Enhanced Computational Frameworks for Implementing Portfolio Optimization

Alan R. Kessler

Anil Koluguri

Northwestern University

Author Note

Alan R. Kessler and Anil Koluguri are students in the School of Professional Studies at Northwestern University.

Abstract

In the investments industry, it is critical to minimize risk for a given return. The theory involved in this task is well studied, but the practical implementation can be burdensome. As the number of securities in a portfolio increases, more advanced tooling is required to reach the optimal solution. Simple coding examples of portfolio optimization are commonly available but do not reflect considerations such as commissions and use of the risk-free asset. We propose a framework that incorporates these considerations and most importantly provide the means to continually update the optimal mix of assets. The business problem we are solving with this framework is taking an existing portfolio and finding the optimal set of transactions that should take place to minimize risk while meeting a minimum return requirement. The resulting portfolio can then be compared to individual assets and other portfolios including the minimum risk and optimal risky portfolio. The results show that this framework could be valuable to personal investors as they track their portfolios.

Keywords: Portfolio, optimization, capital allocation line, Stocks, Markowitz

Enhanced Computational Frameworks for Implementing Portfolio Optimization

When investing in the financial markets, the goal of the investor is to maximize their individual utility which is often a function of risk and return on investment. The resulting optimization can be defined as quantitative approach for decision making where one seeks the best decision out of available set of choices. Portfolio optimization is the process of selecting the optimal utility-maximizing asset distribution out of all the assets that are being considered.

For this project, we consider the Modern Portfolio Theory or mean-variance analysis for constructing the portfolio to optimize or minimize risk based on a required return. Simple portfolio optimization is discussed in great detail in the course for which this project is being completed. The focus of the course discussion lies in determining an efficient frontier of minimum variance for a specific return on investment and on that frontier, an overall minimum risk portfolio.

The goal of this project is to move this framework forward and address the practical limitations of simply considering the efficient frontier. The most obvious limitation is that under the research discussed in the literature review, there are actually two aspects of the portfolio that need to be optimized. First, the optimal risky portfolio must be determined which is identical for all investors regardless of risk appetite. Second, the investor splits assets between the optimal risky portfolio and the risk-free asset based on risk aversion or return expectation. Another limitation is that in reality, there are costs associated with transactions that must be considered in creating the optimized portfolio.

As a business problem, suppose an investor has an existing portfolio and seeks to minimize risk subject to a minimum required daily return. The investor can buy or sell stocks at a fixed percentage commission and can also invest in a risk-free asset. To solve this business

problem, we propose a framework that addresses these limitations and generates optimal portfolios in a consumable format for actual implementation. The details of this problem are discussed later in the paper. The features of this framework include:

- The ability to import and export portfolios.
- The ability to incorporate actual historical returns based on specified dates.
- The ability to generate an optimized portfolio considering the current portfolio, transaction costs, the risk-free return, and minimum growth expectations.
- The ability to plot the efficient frontier, individual stock data, the optimal risky portfolio, the capital allocation line, and the current portfolio's risk and return.

Literature Review

The first step to understanding the problem is determining what drives risk and why a mix of assets is key to determining the optimal portfolio. According to Meir Statman, adding stocks to a portfolio reduces risk marginally (1987). His research shows that as the number of stocks in a portfolio increases, the standard deviation of returns decreases at a marginally decreasing rate. This is due to some level of independence in the risks between two securities, but due to systematic risks, an infinite number of securities still has a standard deviation in returns greater than zero.

Harry Markowitz describes how the expected return and variance of a portfolio is calculated (1952). The respective formulas are shown below and demonstrate that minimizing variance for a given return is a nonlinear optimization problem.

$$Expected\ Return: R = \sum X_i R_i\ , where\ \sum X_i = 1, when\ X\ is\ the\ portfolio\ weights$$

$$Expected\ Variance: V = \sum_{i=1}^{N} \sum_{j=1}^{N} \sigma_{ij} X_i X_j\ , where\ \sigma_{ij}\ is\ the\ covariance\ of\ stocks\ i, j$$

Markowitz goes on to describe a set of attainable return and variance combinations that he considers efficient in that variance is minimized for a given return. This research forms the basis for the optimization statement but does not distinctly identify the point to select on the efficient frontier. Another limitation is that Markowitz does not go into detail on how to calculate the efficient frontier and the number of calculations increases exponentially with the number of stocks considered.

