The Cooper Union for the Advancement of Science and Art

An Optimal NYSE Stock Portfolio Utilizing the Sharpe Ratio

By

Grant Aarons & William Biesiadecki

Professor Benjamin Davis EID488 - Fall 2013

Abstract

This report aims to determine optimal asset allocation in a portfolio of stocks using the Sharpe ratio. In search of greater returns on their investments, all investors must take on additional portfolio volatility. Stocks considered in this optimization project were retrieved from a five year historic New York Stock Exchange (NYSE) data service.

The Sharpe ratio objective function, closely resembles a linear regression problem with a sum of residuals leading to a quadratic variance term. Minimizing volatility is equivalent to minimizing residuals for a linear regression. The diversity of stock allocation must be linearly constrained. The optimization routine currently implements MATLAB's fmincon solver and handles the quadratic programming problem (QP) well with certain input parameters, such as number of stocks considered and number of days in the time series to be regressed.

Quadratic programming problems (QPs) and specifically this Sharpe ratio problem can take an extraordinary amount of computational time when input parameters are applied irrationally. An increase to the number of days in the historical time series, which causes a couple of extra residual calculations, actually decreases the computational time. However, when we increase the number of stocks initially considered we see an exponential rise in the necessary solver computational time (Table 1).

Table 1: Computation Time With Respect To Input Parameters*

Time [s]	Days in Series	Considered Stocks	Allocated Stocks
182	40	200	$15-20$
174	80	200	$15-20$
173	120	200	$15-20$
356	40	400	$15-20$
435	40	600	$15-20$

In our report we work to understand how the days in the historical time series and the number of stocks in which we consider investment allocation may effect the solver computation time. Increasing the historical series and the number of stocks considered helps the portfolio track long market trends and greater diversify the investment of assets, but seemingly misses out on short term trends. Most importantly the portfolio needs to select optimal stock choices from the entire NYSE market, but the challenge remains in finding an optimal solution in a reasonable time frame if all NYSE stocks are initially considered. With this in mind we explore possible filters that selectively sample the broad stock market and potentially realize greater returns from an entire market representation in the optimization routine.

Contents

1 Introduction

In the past and still lingering around today is the notion that particular asset managers have enhanced skills in heuristic judgment and can consistently bet big and at the right moment in time, for their respective client portfolios. This notion has been crumbling away in the wake of the 1970s stagflation period, and crises like the most recent financial collapse of 2008. In the 1970s, unhappy pension fund managers learned this lesson the hard way by failing to outperform the market, despite their major expense of time and effort. [3] In response, some of these pension fund managers began investing in indexed funds that track a particular market, such as the S&P500, while concurrently pinning investment risk to the quantifiable broad market risk. Even the most successful old school asset managers overlook what investors in crisis desperately demand, and that is a quantifiable understanding of the current risk-reward tradeoff. Investors want to understand the fluctuations in their portfolio's value. In the words of John Maynard Keynes "Worldly wisdom teaches that it is better for reputation to fail conventionally, than to succeed unconventionally." [6]

Modern Portfolio Theory (MPT) is a field of work that was primarily motivated by the *Journal of Finance* article published in 1952, titled "Portfolio Selection" and written by Harry Markowitz. In "Portfolio Selection" Markowitz develops one of the first mathematical models to relate covariant risk of an investment allocation to the total return of that investment. [5] In this way Markowitz was able to introduce the notion of an efficient frontier, which could be used as a selection criteria in portfolios. In Modern Portfolio Theory, as it is now termed, the establishment of an efficient frontier gives the average investor a concrete sense of the tradeoff between risk and reward. William F. Sharpe expanded upon the work of Harry Markowitz, and developed a scalar value function that relates expected return to the level of risk in a portfolio. [8] One extremely popular visualization that grew out of the Sharpe ratio, is the efficient frontier, which maps the spectrum of expected return values and the level of risk in a portfolio. The Sharpe Ratio is represented by: [1] [8]

$$S = \frac{E(R_p - R_f)}{\sigma_p} \tag{1}$$

Where:

$E(R_p - R_f)$: represents the difference between the expected return of the portfolio and the expected return of a "risk-free" asset (such as the US Treasury or cash)

R_f: the expected return of a "risk-free" asset is set equal to zero. We are using cash as our risk-free asset and holding cash does not produce any return

σ_p: represents the standard deviation in the values of time series points and is the primary metric of volatility in (1)

We select a level of diversification to measure the risk in our portfolio, maximum percent allocation into one particular stock, and use the allocation of assets specified by an optimal Sharpe ratio. Every decrease in diversification, increase of risk and maximum percent allocation into one stock, has an associated optimal return. The mapping of optimal return for the sweep of diversification risks gives us the efficient frontier.

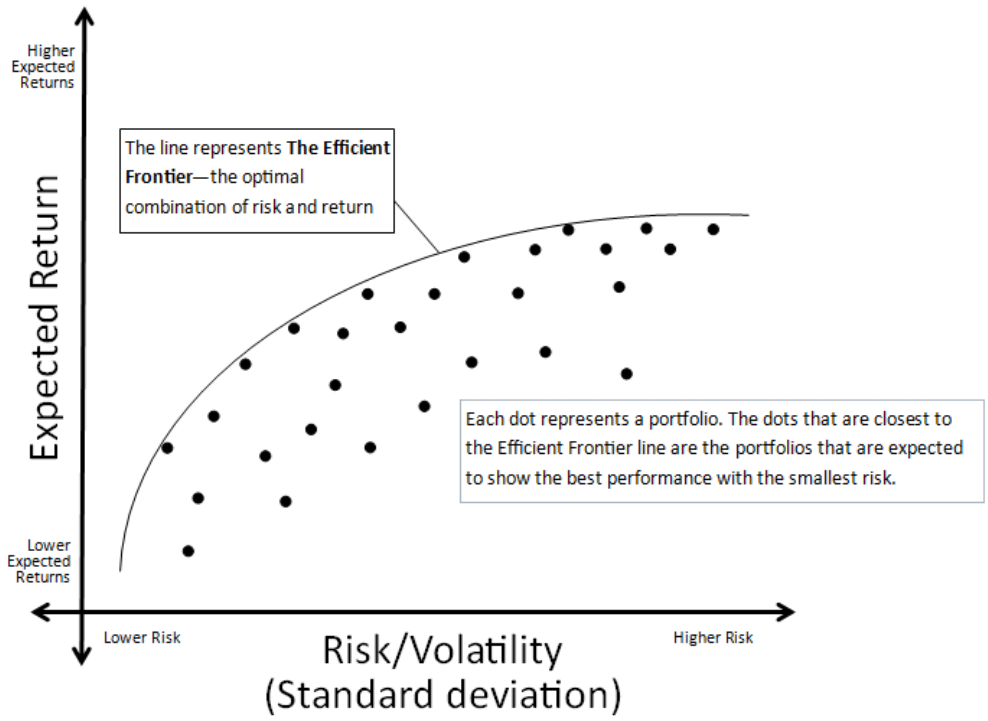

Figure 1: Efficient Frontier

Ideally an asset manager should only be investing at points along this efficient frontier, because these points represent the optimal return for a given level of risk. The efficient frontier is the type of quantifiable risk-reward tradeoff that wise investors expect from their asset managers. Investors feel safer with loses that can be justified conventionally through an empirical tradeoff curve, than with loses incurred from the heuristic judgment of a single manager in today's data crazed world of trading.

It is entirely unrealistic to believe that asset managers can heuristically parse the growing mountain of financial market data without tools like the efficient frontier. Heuristics of the mind and fundamental market knowledge are no-longer good enough to protect the wealth that clients trust asset managers to develop further. Asset managers in this day and age must use computerized optimization routines to best protect the wealth of, and meet the needs of, all their investors. [6] Early "Modern Portfolio Theory" and the work of William Sharpe was recognized in economic literature dating back to 1995 as being advantageous to Wall Street portfolio managers. The authors of this particular 1995 article, Samuelson and

Nordhaus, conclude "every good portfolio manager on Wall Street uses these techniques to bolster his or her intuition in choosing stocks and bonds." [7] Not only is MPT helping to serve investors' sense of conventionality, but it also helps to curb the enthusiasm and focus the occasionally misaligned heuristics of a portfolio manager.

The Sharpe ratio embodies the trade off between portfolio return and portfolio volatility. In our project we implement the Sharpe ratio on a variety of stocks from the New York Stock Exchange (NYSE), and focus on how the expected return of our stock portfolio varies with the volatility risk of our portfolio. Quite current in the investing world is the flow of capital into mutual and index fund portfolios. After sustaining massive losses in the crises of '08, investors are again eager to invest and yet keep risk moderately low. It is tough for investors to look at the equity market's great performance, boosted by accommodative monetary policy, and not want at least a small piece of that action. We believe the current footing of the market is ripe for mutual and index funds. Investors want to receive some of the substantial market growth, but we are all still wary of the volatility seen in the last crises; volatility that has not yet faded from our memories. If investors are going to take on a state of elevated risk it has to be quantifiable, now more than ever before.

With an overwhelming demand for quantitative risk reward tradeoff schemes we would like to assess the Sharpe ratio allocation model. We will observe the efficient frontier, as it is constructed from varying our diversification measure of risk. As we loosen the proportion of a single stock that we can hold, we decrease our diversification and increase the risk in our portfolio. The decreased diversification can help really lift the Sharpe ratio, and this is what the efficient frontier depicts.

We also wanted to understand the computational time dependence, given certain input parameters. We studied the effect of changing the number of days observed in the time series and the number of stocks being considered.

2 Problem Description

2.1 Data Mining

In order to evaluate the performance of multiple stocks over given time periods, we needed to obtain a lengthy series of data. We were able to obtain raw comma-separated value (csv) documents that detailed the end of day trading prices of all stocks listed on a given day for the New York Stock Exchange (NYSE). The online vendor had to process these files and we waited about a day for them, but this was just the start. [10] After receiving the csv data files we quickly moved to optimize portfolios over time horizons of greater than 20 days. Within 20 days many stocks are added and removed from the exchange. Ordering a matrix of prices for all of our stocks over the extended period of about 5 years in data we had purchased quickly became a daunting task. We created an algorithm to handle the parsing of daily csv trading data that would be compiled in a centralized matrix of all stocks ever traded in the 5

year period. More information on how we built our compiler can be found in the Appendix Section 1, and the exact MATLAB script can be found in Appendix Section 4.

We value the steps taken to organize the considerable time series of data. When this optimization is executed over an extended and rolling time frame, having the 5 year data set will be tremendously important in testing of our optimal portfolio techniques. Having an extended matrix of prices, also enabled us to observe multiple periods in the economy discretely. We could choose to observe our optimization routine, on any month of daily price close data from January 2008 through October 2013. That set of data is being kept and can be produced at request.

