MIE376

Analysis on Stochastic Programming in Portfolio Optimization

Hang Zuo Xiang
hang.xiang@mail.utoronto.ca
1002376476

and Yiqing (Louis) Luo
louis.luo@mail.utoronto.ca
1002449059

Due: April 11th, 2018

Contents

1 Introduction

The motivation is to formulate and optimize investment strategies to reach some monetary targets using a two-stage stochastic programming model. Twenty assets from the S&P500 were considered in the investment environment, and the data spanned from January 2013 to Dec 2015. The generation of scenarios was performed through modeling future returns as a normal distribution of the past. For the experiment, fifty scenarios were used for different target values. The behavior of the stochastic program was examined by analyzing weights with different target values, and the stochastic solution was compared to the deterministic model of the same prompt.

2 Methodology

2.1 Data Process

The investment universe consisted of 20 stocks (n = 20), all constituents of the S&P500. The list of stocks can be found in Table 1. The weekly adjusted closing price data for these assets (Quandl.com, 2017) collected spans 3 years from January 2013 to December 2015. The use of adjusted closing price is standard in the industry and is adjusted for stock splits and dividend payments, which could otherwise distort the time-series. The data is split in preparation of both in-sample and out-of-sample training and testing. The in-sample process uses data from January 2013 to October 2015 while the out-of-sample process uses data from November 2015 to December 2015.

Table 1: List of assets in Investment Environment

Ford (F)	Caterpillar (CAT)	Disney (DIS)
Apple(APPL)	IBM (IBM)	Pfizer Inc (PFE)
McDonald (MCD)	The Coca-Cola Co (KO)	Pepsi (PEP)
Johnson and Johnson (JNJ)	Exxon Mobil Corporation (XOM)	Marathon Oil Corporation (MRO)
Walmart (WMT)	CitiGroup Inc (C)	Wells Fargo Co (WFC)
Consolidated Edison, Inc. (ED)	Tesla (T)	Verizon Communications Inc (VZ)
JP Morgan (JPM)	Newmont Mining Corp (NEM)	-

2.1.1 In-Sample Data Processing

In-sample data processing consists of the calculation of monthly returns, and the generation of the first two statistical moments.

Monthly Returns:

The monthly return at any given time period is given by the following:

$$r_{t,monthly} = \prod_{n=0}^{3}(1 + r_{t+n,weekly}) \approx 1 + \sum_{n=0}^{3} r_{t+n,weekly} \tag{1}$$

where t represent the date index at the start of each month

However, a major challenge encountered in the in-sampling process is the lack of quality monthly return data available. While historical price data prior to 2013 is easily accessible, only data after 2013 is selected because data too far from the past may contain unjustified historical bias to them and would not accurately depict the true sample mean expected in the nearby future.

Another approach to generate more data points is to use weekly returns within a 4 week interval for every week available. By doing so, we can generate approximately the same number of monthly returns as the weekly returns.

$$r_{t,monthly} = \prod_{n=0}^{3}(1 + r_{t+n,weekly}) \approx 1 + \sum_{n=0}^{3} r_{t+n,weekly} \tag{2}$$

where t represent the date index for every available week.

However, while the number of sample data increased, so does the correlation among each returns. Since there are significant overlaps for each monthly return with subsequent monthly returns, this approach induces a high correlation factor between data points, hence overfitting the past distributions.

$$r_{t,monthly} = \frac{P_{t+4} - P_t}{P_t} \tag{3}$$

where P_t is the weekly price with time index t.

Statistical Moments

The first moment, also known as the expectation value, can either be obtained through arithmetic mean and geometric mean. For raw price data, as well as numerical portfolio values the arithmetic mean is taken. For returns in percentage, the geometric mean is taken.

$$E[X] = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{4}$$

The second moment, also known as the variance and co-variance, is obtained through the

sample covariance formula.

$$Cov[X,Y] = \frac{1}{n-1} E[(X - E[X])(Y - E[Y])] \tag{5}$$

2.1.2 Out-of-Sample Data Processing

The out of sample data consists of the last two month of the data collected, spanning from Nov 2015 to Dec 2015. The monthly return is calculated from the weekly prices using the same techniques described in in-sample.

To analyze the behavior of the portfolios with respect to different targets, the portfolio values with the weightings determined by the stochastic and deterministic programs are plotted against time in monthly intervals. The scenarios were also plotted against a monthly timeline to see the stochastic behavior.

2.2 Stochastic Programming Formulation

As a two stage stochastic programming question, the optimization contains two parts which are the deterministic stage and the in-deterministic stage. Two time periods were considered.