Sharpe takes the next step in defining a ratio of his namesake that includes the concept of the risk free-rate, the rate of return for an investment with zero standard deviation in its returns (1966). The Sharpe ratio formula is shown below. It is the ratio of the expected return in excess of the risk-free rate for an asset divided by its standard deviation.

$$S_i = \frac{E(r_i) - r_f}{\sigma_i}$$

By maximizing the Sharpe ratio for the set of combinations that make up the efficient frontier, we can determine the optimal risky portfolio, that which minimizes variance and maximizes the Sharpe ratio. Sharpe elaborates to describe a line called the Capital Allocation Line, that reflects the combinations of the optimal risky portfolio and the risk-free asset. This line runs tangent to the efficient frontier at this point such that rather than an investor selecting a point on the efficient frontier based on their risk preferences, they are actually selecting a point on this line. This paper also does not discuss the calculations in as much detail, making the efficient application of this theory the main motivation of our research.

Methodology

Portfolio optimization is considered as a class of non-linear programming problems as the optimal solution is a point on curve known as the efficient frontier. We developed several model variations from a fixed dataset that will describe in greater detail later in the paper. The goal is to

minimize the variance, V, of the daily portfolio returns. The variable X_i is the amount invested in one of the *N* available stocks. The variance can be described as the dot product covariance matrix and the amount invested twice.

$$V = \sum_{i=1}^{N} \sum_{j=1}^{N} \sigma_{ij} X_i X_j, where\ \sigma_{ij}\ is\ the\ covariance\ between\ stocks\ i, j$$

The first constraint reflects that the sum of the individual investments and commissions must not exceed the original value of the portfolio. In this case B_i and S_i are the amount of a stock bought and sold respectively with commission percentage C and amount originally invested O_i including any initial investment in the risk-free asset. The amount held, bought, and sold are the decision variables which are all assumed to have a lower bound at zero. We assume there is no limitation in the number of trades or the investments able to be made. Additionally, we assume that borrowing is not allowed and that there is no bid-ask spread. This constraint matches the familiar concept that the percentage weights for stocks in a portfolio must add up to one.

$$\sum_{i=1}^{N} X_i + C(B_i + S_i) \leq \sum_{i=1}^{N} O_i$$

The next set of constraints reflects that the amount of the stock held, bought, and sold must balance to the amount originally invested. Additionally, we assume that short-selling is not allowed, required the amount sold to be less than or equal to the amount originally invested.

$$X_i - B_i + S_i = O_i$$

$$S_i \leq O_i$$

Optionally, the user can specify a risk-free rate of return. Otherwise, there is a constraint that the amount in the risk-free asset is zero. The amount invested in the risk-free asset is itself

not a decision variable but a function of decision variables. The conditional constraint for no risk-free asset is shown below and reflects that the amount held must equal the total original portfolio value less any commissions.

$$\sum_{i=1}^{N} X_i = \sum_{i=1}^{N} O_i - C(B_i + S_i)$$

The last conditional constraint is active if the user specifies a minimum expected daily return G. The basis for this return is the original portfolio value and includes any return from the risk-free asset if available. This analysis does not identify any single utility function from which to define an investor's risk appetite, so a simple return goal is used instead. Without this goal, the focus of the optimization is entirely on the benefit that diversification provides when minimizing variance.

$$\sum_{i=1}^{N} X_i R_i + X_{rf} R_{rf} = G \sum_{i=1}^{N} O_i$$

For values on the efficient frontier, we can also calculate the optimal risky portfolio as the combination of risk and return that maximizes the Sharpe ratio discussed earlier. This calculation is performed by iterating over the efficient frontier space rather than being a separate Gurobi Python optimization. This is possible because the efficient frontier significantly reduces the number of options that require consideration. The results from that additional analysis is the optimal risky portfolio which takes into consideration both return and risk.

The underlying weakness of the Sharpe Ratio is that return (r_X) must be normally distributed. Ratio cannot perform well if distribution is skewed as the standard deviation does not have same effect. The return can be measured in any frequency as long as it is normally distributed.

Computational Experiment and Results

Within the Python framework created, we create a simple portfolio comprised of the following investments formatted as JSON key value pairs and imported as a Python dictionary. In our business problem, this is the portfolio provided by the investor that is not currently optimized for any objective.

Investment	Current Portfolio
Risk-Free	$ -
AAPL	$ 1,000.00
MSFT	$ 2,000.00
INTC	$ 1,000.00
AMZN	$ -
GOOGL	$ 3,000.00
NFLX	$ 3,000.00

Historical daily returns are calculated for the possible stocks in 2018. Returns are based on the log-difference of the adjusted closing prices calculated by Yahoo! Finance (Yahoo!). This historical data serves as the basis for the expected return and the covariance matrix of the returns. A single year was selected to reduce the impact of any seasonality when selecting a part of the year.

