected returns. From these figures, it is evident how the increased expected return rate produced an increased variance in the actual performance of the portfolio. Table 6 shows the large profits the theoretical portfolios made in the second half of 1999.

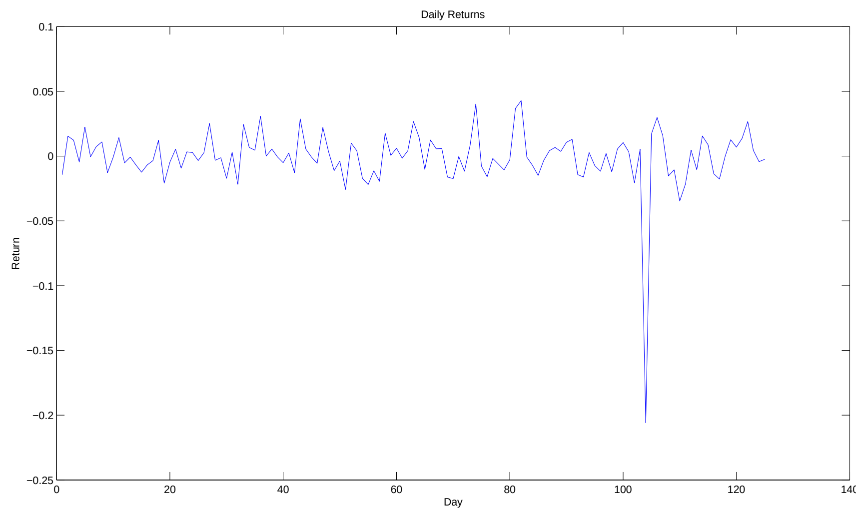


Figure 5: Daily Returns for Portfolio with Minimum Variance

Figures 10 and 11 show the performance of the S&P 500 index during 1999. The stock market was booming during this time, and the S&P 500 was doing very well. One of the reasons for the boom was that the dot-com bubble was rapidly growing at this time. Internet companies and other technologically based stocks were thriving during this year. Stock prices were rising, and this led to our portfolios to doing extremely well. Also, this was the period right before Y2K, when the U.S. spent an estimated $300 billion [6] on preparations for the potential problems that the new year could have caused.

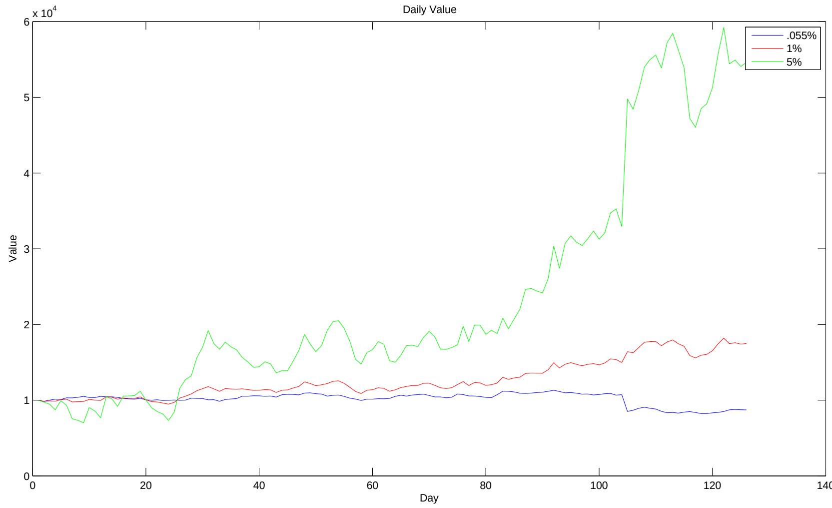


Figure 6: Daily Value

## 4.2 Constrained Problem

### 4.2.1 Problem Formulation

In the second optimization problem we add upper and lower bounds to the portfolio weights, $\Pi_i$, to prevent the portfolio from taking large long and short positions. It also allows us to disallow short selling. The process of forming a Lagrangian is similar to the unconstrained problem. Now we have additional constraints and additional unknowns which make the problem more involved. Our goal is still to minimize the portfolio variance, $\sigma_\Pi^2$. $u_i$ is the upper bound, or the maximum long position, and $l_i$ is the lower bound, or the maximum short position. If we set the lower bounds equal to zero for all stocks, this eliminates short selling because we will never have a negative proportion of

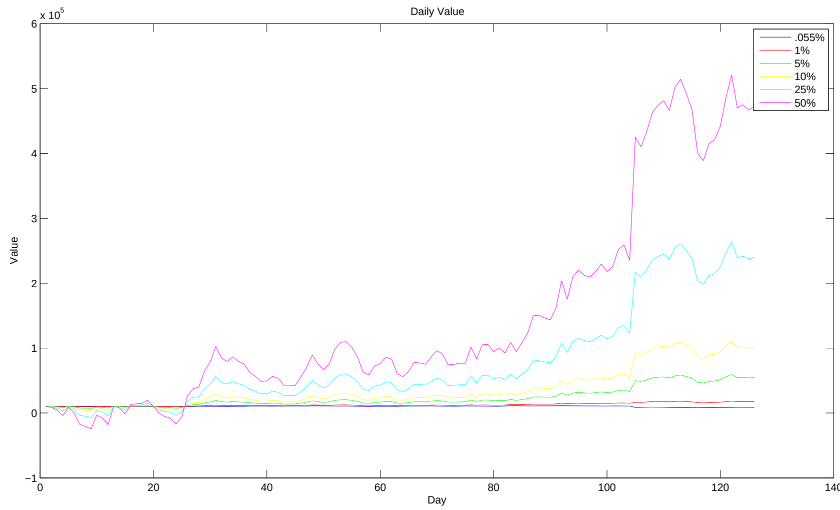


Figure 7: Daily Value2

one stock in our portfolio. Our new optimization problem is

$$\min \sigma_{\Pi}^2 = \sum_{i=1}^{N} \sum_{j=1}^{N} \Pi_i \Pi_j \sigma_{ij} \tag{20}$$

s.t.