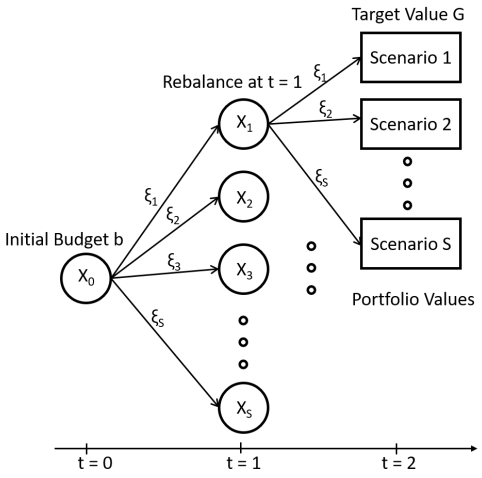

Figure 1: A graphical representation of the portfolio value corresponding to different scenarios

The exact formulation of the stochastic program was as following:

$$
\begin{aligned}
\min_{X} \quad & \mathbf{1}^T X_0 + \mathbb{E}(q \times q^+_{surplus} - r \times q^-_{shortfall}) \\
\text{s.t.} \quad & \mathbf{1}^T X_0 \leq b \\
& \sum_{i=1}^{S} \xi_i X_1 - q^+_{surplus} + q^-_{shortfall} = G \\
& \sum_{i=1}^{S} \xi_i X_0 - X_1 = 0 \\
& \sum_{i=1}^{S} \xi_i X_1 - X_2 = 0 \\
& \forall X_t \geq 0, q^+_{surplus} \geq 0, q^-_{shortfall} \geq 0
\end{aligned}
$$

where the parameters to the optimizer are as follows:

- b is a scalar for the initial funding.
- G is a scalar for the target ending value at t = 2.
- q and r are both scalars corresponding to the reward and penalty associated with surplus and slack. In this experiment, q = 1 and r = 2.

the stochastic variable to the optimizer is as follows:

- ξ_i corresponds to the i^{th} scenario.

and the decision variables to the optimizer are as follows:

- $X_i \in R^n$ is a vector of n assets representing the monetary wealth invested in each of the assets at time = 0. Specifically, $t = 0$ refers the first stage, and X_i with $t > 0$ represents the wealth at scenarios at t = 1 month.
- $q^+_{surplus}$ is the surplus variable.
- $q^-_{shortfall}$ is the slack variable.

2.2.1 Deterministic Stage

The variables representing the initial amount of money invested into each n assets (t = 0) were considered in this stage.

The objective function $\mathbf{1}^T X_0$ corresponded to the deterministic stage, which was to minimize the total value of money invested. Given the target return of the portfolio at the end of the second time period was set to G, this objective function minimizes the total amount

of money invested so that the highest rate of return could be obtained.

On top of the objective function, the first constraint, $\mathbf{1}^T X_0 \leq b$, set the upper bound of the total amount of money invested at the beginning. This was to make sure that the starting budget at every period was no greater than the total amount of money given.
Overall, the formulation is as follows:

$$\begin{aligned} \min_{X} \quad & \mathbf{1}^T X_0 \\ \text{s.t.} \quad & \mathbf{1}^T X_0 \leq b \\ & \forall X_t \geq 0 \end{aligned}$$

2.2.2 Scenario Generation

Different scenarios were generated based on the multi-variate normal distribution with mean μ and variance σ^2, which were generated from previous years' data and were assumed to be the population mean and variance for the multi-variate normal distribution.

$$\xi_{\mathbf{i}} \sim \mathcal{N}(\mu, Q) \quad \forall \text{i} = \{1, 2, ..., \text{Number of Scenarios}\} \text{ and } \quad \xi_{\mathbf{i}} \in \mathbb{R}^{20}$$

In our program, results were generated based on 10 scenarios, 100 scenarios, and 1000 scenarios, respectively, to further analyze the effect of the number of scenarios on the final result.

2.2.3 Non-deterministic Stage

As all scenarios are randomly generated from the multi-variate normal distribution, they were considered to be equally likely. Hence, the probability of each scenario was $p(s) = 1/S$, where S was the total number of scenarios generated. The expectation in the objective function, $\mathbb{E}(q \times q^+_{surplus} - r \times q^-_{shortfall})$, can then be expressed as $\sum_{i=1}^{S} \frac{q \times q^+_{surplus} - r \times q^-_{shortfall}}{S}$.

In the objective function, the surplus variables and shortfall variables representing every scenarios were multiplied by their respective reward and penalty ratio, then added up and multiplied by their corresponding probabilities. As shown in $\sum_{i=1}^{S} \frac{q \times q^+_{surplus} - r \times q^-_{shortfall}}{S}$, the reward and penalty ratio indicates the slope of the actual price from below G and from above G leading to G, respectively. For example, in figure 2, the slope of the red line representing the shortfall ratio, r, was larger than the slope of the blue line representing the surplus ratio, q, which indicated that the user cared more about reaching G from below than getting above G.