2.2 Formulation

It was very helpful at first glance to see the formulation of this Sharpe ratio problem, as it was posed by a prior Cooper Union student Jorge Aguerrevere (ChE '12). [1] In getting this problem formulated there were many vector form substitutions we had to rationalize for ourselves and we contemplated the variance calculation many times. To bring the formulation to light, let us start with the Sharpe ratio and begin defining where some of the optimization variables and parameters might live. Recall (1) from page 1 of the report:

$$S = \frac{E(R_p - R_f)}{\sigma_p} = \frac{E(R_p)}{\sigma_p}$$

We also know that the Sharpe ratio being optimized, should be a scalar, and this means that both σ_p and $E(R_p)$ are scalar quantities. R_f is set to zero. We will now create the vector quantities, and pose our problem to maximize the value of S.

2.2.1 Problem Set Up

The expected portfolio return is the linear combination of the vector $c^\top$ with the n-column vector x, where we have defined the new vector c to represent the total return of each individual stock over the entire time series being considered. $c \epsilon \Re^{n \times 1}$. The daily return for each individual stock is constructed by taking the difference of the current day's price and the starting day price and then dividing it by the starting day price. The daily return matrix $R \epsilon \Re^{m \times n}$. The returns on an individual day can be expressed by a single row from the R matrix. The stock returns for a single day are expressed as the r_i row vector; $r_i \epsilon \Re^{1 \times n}$, $\forall i = 1, .., m$. More clarification on the design of the matrices c and R, can be found in the Appendix Section 2.

$$\text{Total Return} = E(R_p) = c^\top x \tag{2}$$

$$\text{Daily Return} : \Re^{n \times 1} \to \Re \text{ defined by } r_i x \tag{3}$$

$$\text{Variance} = \sum_{i=1}^{m\ days} (Residuals_i)^2 \tag{4}$$

$$\text{Variance} = \sum_{i=1}^{m} ((rx)_i - c^\top x)^2 \tag{5}$$

$$\text{Variance} = \sum_{i=1}^{m} [(rx)_i - c^\top x)((rx)_i - c^\top x)] \tag{6}$$

$$\text{Variance} = \sum_{i=1}^{m} [(rx)_i (rx)_i - (rx)_i (c^\top x) - (c^\top x)(rx)_i + (c^\top x)(c^\top x)] \tag{7}$$

$$\text{Variance} = \sum_{i=1}^{m} [(x^\top r^\top)_i (rx)_i] - \sum_{i=1}^{m} [(rx)_i (c^\top x) - (c^\top x)(rx)_i] + \sum_{i=1}^{m} [(c^\top x)(c^\top x)] \tag{8}$$

$$\text{Variance} = x^\top R^\top Rx - \sum_{i=1}^{m} [(x^\top crx)_i + (x^\top r^\top c^\top x)_i] + mx^\top cc^\top x \tag{9}$$

$$\text{Variance} = x^\top R^\top Rx + mx^\top cc^\top x - 2\sum_{i=1}^{m} [(x^\top crx)_i] \tag{10}$$

$$\text{Variance} = x^\top R^\top Rx + mx^\top cc^\top x - 2(x^\top c)\sum_{i=1}^{m} [r_i]x \tag{11}$$

$$c^\top = \sum_{i=1}^{m} [r_i] \tag{12}$$

$$\text{Variance} = x^\top R^\top Rx + mx^\top cc^\top x - 2(x^\top cc^\top x) \tag{13}$$

$$\text{Variance} = x^\top R^\top Rx + (m-2)x^\top cc^\top x \tag{14}$$

After defining the parameter matrices R and c, and how they relate to the total return of the portfolio and variance of the portfolio we were able to set up the final optimization problem.

$$\text{Problem 1} = \max_{x \epsilon \Omega} \frac{c^\top x}{\sqrt{x^\top R^\top Rx - (m-2)x^\top CC^\top x}} \tag{15}$$

Such that:

$$\Omega = \{x \epsilon \Re^{n \times 1} : 1^\top x = 1, 0 \leq x \leq 1\}$$

Ω represents the compact set for this problem on which there are linear inequality and equality constraints on x. The n-column vector x represents the proportion of portfolio investment that go to each of n stock choices. The matrix 1 is a constant ones column vector and lives in $\Re^{n\times 1}$. $1^\top x$ is equivalent to the verbal statement that the sum of all the investment proportions must total to 1. The inequality constraints on x tell us that all the elements of the vector are non-negative, and that all the elements of the n-column vector of x are bounded above by 1.

Where:

R = m x n dimension matrix with elements corresponding to the daily returns from a stock j, from day i to i+1; j = 1 ... n and i = 1 ... m

r_i = 1 x n dimension matrix with elements corresponding to the daily return on day i, from a stock j; j = 1 ... n

c = n-column vector with every element corresponding to the sum of return for a given stock j over the given time series; j = 1 ... n

x = n-column vector for the proportion allocation of every stock j to be included in the portfolio; j = 1 ... n

m = number of days in the time series, time horizon

(See Appendix Section 2 for information on generating the parameter matrices, R and c)

In optimization routines it is common to minimize the objective function over the constrained set. In addition, all of our techniques are developed with respect to minimization. Taking the minimum of our Problem 1 argument multiplied by negative one, is equivalent to taking the maximum of our original Problem 1 argument. The equivalent form of the problem stated in (15) follows:

$$\text{Problem 2} = \min_{x\epsilon\Omega} \frac{-c^\top x}{\sqrt{x^\top R^\top R x - (m-2)x^\top C C^\top x}} \tag{16}$$

Quick note on shorting: The constraint on the decision variables puts into effect an upper and lower bound. Using 0 as a lower bound simply limits the problem to the situation where money is being spent on stocks, and no stocks are being shorted (shorting a stock means borrowing a stock and selling it, expecting that the stock's value will drop and it will be available for purchase at a lower price in the future). Shorting will not be considered in the portfolios in this project, because the risks involved in shorting stocks are very different to those involved in purchasing stocks. A portfolio that only has purchased stocks, in the worst-case scenario will lose its value completely. The value of the stock is bounded below. A portfolio that involves shorted stocks can leave the investor in debt, because there is no upper bound on the stock's value and, thus a shorted stock can result in a loss larger than the initial investment. [1]

2.2.2 Problem Analysis

This portfolio optimization problem utilizes the Sharpe ratio. The Sharpe ratio as we have just posed it is a quadratic programming problem subject to linear constraints on the variable x. The objective function of this problem is quadratic, because just as with linear regression, the denominator of this function aims to sum together the residuals about a constant sloped line. The function uses variance to assess risk in the portfolio allocation, and variance is inherently a quadratic sum. Variance is the square of all the residuals and must be impartial to both the positive and negative half-spaces. The non-linearity stemming from the variance calculation really slows down the fmincon algorithm, but the problem can be reformulated into a quadratically constrained linear programming problem that could possibly speed up the computation. This reformulation is discussed in the Appendix section 3, but is not implemented because MATLAB does not provide quadratically constrained solvers. We continue to use fmincon for this optimization, and look to other methods, such as filtering, to reduce the computation time.

The objective function has the variable x in the numerator, and the square root of this variable x in the denominator. The objective function handles both the assessment of risk and the assessment of return. This is a very nice property of optimizing with respect to the Sharpe ratio. The constraints in this question are only used to ensure that the set is compact. The set over which the objective function acts must be convex because the Rx matrix, representing daily return for the entire portfolio is squared and added to a multiple of the $c^\top x$ scalar squared. The sum of two squared quantities is always positive and the square root operator always acts on this positive quantity in the denominator of the objective function. The objective function is defined for the linearly constrained variable vector x and will always exist as the problem has been posed.

2.2.3 Existence Proof

To prove the existence of a solution to the Sharpe ratio problem, we invoke Weierstrass' theorem. Weierstrass' says that if the objective function is a continuous mapping of the set Ω to $\Re$ values, where $\Omega \epsilon \Re^n$ is a compact set, then $\exists$ a point $x^* \epsilon \Omega$ such that $f(x^*) \leq f(x)\ \forall x \epsilon \Omega$.

We have shown in the problem analysis section that the value in the square root operator is always positive. The denominator of the function is therefore always real valued and continuous. The numerator of the function is also real valued and continuous, because it is a linear combination of real values. Finally, the objective function is always real valued and continuous, because it is a fraction of two real values.

In order to satisfy the compactness required by Weierstrass', the objective function scalar value output must be both continuous and bounded. The output of the objective function is bounded above and below, such that $f(x) \leq a$ and $f(x) \geq b$, where a and b are scalar values.

2.2.4 Method of Solving

[9]

MATLAB has a general purpose solver, that can handle both linear and non-linear programming problems. The fmincon algorithm is implemented and accepts the linear constraints that we have posed. The fmincon algorithm is great for beginners because it will assess what type of "scaled" algorithm to apply to a particular problem, based on the parameter matrices. The fmincon algorithm has two large-scale algorithms, used for functions taking advantage a very sparse matrices. An algorithm is large scale when it uses linear algebra that does not need to store, nor operate on, full matrices. This may be done internally by storing sparse matrices, and by using sparse linear algebra for computations whenever possible. MATLAB says to "Choose a medium-scale algorithm to access extra functionality, such as additional constraint types, or possibly for better performance". A trust-region reflective algorithm may be implemented if the gradient of the function is accessible. Interior-point algorithms in fmincon, quadprog, and linprog have many good characteristics, such as low memory usage and the ability to solve large problems quickly. The answers to interior-point may be less accurate because the Interior point algorithm stays clear of the barrier functions, i.e. the boundary constraints. Trust-region-reflective does not solve problems with the linear constraints we have specified, and as such MATLAB is implementing the Active-set algorithm, which is outdated and slower than most other models. We will attempt to improve this clear inefficiency in the future, when necessary.