$$\sum_{i=1}^{N} \Pi_i R_i = \mu \tag{21}$$

$$\sum_{i=1}^{N} \Pi_i = 1 \tag{22}$$

$$\Pi_i \geq l_i, \qquad i = 1, ..., N \tag{23}$$

$$\Pi_i \leq u_i, \qquad i = 1, ..., N \tag{24}$$

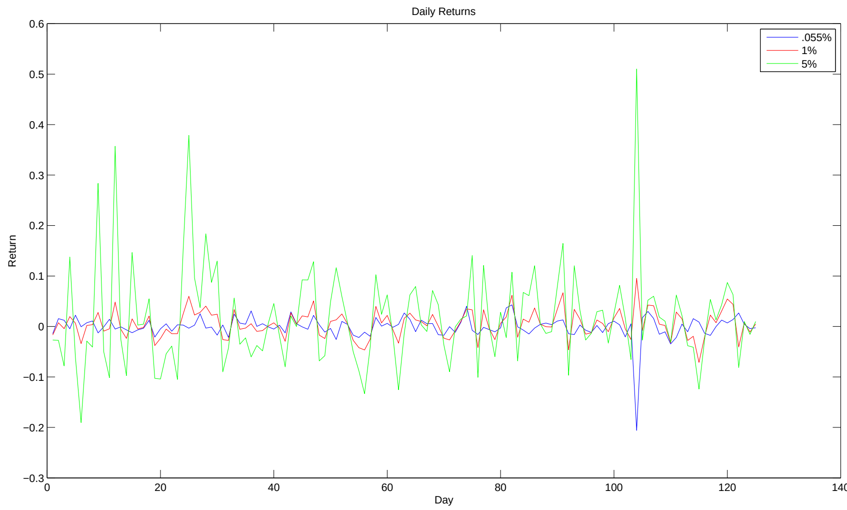


Figure 8: Daily Returns

We then introduce Lagrange Multipliers $\lambda_1$, $\lambda_2$, $\gamma_i$, and $\theta_i$ to form a Lagrangian function [1].

$$L = \sum_{i=1}^{N}\sum_{j=1}^{N}\Pi_i\Pi_j\sigma_{ij} - \lambda_1(\sum_{i=1}^{N}\Pi_i R_i - \mu) - \lambda_2(\sum_{i=1}^{N}\Pi_i - 1) - \sum_{i=1}^{N}\gamma_i(\Pi_i - l_i) - \sum_{i=1}^{N}\theta_i(u_i - \Pi_i) \tag{25}$$

Now to differentiate we need to factor out a $\Pi_k$,

$$L = \sum_{i=1}^{N}(\Pi_i\sum_{j=1}^{N}\Pi_j\sigma_{ij}) - \lambda_1(\sum_{i=1}^{N}\Pi_i R_i - \mu) - \lambda_2(\sum_{i=1}^{N}\Pi_i - 1)$$

$$- \sum_{i=1}^{N}\gamma_i(\Pi_i - l_i) - \sum_{i=1}^{N}\theta_i(u_i - \Pi_i)$$

$$= \sum_{\substack{i=1\\ i\neq k}}^{N}(\Pi_i(\sum_{\substack{j=1\\ j\neq k}}^{N}\Pi_j\sigma_{ij} + \Pi_k\sigma_{ik})) + \Pi_k(\sum_{\substack{j=1\\ j\neq k}}^{N}\Pi_j\sigma_{kj} + \Pi_k\sigma_{kk})$$

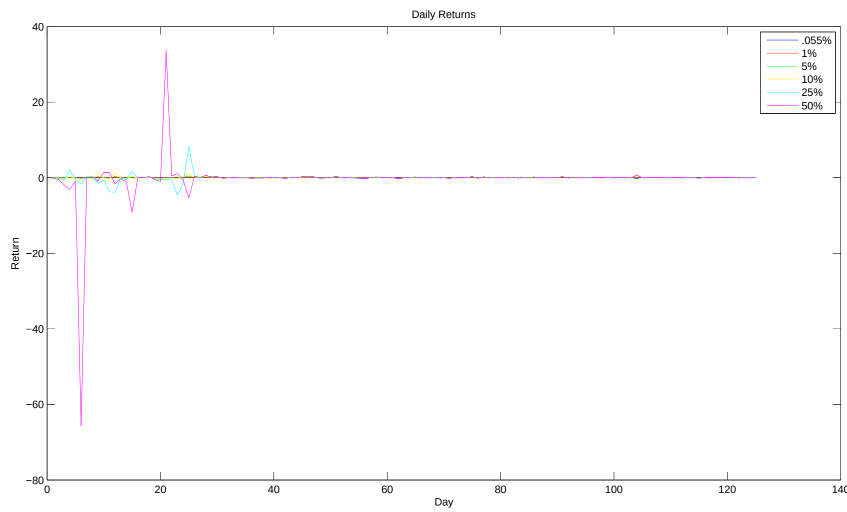


Figure 9: Daily Returns2

$$- \lambda_1 (\sum_{\substack{i=1 \\ i \neq k}}^{N} \Pi_i R_i + \Pi_k R_k - \mu) - \lambda_2 (\sum_{\substack{i=1 \\ i \neq k}}^{N} \Pi_i + \Pi_k - 1)$$

$$- (\sum_{\substack{i=1 \\ i \neq k}}^{N} \Pi_i \gamma_i + \Pi_k \gamma_k - l_i \gamma_i) - (\sum_{\substack{i=1 \\ i \neq k}}^{N} u_i \theta_i - \Pi_i \theta_i + \Pi_k \theta_k)$$

Now we need to differentiate with respect to $\Pi_k$,

$$\frac{\partial L}{\partial \Pi_k} = \sum_{\substack{i=1 \\ i \neq k}}^{N} \Pi_i \sigma_{ik} + \sum_{\substack{j=1 \\ j \neq k}}^{N} \Pi_j \sigma_{kj} + 2\Pi_k \sigma_{kk} - \lambda_1 R_k - \lambda_2 - \gamma_k - \theta_k$$

$$= 2 \sum_{i=1}^{N} \Pi_i \sigma_{ik} - \lambda_1 R_k - \lambda_2 - \gamma_k - \theta_k$$