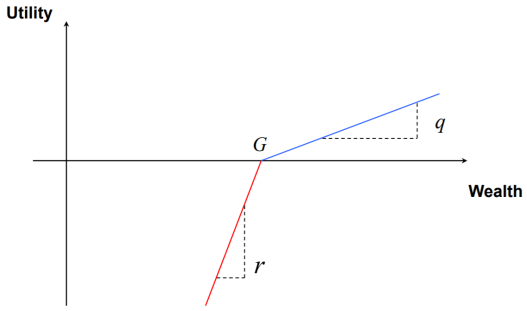

Figure 2: A graphical representation of the reward and penalty ratio

The second to forth constraints in the formulation corresponded to this stage. In the second constraint, the surplus and shortfall variables were introduced to complement the fact that the actual value of the portfolio, which was $\sum_{i=1}^{S} \xi X_1$, may be greater or less than the target return G. Hence, the surplus variable, $q^+_{surplus}$, was subtracted and the shortfall variable, $q^+_{surplus}$, was added to the actual value of the portfolio, respectively. As two balancing periods were considered in our program, the third constraint represented the fact that the initial budget at time 1 equaled to the initial budget at time 0 multiplied by the rate of return corresponding to each scenario, while the forth constraint applied the same logic to the second time period.

By design, all surplus and shortfall variables are greater or equal to zero.

Overall, this formulates to:

$$
\begin{aligned}
\min_{X} \quad & \sum_{i=1}^{S} \frac{q \times q^+_{surplus} - r \times q^-_{shortfall}}{S} \\
\text{s.t.} \quad & \sum_{i=1}^{S} \xi_i X_1 - q^+_{surplus} + q^-_{shortfall} = G \\
& \sum_{i=1}^{S} \xi_i X_0 - X_1 = 0 \\
& \sum_{i=1}^{S} \xi_i X_1 - X_2 = 0 \\
& \forall X_t \geq 0, q^+_{surplus} \geq 0, q^-_{shortfall} \geq 0
\end{aligned}
$$

2.2.4 Testing Stage

After setting up all the parameters and formulating the optimization question, the two stage stochastic program was tested against different conditions including changing number of scenarios, different initial budget, and different target returns. More details are discussed

in the result section.

3 Result

The results of $G = 1100$ from the stochastic problem, as well as the result for the deterministic problem with $G = 1100$, are presented in Appendix A.

Furthermore, the weight distribution of portfolios with different G values were examined at each time period. The percentages each asset occupy the portfolio are summarized in the Figure 3. Note that the weights shown are the averages of the weights across scenarios and would be the recommendation presented to clients.

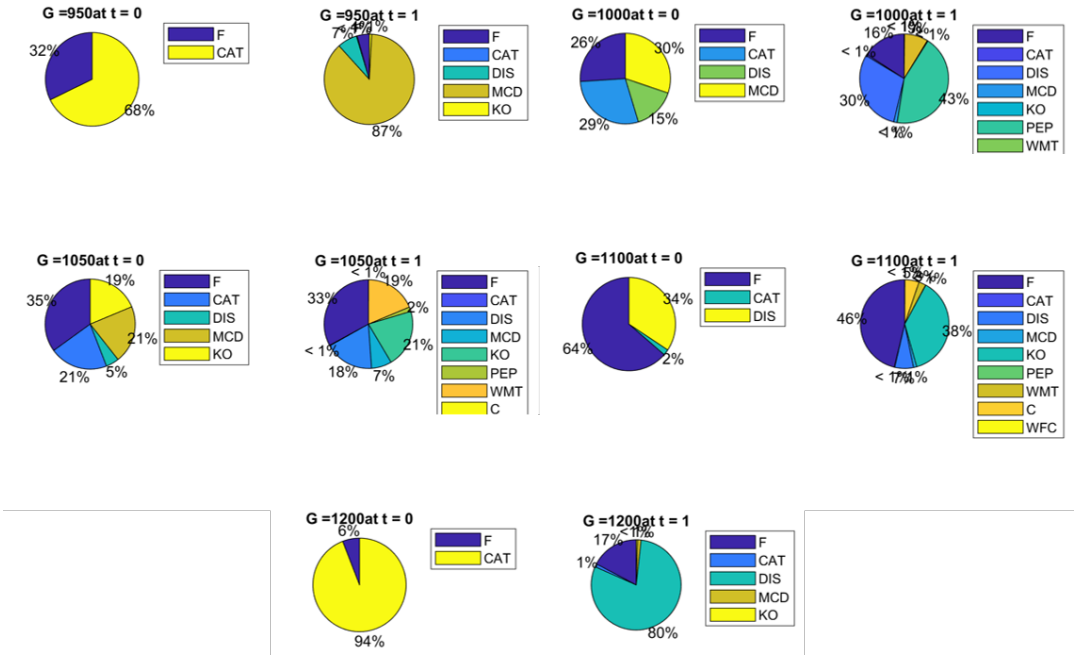

Figure 3: Weight distribution for varying target values (G) at time steps 0 and 1.

By inspection, a trend in regards to the concentration of the portfolio is observed. Portfolios with G values ranging from 1000 to 1100 all exhibited an appreciable diversification compared to both the $G = 950$ case and $G = 1200$ case. This is like due to the modest expectation growth for the 20 assets selected over the testing period. In fact, for $G = 1200$ case, in order to meet a target that was far beyond the expected return of the assets, the stochastic model elected to concentrate most of its asset into one or two assets with a high expected return, such as Disney. This would likely imply that these portfolio, as it attempts to maximize its return, are much more sensitive to any changes in the market.