3 Results

3.1 Solution to Portfolio Allocation

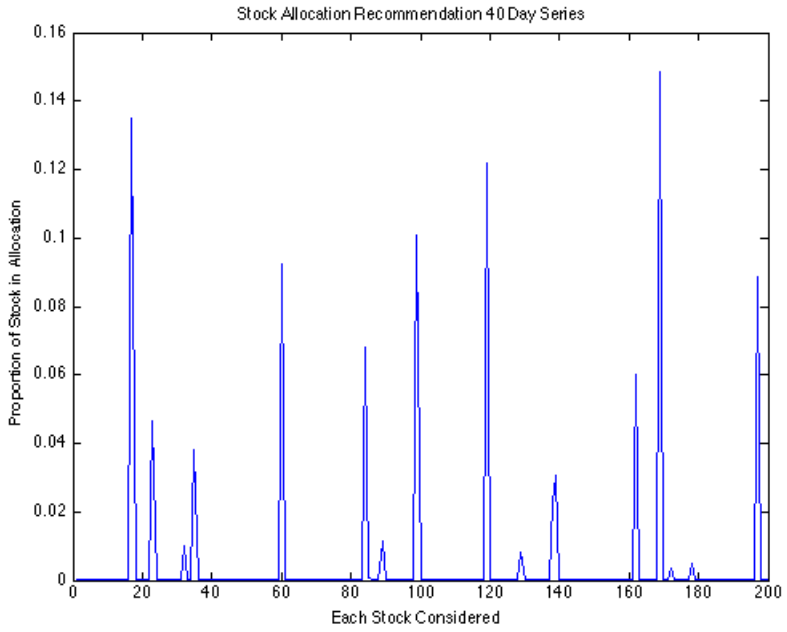

Figure 2: 200 Stocks and 40 Day Time Series

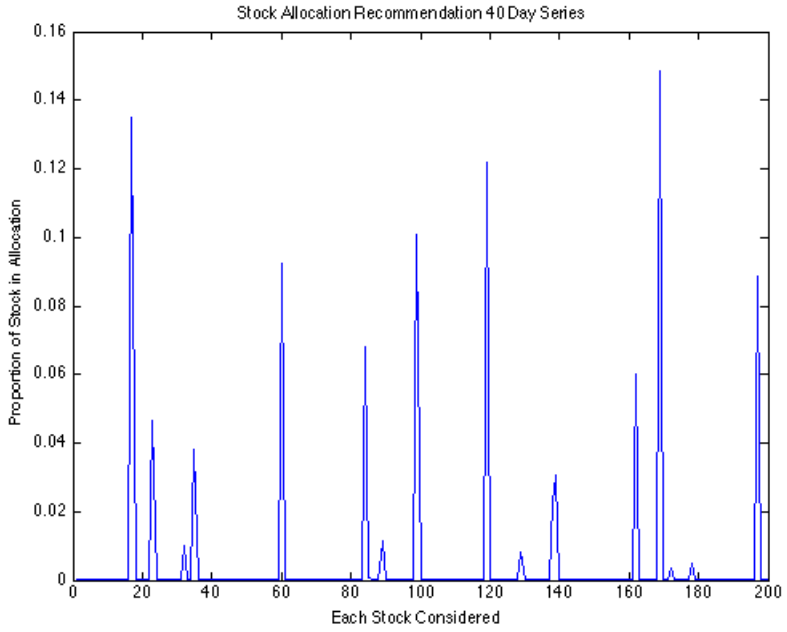

Figure 3: 200 Stocks and 80 Day Time Series

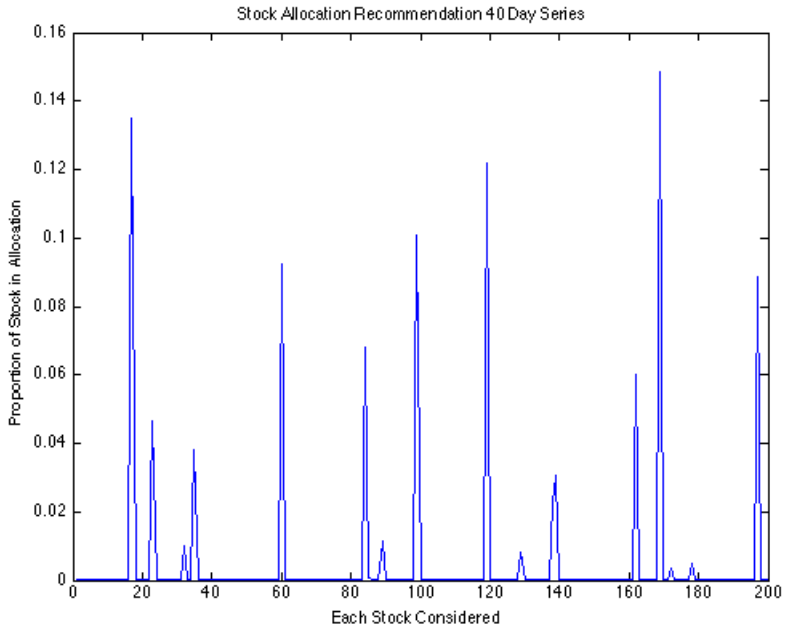

Figure 4: 200 Stocks and 120 Day Time Series

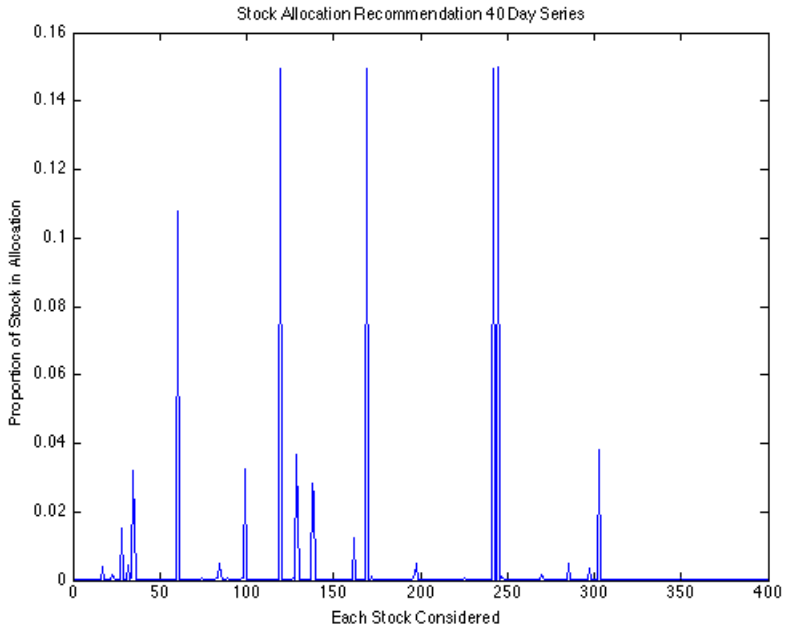

Figure 5: 400 Stocks and 40 Day Time Series

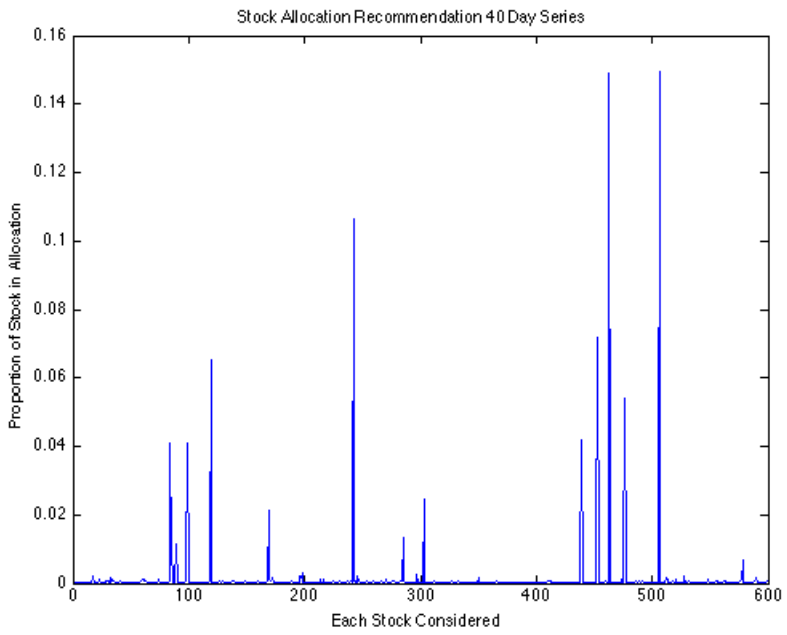

Figure 6: 600 Stocks and 40 Day Time Series

Table 2: Computation Time With Respect To Input Parameters*

Time [s]	Days in Series	Considered Stocks	Allocated Stocks
182	40	200	$15-20$
174	80	200	$15-20$
173	120	200	$15-20$
356	40	400	$15-20$
435	40	600	$15-20$

*Maximum individual stock tolerance of 15% used in above cases

Quadratic programming problems (QPs) and specifically this Sharpe ratio problem can take an extraordinary amount of computational time when input parameters are applied irrationally. Computational time decreases with each additional day considered in the time series. It is easier for the optimization routine to find a solution to longer term trends than it is to solve for short run solutions. The decrease in computation time for longer time series, may be the result of less well-performing stocks in the long run. If longer run trends are solved faster by the optimization, that could mean a better search direction and/or a smaller set of good Sharpe ratio candidates in the long run. The computation of an extra

day's residual for an increase of days in the time series, does not seem to be a source in slowing down the optimization routine. The simulation time needed for 200 initial stocks considered was marginally less in the case of an 120 day time series than in the case of the 80 day series, and the 80 day series takes marginally less computation time than the 40 day time series. The difference in computational time from the 120 day series and 40 day series is less than 10 seconds. This tells us that within the constructs of this problem we will be able to look at very large time horizons if we so deem it necessary.

Regardless of the number of stocks initially being considered for allocation, the number of stocks that are chosen for optimality are fairly similar. If choosing to allocate an investment over 200 stocks in a 40 day series, the optimization chooses a total of 16 stocks for allocation, as can be seen by the peaks in figure 2 on page 9. Both the 80 day and 120 day series considering 200 stocks have 16 peaks. If the optimization had to pick stocks from an initial group of 400 stocks, still in a 40 day series, it seems fairly plausible that the algorithm would find better stock choices. Interestingly the number of stocks chosen for allocation remains at 16; looking at the sizable peaks in the plot. The 16 peaks in figure 5 on page 10, are interesting because we have just doubled the range of stock choices, and yet the optimization chooses only 16 stocks for allocation. Figure 6 on page 11 also validates this finding. In this plot there are 18 peaks, but many of them are small might well continue dying out as the number of stock being initially considered increases to the 3000+ stocks on the NYSE. Small peaks of allocation may also die out if a longer time horizon is chosen.

The consistently low number of stocks chosen for allocation tells us something about the solution to this very real world problem. The solutions show that if you increase the number of stocks initially considered, the optimization will somehow continue to allocate into only a small number of stock choices. This trend is very strange in the context of two increases of 200 stocks for initial consideration in the solver. We postulate that the optimization only needs to chose about 10-30 well performing stocks to reduce the variance considerably. The appropriate diversification of stocks, such that the variance is reduced, does not need the full spectrum of additional initial stocks being offered. In essence the solver is being handed far too many stocks for the reduced variance and increased return it is trying to accomplish. We see that the optimization routine takes about twice the computation time to sift through 600 initial stocks then it does for the 200 initial stock case.

3.2 Efficient Frontier

3.2.1 Varied Stocks Considered

Figure 7 shows the efficiency frontiers for various sets of data over a 40 day history. These stocks were the first of the set of stocks ordered alphabetically by ticker symbol. This was chosen for convenience, and as a semi-random collection of stocks (in terms of performance) over this time period. Large sets of stocks proved to be infeasible to solve in a reasonable amount of time, and so smaller data sets were considered to characterize the sensitivity of the solution.