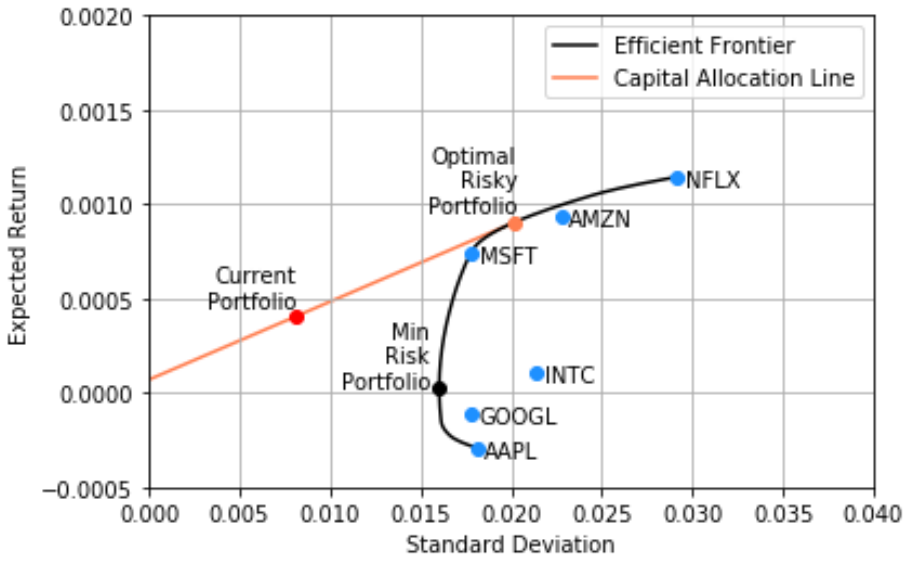
The portfolio is optimized when given parameters regarding cost of transactions assumed 1%, daily risk-free rate assumed 0.007%, and daily return minimum assumed 0.04%. Optimization is handled by Gurobi Python contained within functions that can be called via other scripts. The program determines the amount of each security to buy and sell in order to minimize variance while generating the minimum return required. The basis for both the growth and volatility calculations are the portfolio. The output of the optimization is shown below.

AAPL: 0.00, Return: -0.0293%, Buy: 0.00, Sell: 1000.00
MSFT: 1740.68, Return: 0.0737%, Buy: 0.00, Sell: 259.32
INTC: 0.00, Return: 0.0106%, Buy: 0.00, Sell: 1000.00

AMZN: 1165.00, Return: 0.0935%, Buy: 1165.00, Sell: 0.00
GOOGL: 0.00, Return: -0.0107%, Buy: 0.00, Sell: 3000.00
NFLX: 1059.58, Return: 0.1144%, Buy: 0.00, Sell: 1940.42
RISK-FREE: 5951.09, Buy: 5951.09, Sell: 0.00
Minimum Daily Volatility = 0.7996%
Expected Daily Return = 0.0400%

The program suggests that to minimize variance, the investor should reallocate their funds between the available stocks and purchase a large amount of the risk-free asset. As discussed earlier, the proportions of the stocks make up an optimal risky portfolio and the amount invested in the risk-free asset is used to reduce the overall risk while meeting the goal return.

The optimization returns the resulting portfolio which can be saved as a JSON file as well. This functionality allows the program to be used on a regular basis to rebalance the portfolio in an optimal way as new data becomes available. The resulting portfolio can also be visualized in relation to key features. The plot for the portfolio discussed is shown below. The y-axis represents daily return while the x-axis shows daily volatility. The plot includes the historical data for individual stocks, the optimal risky portfolio, the minimum risk portfolio of stocks, the current portfolio, the efficient frontier, and the capital allocation line.



The code to define the functions needed for Python optimization is found in Appendix A. The specific use of the functions to solve the business problem that we have described is found in Appendix B.

We also analyzed other options for optimization to perform sensitivity analysis for the model excluding transaction costs. The table below considers the optimization of maximizing return without the risk-free asset, minimizing variance without the risk-free asset, and maximizing Sharpe Ratio respectively. The resulting portfolio weights vary greatly based on the selection of the goal. That reinforces the importance of using an optimal risky portfolio that is on the efficient frontier because it considers multiple goals in a structured way.

Investment	Max Return	Mean-Variance	Sharpe Ratio
NFLX	100%	0%	27%
GOOGL	0%	30%	0%
AMZN	0%	0%	30%
INTC	0%	15%	0%
MSFT	0%	20%	44%
AAPL	0%	35%	0%
Portfolio Variance	0.0847%	0.0256%	0.0405
Sharpe Ratio	0.0369	-0.0028	0.0414

Appendix C contains screenshots of the model definitions as defined in Excel while Appendix D shows the respective sensitivity reports for maximizing returns and minimizing variance. The results of the sensitivity analysis show minimal allowable increases and decreases which makes intuitive sense based on the considerations of the models.

As a result of the entire analysis, the investor now has a good indication that they are achieving minimum variance results for this portfolio while meeting their goals. They also know what actions are needed by looking at the decision variables.