Figure 10: S&P 500, all of 1999

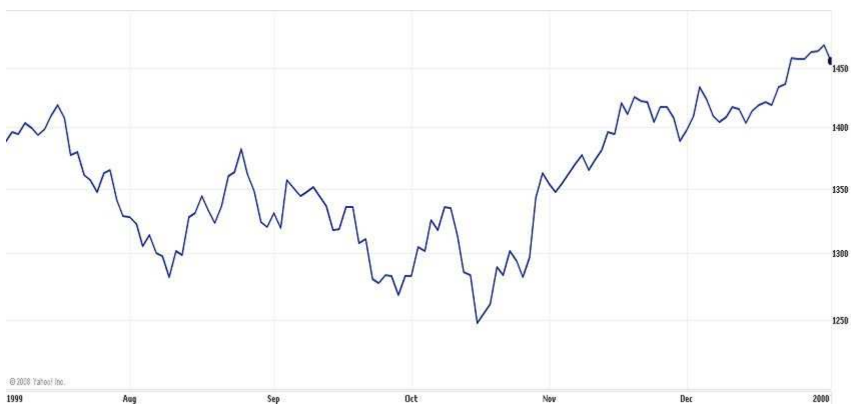


Figure 11: S&P 500, second half of 1999

If we repeat this process for all k from 1 to N and set the derivative equal to 0, we get our first order constraints

$$\frac{\partial L}{\partial \Pi_i} = 2\sum_{j=1}^{N} \Pi_j \sigma_{ij} - \lambda_1 R_i - \lambda_2 - \gamma_i - \theta_i = 0, \qquad i = 1, ..., N \tag{26}$$

This gives us 3N equations, but we have 3N+2 unknowns ($\Pi_i$, $\lambda_1$, $\lambda_2$, $\gamma_i$, and $\theta_i$). Therefore, we need two more equations to be able to solve this optimization problem. These two equations come from the initial constraints [1]. So, our first order constraints are

$$2\sum_{j=1}^{N} \Pi_j \sigma_{ij} - \lambda_1 R_i - \lambda_2 - \gamma_i - \theta_i = 0, \qquad i = 1, ..., N \tag{27}$$

$$\sum_{i=1}^{N} \Pi_i R_i = \mu \tag{28}$$

$$\sum_{i=1}^{N} \Pi_i = 1 \tag{29}$$

$$\gamma_i(\Pi_i - l_i) = 0, \qquad i = 1, ..., N \tag{30}$$

$$\theta_i(u_i - \Pi_i) = 0, \qquad i = 1, ..., N \tag{31}$$

$$\Pi_i \geq l_i, \qquad i = 1, ..., N \tag{32}$$

$$\Pi_i \leq u_i, \qquad i = 1, ..., N \tag{33}$$

$$\gamma_i \geq 0, \qquad i = 1, ..., N \tag{34}$$

$$\theta_i \geq 0, \qquad i = 1, ..., N \tag{35}$$

$$\lambda_1 \geq 0 \tag{36}$$

$$\lambda_2 \geq 0 \tag{37}$$

or

$$2(\Pi_1 \sigma_{1\,1} + \Pi_2 \sigma_{1\,2} + ... + \Pi_N \sigma_{1\,N}) - \lambda_1 R_1 - \lambda_2 - \gamma_1 - \theta_1 = 0$$

$$2(\Pi_1\sigma_{2\,1} + \Pi_2\sigma_{2\,2} + ... + \Pi_N\sigma_{2\,N}) - \lambda_1 R_2 - \lambda_2 - \gamma_2 - \theta_2 = 0$$

$$\vdots$$

$$2(\Pi_1\sigma_{N\,1} + \Pi_2\sigma_{N\,2} + ... + \Pi_N\sigma_{N\,N}) - \lambda_1 R_N - \lambda_2 - \gamma_N - \theta_N = 0$$

$$\Pi_1 R_1 + \Pi_2 R_2 + ... + \Pi_N R_N = \mu$$

$$\Pi_1 + \Pi_2 + ... + \Pi_N = 1$$

$$\gamma_1(\Pi_1 - l_1) = 0$$

$$\vdots$$

$$\gamma_N(\Pi_N - l_N) = 0$$

$$\theta_1(u_1 - \Pi_1) = 0$$

$$\vdots$$

$$\theta_N(u_N - \Pi_N) = 0$$

$$\Pi_1 \geq l_1$$

$$\vdots$$

$$\Pi_N \geq l_N$$

$$\Pi_1 \leq u_1$$

$$\vdots$$

$$\Pi_N \leq u_N$$

$$\gamma_1 \geq 0$$

$$\vdots$$

$$\gamma_N \geq 0$$

$$\theta_1 \geq 0$$

$$\vdots$$

$$\theta_N \geq 0$$

$$\lambda_1 \geq 0$$

$$\lambda_2 \geq 0$$

# 5 Conclusions

Our optimization proved very successful with the extremely high realized returns. Part of this was because of the booming stock market at the time, but another part was because of the effectiveness of the risk minimization.

The data cleaning and fixing of all of the errors was essential to get an accurate representation of the stock market at the time. We needed to delete some stocks from the project because of the errors, but saved others by fixing their stock splits. After formulating the portfolio optimization problem using Lagrange multipliers, Matlab was used to calculate the minimum variance portfolios and other portfolios with various expected returns. After the efficient frontier was graphed, the future performances of some of the portfolios were calculated using the next six months of data. We used a buy and hold strategy. The minimum variance portfolio lost money, but the rest greatly outperformed their respective expected returns.

This project went through the whole process of gathering data, data cleaning, problem formulation, optimization, and calculating future performances. All of the aspects are essential parts of the process. After completing this project I have a better understanding of optimization and the tasks that are involved with it.

# References


[1] Cvitanić, J. and F. Zapatero, *Introduction to the Economics and Mathematics of Financial Markets*, MIT Press: Cambridge, 2004.

[2] Nocedal, J. and S. Wright, *Numerical Optimization*, Springer: New York, 1999.

[3] Peressini, A., F. Sullivan, and J. Uhl, *The Mathematics of Nonlinear Programming*, Springer, 1988.

[4] Daye, Z., K. Leow, and S. Ding, "Empirical Evaluation of Volatility Estimation", paper, 2001.

[5] Stewart, J., *Calculus*, Brooks Cole, 5th Edition, 2002.