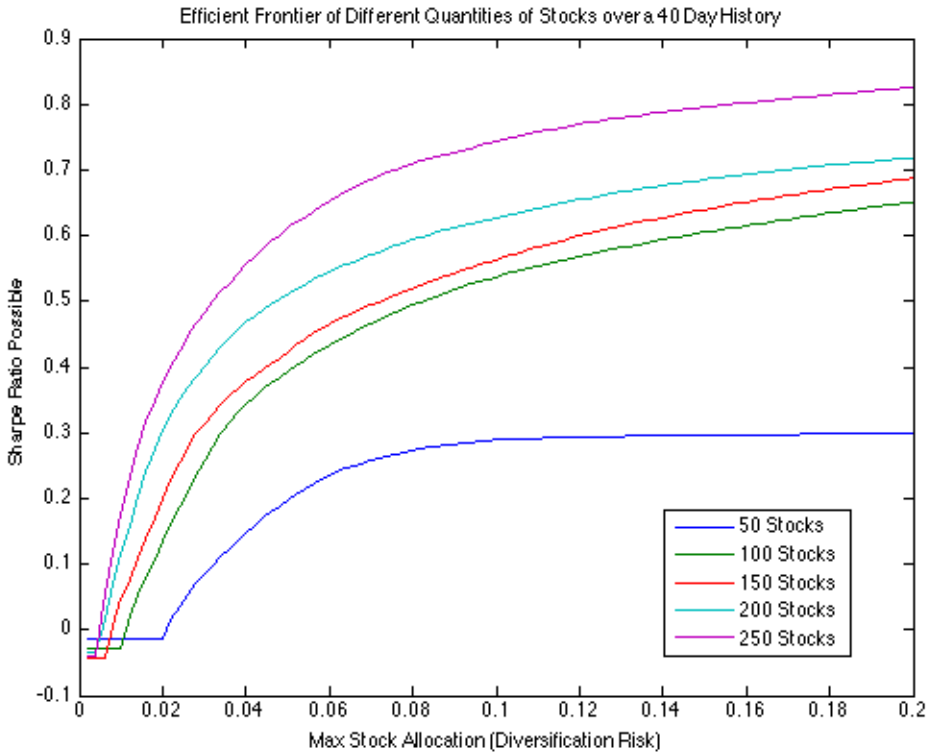

Figure 7: Efficient Frontier

The simulated efficient frontiers in figure 7 show a diminishing return in Sharpe Ratio as diversification risk is increased; as is expected from the theory of efficiency frontiers. Diversification risk is set by varying the maximum allowable stock allocation in the portfolio. Finding the efficiency frontier for various subsets of stocks over this specific time history, provides insight into a portfolios sensitivity to the increased diversification risk. For all subsets of stocks between 50 and 250 stocks initially considered, there is a diminishing return in the Sharpe ratio once the diversification risk is pushed past about 10%, providing a value that can be used for future simulations.

3.2.2 Varied Time Series Considered

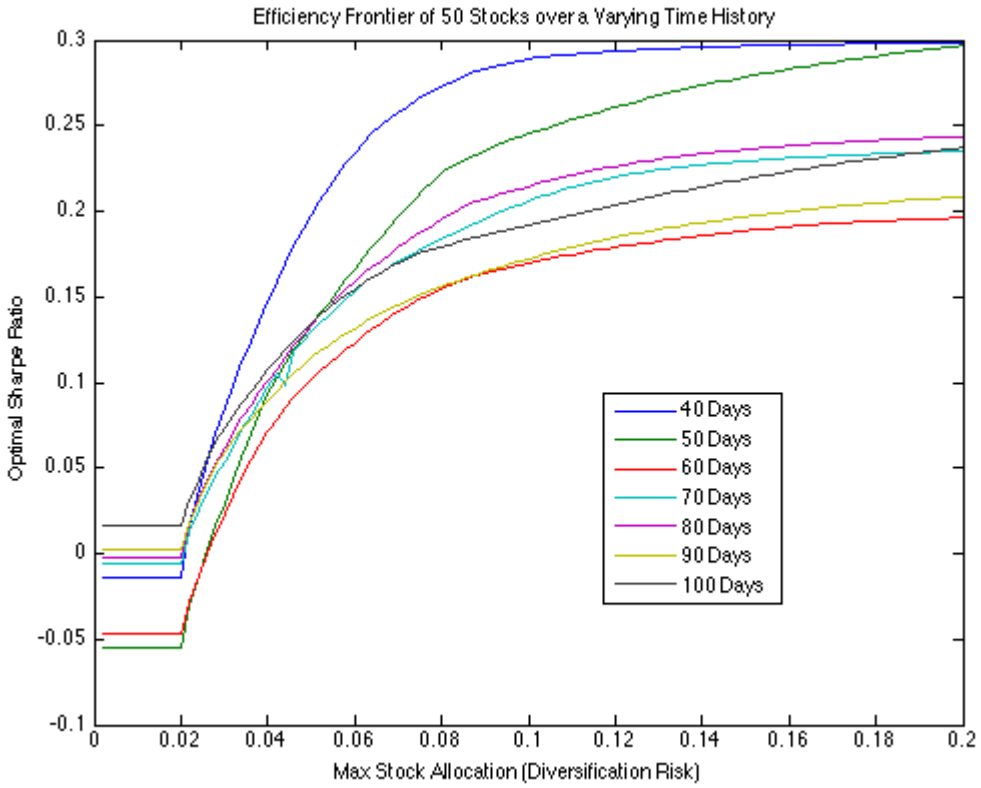

Figure 8: Efficient Frontier

Figure 8 shows the change in optimal Sharpe ratio as a function of diversification risk, while changing the length of the time history considered. Only 50 stocks were considered in this analysis to decrease computation time over the large number of runs of the optimization algorithm. It is apparent from the figure that as the time history considered for the optimization is increased, there is only a small increase in optimal Sharpe Ratio past about10% allocation per stock, in agreement with the variation in the similar plot where the number of stocks considered was varied. This suggests that the efficiency frontier depicting tradeoff in Sharpe ratio and the diversification risk may have an absolute curvature that is dictated by the historic period being sampled.

4 Application to Real Data

4.1 Sharpe Ratio Applied to Individual Stocks: Filter

It is difficult to optimize a portfolio that considers all possible stock allocations in the NYSE exchange. We have not been able to complete an optimization that considers the entire NYSE, because the computation time necessary increases exponentially with initial stocks passed to the solver. In order to obtain a broad market allocation optimization, the use of an independent Sharpe ratio filter is applied to the entire NYSE. The solver is not needed for an independent Sharpe ratio calculation wherein all allocation is given to one stock. Each stock has its independent Sharpe ratio calculated, as if it were the only possible stock choice.

The independent Sharpe ratio will yield a large positive value for those stocks that have shown positive return and minimal variance about their own price history in a given time series period. Stocks that have experienced price losses in the given time series will have negative Sharpe ratios. An investor looking for an optimal diversified portfolio is willing to pick negative returning stocks, if those stocks offset daily residuals in the return of their entire portfolio. After running some initial portfolio solutions on small sets of stocks, we noticed that occasionally an allocated stock selection lost value over the given time series (green line); not seen in the plot because it is in the negative range.

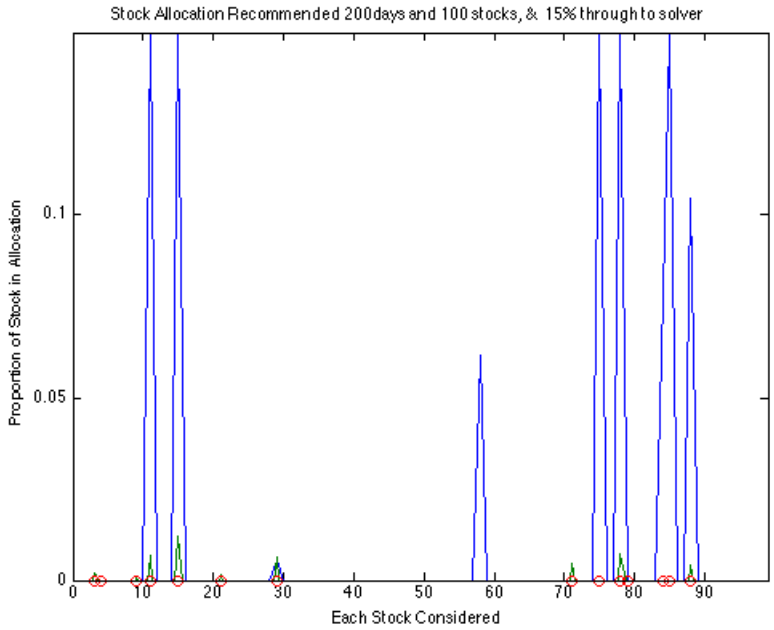

Figure 9: Filtering with Independent Sharpe Ratios

In figure 9, with 200 stocks considered, the optimization routine (blue line) choose stock number 58 even though it returns a negative value over the time series (green line; not seen), because stock number 58 was capable of reducing the overall daily residuals and variance of

the portfolio. We postulate that running the solver over the entire NYSE would diminish the amount of negative return stocks that need to be chosen in order to lower daily residuals and portfolio variance. In the entire NYSE we postulate that there *must* be positive returning stocks (red circles: chosen by filter), or hopefully less negative returning stocks, that can reduce daily residuals in the same way as the solver selected negative returning stocks do in our smaller sample of choices.

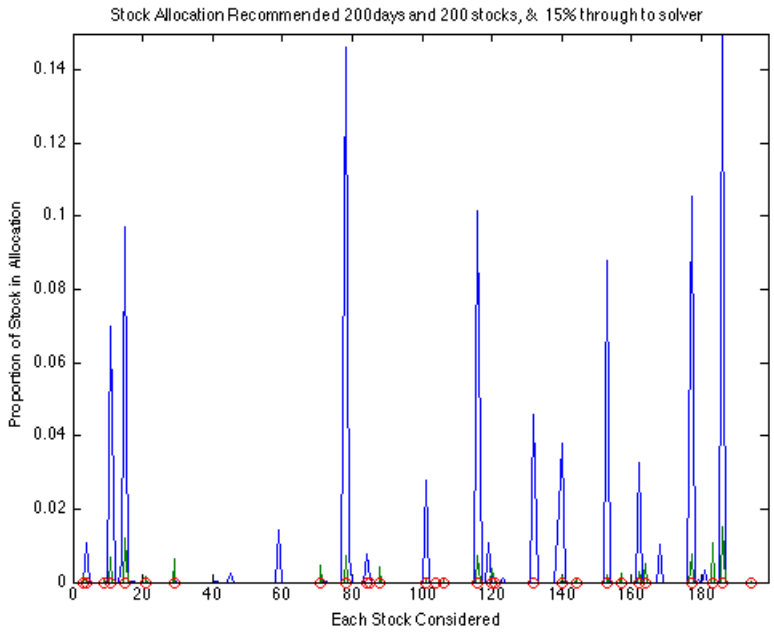

Figure 10: Filtering with Independent Sharpe Ratios

In figure 10, with 300 stocks considered, the allocation chosen for the negative returning stock number 58 has decreased from 7% to around 1%. In the limit of considering the entire exchange, we postulate that the negative returning stocks will be rarely used and can in fact be filtered out.