Discussion and Conclusions

Our research for this project highlights that there are many considerations and options available to optimize a portfolio. If we look at the results from the optimization models, the preferred solution to the business problem is a mix between the optimal risky portfolio based on maximizing the Sharpe ratio and the risk-free asset. This is true even for the minimum risk combination of stocks. It has the higher portfolio return compared to all other models and the cost for greater overall performance is either significantly higher variance or would require borrowing. The key take way from this analysis is that these models provide a solid framework for portfolio optimization. However, that the goals of the investment and risk appetite of the investor must be considered when setting up the model for portfolio optimization. Our research into literature and exploration of the model gave us much better understanding of the methods and underlying concepts of portfolio optimization. The implication of the work to the business problem, is that we were able to take an existing portfolio and provide the investor with insights on how to minimize variance while meeting a number of constraints related to returns, transaction costs, and the risk-free asset.

Although, the dataset we chose was quite small to provide the realistic portfolio, it was sufficient enough for us to understand the existing frameworks. The limited sample of the data and performance of the well-known stocks meant that we had overall low variance and high returns on the investment. Also, the time horizon is an important factor for building a portfolio optimization model, our dataset spawns only one-year timeframe. Using a longer period of data might have captured more volatility over an economic cycle. Additionally, the model does not allow for short-selling, bid-ask spread, or borrowing which could be enhanced for future work.

References

Markowitz, H. M. (1952). Portfolio Selection/Harry Markowitz. *The Journal of Finance*, *7*(1), 77-91

Sharpe, W. F. (1966). Mutual fund performance. *The Journal of business*, *39*(1), 119-138.

Statman, M. (1987). How many stocks make a diversified portfolio?. *Journal of financial and quantitative analysis*, *22*(3), 353-363.

Yahoo! Finance. (n.d.). Historical data. Retrieved from https://finance.yahoo.com/.