[6] http://news.bbc.co.uk/2/hi/talking_point/586938.stm

[7] http://www.riskglossary.com/link/portfolio_theory.htm

# 6 Appendix

## 6.1 Daily Prices Routine

```
function [priceMatrix] = daily(stock,numberOfDays)
%load stock data
stockMatrix = load(stock);

%start on 1/4/99, end after specified number of days
day = 36164;
dayEnd = 36164+numberOfDays;

%take prices between 10am and 3pm
timeStart = 100000;
timeEnd = 150000;

%vector of prices between 36164 and dayEnd, and between timeStart and
%timeEnd
validPrices = 0;

%column1 is the mean price, column2 is the std dev of the prices, and each
%row is a day
priceMatrix = [0;0];

%counters
dayCounter = 1;
priceCounter = 1;
finalPriceCounter = 1;
trade = 0;

%while day is within timeframe
while (day < dayEnd),
    %find all prices on certain day
    while (stockMatrix(dayCounter,1) == day),
        %find all prices within timeframe
        if ((stockMatrix(dayCounter,2)>=timeStart)&&
        (stockMatrix(dayCounter,2)
```

```
    <=timeEnd)),
        %take out error prices that are greater than $1000
        if (stockMatrix(dayCounter,4) < 1000),
            %store price in a vector
            validPrices(priceCounter) = stockMatrix(dayCounter,4);
            priceCounter = priceCounter+1;
            trade = 1;
        end;
    end;
    dayCounter = dayCounter+1;
end;

%exclude this error
if (strcmp(stock,'HON.dat')),
    if (day == 36495),
        trade = 0;
    end;
end;

%exclude these errors
if (strcmp(stock,'VIAB.dat')),
    if ((day >= 36242)&&(day <= 36251)),
        trade = 0;
    end;
end;

%if there was a trade on this day, store mean/std dev in the price
%matrix
if (trade == 1),
    priceMatrix(finalPriceCounter,2) = mean(validPrices);
    priceMatrix(finalPriceCounter,1) = stockMatrix(dayCounter-1,1);
    %priceMatrix(finalPriceCounter,3) = day;
    finalPriceCounter = finalPriceCounter+1;
end;
%reset variables
trade = 0;
validPrices = 0;
priceCounter = 1;
```

```
    day = day+1;
end;

%clear stock from workspace
clear stockMatrix;
```

## 6.2 Calculate All Prices and Std Dev's

```
function allStocks(excludeStocks, numOfDays)

%stock names
ticker1  =  'AA' ;
ticker2  =  'AEP' ;
ticker3  =  'AES' ;
ticker4  =  'AGC' ;
ticker5  =  'AIG' ;
ticker6  =  'AMGN' ;
ticker7  =  'AOL' ;
ticker8  =  'ATI' ;
ticker9  =  'AVP' ;
ticker10  =  'AXP' ;
ticker11  =  'BA' ;
ticker12  =  'BAC' ;
ticker13  =  'BAX' ;
ticker14  =  'BCC' ;
ticker15  =  'BDK' ;
ticker16  =  'BHI' ;
ticker17  =  'BMY' ;
ticker18  =  'BNI' ;
ticker19  =  'C' ;
ticker20  =  'CCU' ;
ticker21  =  'CI' ;
ticker22  =  'CL' ;
ticker23  =  'CPB' ;
ticker24  =  'CSC' ;
ticker25  =  'CSCO' ;
ticker26  =  'DAL' ;
ticker27  =  'DD' ;
```

```
ticker28  =  'DIS' ;
ticker29  =  'DOW' ;
ticker30  =  'EK' ;
ticker31  =  'EMC' ;
ticker32  =  'ENE' ;
ticker33  =  'EPG' ;
ticker34  =  'ETR' ;
ticker35  =  'EXC' ;
ticker36  =  'F' ;
ticker37  =  'FDX' ;
ticker38  =  'G' ;
ticker39  =  'GD' ;
ticker40  =  'GE' ;
ticker41  =  'GM' ;
ticker42  =  'GX' ;
ticker43  =  'HAL' ;
ticker44  =  'HCA' ;
ticker45  =  'HD' ;
ticker46  =  'HET' ;
ticker47  =  'HIG' ;
ticker48  =  'HNZ' ;
ticker49  =  'HON' ;
ticker50  =  'HWP' ;
ticker51  =  'IBM' ;
ticker52  =  'INTC' ;
ticker53  =  'IP' ;
ticker54  =  'JNJ' ;
ticker55  =  'JPM' ;
ticker56  =  'KO' ;
ticker57  =  'LEH' ;
ticker58  =  'LTD' ;
ticker59  =  'LU' ;
ticker60  =  'NSC' ;
ticker61  =  'NSM' ;
ticker62  =  'NT' ;
ticker63  =  'NXTL' ;
ticker64  =  'ONE' ;
ticker65  =  'ORCL' ;
```

```
ticker66  =  'PEP' ;
ticker67  =  'PFE' ;
ticker68  =  'PG' ;
ticker69  =  'PHA' ;
ticker70  =  'RAL' ;
ticker71  =  'ROK' ;
ticker72  =  'RSH' ;
ticker73  =  'RTNB' ;
ticker74  =  'S' ;
ticker75  =  'SLB' ;
ticker76  =  'SLE' ;
ticker77  =  'SO' ;
ticker78  =  'T' ;
ticker79  =  'TOY' ;
ticker80  =  'TXN' ;
ticker81  =  'TYC' ;
ticker82  =  'UIS' ;
ticker83  =  'USB' ;
ticker84  =  'UTX' ;
ticker85  =  'VIAB' ;
ticker86  =  'VZ' ;
ticker87  =  'WFC' ;
ticker88  =  'WMB' ;
ticker89  =  'WMT' ;
ticker90  =  'WY' ;
ticker91  =  'XOM' ;
ticker92  =  'XRX' ;

%matrix with stddev's of each stock's prices
volatilityMatrix = [0,0];

volatilityCounter=0;
index = 1;

%92 stocks to run through
while index<=92,
    %used to look ticker1-ticker92
    currentIndexedTicker = cat(2,'ticker',num2str(index));
```

```
    %finds if stock should be excluded
    skipStock=0;
    for excludeCounter=1:length(excludeStocks),
        if index==excludeStocks(excludeCounter),
            skipStock = 1;
        end
    end

    if skipStock==1,
        index=index+1;
    else
        volatilityCounter=volatilityCounter+1;
        stockName = [eval(currentIndexedTicker),'.dat'];
        %create matrix of stock's daily prices and stddev's
        prices = daily(stockName,numOfDays);
        stockSave = [eval(currentIndexedTicker),'Daily.dat'];
        %save price matrix as i.e. AADaily.dat
        save(stockSave,'prices','-ASCII');
        %volatilityMatrix(volatilityCounter,1) = eval(currentIndexedTicker);
        %create matrix where first column if stock number, second column is
        %stddev of all its prices
        volatilityMatrix(volatilityCounter,1) = index;
        volatilityMatrix(volatilityCounter,2) = std(prices(:,2));
        save('volatilityMatrix.dat','volatilityMatrix','-ASCII');
        %keep track of which stocks are completed
        fprintf('stock %s is done.\n',eval(currentIndexedTicker));
        index=index+1;
    end
end
```