The independent Sharpe ratio filter (red circles) lets the solver look at the broad market, while simultaneously meeting the needs of a typical investor. Let us recall the wise words of Keynes once again, "Worldly wisdom teaches that it is better for reputation to fail conventionally, than to succeed unconventionally." [6] Investors don't want to diversify in negative returning stocks, and the filter passes through to the solver only a top percentile of the independent Sharpe ratios. The independent Sharpe ratio filter is able to compute independent stock Sharpe ratios for each of the 2000+ stocks on the NYSE in about a second. Once the ratios have been computed the filter passes only a top percentile of those stocks through to the solver.

In the previous two figures only the top 15% of the initial stocks considered are passed to the solver. When applied to the entire NYSE, this allows us to complete a broad market allocation solve. Figure 11 demonstrates the broad market NYSE allocation that has made use of the filter.

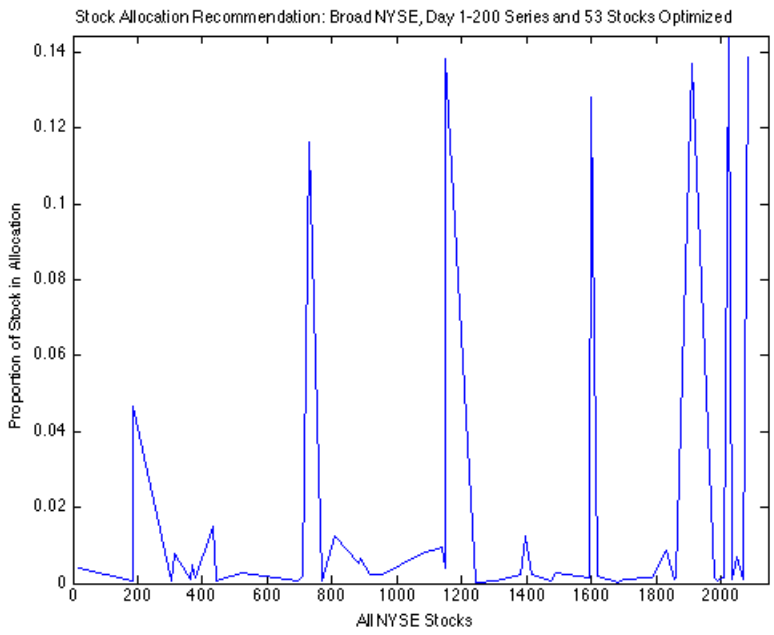

Figure 11: Broad Market Allocation Solution

The allocation in figure 11 took the solver 77 seconds:
"ACC' 'BBT' 'BBX' 'BXS' 'CACI' 'CFR' 'CHD' 'CIA' 'CPB' 'CRD.A' 'CRD.B' 'DEL' 'FBP' 'FCF' 'FDO' 'FLO' 'GB' 'GIS' 'HCN' 'HE' 'HME' 'HR' 'HVT' 'HVT.A' 'K' 'LDR' 'LF' 'LG' 'MHO' 'MTH' 'NHI' 'NMK-C' 'NPK' 'OCN' 'OHI' 'PNC' 'PNY' 'PSA' 'REV' 'RKT' 'SKT' 'STE' 'SWS' 'SXI' 'THS' 'UBA' 'UFI' 'USB' 'VEL-E' 'VLY' 'VRX' 'WFC' 'WM'

Any stocks that do not have history for every day in the time series are discarded and this brings the exchange down to the near 2000 stocks indexed that can be seen in figure 11. In order to view the ticker name of the stock you should invest in, you have to look at a processed version of the price matrix that only shows the stocks with full histories for the optimization. The first processing that checks adequate price history for all stocks to be considered, and then filtered, can be viewed in greater detail in the MATLAB code in Appendix Section 4.5 and 4.6.

4.2 Historic Volatility

The Sharpe ratio that corresponds to a particular diversification risk, by itself tells the investor little about the tradeoff between additional risk and additional return. In order to understand the tradeoff between additional risk and return, the efficient frontier was developed.

A single efficient frontier assess the added risk to return tradeoff over a particular

time series history. Analogous to the singular Sharpe ratio, a single efficient frontier at a snapshot in time will not tell much about the *future of risk and return tradeoffs*. In order to understand the future risk and return tradeoff the investor must analyze a collection of efficient frontiers. Figure 12 shows a collection of efficient frontiers, with the 1350 index representing the beginning of the year 2012 and the 1500 index representing the end of 2012.

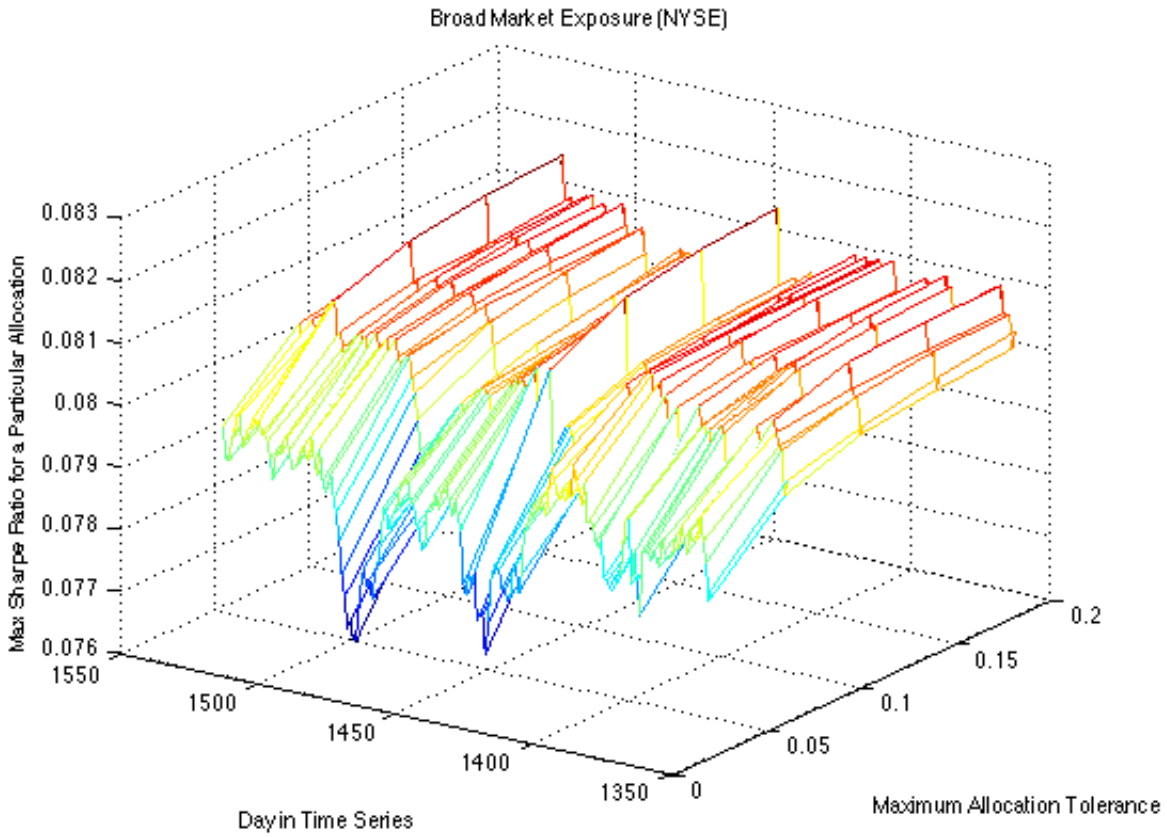

Figure 12: Efficient Frontiers Throughout 2012: Each Frontier on a 150 Day Series

In order to best understand the evolving tradeoff between risk and return, the investor must look at the curvature alterations within the set of efficient frontiers. The curvature of an efficient frontier informs the investor of the diminishing returns on a particular trading day, with each successive efficient frontier looking back on a historic time series (our case 150 days). A good investor will take advantage of a situation in which the return per an incremental increase in risk is above the average. Relaxing the diversification risk should be done in a quantifiable way. In order to get a better sense of the changing slope of the efficient frontier, we take 2-dimensional slices at incremental diversification risk tolerances.

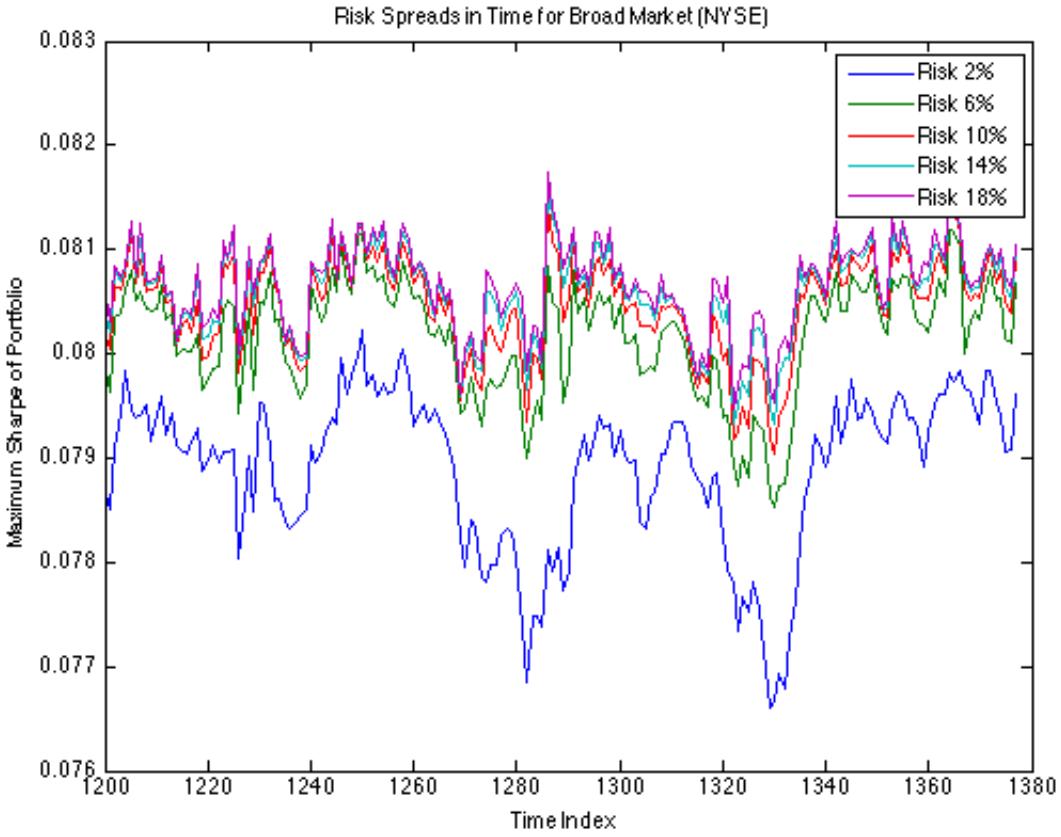

Figure 13: Risk Spreads

In figure 13, the return for all diversification risks move together with the swings in the NYSE. This intuitively makes sense, because bad market performance will lower the maximum Sharpe ratio attainable in every diversification risk level. The most interesting thing to note in these risk spreads is that the spread between any two risk levels constantly fluctuates. The average curvature of the efficient frontier can be empirically quantified by averaging the spreads between different risk levels. In this way, if a spread is above average then the investor should take note that an incremental increase in risk will yield a more than average increase in the value of Sharpe ratio, and consequently the strength of the portfolio. An example of fluctuating spreads can be seen in figure 14; which is simply a blown up view of 13 with MATLAB data tips added.