Appendix

A. Supporting Python Script (optimization.py) – GitHub Link

```
"""Execute complex portfolio optimization tasks."""

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from gurobipy import Model, GRB
import fix_yahoo_finance as yf


def calc_return(stock_payload, symbol):
    """Calculate daily stock returns given a pandas DataFrame of price data."""
    # Use difference in natural log as return
    stock_payload['Return'] = np.log(stock_payload['Adj Close'])
    stock_return = stock_payload[['Return']].diff().dropna()
    # Rename column as its symbol
    stock_return.columns = [symbol]
    return stock_return


def calc_port_returns(symbols, start_date, end_date):
    """Combine daily returns from all possible stocks in the portfolio.

    Symbol of RISK-FREE will be ignored for return data.

    Args:
        symbols (list): strings of each stock symbol e.g. 'AAPL'.
        start_date (str): first day of returns to select YYYY-MM-DD
        end_date (str): last day of returns to select YYYY-MM-DD
    Returns:
        a pandas dataframe of daily returns with date as the index.
    """
    try:
        symbols.remove('RISK-FREE')
    except ValueError:
        pass
    for i, symbol in enumerate(symbols):
        # Download daily stock price data
        stock_payload = yf.download(symbol, start_date, end_date,
                                    progress=False).dropna()
        # First symbol will be base with subsequent joining
        if i == 0:
            base = calc_return(stock_payload, symbol)
        else:
            base = base.join(calc_return(stock_payload, symbol))
    return base


def optimize_portfolio(portfolio, hist_data, cost=0, risk_free=None,
                       growth=None, verbose=False):
    """Optimize a given stock portfolio to minimize risk.

    This model assumes no short salesself.
    Risk free asset has assumed symbol: 'RISK-FREE'.

    Args:
        portfolio (dict): stock symbol and balance key-value pairs.
        hist_data (DataFrame): historical stock returns
        cost (float): % cost of transactions (default=0)
        risk_free (float): risk free rate (default no risk free assset)
        growth (float): minimum daily return required (default=None)
        verbose (bool): whether to print results (default=False)
    Returns:
        a dictionary of the new portfolio and risk statistics.

    """
```

```
# Initialize model
model = Model('portfolio')

# Decision Variables
# Variables for amount to invest in each stock (continuous with lb=0)
xvars = pd.Series(model.addVars(hist_data.columns.tolist(), name='x'),
                  index=hist_data.columns.tolist())
# Variables for amount to buy and sell of each stock
bvars = pd.Series(model.addVars(hist_data.columns.tolist(), name='b'),
                  index=hist_data.columns.tolist())
svars = pd.Series(model.addVars(hist_data.columns.tolist(), name='s'),
                  index=hist_data.columns.tolist())

# Objective - Minimize portfolio risk
portfolio_risk = hist_data.cov().dot(xvars).dot(xvars)
model.setObjective(portfolio_risk, GRB.MINIMIZE)

# Calculated Variables
# Budget is sum of current holdings
budg = sum(portfolio.values())
# Amount in risk free investments is the budget less commission and stocks
amt_risk_free = budg - cost*(bvars.sum()+svars.sum()) - xvars.sum()

# Constraints
# Enforce being at or under current holdings
model.addConstr(xvars.sum() + cost*(bvars.sum()+svars.sum()) <= budg,
                "budget")
# Enforce that transactions balance and no short sales
for i, stock in enumerate(hist_data.columns.tolist()):
    stock_budget = portfolio[hist_data.columns.tolist()[i]]
    model.addConstr(xvars[i] - bvars[i] + svars[i] == stock_budget,
                    f"budget {stock}")
    model.addConstr(svars[i] <= stock_budget, f"no short {stock}")
# Optionally require no risk free assets
portfolio_return = np.dot(hist_data.mean(), xvars)
if risk_free is not None:
    portfolio_return += risk_free*amt_risk_free
else:
    model.addConstr(xvars.sum() == (budg - cost*(bvars.sum()+svars.sum())),
                    "no rf")
# Optionally require a minimum return
if growth is not None:
    model.addConstr(portfolio_return == growth*budg, "return")
model.update()

# Run the model
model.setParam('OutputFlag', 0)
model.optimize()

# Save xvars to new portfolio
new_portfolio = {}
for i, stock in enumerate(hist_data.columns.tolist()):
    if verbose:
        print(f"{stock}: {xvars[i].x:0.2f}, "
              f"Return: {100*hist_data.mean()[i]:0.4f}%, "
              f"Buy: {bvars[i].x:0.2f}, "
              f"Sell: {svars[i].x:0.2f}")
    new_portfolio[stock] = np.round(xvars[i].x, 2)
minrisk_volatility = np.sqrt(portfolio_risk.getValue()) / budg
minrisk_return = portfolio_return.getValue() / budg

try:
    amt_rf_change = amt_risk_free.getValue() - portfolio['RISK-FREE']
except KeyError:
    amt_rf_change = amt_risk_free.getValue()
if verbose:
    if amt_rf_change == 0:
        print(f"RISK-FREE: {amt_risk_free.getValue():0.2f}, "
              f"Buy: {0:0.2f}, "
              f"Sell: {0:0.2f}")
    elif amt_rf_change < 0:
```

```
            print(f"RISK-FREE: {amt_risk_free.getValue():0.2f}, "
                  f"Buy: {0:0.2f}, "
                  f"Sell: {-amt_rf_change:0.2f}")
        else:
            print(f"RISK-FREE: {amt_risk_free.getValue():0.2f}, "