It is interesting to note that as the number of splits at each node shown in figure Figure 1 increases, so does the diversification of portfolio weights at the two time steps. These weights are presented in the appendix. A likely cause to this is that with the increase in possible scenarios, concentrating in few assets that expected a higher return is not possible as these selected assets can also under-perform, although not as likely. To account for these sub-cases, the optimizer is forced into allocating various proportions of selected assets into other previously not selected ones. Another possible cause to this behavior is that the with fewer scenarios, the mean of these scenarios is not representative of the true sample mean. Hence, with increase to the number of scenarios, the parameters fed into the optimizer more closely resemble the true distribution.

Another interesting visualization is the portfolio values under the out-of-sample tests. For the 5 values of G, the portfolio values at Nov 2015 (Period 2) and Dec 2015 (Period 3) are displayed in figure Figure 4. This initial asset allocation for each value of G is the same for its respective subclass, but results for each scenario differed significantly. One possible explanation is that due to the stochasticity of the model, the expected value of the asset returns varied significantly. More importantly, this behavior should be attributed to the volatile stock price, as the actual return for certain assets differed significantly from the expected return. While the weights for scenarios are somewhat diversified relatively to each G, overall the number of assets included in the portfolios is still quite small. Hence, any disturbance in the market, such as a drop in Disney return, would significantly impact the portfolios generated by the model, which is indeed the case as shown in the result obtained.

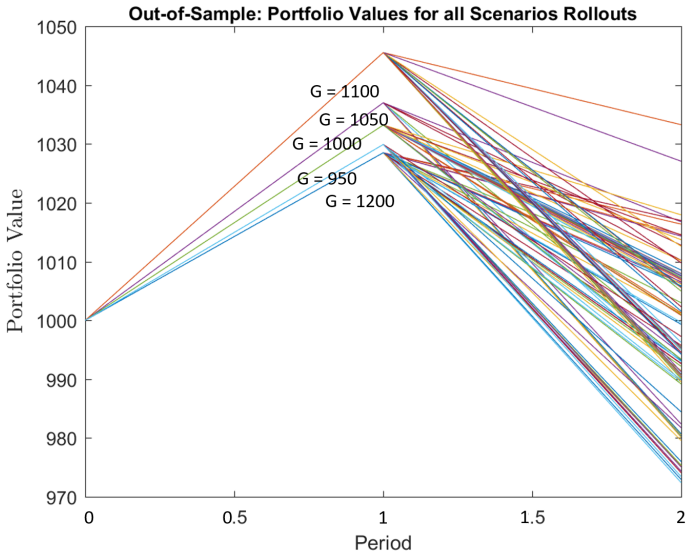

Figure 4: Portfolio Values at Each Month for Different Scenarios

The portfolio values of the average portfolio weights for each G is presented in Figure 5. Again, the average portfolio weights are determined by arithmetic mean taken over all scenario weights given the target goal.

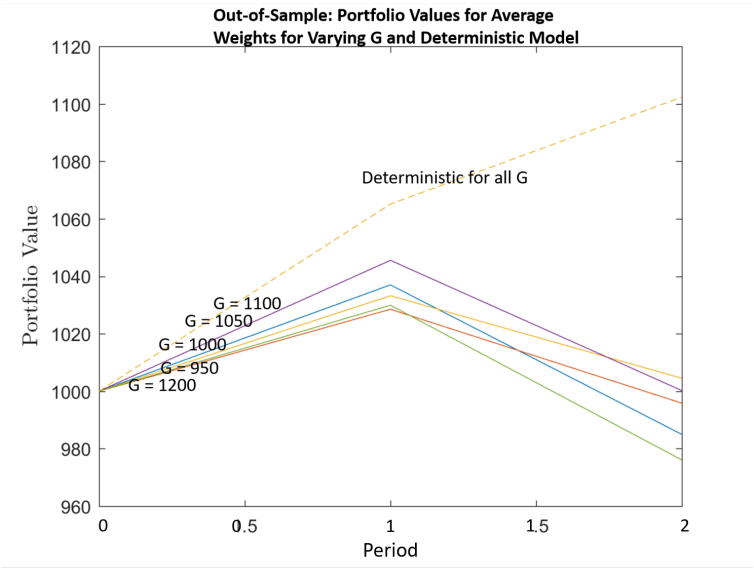

Figure 5: Avg Portfolio Values from Stochastic Model and Deterministic Model at each month

The expected portfolio value with perfect information for the deterministic problem significantly outperformed the portfolio from stochastic programming, which is indeed expected. The EVPI portfolio concentrated all of its stocks into Apple, which had high returns in the testing months.However, it is worth noting that all values of the portfolios generated by stochastic programming all dropped after first rebalance. This suggests that the distribution of wealth, even after the scenario generation, was not fully able to predict the future return. In fact, non of the portfolios favored Apple stocks, but instead preferred Ford stocks. Hence, this demonstrates that while stochastic programming improves robustness of the model, it is highly dependent on the predicted return of the market. If the stocks selected by the stochastic programming failed to perform as expected, it is unable to optimize the portfolio as well as investors would hope.