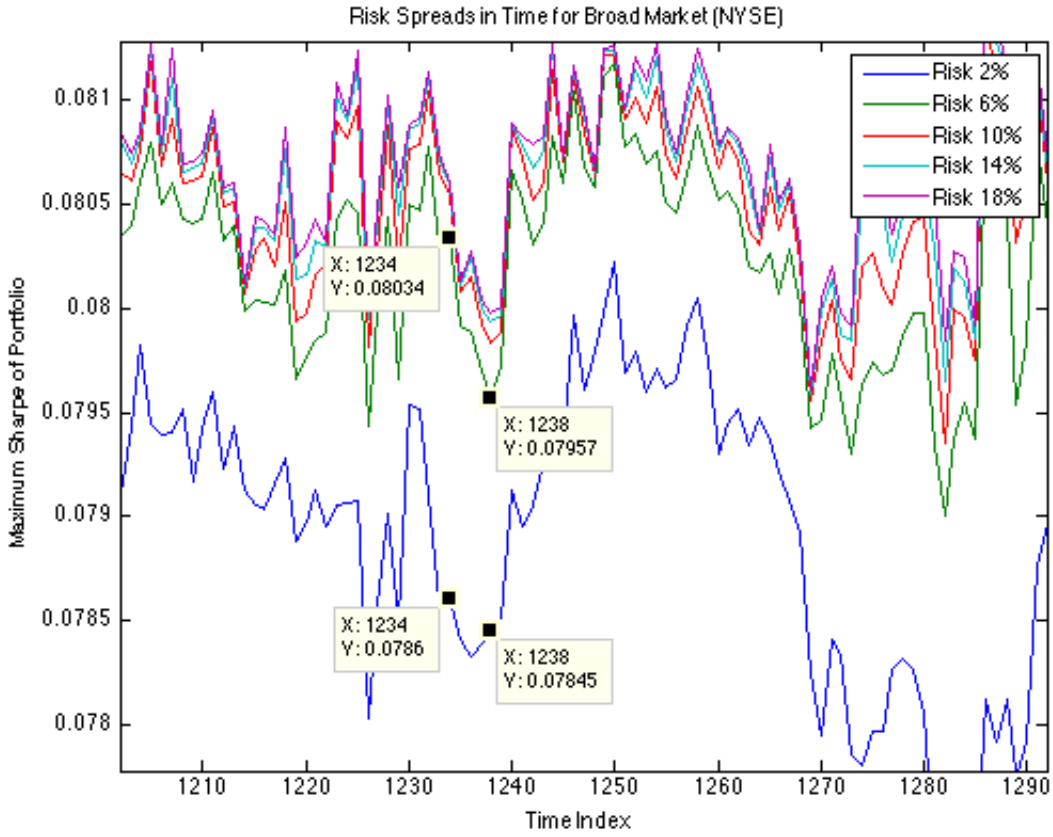

Figure 14: Risk Spreads

On day 1234 looking back at 150 days of time series history, the efficient frontier will depict a jump in Sharpe ratio from 0.0786 to 0.0803, for an increase in diversification from 2% to 6%. That jump in Sharpe ratio is equal to 0.0017. On day 1238 looking back at 150 days of time series history, the efficient frontier will depict a jump in Sharpe ratio from 0.0785 to 0.0796, for the same increase in diversification from 2% to 6%. That jump in Sharpe ratio is equal to 0.0011. The efficient frontier curvature, and corresponding return per increase in risk, is greater on day 1234 than on day 1238. It is clear that these spreads, which are derived from the curvature of the efficient frontier, do not stay constant and that there is opportunity for investors willing to act when the spreads are in their favor.

The return per incremental increase in risk spreads, will vary in magnitude and volatility with the time series chosen for construction of the efficient frontiers (our case 150 day). There is also the question of how many days of these risk spreads to observe when making a diversification risk decision. In future work trading algorithms could potentially be built from these types of risk spreads. The trading algorithms operate on top of the efficient frontier set, and before going further with a trading algorithm the time series used in constructing the efficient frontier must be chosen with more rigorous analysis; efficient frontiers constructed with 150 day history.

Table 3 outlines the averages and standard deviations of all the risk spreads we have visually observed in figures 13 and 14.

Table 3: Data for Trading Algorithm: Observing 177 Efficiency Curves

Starting Risk	Ending Risk	Average Sharpe Jump [1e-5]	Standard Deviation Sharpe Jump [1e-5]
1	2	130	37.8
1	3	160	48.6
1	4	170	54.7
1	5	180	58.3
2	3	30.1	15.0
2	4	43.7	23.0
2	5	51.0	27.8
3	4	13.6	9.09
3	5	21.0	14.4
4	5	7.31	5.67

5 Conclusions

While solving our problem of determining an optimal portfolio for the NYSE using the Sharpe ratio, we encountered a problem with our variance calculation. After reworking the variance calculation back into the optimization routine we no longer had infeasible points, but the optimization routine did take longer to solve the problem. Our analysis of this problem centered on the computation time needed given a set of input parameters. Specifically, these parameters are the length of the time series considered for each stock, and the number of stocks considered in the analysis. We found that the Sharpe ratio could easily be solved with a small set of initial stocks considered, but that the computation time for the solver increased exponentially with the number of stocks being initially considered. After finding out that the stocks initially considered parameter weighed most heavily on our solution computation time, we determined a method to reduce this parameter while at the same time preserving the quality of our solution.

To expand on the project, we applied an independent Sharpe ratio filter on the entire NYSE in order to reduce the number of stocks passed to the solver, and get an entire market solution. Reducing the number of stocks before the portfolio optimization takes place is important when considering the time constraints on solving this problem daily. The

optimization routine would need to consistently find an optimal solution in the 15 hour time window between when the market closes, and when the market opens the subsequent morning. The individual Sharpe ratio of many stocks can be calculated extremely quickly without the need for a solver. By filtering the initial stocks, we can remove some of the challenging computational sifting that the solver is forced into by the abundance of stock choices.

Another extension of this project, that we did not get to, would be the use of our five year historical data set to test the ability of our algorithm to make money on the NYSE over a given time period. This would involve creating a method to decide whether or not to change the stock allocation based on how much the objective function (Sharpe ratio) changed from day-to-day, and determining an adequate tolerance for this change before reallocating funds. If this extension were to be tested, we would need to incorporate the cost of executing trades, and factor that into net asset value at the close of every day. If the algorithm is shown to be able to return a profit over different historical data sets, then it would prove the effectiveness of using the Sharpe ratio as a tool for portfolio allocation.

6 Bibliography

6.1 Modern Portfolio Theory Research

[1] Aguerrevere, Jorge. "Portfolio Optimization of Expected Return to Risk Using the Sharpe Ratio and the Capital Asset Pricing Model." ChE488 - Convex Optimization, December 19, 2011.

[2] Guia J, Kim E., Raj V., Saleem S., Shah A., Synnott T. "Performance Analysis of Sector Specific ETFs." EID300 - Special Research Projects, May 13, 2013.

[3] H.Fromlet, "Behavioral Finance - Theory and Application." *Business Economics*, July 2001.

[4] Keynes, J. "The General Theory of Employment, Interest and Money" *Macmillan*, London, 1936.

[5] Markowitz, Harry. "Portfolio Selection." *The Journal of Finance* 7.1 (1952): 77-91.JSTOR. Web. 12 November 2011.

[6] P.P. Hogan, "Portfolio Theory Creates New Investment Opportunities." *Journal of Financial Planning*, January, 1994.

[7] Samuelson, Paul and William Nordhaus. *Economics*. McGraw-Hill, New York. 1995.

[8] Sharpe W. "Capital Asset Prices: A Theory of Market Equilibrium Under Conditions of Risk." *Journal of Finance*, Vol. XIX, No. 3, September 1964, pp. 425-442.

6.2 Data & Analytics

[9] "Choosing a Solver." *The MathWorks, Inc.* http://www.mathworks.com/help/, November 1, 2013: Web.

[10] "New York Stock Exchange Data." *EODData, LLC.* http://www.eod data.com/ October 15, 2013: Web.

[11] "SPDR S&P 500 Growth ETF." *State Street Global Advisors.* https://www.spdrs .com/, November 15, 2013: Web.

6.3 External Figures

[12] "Modern Portfolio Theory and The Efficient Frontier." http://www.smart401k .com/, November 17, 2013: Web.

7 Appendix

7.1 Building Data Compiler

The data sets for this type of problem can grow to be quite large. In good spirits about potentially using our algorithm at the end of this project, to trade in the market, we made the decision to create a data compiler. In retrospect this aspect of the project was quite difficult, but we are happy to have executed a successful sorting routine for our raw daily prices.

Before testing our algorithm in the present market, we will be running it historically over the 5 years of data we have organized. In this vetting stage for our algorithm, we will be able to detect any flaws and pitfalls possibly hidden in the objective function, the parameters, or in the type of filter we apply.

7.2 Defining Parameters

Defining R - Daily Return Matrix $S(i,j)$ = Price matrix for each Stock j and all days i being optimized $R(i,j)$ = Return of Stock j, Up Until a Given Closing Day i

i = Day Index = $1...m$; j = Stock Index = $1...n$

for $i = 1 : (size(S,1) - 1)$

for $j = 1 : size(S,2)$

$$R(i,j) = \frac{(S(i+1,j) - S(1,j))}{(S(1,j)}$$

end

end

Defining c - Total Return

$c(i,j)$ = Stock Return Over Entire Series

k = Stock Index = $1...n$

for $k = 1 : size(S,2)$

$$c(k,:) = \frac{(S((size(S,1)-1),k) - S(1,k))}{S(1,k)}$$

end

7.3 Reformed Problem

The objective function for Problem 2 is quadratic; with a linear numerator and a quadratic denominator. Note that the value associated with solution $x = [x_1, \ldots, x_n]^\top$ is equivalent to the value associated with a solution of the form $tx = [tx_1, \ldots, tx_n]^\top$, where $t \epsilon \Re$ is any non-zero scalar.

$$\frac{c^\top (tx)}{\sqrt{((tx)R^\top R(tx)) + (m-2)(tx)^\top cc^\top (tx)}} = \frac{tc^\top x}{\sqrt{t^2(xR^\top Rx) + t^2(m-2)x^\top cc^\top x}}$$

$$= \frac{tc^\top x}{t\sqrt{(xR^\top Rx) + (m-2)x^\top cc^\top x}} = \frac{c^\top x}{\sqrt{(xR^\top Rx) + (m-2)x^\top cc^\top x}}$$

If the optimal value to Problem 2 remains constant after this non-zero scalar multiple to the numerator and denominator, then we have found an equivalent form for the problem. Of particular interest is the reformed problem and equality constraint that follow.