                  f"Buy: {amt_rf_change:0.2f}, "
                  f"Sell: {0:0.2f}")
    new_portfolio['RISK-FREE'] = np.round(amt_risk_free.getValue(), 2)
    if verbose:
        print(f"Minimum Daily Volatility = {100*minrisk_volatility:0.4f}%")
        print(f"Expected Daily Return = {100*minrisk_return:0.4f}%")

    # Save model statistics for output
    model_stats = {'stock_return': hist_data.mean(),
                   'stock_volatility': hist_data.std(),
                   'minrisk_volatility': minrisk_volatility,
                   'minrisk_return': minrisk_return}

    return new_portfolio, model_stats


def calc_current_stats(portfolio, hist_data, risk_free):
    """Calculate current portfolio stats."""
    stock_amt = {k: portfolio[k] for k in hist_data.columns.tolist()}.values()
    budg = sum(portfolio.values())

    amt_risk_free = budg - sum(stock_amt)
    portfolio_return = np.dot(hist_data.mean(), list(stock_amt))
    portfolio_return += risk_free*amt_risk_free
    portfolio_risk = hist_data.cov().dot(list(stock_amt)).dot(list(stock_amt))
    return np.sqrt(portfolio_risk) / budg, portfolio_return / budg


def efficient_frontier(portfolio, hist_data, risk_free=0):
    """Plot efficient frontier."""
    non_rf_port = {i: portfolio[i] for i in portfolio if i != 'RISK-FREE'}
    _starting_portfolio, starting_stats = optimize_portfolio(non_rf_port,
                                                             hist_data)
    # Plot optimal risky portfolio
    curr_risk, curr_return = calc_current_stats(portfolio, hist_data,
                                                risk_free)
    # Solve for efficient frontier by varying target return
    eff_frontier = pd.Series()
    # Save optimally risky portfolio
    max_sharpe = 0
    for target in np.linspace(starting_stats['stock_return'].min(),
                              starting_stats['stock_return'].max(), 500):
        _loop_port, loop_stats = optimize_portfolio(non_rf_port, hist_data,
                                                    growth=target)
        eff_frontier.loc[loop_stats['minrisk_volatility']] = target
        risk_prem = loop_stats['minrisk_return'] - risk_free
        if risk_prem / loop_stats['minrisk_volatility'] > max_sharpe:
            optimal_risk = loop_stats['minrisk_volatility']
            optimal_return = loop_stats['minrisk_return']
            max_sharpe = risk_prem / loop_stats['minrisk_volatility']

    # Solve for capital allocation line
    ca_line = pd.Series()
    for risk_level in np.linspace(0, optimal_risk, 500):
        slope = (optimal_return - risk_free) / optimal_risk
        ca_line.loc[risk_level] = risk_free + risk_level*slope

    # Plot volatility versus expected return for individual stocks
    ax_rr = plt.gca()

    # Plot efficient frontier
    eff_frontier.plot(color='Black', label='Efficient Frontier', ax=ax_rr,
                      zorder=2)

    # Plot capital allocation line
    ca_line.plot(color='Coral', label='Capital Allocation Line', ax=ax_rr,
```

```
                zorder=1)

    ax_rr.scatter(x=starting_stats['stock_volatility'],
                  y=starting_stats['stock_return'],
                  color='dodgerblue', zorder=3)
    for i, stock in enumerate(hist_data.columns.tolist()):
        ax_rr.annotate(stock,
                       (starting_stats['stock_volatility'][i] + 0.0004,
                        starting_stats['stock_return'][i] - 0.00005))

    # Plot volatility versus expected return for minimum risk portfolio
    ax_rr.scatter(x=starting_stats['minrisk_volatility'],
                  y=starting_stats['minrisk_return'], color='Black', zorder=4)
    ax_rr.annotate('Min\nRisk\nPortfolio',
                   (starting_stats['minrisk_volatility'] - 0.0005,
                    starting_stats['minrisk_return']),
                   horizontalalignment='right')

    # Plot optimal risky portfolio
    ax_rr.scatter(x=optimal_risk, y=optimal_return, color='Coral', zorder=5)
    ax_rr.annotate('Optimal\nRisky\nPortfolio',
                   (optimal_risk + 0.0001, optimal_return + 0.00005),
                   horizontalalignment='right')

    ax_rr.scatter(x=curr_risk, y=curr_return, color='Red', zorder=6)
    ax_rr.annotate('Current\nPortfolio',
                   (curr_risk + 0.0001, curr_return + 0.00005),
                   horizontalalignment='right')

    # Format and display the final plot
    ax_rr.axis([0, 0.04, -0.0005, 0.002])
    ax_rr.set_xlabel('Standard Deviation')
    ax_rr.set_ylabel('Expected Return')
    ax_rr.legend()
    ax_rr.grid()
    plt.show()


if __name__ == '__main__':
    # Starting stock portfolio
    HOLDINGS = {'AAPL': 1000, 'MSFT': 2000, 'INTC': 1000, 'AMZN': 0,
                'GOOGL': 3000, 'NFLX': 3000}
    # Download historical returns data
    STOCK_DATA = calc_port_returns(list(HOLDINGS.keys()), '2018-01-01',
                                   '2018-12-31')
    # Complex model example
    NEW_PORTFOLIO, MODEL_STATS = optimize_portfolio(HOLDINGS, STOCK_DATA,
                                                    cost=0.01,
                                                    risk_free=0.00007,
                                                    growth=0.0004,
                                                    verbose=True)
    # Example efficient frontier
    efficient_frontier(NEW_PORTFOLIO, STOCK_DATA, risk_free=0.00007)
```