4 Next Steps and Improvements

Accuracy of Future Predictions The sampling process uses a normal distribution to generate scenarios using historical data. However, a major drawback is that this distribution model does not account for the sudden changes in market trend for each asset. As a result of this change in trend, the evaluation for the effectiveness of the models are not always

consistent. In future, techniques such as Monte Carlo Simulation should be attempted in the scenario generation.

Varying Assets
The only varying variable in this project is G, the target value. Other parameters could be varied in the future as well, such as the number of scenarios, and reward and penalty values.

Expected value of Scenarios In the objective function, the expected value of different scenarios are taken over a uniform distribution. This is a naive approach and in future, different variations of likelihood should also be attempted to further test the behavior of portfolios generated by the stochastic program.

5 Instruction on running Matlab Files

All function calls and code are ran in the *main.m* file, which generates all results and figures aforementioned. The stochastic optimizer is in the *solver.m*. Both files are reproduced in Appendix B.1 and B.2. Note that as is the nature of stochastic programming, the exact replication of figures and results is not likely, but the trends and insights should still hold.

References

Quandl.com (2017). Wiki – various end-of-day stock prices. `https://www.quandl.com/product/WIKIP/usage/export`. [Online; accessed 07-Nov-2017].

A Five portfolios generated from five distinct scenarios

Table 2: Scenario 1

Ticker Name	SS (t=0)	SS(t=1)	DS(t=0)	DS(t=1)
F	0	0	0	0
CAT	0	0	0	0
DIS	1000	506.0489	0	0
APPL	0	0	1000	1000
IBM	0	0	0	0
PFE	0	0	0	0
MCD	0	0	0	0
KO	0	0	0	0
PEP	0	379.5565	0	0
JNJ	0	0	0	0
XOM	0	0	0	0
MRO	0	0	0	0
WMT	0	0	0	0
C	0	0	0	0
WFC	0	0	0	0
ED	0	0	0	0
T	0	177.5681	0	0
VZ	0	0	0	0
JPM	0	0	0	0
NEM	0	0	0	0
surplus variable	82.212	52.937	-	-
shortfall variable	0	0	-	-
E(surplus)	-	-	115.11	84.943
E(shortfall)	-	-	0	0
EVPI	-	-	884.9	915.1

Table 3: Scenario 2

Ticker Name	SS (t=0)	SS(t=1)	DS(t=0)	DS(t=1)
F	0	0	0	0
CAT	0	0	0	0
DIS	332.5625	403.3338	0	0
APPL	0	0	1000	1000
IBM	0	0	0	0
PFE	0	0	0	0
MCD	0	0	0	0
KO	0	0	0	0
PEP	265.1164	276.947	0	0
JNJ	0	0	0	0
XOM	95.2447	32.7646	0	0
MRO	0	0	0	0
WMT	0	46.4722	0	0
C	0	0	0	0
WFC	0	0	0	0
ED	0	0	0	0
T	288.3638	187.244	0	0
VZ	0	0	0	0
JPM	0	0	0	0
NEM	18.7125	29.903	0	0
surplus variable	10.342	5.323	-	-
shortfall variable	0	0	-	-
E(surplus)	-	-	65.1078	34.943
E(shortfall)	-	-	0	0
EVPI	-	-	934.9	965.1

Table 4: Scenario 3

Ticker Name	SS (t=0)	SS(t=1)	DS(t=0)	DS(t=1)
F	0	0	0	0
CAT	0	0	0	0
DIS	639.9324	288.7949	0	0
APPL	0	23.4776	1000	1000
IBM	0	0	0	0
PFE	0	89.6384	0	0
MCD	0	0	0	0
KO	0	0	0	0
PEP	246.4183	246.7946	0	0
JNJ	0	0	0	0
XOM	0	158.1036	0	0
MRO	0	0	0	0
WMT	0	46.4722	0	0
C	0	0	0	0
WFC	0	0	0	0
ED	0	0	0	0
T	113.6493	247.965	0	0
VZ	0	0	0	0
JPM	0	0	0	0
NEM	0	0	0	0
surplus variable	54.685	43.734	-	-
shortfall variable	0	0	-	-
E(surplus)	-	-	15.1078	15.057
E(shortfall)	-	-	0	0
EVPI	-	-	984.9	1060.2