Original Problem:

$$Problem2 = \max_{x \epsilon \Omega} \frac{c^\top x}{\sqrt{(xR^\top Rx) + (m-2)x^\top cc^\top x}}$$

Reformed Problem:

$$Problem3 = \max_{x \epsilon \Omega} tc^\top x$$

Non-zero scalar t is removed from the maximization because it is a scalar and will not alter the argument solution:

$$Problem3 = \max_{x \epsilon \Omega} c^\top x$$

Such that:

$$\mathrm{t}[(xR^\top Rx) + (m-2)x^\top cc^\top x] = 1 \tag{17}$$

Reformulation yields the linear Problem 3, subjected to the quadratic constraint that the original Problem 2 denominator multiplied by scalar t is set equal to 1. Problem 3 is an equivalent form of Problem 2, and could result in a fast solver computation time. The trouble is that MATLAB does not support quadratically constrained optimization.

7.4 MATLAB Scripts

7.4.1 CSV Parser

```
% Grant Aarons & William Biesiadecki
% Convex Optimization Project
% Portfolio Optimization
% Data Parsing Script
% October 8, 2013

clear all; close all

%Figure out how many Years there are to Parse
rawData = dir('Raw Data');
numYears = numel(rawData);
nameYears = cell(numYears-2,1,1);
for i = 3:numYears
    nameYears{i-2,1,1} = rawData(i).name;
end

%Figure out how many days are available for a given year
rawDays = cell(numYears-2,1,1);
for j = 1:numel(nameYears)
    rawDays{j,1,1} = dir(sprintf('Raw Data/%s',nameYears{j}));
    numDays(j) = numel(rawDays{j,1,1});
    maxDays = max(numDays)-2;
end

    nameDays = cell(maxDays-2,numel(nameYears),1);
        %Creates a Cell Array of every .csv file
for j = 1:numel(nameYears)
    for k = 3:numDays(j)
        nameDays{k-2,j,1} = rawDays{j,1,1}(k).name;
    end
end

rawStockData = cell(numel(nameDays),1,1);
k = 1;

%Takes all the Data out of csv's and puts it into Large Cell Array
for m = 1:numel(nameYears)
    for n = 1:numDays(m)-2
        rawStockData{k} = importfile(sprintf('Raw ...
            Data/%s/%s',nameYears{m}, nameDays{n,m,1}),1,2562);
        k = k+1;
    end
end
```

7.4.2 Organizational Compiler

```
% Grant Aarons & William Biesiadecki
% 11-20-2013
% Compiler - Sorts stocks into their own individual column for the ...
    period going back to January 2008

% Pre-allocate cells for the initial stocks for all time (5 years)
A = cell(1527,2167);
P = cell(1527,2167);

% Initialize sorting at the first row, first day
i = 1;
% Load the first day stock names and prices
for j = 1:length(rawStockData{i})
    % Ticker Names
    A(i,j) = rawStockData{i}(j,1);
    % Stock Prices
    P(i,j) = rawStockData{i}(j,3);
end

% Find the length of days in the time series
N = size(rawStockData,1);
for i = 2:N
    % Set a variable amount of stocks on the exchange for any given day
    M(i) = length(rawStockData{i});
    for j = 1:M(i)
        A(i,j) = rawStockData{i}(j,1);
        P(i,j) = rawStockData{i}(j,3);
    end

    % After every day is sorted, come back to here and start next day at
    % the first stock to appear on the exchange.
    j = 1;
    while j < length(A(i,:))
        while j < length(A(i,:))
            % If on this given day, i, this stock index matches the one
            % above it, then move on to the next stock to sort (stock
            % placed correctly in a column)
            if (strcmp(A(i,j),A(i-1,j))>0.9)
                j = j + 1;

                % Check if the cell above is empty (special attention)
            else if isempty(A{i-1,j})
                    clear emp
                    % Check all cells above this one, in this particular
                    % column, if they are empty return a 0.  Sum all evals
                    for a = 1:(i-1)
                        emp(a) = ~isempty(A{a,j});
                    end
                    % If the sum is less than 0.9 than the entire ...
                       column is
```

```
49                      % empty and it is okay to place a new stock here
50                      if (sum(emp)<0.9)
51                          j = j+1;
52                          % If the column has been used before, than ...
                               place a zero
53                          % in this cell location to continue showing ...
                               that this
54                          % ticker has gone private, bankrupt, etc.
55                      else if (sum(emp)>0.9)
56                              clear S1 S2 S3 SP1 SP2 SP3
57                              S1 = A(i,1:j-1);
58                              SP1 = P(i,1:j-1);
59                              S2 = {[]};
60                              SP2 = {0};
61                              S3 = A(i,j:end);
62                              SP3 = P(i,j:end);
63                              A(i,1:end+1) = [S1 S2 S3];
64                              P(i,1:end+1) = [SP1 SP2 SP3];
65                              j = j+1;
66                          else
67                              disp('Error in new segment')
68                          end
69                      end
70                      % Check if the stock that should be in this column, ...
                           is anywhere in this row.
71                  else if (sum(strcmp(A(i,j:end),A(i-1,j)))>0.9)
72                          clear S1 S2 S3 SP1 SP2 SP3
73                          S1 = A(i,1:j-1);
74                          SP1 = P(i,1:j-1);
75                          % Reorganize as usual
76                          if (j+1 ≠ length(A(i,:)))
77                              S2 = A(i,j+1:end);
78                              SP2 = P(i,j+1:end);
79                              % Special case is a flip of cells, if you ...
                                   are at the end of a row
80                          else if (j+1 == length(A(i,:)))
81                                  S2 = A(i,j+1);
82                                  SP2 = P(i,j+1);
83                              end
84                          end
85                          S3 = A(i,j);
86                          SP3 = P(i,j);
87                          A(i,:) = [S1 S2 S3];
88                          P(i,:) = [SP1 SP2 SP3];
89
90                          % Check if this cell i,j is shifted left one ...
                               column,
91                          % This would indicate the first time the stock ...
                               to the immediate
92                          % left has left the NYSE/goes bankrupt/goes private
93                      else if (strcmp(A(i,j),A(i-1,j+1))>0.9)
94                              clear S1 S2 S3 SP1 SP2 SP3
95                              S1 = A(i,1:j-1);
96                              SP1 = P(i,1:j-1);
```

```
                                S2 = {[]};
                                SP2 = {0};
                                S3 = A(i,j:end);
                                SP3 = P(i,j:end);
                                A(i,1:end+1) = [S1 S2 S3];
                                P(i,1:end+1) = [SP1 SP2 SP3];
                                j = j+1;

                                % Check to see if the 1st day in our series ...
                                    is non-empty,
                                % if so, and immediately above is empty, ...
                                    than we
                                % ticker name continues on as a 0 value
                            else if (isempty(A(i-1,j)) && ~isempty(A(1,j)))
                                    clear S1 S2 S3 SP1 SP2 SP3
                                    S1 = A(i,1:j-1);
                                    SP1 = P(i,1:j-1);
                                    S2 = {[]};
                                    SP2 = {0};
                                    S3 = A(i,j:end);
                                    SP3 = P(i,j:end);
                                    A(i,1:end+1) = [S1 S2 S3];
                                    P(i,1:end+1) = [SP1 SP2 SP3];
                                    j = j+1;

                                else
                                    disp('Error in Daily Stock Reorder')
                                end
                            end
                        end
                    end
                end
            end
        end
    end
    % Remove any extra columns that are completely empty
    q = length(A(i,:))-2156;
    for k = 1:q
        for n = 1:i
            C(n,k) = ~isempty(A{n,2156+k});
        end
    end
    iter = 0;
    for m = 1:q
        if (sum(C(:,m))<0.9)
            f = 2156-iter;
            A(:, f+m) = [];
            P(:, f+m) = [];
            iter = iter +1;
        else
        end
    end
    clear C
end
```

7.4.3 Concatenate

```
% Grant Aarons & William Biesiadecki
% Convex Optimization Project
% Portfolio Optimization
% Concatenating Script
% October 8, 2013

%Fills in Empty Matricies and Converts cells to matrix

PIndex = cellfun(@isempty,P);  % Find indices of empty cells
Pready = P;
Pready(PIndex) = {0};          % Fill empty cells with 0
Pmat = cell2mat(Pready);       % Convert the cell array
```

7.4.4 Objective Function

```
% Grant Aarons & William Biesiadecki
% 11-20-2013
% Objective function and parameters

function f = myfun3(x)

% Call necessary global variables from the main script
global S

% Pre-allocate space for the daily price gains
R = zeros(size(S,1)-1,(size(S,2)));

% Iterate each individual stock's gains, over everyday in the time series
for i = 1:(size(S,1)-1)
    for j = 1:size(S,2)
        % Return for a stock j, from one day's closing price to the next
        % day's closing price
        R(i,j) = (S(i+1,j) - S(1,j))/S(1,j);
    end
end

% Pre-allocate space for total return of each stock over the entire series
C = zeros(size(S,2),1);
for k = 1:size(S,2)
    % Difference in price at the end and begining of the Time Series,
    % divided by initial price of stock, input capital requirement
    C(k,:) = (S((size(S,1)-1),k)-S(1,k))/S(1,k);
end

% How many days were considered in the time series
m = size(R,1);

```

```
% Negative Sharpe's ratio in vector form
% Sharpe's ratio is intended for maximization, but we can formulate the
% problem as an equivalent minimization of an argument that is the ...
    Negative function
f = -(C'*x) / sqrt(m*x'*R'*R*x - x'*C*C'*x);
end
```

7.4.5 Single Case Optimal Portfolio

```
% Grant Aarons & William Biesiadecki
% Convex Optimization Project
% Portfolio Optimization
% Allocation Script
% November 20, 2013


clear Sprep checkZeros anyStockZeros SpostProcess S SharpeI x0 ...
    stocksDropped ...
    SharpeI R C m arrayFilteredStocks indSharpes indAllocations
% Define global variables, which are passed to the function to be optimized
global S R C m
clear tolStock endDay startDay endStock startStock ...
    g i amountThroughFilter arrayStocks %indSharpes indAllocations % ...
        maxSharpeRatio

startStock = 1;
endStock = size(Pmat,2);
% Length of Time Series being Used!!! (Perfect Size unknown; yet)
length = 150;  % 200 is good because it eliminates short term
% trends and pays greater attention to consistently well
% performing stocks.  We see this in stock 100 from our report
% Step Size through Time
stepSize = 1;
lastDay = 1528-length;

for g = 1200:lastDay
        clear x x0 fval

    % Keep Time
    tic
    endDay(g) = (length+1) + (g-1)*stepSize;
    startDay(g) = 1 + (g-1)*stepSize;

    % Load a concatenated price matrix whose size is dictated above
    Sprep = Pmat(startDay(g):endDay(g),startStock:endStock);
    Aprep = A(startDay:endDay,startStock:endStock);