B. Screenshots from Jupyter Notebook Solving Business Problem

Portfolio Optimization Examples

Alan Kessler and Anil Koluguri, MSDS 460 Section 56

Import **optimization.py** created for the project.

In [1]:
```
%matplotlib inline
import json
import optimization
```

A portfolio can be saved as a JSON file for documentation purposes. It represents the amount invested in each asset. The amount stored in an assumed risk-free asset is shown with symbol `RISK-FREE`.

In [2]:
```
current_file = open("portfolio_example.json")
current_port = json.load(current_file)
current_file.close()
current_port
```

Out[2]:
```
{'AAPL': 1000,
 'MSFT': 2000,
 'INTC': 1000,
 'AMZN': 0,
 'GOOGL': 3000,
 'NFLX': 3000}
```

The script downloads historical daily returns for the investments in the portfolio, ignoring the risk free asset which is assumed theoretical. The data is from Yahoo Finance and the connection may be restricted, so restart and try again if an error is returned.

In [3]:
```
stock_data = optimization.calc_port_returns(list(current_port.keys()),
                                            start_date='2018-01-01',
                                            end_date='2018-12-31')
stock_data.head()
```

Out[3]:

	AAPL	MSFT	INTC	AMZN	GOOGL	NFLX
Date						
2018-01-03	-0.000174	0.004643	-0.034527	0.012694	0.016917	0.019601
2018-01-04	0.004634	0.008763	-0.018509	0.004466	0.003877	0.002825
2018-01-05	0.011321	0.012322	0.006953	0.016033	0.013173	0.020981
2018-01-08	-0.003721	0.001020	0.000000	0.014322	0.003524	0.009762
2018-01-09	-0.000115	-0.000680	-0.025352	0.004665	-0.001275	-0.013006

The optimization script minimizes the standard deviation of the daily returns. It is possible to incorporate a percentage commission for transactions as well as a minimum return requirement.

In [4]:
```
opt_port, _port_stats = optimization.optimize_portfolio(current_port,
                                                        stock_data,
                                                        cost=0.01,
                                                        risk_free=0.00007,
                                                        growth=0.0004,
                                                        verbose=True)
```

```
Academic license - for non-commercial use only
AAPL: 0.00, Return: -0.0293%, Buy: 0.00, Sell: 1000.00
MSFT: 1740.68, Return: 0.0737%, Buy: 0.00, Sell: 259.32
INTC: 0.00, Return: 0.0106%, Buy: 0.00, Sell: 1000.00
AMZN: 1165.00, Return: 0.0935%, Buy: 1165.00, Sell: 0.00
GOOGL: 0.00, Return: -0.0107%, Buy: 0.00, Sell: 3000.00
NFLX: 1059.58, Return: 0.1144%, Buy: 0.00, Sell: 1940.42
RISK-FREE: 5951.09, Buy: 5951.09, Sell: 0.00
Minimum Daily Volatility = 0.7996%
Expected Daily Return = 0.0400%
```

The new minimal risk portfolio is also stored as a dictionary.

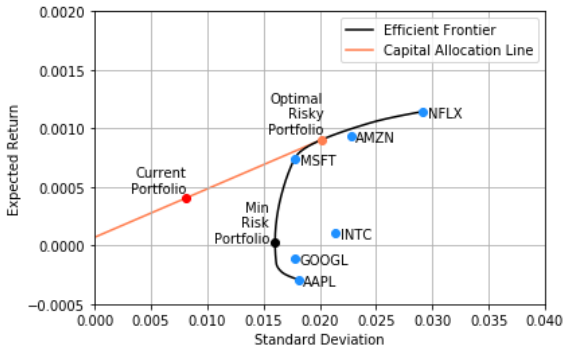
This portfolio can also be saved as a JSON file for documentation. This makes it possible to update the optimal portfolio as new data becomes available.

In [5]:
```
with open("updated_portfolio.json", "w") as opt_port_file:
    json.dump(opt_port , opt_port_file)
```

There is also the option to visualize the various portfolio options including individual stocks, the efficient frontier, and the capital allocation line.

When the risk free asset is used, the resulting portfolio be on the capital allocation line. The mix between the optimal risky porfolio and risk free asset is based on risk preferences.

In [6]:
```
optimization.efficient_frontier(opt_port,
                                stock_data,
                                risk_free=0.00007)
```



C. Screenshots of the Excel Models

Maxmizing Return

	NFLX	GOOGL	AMZN	INTC	MSFT	AAPL	Total
Portfolio %	100%	0%	0%	0%	0%	0%	100%
Mean Return	0.001144258	-0.000106702	0.000934626	0.000106236	0.000736583	-0.000292621	

Covariance Matrix

	NFLX	GOOGL	AMZN	INTC	MSFT	AAPL
NFLX	0.000847118	0.000335008	0.000485747	0.000324911	0.000348388	0.000289257
GOOGL	0.000335008	0.000316477	0.000299385	0.000222334	0.000259741	0.000215629
AMZN	0.000485747	0.000299385	0.000517785	0.000242106	0.000313470	0.000271229
INTC	0.000324911	0.000222334	0.000242106	0.000455543	0.000241684	0.000207401
MSFT	0.000348388	0.000259741	0.000313470	0.000241684	0.000316294	0.000224406
AAPL	0.000289257	0.000215629	0.000271229	0.000207401	0.000224406	0.000327264

Variance	0.000847118	0.000000000	0.000000000	0.000000000	0.000000000	0.000000000
Weights	0.001144258	0.000000000	0.000000000	0.000000000	0.000000000	0.000000000

Portfolio Variance	0.000847118
Mean	0.001144258

Standard Deviation	0.02910529

Minimizing Variance

	NFLX	GOOGL	AMZN	INTC	MSFT	AAPL	Total
Portfolio %	0%	30%	0%	15%	20%	35%	100%
Mean Returns	0.001144258	-0.000106702	0.000934626	0.000106236	0.000736583	-0.000292621	