Table 5: Scenario 4

Ticker Name	SS (t=0)	SS(t=1)	DS(t=0)	DS(t=1)
F	0	0	0	0
CAT	0	0	0	0
DIS	639.9324	653.8367	0	0
APPL	0	0	1000	1000
IBM	0	0	0	0
PFE	0	0	0	0
MCD	0	0	0	0
KO	0	0	0	0
PEP	246.4183	223.1073	0	0
JNJ	0	0	0	0
XOM	0	0	0	0
MRO	0	0	0	0
WMT	0	46.4722	0	0
C	0	0	0	0
WFC	0	0	0	0
ED	0	0	0	0
T	113.6493	125.9403	0	0
VZ	0	0	0	0
JPM	0	0	0	0
NEM	0	0	0	0
surplus variable	36.987	76.876	-	-
shortfall variable	0	0	-	-
E(surplus)	-	-	34.8922	65.057
E(shortfall)	-	-	0	0
EVPI	-	-	1139.6	1260.2

Table 6: Scenario 5

Ticker Name	SS (t=0)	SS(t=1)	DS(t=0)	DS(t=1)
F	0	0	0	0
CAT	0	0	0	0
DIS	1000	992.3784	0	0
APPL	0	0	1000	1000
IBM	0	0	0	0
PFE	0	0	0	0
MCD	0	0	0	0
KO	0	0	0	0
PEP	0	0	0	0
JNJ	0	0	0	0
XOM	0	0	0	0
MRO	0	0	0	0
WMT	0	0	0	0
C	0	0	0	0
WFC	0	0	0	0
ED	0	0	0	0
T	0	0	0	0
VZ	0	0	0	0
JPM	0	0	0	0
NEM	0	0	0	0
surplus variable	90.432	103.432	-	-
shortfall variable	0	0	-	-
E(surplus)	-	-	134.8922	165.057
E(shortfall)	-	-	0	0
EVPI	-	-	1539.6	1660.2