```

```
39      % Check for any stocks with problems ((((If they go to zero, they ...
            throw off
40      % the algorithm))))
41
42      checkZeros = Sprep < 0.001;
43
44      for k = 1:(size(Sprep,2))
45          anyStockZeros(k) = sum(checkZeros(:,k));
46      end
47      stocksDropped = 0;
48      for y = 1:(size(Sprep,2))
49          if anyStockZeros(y) < 0.9
50              ApostProcess(:,y-stocksDropped) = Aprep(:,y);
51              SpostProcess(:,y-stocksDropped) = Sprep(:,y);
52          else
53              stocksDropped = stocksDropped +1;
54          end
55
56      end
57
58
59      %  Independent Sharpe Filter!!!!
60
61
62           [arrayFilteredStocks, indSharpe] = indFilter(SpostProcess);
63
64
65              S = SpostProcess(1:(endDay-startDay),arrayFilteredStocks');
66          % Load a concatenated price matrix whose size is dictated above
67          S = Pmat(startday:endday,1:stocks);
68
69
70
71
72          arrayStocks(g,1:size(arrayFilteredStocks,1)) = ...
                arrayFilteredStocks';
73
74
75      for i = 1:5
76          i
77          g
78          tic
79          clear Bones B
80          tolStock(i) = 0.02+(i-1)*(0.04); % 2, 6, 10, 14, 18.
81
82          % Set an initial condition that weights all stock choices equally
83          x0 = (1/size(S,2))*ones(size(S,2),1);
84
85          % Define matrices for inequality constraints
86          % The sum of all stock allocations must be greater than zero
87          A1 = -ones(1,size(S,2));
88          % Each individual stock allocation is either zero,
89          % or less than a max allocation tolerance
90          A2 = eye(size(S,2));
```

```
91          % Entire Inequality Coefficient matrix
92          A = [A1;A2;-A2];
93
94          % Max percentage of one particular stock in allocation
95          tolStock = 0.15;
96          % Stock allocations are bounded above by the tolerance
97          Bones = tolStock*ones(1,size(S,2));
98          % Stock allocation are bounded below by zero
99          Bzeros = zeros(1,size(S,2));
100         % Entire Inequality Coefficient matrix
101         B = [0 Bones Bzeros]';
102
103         % The sum of all stock allocations
104         Aeq = ones(1,size(S,2));
105         % Must add up to the whole to be invested (==1)
106         Beq = 1;
107
108         options = optimset('MaxFunEvals',50000, 'MaxIter', 1000);
109
110         % Calling and executing minimization of the objective function in
111         % MATLAB function myfun3
112         [x, fval,exitflag,output] = ...
                fmincon(@myfun3,x0,A,B,Aeq,Beq,[],[],[],options);
113
114         % Index all stocks being optimally allocated
115         allStocks = [1:size(S,2)];
116
117         % Store time
118         timeTaken = toc;
119     end
120 end
121
122         figure(1)
123         plot(allStocks,x)
124         title('Stock Allocation Recommended (40 Days)')
125         xlabel('Each Stock Considered')
126         ylabel('Proportion of Stock in Allocation')
127         axis([0 stocks 0 max(x)])
```

7.4.6 Filter Applied Optimal Portfolio

```
1  % Grant Aarons & William Biesiadecki
2  % Convex Optimization Project
3  % Portfolio Optimization
4  % Independent Sharpe Filter
5  % December 19, 2013
6
7  clear Sprep checkZeros anyStockZeros SpostProcess S SharpeI x0 ...
       stocksDropped ...
8      SharpeI R C m arrayFilteredStocks indSharpes indAllocations
```

```
9
10 % Define global variables, which are passed to the function to be optimized
11 global S R C m
12 clear tolStock endDay startDay endStock startStock ...
13    g i amountThroughFilter arrayStocks indSharpes indAllocations ...
         maxSharpeRatio
14
15 startStock = 1;
16 endStock = size(Pmat,2);
17
18 % Length of Time Series being Used (Perfect Size unknown; yet)
19 length = 150;  % 200 is good because it eliminates short term
20
21 % Step Size through Time
22 stepSize = 1;
23 lastDay = 1528-length;
24
25 for g = 1200:lastDay
26     clear x x0 fval
27
28     endDay(g) = (length+1) + (g-1)*stepSize;
29     startDay(g) = 1 + (g-1)*stepSize;
30
31     % Load a concatenated price matrix whose size is dictated above
32     Sprep = Pmat(startDay(g):endDay(g),startStock:endStock);
33     Aprep = A(startDay:endDay,startStock:endStock);
34
35     % Check for any stocks with problems (If they go to zero, they ...
          throw off
36     % the algorithm)
37     checkZeros = Sprep < 0.001;
38     for k = 1:(size(Sprep,2))
39         anyStockZeros(k) = sum(checkZeros(:,k));
40     end
41     stocksDropped = 0;
42     for y = 1:(size(Sprep,2))
43         if anyStockZeros(y) < 0.9
44             ApostProcess(:,y-stocksDropped) = Aprep(:,y);
45             SpostProcess(:,y-stocksDropped) = Sprep(:,y);
46         else
47             stocksDropped = stocksDropped +1;
48         end
49
50     end
51
52     %  Independent Sharpe Filter!!!!
53     [arrayFilteredStocks, indSharpe] = indFilter(SpostProcess);
54     arrayStocks(g,1:size(arrayFilteredStocks,1)) = arrayFilteredStocks';
55
56     S = SpostProcess(1:(endDay-startDay),arrayFilteredStocks');
57
58     for i = 1:5
59         i
60         g
```

```
61          % Keep Time
62          tic
63          clear Bones B
64          tolStock(i) = 0.02+(i-1)*(0.04); % 2, 6, 10, 14, 18.
65
66
67          % Set an initial condition that weights all stock choices equally
68          x0 = (1/size(S,2))*ones(size(S,2),1);
69
70          % Define matrices for inequality constraints
71          % The sum of all stock allocations must be greater than zero
72          AE1 = -ones(1,size(S,2));
73          % Each individual stock allocation is either zero,
74          % or less than a max allocation tolerance
75          AE2 = eye(size(S,2));
76          % Entire Inequality Coefficient matrix
77          AE = [AE1;AE2;-AE2];
78
79          % Max percentage of one particular stock in allocation
80          % Stock allocations are bounded above by the tolerance
81          Bones = tolStock(i)*ones(1,size(S,2));
82          % Stock allocation are bounded below by zero
83          Bzeros = zeros(1,size(S,2));
84          % Entire Inequality Coefficient matrix
85          B = [0 Bones Bzeros]';
86
87          % The sum of all stock allocations
88          Aeq = ones(1,size(S,2));
89          % Must add up to the whole to be invested (==1)
90          Beq = 1;
91
92          amountThroughFilter(g,i) = size(S,2);
93
94          % Pre-allocate space for the daily price gains
95          R = zeros((size(S,1)-1),(size(S,2)));
96          % Pre-allocate space for total return of each stock over the ...
               entire series
97          C = zeros(size(S,2),1);
98          % How many days were considered in the time series
99          m = size(R,1);
100
101         options = optimset('MaxFunEvals',50000, 'MaxIter', 1000, ...
               'Display','off','Algorithm','interior-point');
102         [x, fval,exitflag,output] = ...
               fmincon(@myfun3,x0,A,B,Aeq,Beq,[],[],[],options);
103         maxSharpeRatio(i,g) = -fval;
104         toc
105     end
106 end
107
108 % Broad Market Exposure Finally
109 figure(1)
110 mesh(tolStock,endDay(1200:1377),maxSharpeRatio(1:5, 1200:1377)')
111 title('Broad Market Exposure (NYSE)')
```

```
xlabel('Maximum Allocation Tolerance')
ylabel('Day in Time Series')
zlabel('Max Sharpe Ratio for a Particular Allocation')
axis([min(tolStock) max(tolStock) min(endDay) max(endDay) 0 ...
    max(maxSharpeRatio)])

timeIdx = [1200:(lastDay-1)];
figure(2)
plot(timeIdx,maxSharpeRatio(1,1200:1377),timeIdx,maxSharpeRatio(2,1200:1377),timeIdx,...
    maxSharpeRatio(3,1200:1377),timeIdx,maxSharpeRatio(4,1200:1377),timeIdx,maxSharpeRat
title('Risk Spreads in Time for Broad Market (NYSE)')
legend('Risk 2%','Risk 6%','Risk 10%','Risk 14%','Risk 18%')
xlabel('Time Index')
ylabel('Maximum Sharpe of Portfolio')
zlabel('Max Sharpe Ratio for a Particular Allocation')
```

7.4.7 Independent Sharpe Filter

```
% Grant Aarons & William Biesiadecki
% 12-19-2013
% Independent Sharpe Filter

function [arrayFilteredStocks, indSharpe] = indFilter(SpostProcess)

clear Rfilter
% Pre-allocate space for the daily price gains
Rfilter = zeros(size(SpostProcess,1)-1,(size(SpostProcess,2)));
clear Rcong Ci Rsum m %SharpeI
% Iterate each individual stock's gains, over everyday in the time series
for ii = 1:(size(SpostProcess,1)-1)
    for jj = 1:size(SpostProcess,2)
        % Return for a stock j, from one day's closing price to the next
        % day's closing price
        Rfilter(ii,jj) = (SpostProcess(ii+1,jj) - SpostProcess(1,jj))/...
            SpostProcess(1,jj);
    end
end

% Pre-allocate space for total return of each stock over the entire series
Ci = zeros(size(SpostProcess,2),1);
for kk = 1:size(SpostProcess,2)
    % Difference in price at the end and begining of the Time Series,
    % divided by initial price of stock, input capital requirement
    Ci(kk,1) = ...
        (SpostProcess((size(SpostProcess,1)-1),kk)-SpostProcess(1,kk))/...
        SpostProcess(1,kk);
end

% How many days were considered in the time series
mm = size(Rfilter,1);
for d = 1:size(SpostProcess,2)
    for s = 1:(size(SpostProcess,1)-1)

        %Try viewing individual Stock Sharpe Values??
        Rcong(s,d) = Rfilter(s,d)*Rfilter(s,d)';
    end
    Rsum(d) = sum(Rcong(:,d));
    SharpeI(d,1) = Ci(d,1)/sqrt((mm*Rsum(d))-(Ci(d,1)^2));
end
% Only take the top 2.5% of candidates, and this is only ever ¬60
[svals,idx] = sort(SharpeI(:),'descend'); % sort to vector
perc = svals(ceil((size(SpostProcess,2)/40))); % 97.5th Percentile value

% Output the index number for good potential Stocks i.e. [1 3 4 7 10 99 ...
    110 ..]
[row,col] = find(SharpeI > perc);  % Find when a single stock ...
    outperforms peers

```

```
48 arrayFilteredStocks = row;
49 indSharpe = SharpeI;
50 end
```