## 6.3 Splits

```
function splits()

%set ticker values
```

```
index = 1;
volatilityCounter = 0;

%92 stocks to run through
while index<=92,
    %load stocks except CSCO and INTC
    if ((index~=25)&&(index~=52)),
        currentIndexedTicker = cat(2,'ticker',num2str(index));
        stockName = [eval(currentIndexedTicker),'Daily.dat'];
        dailyMatrix = load(stockName);
    end;

    %don't include stocks that appear to have a split but don't
    if ((index==73)||(index==25)||(index==52)||(index==65)),
        split=0;
    else
        split = 1;
    end;

    while (split > 0),
        split = 0;
        previous = 1;
        splitDate=10000;

        for counter=2:length(dailyMatrix),
            %to avoid false-positive split detections
            if (dailyMatrix(previous,2)>25),
                %detect 3-2 split
                if (dailyMatrix(counter,2)-5 < dailyMatrix(previous,2)/1.5)&&
                (dailyMatrix(counter,2)+5 > dailyMatrix(previous,2)/1.5),
                    split=32;
                    splitDate=counter;
                    counter=length(dailyMatrix)+1;
                end;

                if (counter <= length(dailyMatrix)),
                    %detect 2-1 split
                    if (dailyMatrix(counter,2)-5 < dailyMatrix(previous,2)/2)&&
```

```
                (dailyMatrix(counter,2)+5 > dailyMatrix(previous,2)/2),
                    split=21;
                    splitDate=counter;
                    counter=length(dailyMatrix)+1;
                end;
            end;

            if (counter <= length(dailyMatrix)),
                %detect 3-1 split
                if (dailyMatrix(counter,2)-5 < dailyMatrix(previous,2)/3)&&
                (dailyMatrix(counter,2)+5 > dailyMatrix(previous,2)/3),
                    split=31;
                    splitDate=counter;
                    counter=length(dailyMatrix)+1;
                end;
            end;
            previous=previous+1;
        end;
    end;

    if (split > 0),
        for index2=1:length(dailyMatrix),
            price = dailyMatrix(index2,2);
            %recalculate prices by reversing splits
            if (index2>=splitDate),
                if (split==32),
                    dailyMatrix(index2,2)=price*1.5;
                end;

                if (split==21),
                    dailyMatrix(index2,2)=price*2;
                end;

                if (split==31),
                    dailyMatrix(index2,2)=price*3;
                end;
            end;
        end;
```

```
        end;
    end;
    if ((index~=25)&&(index~=52)),
        %save price matrix as i.e. AADailySplits.dat
        stockSave = [eval(currentIndexedTicker),'DailySplits.dat'];
        save(stockSave,'dailyMatrix','-ASCII');

        %save std's of prices of each stock in one matrix
        volatilityCounter=volatilityCounter+1;
        volatilityMatrix(volatilityCounter,1) = index;
        volatilityMatrix(volatilityCounter,2) = std(dailyMatrix(:,2));
        save('volatilityMatrixSplits.dat','volatilityMatrix','-ASCII');
        fprintf('stock %s is done.\n',eval(currentIndexedTicker));
    end;
    %loop to next stock
    index=index+1;
    clear dailyMatrix;
end;
```

## 6.4 Half

```
function half()
%set ticker values

index=1;
%92 stocks to run through
while index<=92,
    %load stocks except CSCO and INTC
    if ((index~=25)&&(index~=52)),
        currentIndexedTicker = cat(2,'ticker',num2str(index));
        stockName = [eval(currentIndexedTicker),'DailySplits.dat'];
        dailyMatrix = load(stockName);
        x=1;
        %split the prices into files for first 6 months and second 6 months
        for counter=1:length(dailyMatrix),
            if (counter<=126),
                half1(counter,1)=dailyMatrix(counter,1);
                half1(counter,2)=dailyMatrix(counter,2);
```

```
            end;

            if (counter>126),
                half2(x,1)=dailyMatrix(counter,1);
                half2(x,2)=dailyMatrix(counter,2);
                x=x+1;
            end;
        end;
        %save price matrix as i.e. AAPrices1.dat
        stockSave = [eval(currentIndexedTicker),'Prices1.dat'];
        save(stockSave,'half1','-ASCII');
        stockSave = [eval(currentIndexedTicker),'Prices2.dat'];
        save(stockSave,'half2','-ASCII');

        fprintf('stock %s is done.\n',eval(currentIndexedTicker));
    end;
    index=index+1;
    clear dailyMatrix;
    clear half1;
    clear half2;
end;
```

## 6.5 Length Check

```
function lengthCheck()

%set ticker values
index=1;

%85 stocks to run through
while index<=85,
    currentIndexedTicker = cat(2,'ticker',num2str(index));
    stockName = [eval(currentIndexedTicker),'Returns2.dat'];
    stock = load(stockName);
    %make sure length is 126 days
    lengthVector(index,1)=index;
    lengthVector(index,2)=length(stock)-126;
```

```
    save('lengthVector.dat','lengthVector','-ASCII');
    fprintf('%d is done.\n',index);
    index=index+1;
end;
```

### 6.6 Timeframe

```
function timeFrame()

%set ticker values
index1=1;
index2=1;
currentIndexedTicker1 = cat(2,'ticker',num2str(index1));
stock1Name = [eval(currentIndexedTicker1),'Prices2.dat'];
stock1 = load(stock1Name);

%85 stocks to run through
while index2<=85,
    currentIndexedTicker2 = cat(2,'ticker',num2str(index2));
    stock2Name = [eval(currentIndexedTicker2),'Prices2.dat'];
    stock2 = load(stock2Name);

    %check if lengths of all stocks are the same
    check(:,index2)=stock1(:,1)-stock2(:,1);

    save('check2.dat','check','-ASCII');
    fprintf('%d is done.\n',index2);
    index2=index2+1;
end;

counter1=1;
counter2=1;
checkStock=zeros(85,1);

%if length isn't the same, identify with a '1'
for counter1=1:85,
    for counter2=1:126,
        if (check(counter2,counter1)~= 0),
```

```
            checkStock(counter1)=1;
            counter2=126;
        end;
    end;
end;

save('checkStock.dat','checkStock','-ASCII');
```