B MATLAB Code

B.1 Main.m

```
clc
clear all
format long

% Program Start
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% 1. Read input files
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Load the stock weekly prices and factors weekly returns
adjClose = readtable('Project1_Data_adjClose.csv');
adjClose.Properties.RowNames = cellstr(datetime(adjClose.Date));
dates = datetime(adjClose.Date);
size_adjClose = size(adjClose);
adjClose = adjClose(:,2:size_adjClose(2));

factorRet = readtable('Project1_Data_FF_factors.csv'); %, 'ReadRowNames', true);
factorRet.Properties.RowNames = cellstr(datetime(factorRet.Date));

yearlyReturn = readtable('yearly_ret.xlsx'); %xlsread('Project1_Data_adjClose_yearly_ret','Project1_Data_adjClose_yearly_r');
yearlyReturn = yearlyReturn(:,2:end);
yearlyReturn.Properties.RowNames = cellstr(dates((size(dates,1)-size(yearlyReturn,1)+1):end, :));

monthlyReturn = readtable('monthly_ret.xlsx'); %xlsread('Project1_Data_adjClose_yearly_ret','Project1_Data_adjClose_yearly_r');
```

```
25 %monthlyReturn = monthlyReturn(:,2:end);
26 % monthlyReturn.Properties.RowNames = cellstr(dates(1:size(
       monthlyReturn,1), :));
27 avgMonthlyReturn = table2array(readtable('averaged_monthly_ret.
       xlsx'));
28
29 % Identify the tickers and the dates
30 tickers = yearlyReturn.Properties.VariableNames';
31
32 % Calculate the stocks' weekly EXCESS returns
33 prices  = table2array(adjClose);
34 returns = ( prices(2:end,:) - prices(1:end-1,:) ) ./ prices(1:end
       -1,:);
35 returns = array2table(returns);
36 returns.Properties.VariableNames = tickers;
37 returns.Properties.RowNames = cellstr(datetime(factorRet.
       Properties.RowNames));
38
39 %% 2. Define initial parameters
40
41
42 % Start of in-sample calibration period
43 calStart = datetime('2013-01-01');
44 calEnd   = calStart + calmonths(12*2) - days(1);
45
46 % Start of out-of-sample test period
47 testStart = datetime('2015-01-01');
48 testEnd   = testStart + calmonths(12) - days(8);
49
50 % Number of investment periods (each investment period is 6 months
       long)
51 NoPeriods = 6;
52
53 % Determine hyperparameters based on adjClose and monthlyReturn
54 [NoTotalDates, NoAssets] = size(adjClose);
55 [NoTotalMonths, NoAssets] = size(monthlyReturn);
56
57 % yearly training returns and prices from 2012 Jan to 2014 Dec
```

```
58 trainingReturns_yr = table2array( yearlyReturn( calStart <= dates
       & dates <= calEnd, :) );
59 lastYearTrainingReturns_yr = table2array( yearlyReturn( calEnd-
       calmonths(12)+days(1) <= dates & dates <= calEnd, :) );
60
61 % yearly testing returns from 2015 Jan to 2015 Dec
62 %testingReturns = table2array( yearlyReturn( testStart <= dates &
       dates <= testEnd, :) );
63
64 % weekly testing returns from 2015 Jan to 2015 Dec
65 testingReturns_weekly = table2array( returns( testStart <= dates &
        dates <= testEnd, :) );
66
67 % monthly training returns and prices from 2012 Jan to 2014 Dec
68 trainingReturns_mth = table2array( monthlyReturn( calStart <=
       dates & dates <= calEnd, :) );
69 lastYearTrainingReturns_mth = table2array( monthlyReturn( calEnd-
       calmonths(12)+days(1) <= dates & dates <= calEnd, :) );
70
71 % monthly testing returns from 2015 Jan to 2015 Dec
72 testingReturns = table2array( monthlyReturn( calStart <= dates &
       dates <= calEnd, :) );
73
74 % Data Mean Year
75 mu_entire_training_yr = geomean(trainingReturns_yr, 1) - 1;
76 mu_last_year_yr = geomean(lastYearTrainingReturns_yr, 1) - 1;
77
78 % Data Mean Month
79 mu_entire_training_mth = geomean(1+trainingReturns_mth, 1) - 1;
80 mu_last_year_mth = geomean(1+lastYearTrainingReturns_mth, 1) - 1;
81
82 % Data Variance Year
83 cov_entire_training_yr = cov(trainingReturns_yr-1);
84
85 % Data Variance Month
86 cov_entire_training_mth = cov(trainingReturns_mth);
87
88 % No of splits and Generation of xi
```

```
89 no_splits = 50;
90 generated_ret = mvnrnd(mu_entire_training_mth,
       cov_entire_training_mth, no_splits); % 20 assets = 20 rows;
       scenarios in columns
91 exp_generated_ret = geomean(1+generated_ret,1)-1; % for VSS first
       term
92
93 % Initial budget
94 b = 1000;
95 % Target Values
96 G = [950, 1000, 1050, 1100, 1200];
97
98 %% 3. Stochastic Program for varying parmeter
99 clear weights weights_mean weights_exp_scenarios
       fval_actual_scenarios fval_avg_scenarios
100
101 % Run solver for different G values
102 for i = 1:size(G, 2)
103     % Call solver function to optimize weights for Stochastic
          Program
104     [x,fval] = Solver_mat(1+generated_ret, b, G(1,i));
105     fval_actual_scenarios(i,:) = -1*(fval);
106
107     % Extract weights corresponding to each rollout of scenarios
108     for j = 1:no_splits
109         weights{i,j} = [x((1:NoAssets), 1) x((NoAssets*j+1):(
              NoAssets*(j+1)), 1)];
110     end
111
112     % Average of weights from stochastic problem
113     weights_mat = cat(3, weights{i,:});
114     weights_mean{i} = mean(weights_mat, 3);
115
116     % Determine the weights for the true sample mean mu
117     [x_expected_scenarios, fval_expected_scenarios] = Solver_mat
          (1+exp_generated_ret, b, G(1,i)); %VSS first term
118     weights_avg_scenarios{i} = [x_expected_scenarios((1:NoAssets),
           1) x_expected_scenarios(((NoAssets+1):NoAssets*2), 1)];
```

```
119     fval_avg_scenarios(i,:) = (fval_expected_scenarios)*-1;
120 end
121
122 %% 4. Deterministic Program for t = 0 to t = 1
123