Covariance Matrix

	NFLX	GOOGL	AMZN	INTC	MSFT	AAPL
NFLX	0.000847118	0.000335008	0.000485747	0.000324911	0.000348388	0.000289257
GOOGL	0.000335008	0.000316477	0.000299385	0.000222334	0.000259741	0.000215629
AMZN	0.000485747	0.000299385	0.000517785	0.000242106	0.000313470	0.000271229
INTC	0.000324911	0.000222334	0.000242106	0.000455543	0.000241684	0.000207401
MSFT	0.000348388	0.000259741	0.000313470	0.000241684	0.000316294	0.000224406
AAPL	0.000289257	0.000215629	0.000271229	0.000207401	0.000224406	0.000327264

Variances	0.000000000	0.000076937	0.000000000	0.000038129	0.000050100	0.000090461
Weights	0.000000000	-0.000032115	0.000000000	0.000015846	0.000144361	-0.000103552

Portfolio Variance	0.000255627
Mean	0.000024540

0.0255627

Standard Deviation	0.01598834

Maximizing Sharpe Ratio

	NFLX	GOOGL	AMZN	INTC	MSFT	AAPL	Total
Portfolio %	27%	0%	30%	0%	44%	0%	100%
Mean Returns	0.001144258	-0.000106702	0.000934626	0.000106236	0.000736583	-0.000292621	

Covariance Matrix

	NFLX	GOOGL	AMZN	INTC	MSFT	AAPL
NFLX	0.000847118	0.000335008	0.000485747	0.000324911	0.000348388	0.000289257
GOOGL	0.000335008	0.000316477	0.000299385	0.000222334	0.000259741	0.000215629
AMZN	0.000485747	0.000299385	0.000517785	0.000242106	0.000313470	0.000271229
INTC	0.000324911	0.000222334	0.000242106	0.000455543	0.000241684	0.000207401
MSFT	0.000348388	0.000259741	0.000313470	0.000241684	0.000316294	0.000224406
AAPL	0.000289257	0.000215629	0.000271229	0.000207401	0.000224406	0.000327264

Variances	0.000139054	0.000000000	0.000125289	0.000000000	0.000141124	0.000000000
Weights	0.000304726	0.000000000	0.000278633	0.000000000	0.000320832	0.000000000

Portfolio Variance	0.000405467
Mean	0.000904191

Standard Deviation	0.020136208	Risk-Free	0.00007
Sharpe Ratio	0.041427424		

D. Screenshots of Excel Sensitivity Analysis

Worksheet: [Group_1_Final_Project-Corrected.xlsx]MAX-Return
Report Created: 3/7/2019 6:09:51 AM
Engine: Gurobi Solver

Objective Cell (Max)

Cell	Name	Final Value
C22	Mean NFLX	0.001144258

Decision Variable Cells

Cell	Name	Final Value	Reduced Cost	Objective Coefficient	Allowable Increase	Allowable Decrease
C4	Portfolio % NFLX	100%	0%	0.001144258	1E+100	0.000209631
D4	Portfolio % GOOGL	0%	0%	-0.000106702	0.00125096	1E+100
E4	Portfolio % AMZN	0%	0%	0.000934626	0.000209631	1E+100
F4	Portfolio % INTC	0%	0%	0.000106236	0.001038022	1E+100
G4	Portfolio % MSFT	0%	0%	0.000736583	0.000407675	1E+100
H4	Portfolio % AAPL	0%	0%	-0.000292621	0.001436878	1E+100

Constraints

Cell	Name	Final Value	Shadow Price	Constraint R.H. Side	Allowable Increase	Allowable Decrease
I4	Portfolio % Total	100%	0%	1	1E+100	1

Worksheet: [Group_1_Final_Project-Corrected.xlsx]Mean-Variance
Report Created: 3/7/2019 6:11:25 AM
Engine: Gurobi Solver

Objective Cell (Min)

Cell	Name	Final Value
C21	Portfolio Variance NFLX	0.000255627

Decision Variable Cells

Cell	Name	Final Value	Reduced Cost	Objective Coefficient	Allowable Increase	Allowable Decrease
C4	Portfolio % NFLX	0%	0%	0	1E+100	0.000128614
D4	Portfolio % GOOGL	30%	0%	0	5.61928E-05	5.99586E-05
E4	Portfolio % AMZN	0%	0%	0	1E+100	5.6022E-05
F4	Portfolio % INTC	15%	0%	0	0.000195843	7.94089E-05
G4	Portfolio % MSFT	20%	0%	0	8.62939E-05	3.31941E-05
H4	Portfolio % AAPL	35%	0%	0	0.00013843	0.000119072

Constraints

Cell	Name	Final Value	Shadow Price	Constraint R.H. Side	Allowable Increase	Allowable Decrease
I4	Portfolio % Total	100%	0%	1	1.825821387	1