## 6.7 Returns

```
function returns()

%set ticker values

index=1;
returnCounter=0;

%85 stocks
while index<=85,
    %don't include CSCO and INTC
    %if ((index~=25)&&(index~=52)),
        %load stock
        currentIndexedTicker = cat(2,'ticker',num2str(index));
        stockName = [eval(currentIndexedTicker),'Prices2.dat'];
        dailySplitsMatrix = load(stockName);


        previous=1;
        for counter=2:length(dailySplitsMatrix),
            %1st column for date, 2nd column for return
            returnMatrix(previous,1)=dailySplitsMatrix(counter,1);
            returnMatrix(previous,2)=(dailySplitsMatrix(counter,2)-
            dailySplitsMatrix(previous,2))/dailySplitsMatrix(previous,2);
            previous=previous+1;
        end;

        %save price matrix as i.e. AAReturns.dat
        stockSave = [eval(currentIndexedTicker),'Returns2.dat'];
```

```
        save(stockSave,'returnMatrix','-ASCII');

        %find mean/std of all returns for each stock, store in a matrix
        returnCounter=returnCounter+1;
        allReturns(returnCounter,1) = index;
        allReturns(returnCounter,2) = mean(returnMatrix(:,2));
        allReturns(returnCounter,3) = var(returnMatrix(:,2));
        save('allReturns2.dat','allReturns','-ASCII');

        fprintf('stock %s is done.\n',eval(currentIndexedTicker));
    %end;
    %loop to next stock
    index=index+1;
    clear dailySplitsMatrix;
    clear returnMatrix;
end;
```

## 6.8 Covariance Matrix

```
function covMatrix()

%set ticker values
covMatrix=0;
index1=1;

%85 stocks to run through
while index1<=85,
    index2=1;
    while index2<=85,
        currentIndexedTicker1 = cat(2,'ticker',num2str(index1));
        currentIndexedTicker2 = cat(2,'ticker',num2str(index2));
        stock1Name = [eval(currentIndexedTicker1),'Returns1.dat'];
        stock2Name = [eval(currentIndexedTicker2),'Returns1.dat'];
        stock1 = load(stock1Name);
        stock2 = load(stock2Name);

        %calculate covariance matrix between each stock's returns
        covValue=cov(stock1(:,2),stock2(:,2));
```

```
        covMatrix(index1,index2)=covValue(1,2);

        save('covMatrix.dat','covMatrix','-ASCII');
        fprintf('%d-%d is done.\n',index1,index2);
        index2=index2+1;
    end;
    index1=index1+1;
end;
```

## 6.9 Positive Definiteness

```
function posDef()

%num of stocks
N=85;

%define A
A=zeros(N+2,N+2);

%load covariance matrix
covar = load('covMatrix.dat');

%assign covariances to matrix A
for row=1:N,
    for column=1:N,
        A(row,column)=2*covar(row,column);
    end;
end;


%load returns
returns = load('allReturns1.dat');

%assign returns to matrix A
for row=1:N,
    A(row,N+1)=returns(row,2);
end;
```

```
for column=1:N,
    A(N+1,column)=returns(column,2);
end;


%finish creating matrix A with proper values
for row=1:N;
    A(row,N+2)=1;
end;

for column=1:N;
    A(N+2,column)=1;
end;

determinants=zeros(N,2);

%calculate principal minors
for counter=1:85,
    determinants(counter,1)=det(A(1:counter,1:counter));
end;

%make sure all principal minors are positive
for counter=1:85,
    if (determinants(counter,1)>0),
        determinants(counter,2)=1;
    end;
end;

save('A.dat','A','-ASCII');
save('determinants.dat','determinants','-ASCII');
```

## 6.10 Optimization Routine

```
function [x,variance] = optRoutine(mu)
%num of stocks
N=85;

%define A
```

```
A=zeros(N+2,N+2);

%load covariance matrix
covar = load('covMatrix.dat');

%assign covariances to matrix A
for row=1:N,
    for column=1:N,
        A(row,column)=2*covar(row,column);
    end;
end;


%load returns
returns = load('allReturns1.dat');

%assign returns to matrix A
for row=1:N,
    A(row,N+1)=-returns(row,2);
end;

for column=1:N,
    A(N+1,column)=returns(column,2);
end;


%finish creating matrix A with proper values
for row=1:N;
    A(row,N+2)=-1;
end;

for column=1:N;
    A(N+2,column)=1;
end;

%create matrix b
b=zeros(N+2,1);
b(N+1,1)=mu;
```

```
b(N+2,1)=1;

%use built-in function to identify unknown proportions and lambda's
x=A^-1*b;

variance=0;
for index1=1:85,
    for index2=1:85,
        variance=variance+x(index1)*x(index2)*covar(index1,index2);
    end;
end;
```

## 6.11 Calculate Proportions

```
function lagrange(expReturnStart,expReturnEnd,numOfReturns)
%mean rate of return
mu=expReturnStart;

for counter=1:numOfReturns,

    %set expected return
    expReturn=mu+(expReturnEnd-expReturnStart)/(numOfReturns-1)*(counter-1);

    %perform optimization
    [proportions,variance]=optRoutine(expReturn);

    %record proportions
    proportionMatrix(:,counter)=proportions;

    %record min variances
    minVariances(counter,1)=expReturn;
    minVariances(counter,2)=variance;

    allProportions(1,counter)=expReturn;
    allProportions(2,counter)=variance;
    for index=3:89,
        allProportions(index,counter)=proportions(index-2);
    end;
```

```
    fprintf('%d %d is done.\n',counter,expReturn);
end;

save('proportionMatrix.dat','proportionMatrix','-ASCII');
save('minVariances.dat','minVariances','-ASCII');
save('allProportions.dat','allProportions','-ASCII');
```