124 clear x_optimal_det det_value
125
126 % First we find the deterministic problem using expected returns
      from
127 % historical data
128
129 % Objective function with surplus reward of 1 and slack penalty of
       4
130 f = [ones(1,NoAssets) -1 4];
131
132 % Budget Constraints
133 A = [ones(1,NoAssets) 0 0];
134 b = 1000;
135
136 lb = zeros(1, NoAssets+2);
137
138 for i = 1:size(G,2)
139     % Determine optimal x values for varying G
140     beq = G(1,i);
141     % Determine optimal x for t = 0
142     Aeq1 = [(avgMonthlyReturn((end-1), :)+1) -1 1];
143     [x_optimal_det{i,1}, det_value{i,1}] = linprog(f, A, b, Aeq1,
          beq, lb, []);
144     % Determine optimal x for t = 1
145     Aeq2 = [(avgMonthlyReturn(end, :)+1) -1 1];
146     [x_optimal_det{i,2}, det_value{i,2}] = linprog(f, A, b, Aeq2,
          beq, lb, []);
147 end
148
149 % Uncomment for saving data to .mat file
150 % save('very_important_table_det.mat', 'x_optimal_det', 'det_value
      ')
151 %% 5. Value of Stochastic Solution
```

```
152
153 % VSS calculation
154 vss = fval_actual_scenarios-fval_avg_scenarios;
155 save('vss.m', 'fval_actual_scenarios','fval_avg_scenarios','vss')
156
157 %% 6. Portfolio Weights and Calculation of Portfolio Return for
      Out-of-Sample Periods
158 clear portfRet
159
160 % Plotting the Pi Plots for each time period with varying G values
161 for i = 1:size(G,2)
162     fig1 = figure();
163
164     X = weights_mean{i}(:,1)';
165     labels = tickers;
166     ax1 = subplot(1,2,1);
167     p{(i-1)*2+1} = pie(ax1,X);
168     title(ax1,strcat('G = ', num2str(G(1,i)),'at t = 0'),  '
          FontSize', 10);
169     legend(labels, 'Location', 'bestoutside', 'Orientation', '
          vertical');
170
171     Y = weights_mean{i}(:,2)';
172     ax2 = subplot(1,2,2);
173     p{(i-1)*2+2} = pie(ax2,Y);
174     title(ax2,strcat('G = ', num2str(G(1,i)),'at t = 1'), '
          FontSize', 10);
175     legend(labels, 'Location', 'bestoutside', 'Orientation', '
          vertical');
176
177     % Uncomment for printing graph to file
178     % print(fig1,strcat('pi_', num2str(i)),'-dpng','-r0');
179 end
180 hold off
181
182
183 %% 7. Visualization of Portfolio Return for each Scenarios for Out
      -of-Sample Months
```

```
clear legend_text fig1

% Plotting of all scenario returns against time horizon
fig2 = figure(2);

mon_ret_for_testing = (1+avgMonthlyReturn((end-1):end,:));
for i = 1:size(G,2)
    for s = 1:no_splits
        if mod(s,2)==1
            % Generate portfolio return for each period
            t1_ret = mon_ret_for_testing(1,:)*weights{i,s}(:,1)/
                sum(weights{i,s}(:,1));
            t2_ret = mon_ret_for_testing(2,:)*weights{i,s}(:,2)/
                sum(weights{i,s}(:,2))*t1_ret;
            portfRet{i,s} = [t1_ret;t2_ret];
            plot([0,1,2],[ones(1,1); portfRet{i,s}]'*1000)
            hold on
        end
    end
end

title('Out-of-Sample: Portfolio Values for all Scenarios Rollouts'
    , 'FontSize', 10);
ylabel('Portfolio Value','interpreter','latex','FontSize',12);
xlabel('Period')

% Uncomment for printing graph to file
% print(fig2,'portfolio_value_all_scenarios','-dpng','-r0');
%% 8. Visualization of Portfolio Return for Averaged Scenarios for
    Out-of-Sample Months

% Plotting of only averaged scenario returns against time horizon
fig3 = figure()
portfRet_mat = cat(3, portfRet{:});
expected_portfRet_across_scenarios = mean(portfRet_mat, 3);
weights_mean_mat = cat(3, weights_mean{:});
for i =1:size(G,2)
    % Generate portfolio return for each period
```

```
218     t_ret = (1+avgMonthlyReturn((end-1):end,:))
219     portfRet_testing(i,1) = (1+avgMonthlyReturn((end-1),:))*(
            weights_mean{i}(:,1)/sum(weights_mean{i}(:,1)));
220     portfRet_testing(i,2) = portfRet_testing(i,1)*(1+
            avgMonthlyReturn((end),:))*(weights_mean{i}(:,2)/sum(
            weights_mean{i}(:,2)));
221     portfRet_testing_det(i,1) = t_ret(1,:)*(x_optimal_det{i,1}(1:
            NoAssets)/sum(x_optimal_det{i}(1:NoAssets)));
222     portfRet_testing_det(i,2) = portfRet_testing_det(i,1)*t_ret
            (2,:)*(x_optimal_det{i,2}(1:NoAssets)/sum(x_optimal_det{i
            }(1:NoAssets)));
223
224 end
225
226 plot([0,1,2],[ones(size(G,2),1) portfRet_testing]'*1000);
227 hold on
228 plot([0,1,2],[ones(size(G,2),1) portfRet_testing_det]'*1000, '--')
        ;
229
230 ylabel('Portfolio Value','interpreter','latex','FontSize',12);
231 xlabel('Month','interpreter','latex')
232
233 % Uncomment for printing graph to file
234 % print(fig3,'portfolio_value_avg_scenarios','-dpng','-r0');
```

B.2 Solver.m

```
1  function [x_optimal, fval] = Solver_mat(xi, initial_funding, G)
2
3      [no_splits, no_assets] = size(xi);
4
5          ones_1asset = ones(1, no_assets);
6      one_set_of_yw = kron(eye(no_splits), [-1 1]);
7
8      f = -1*[ones_1asset, zeros(1, no_assets*no_splits), repmat([1
           -100], 1, no_splits^2)/(no_splits^2)];
9      total_no_var = size(f, 2);
10
11         A = [ones_1asset, zeros(1,total_no_var - no_assets)];
```

```
12          b = [initial_funding];
13
14      Aeq = [xi, -1*kron(eye(no_splits), ones_1asset), zeros(
            no_splits, 2*no_splits^2);
15           zeros(no_splits^2, no_assets), kron(eye(no_splits), xi),
                kron(eye(no_splits), one_set_of_yw)];
16      beq = [zeros(no_splits, 1); G*ones(no_splits^2, 1)];
17
18      [x_optimal, fval] = linprog(f, A, b, Aeq, beq, zeros(1,
            total_no_var), []);
19
20  end
```