## 6.12 Future Performance

```
function futurePerformance2(expReturn)
%set ticker values

N=85;
dailyValue=zeros(126,1);
dailyReturns=zeros(125,1);
totalReturn=0;

allProportions = load('allProportions.dat');

%10000 runs to find min variance, set efficient frontier
for counter1=1:10000,
    if allProportions(1,counter1)==expReturn,
        for counter2=1:85,
            proportions(counter2,1)=allProportions(counter2+2,counter1);
        end;
        counter1=10000;
    end;
end;

clear allProportions;

for index=1:N,
    currentIndexedTicker = cat(2,'ticker',num2str(index));
    stockName = [eval(currentIndexedTicker),'Prices2.dat'];
    stockPrices = load(stockName);
    %set future prices
    futurePrices(:,index)=stockPrices(:,2);
```

```
end;


%for index=1:N,
 %   dailyValue(1) = dailyValue(1) + futurePrices(1,index)*
 (10000*proportions(index))/futurePrices(1,index);
%end;


for day=1:126,
    %calculate daily value
    for stockNumber=1:85,
        dailyValue(day)=dailyValue(day) + futurePrices(day,stockNumber)*
        (10000*proportions(stockNumber))/futurePrices(1,stockNumber);
    end;

    %calculate daily return
    if day>1,
        dailyReturns(day-1)=(dailyValue(day)-dailyValue(day-1))/
        dailyValue(day-1);
    end;

    fprintf('day %d is done.\n',day);
end;

%calculate total return
totalReturn=(dailyValue(126)-dailyValue(1))/dailyValue(1);

save('dailyValue2.dat','dailyValue','-ASCII');
save('dailyReturns2.dat','dailyReturns','-ASCII');
save('totalReturn2.dat','totalReturn','-ASCII');
```

Table 3: Portfolio Proportions Part1

| Expected Return | 0.0006 | 0.01 | 0.05 | 0.1 | 0.25 | 0.5 |
|---|---|---|---|---|---|---|
| Variance | 0 | 0.0001 | 0.0012 | 0.0046 | 0.0291 | 0.1166 |
| AA | 0.0874 | 0.1565 | 0.4491 | 0.8148 | 1.9118 | 3.7401 |
| AEP | 0.1958 | -0.6016 | -3.9768 | -8.1959 | -20.853 | -41.9482 |
| AES | -0.0279 | 0.046 | 0.3589 | 0.75 | 1.9232 | 3.8787 |
| AGC | 0.0281 | 0.0926 | 0.3658 | 0.7074 | 1.7319 | 3.4395 |
| AIG | 0.131 | 0.1009 | -0.0261 | -0.1849 | -0.6613 | -1.4553 |
| AMGN | 0.1056 | 0.0571 | -0.1484 | -0.4053 | -1.1758 | -2.46 |
| AOL | -0.0201 | -0.0481 | -0.1666 | -0.3148 | -0.7592 | -1.4998 |
| AVP | -0.0406 | 0.0357 | 0.3589 | 0.7628 | 1.9745 | 3.994 |
| AXP | -0.106 | 0.3208 | 2.1276 | 4.386 | 11.1613 | 22.4534 |
| BA | -0.0581 | -0.2099 | -0.8525 | -1.6558 | -4.0656 | -8.0818 |
| BAC | 0.0272 | -0.0619 | -0.4394 | -0.9112 | -2.3267 | -4.6857 |
| BAX | -0.0989 | -0.064 | 0.0839 | 0.2687 | 0.8232 | 1.7474 |
| BCC | 0.0496 | 0.4991 | 2.4017 | 4.7799 | 11.9147 | 23.806 |
| BDK | 0.0062 | 0.2138 | 1.0922 | 2.1902 | 5.4843 | 10.9744 |
| BHI | 0.064 | 0.1639 | 0.5867 | 1.1153 | 2.701 | 5.3439 |
| BMY | -0.0034 | -0.012 | -0.0486 | -0.0944 | -0.2316 | -0.4604 |
| BNI | 0.1143 | 0.1375 | 0.2356 | 0.3583 | 0.7264 | 1.3398 |
| C | -0.0992 | 0.067 | 0.7707 | 1.6503 | 4.2889 | 8.6868 |
| CCU | 0.074 | 0.011 | -0.256 | -0.5897 | -1.5907 | -3.2591 |
| CI | -0.0291 | 0.1837 | 1.0843 | 2.2101 | 5.5874 | 11.2163 |
| CL | 0.1189 | -0.1503 | -1.2896 | -2.7139 | -6.9866 | -14.1077 |
| CPB | 0.0418 | 0.115 | 0.4247 | 0.8119 | 1.9735 | 3.9094 |
| CSC | 0.0354 | 0.0993 | 0.3698 | 0.7079 | 1.7222 | 3.4128 |
| DAL | 0.0373 | -0.0656 | -0.5015 | -1.0463 | -2.6807 | -5.4048 |
| DD | -0.0044 | -0.3856 | -1.9991 | -4.016 | -10.0666 | -20.1509 |
| DIS | -0.008 | 0.0584 | 0.3393 | 0.6905 | 1.7441 | 3.5 |
| DOW | 0.1145 | 0.4141 | 1.6822 | 3.2674 | 8.0231 | 15.9491 |
| EK | -0.0472 | -0.129 | -0.4754 | -0.9084 | -2.2073 | -4.3722 |
| EMC | 0.04 | 0.0797 | 0.2478 | 0.4579 | 1.0883 | 2.139 |
| ENE | 0.0934 | 0.2248 | 0.7809 | 1.4761 | 3.5615 | 7.0372 |
| EPG | -0.0139 | -0.0764 | -0.3414 | -0.6725 | -1.6659 | -3.3217 |
| ETR | -0.0823 | -0.246 | -0.9393 | -1.8058 | -4.4055 | -8.7384 |
| EXC | -0.0084 | 0.3212 | 1.7161 | 3.4597 | 8.6906 | 17.4087 |
| F | -0.0387 | -0.3714 | -1.7798 | -3.5402 | -8.8215 | -17.6237 |
| FDX | 0.045 | 0.0741 | 0.1976 | 0.3519 | 0.8149 | 1.5865 |
| G | 0.0296 | 0.0825 | 0.3066 | 0.5868 | 1.4271 | 2.8278 |
| GD | 0.068 | -0.0052 | -0.3151 | -0.7025 | -1.8647 | -3.8017 |
| GE | -0.1319 | -0.3156 | -1.0935 | -2.0657 | -4.9825 | -9.8438 |

Table 4: Portfolio Proportions Part2

| Expected Return | 0.0006 | 0.01 | 0.05 | 0.1 | 0.25 | 0.5 |
|---|---|---|---|---|---|---|
| Variance | 0 | 0.0001 | 0.0012 | 0.0046 | 0.0291 | 0.1166 |
| GM | 0.0118 | 0.0107 | 0.0063 | 0.0008 | -0.0159 | -0.0436 |
| GX | -0.0134 | -0.0453 | -0.1802 | -0.349 | -0.8552 | -1.6989 |
| HAL | -0.1254 | -0.2098 | -0.567 | -1.0134 | -2.3526 | -4.5847 |
| HCA | 0.0484 | -0.0793 | -0.62 | -1.2958 | -3.3233 | -6.7025 |
| HD | 0.1357 | -0.0979 | -1.0864 | -2.3221 | -6.0292 | -12.2077 |
| HET | -0.0749 | 0.0571 | 0.6154 | 1.3134 | 3.4074 | 6.8973 |
| HIG | -0.0346 | 0.0245 | 0.2745 | 0.5871 | 1.5246 | 3.0872 |
| HNZ | 0.0658 | 0.1463 | 0.4873 | 0.9134 | 2.1919 | 4.3227 |
| HWP | -0.0355 | -0.1293 | -0.5263 | -1.0225 | -2.5113 | -4.9925 |
| IBM | 0.0722 | 0.2001 | 0.7416 | 1.4184 | 3.4489 | 6.833 |
| IP | 0.0535 | -0.0713 | -0.5994 | -1.2595 | -3.2398 | -6.5404 |
| JNJ | 0.024 | 0.5907 | 2.9894 | 5.9877 | 14.9826 | 29.9741 |
| JPM | -0.024 | 0.0934 | 0.5903 | 1.2114 | 3.0746 | 6.18 |
| KO | -0.0049 | 0.0454 | 0.2586 | 0.5251 | 1.3245 | 2.657 |
| LEH | 0.0308 | -0.1338 | -0.8305 | -1.7014 | -4.314 | -8.6684 |
| LTD | -0.0002 | 0.3181 | 1.6656 | 3.3499 | 8.4029 | 16.8246 |
| LU | -0.0062 | -0.1702 | -0.8647 | -1.7328 | -4.3372 | -8.6778 |
| NSC | -0.0536 | -0.1121 | -0.3593 | -0.6683 | -1.5954 | -3.1406 |
| NSM | 0.0529 | 0.057 | 0.0744 | 0.096 | 0.161 | 0.2693 |
| NT | -0.036 | 0.3492 | 1.9798 | 4.018 | 10.1326 | 20.3236 |
| NXTL | 0.061 | 0.1897 | 0.7348 | 1.4161 | 3.46 | 6.8665 |
| ONE | -0.0979 | -0.1278 | -0.2545 | -0.4129 | -0.888 | -1.6798 |
| ORCL | -0.0482 | -0.0994 | -0.3162 | -0.5872 | -1.4002 | -2.7551 |
| PEP | 0.0632 | 0.0376 | -0.0706 | -0.2059 | -0.6118 | -1.2882 |
| PFE | 0.069 | -0.1412 | -1.0313 | -2.1439 | -5.4816 | -11.0445 |
| PG | 0.0259 | 0.0506 | 0.1551 | 0.2857 | 0.6774 | 1.3303 |
| PHA | 0.0138 | 0.0573 | 0.2412 | 0.471 | 1.1607 | 2.3101 |
| RAL | 0.022 | -0.204 | -1.1607 | -2.3565 | -5.944 | -11.9232 |
| ROK | -0.0747 | 0.0264 | 0.454 | 0.9885 | 2.592 | 5.2646 |
| RSH | 0.0303 | 0.1925 | 0.8789 | 1.7369 | 4.3109 | 8.601 |
| S | -0.0574 | -0.2854 | -1.2503 | -2.4565 | -6.0749 | -12.1057 |
| SLB | 0.0628 | 0.0893 | 0.2015 | 0.3417 | 0.7625 | 1.4637 |
| SLE | 0.0304 | 0.1497 | 0.6549 | 1.2864 | 3.1807 | 6.338 |
| SO | 0.2011 | 0.4247 | 1.371 | 2.5538 | 6.1024 | 12.0168 |
| T | -0.0246 | -0.1098 | -0.4703 | -0.921 | -2.273 | -4.5265 |
| TOY | -0.0536 | -0.0859 | -0.2222 | -0.3927 | -0.9041 | -1.7564 |
| TXN | -0.0081 | 0.0991 | 0.5527 | 1.1197 | 2.8207 | 5.6557 |
| TYC | -0.0935 | -0.0104 | 0.3416 | 0.7815 | 2.1013 | 4.3009 |

Table 5: Portfolio Proportions Part3

| Expected Return | 0.0006 | 0.01 | 0.05 | 0.1 | 0.25 | 0.5 |
|---|---|---|---|---|---|---|
| Variance | 0 | 0.0001 | 0.0012 | 0.0046 | 0.0291 | 0.1166 |
| UIS | 0.0088 | -0.0264 | -0.1755 | -0.362 | -0.9212 | -1.8533 |
| USB | 0.0505 | -0.1832 | -1.1727 | -2.4095 | -6.12 | -12.3042 |
| UTX | -0.0644 | -0.204 | -0.7949 | -1.5336 | -3.7496 | -7.443 |
| VZ | 0.0868 | -0.0409 | -0.5812 | -1.2567 | -3.2832 | -6.6606 |
| WFC | -0.0637 | -0.1392 | -0.4588 | -0.8582 | -2.0564 | -4.0534 |
| WMB | 0.0553 | 0.0851 | 0.2116 | 0.3697 | 0.844 | 1.6345 |
| WMT | 0.0217 | -0.0255 | -0.2252 | -0.4748 | -1.2237 | -2.4718 |
| WY | 0.0013 | -0.1365 | -0.72 | -1.4493 | -3.6374 | -7.2841 |
| XRX | 0.0067 | -0.2379 | -1.2731 | -2.5671 | -6.4491 | -12.9191 |

Table 6: Future Performance

| Expected Return | Starting Value | Ending Value | Value Difference | Actual Return |
|---|---|---|---|---|
| .055% | $10000 | $8711.52 | -$1288.48 | -12.88% |
| 1% | $10000 | $17484.11 | $7484.11 | 74.84% |
| 5% | $10000 | $54616.77 | $44616.77 | 446.17% |
| 10% | $10000 | $101032.58 | $91032.58 | 910.33% |
| 25% | $10000 | $240280.04 | $230280.04 | 2302.80% |
| 50% | $10000 | $472359.14 | $462359.14 | 4623.59% |