Problem Chosen	**2022**	**Team Control Number**
C	**MCM/ICM**	2226255
	Summary Sheet	

In the chosen problem, we are given an initial balance of $1000 in cash and asked to develop an optimal trading strategy with gold, cash and bitcoin. For our solution, we applied Henry Markowitz's Mean-Variance optimization model, which is the foundation method of portfolio theory. The principle behind Mean-Variance Optimization is to maximize the expected return of a portfolio while minimizing the risk. This ratio between expected return and risk of each asset can be quantified by the *Sharpe Ratio* calculated for that asset. In preparing the dataset for this algorithm, we decided to exclude weekend trading since while bitcoin is traded every day, gold is only traded during market hours. Thus we aligned the historical data provided by the problem in order to run all our methods on purely true market data and avoiding estimates.

We propose a relatively conservative approach to trading gold and bitcoin that guarantees favorable returns by minimizing the risk of investment. We implement our approach by calculating the maximum Sharpe ratio for a given portfolio and finding the optimal distribution, or weights of each asset. The optimal distribution we find through this process describes a portfolio on the efficient frontier. An efficient portfolio minimizes the risk i.e. volatility while maximizing the potential returns given the risk minimization constraints. While the optimal asset distribution changes daily with the data updates, our model found it inefficient to trade daily due to transaction fees involved in re-balancing the portfolio. This suggested that it is more efficient to trade at a slower frequency.

A portfolio attaining the max return possible with a given variance is deemed an *efficient portfolio* with all others under considered *inefficient*. The graph of all efficient portfolios can be used to construct an *efficient frontier graph*. At the specified re-balancing interval, the algorithm takes in as an input the gold and bitcoin time-series data up to that day and through the Sharp Ratio maximization, returns an optimal weight vector w_o. w_o and current weighting w_c, as well as the current holdings in cash are given as an input to our designed transaction handler in which the re-balancing is performed. Assets with a negative change between the optimal and previous weights are *sold* with the value being transferred to the cash, and those requiring over-weighting are *bought* with the value removed from the cash. The calculation of fees is incorporated with the assumption that fees slightly reduce the value of asset transaction. In order for the portfolio weighting to remain optimal in spite of fees, the new holdings with fees taken into account is again recomputed to match the optimal weighting. Due to transaction fees, achieving the optimal weights exactly as outputted becomes impossible. Moreover, frequent trading incurred steep transaction fees that offset the returns, while infrequent trading failed to optimize the portfolio weights and missed potential returns. To find the optimal trading frequency given the transaction costs, we iterated through trading frequencies ranging from 1 to 300 days, calculating the final portfolio value after the trading period for each frequency. We took the maximum return achieved in that data to be the most optimal trading frequency given the transaction costs.
The rebalancing interval analysis described had a maximum value at 164 days, representing the highest return achieved across all interval lengths. This frequency appears to minimize the cost of transaction fees while still producing near-optimal portfolio re-balancing based on the historical data.

To analyze results, the algorithmic risk and return was compared against gold and bitcoin single-asset benchmarks. The final algorithm portfolio value was $28,746, a return of 2774.6%. While this return is less than the bitcoin single-asset portfolio return, the significantly smaller volatility ensures the risk is minimized in a live-trading setting with no foresight.

Contents

1 Introduction

Algorithmic trading is the utilization of an algorithm following a specified mathematical model to trade with higher frequency, accuracy, and efficiency than is possible traditionally. A well designed trading algorithm takes into account the provided data in real time and makes an informed trade that aligns with the optimization specified in the mathematical model. Different trading algorithms may be designed with different intended optimizations in mind, but all share the common goal of maximizing profit and minimizing risk.

A significant benefit to algorithmic trading as compared to traditional trading methods is that an algorithm can be *backtested* on historical market data. *Backtesting* refers to the practice of running a trading algorithm through the historical market data time-series as if it were trading on that data in real-time. The results of a backtest indicate whether the model/algorithm would have been viable if it had been implemented across the historical time period. Such results are then used to assess the usability and viability of the algorithm in the present time frame. The results of a backtest are also vital in adjusting the algorithm's parameters and overall strategy. Performing well or poorly on a backtest is not necessarily the only ruler by which the viability of an algorithm can be assessed—as even the most inefficient algorithm may appear to succeed in a general bull market—but performing well across a long period of time in a variety of market conditions demonstrates resilience that bodes well for live implementation.

In designing a viable trading algorithm, the abstractions and assumptions of the chosen mathematical model must also be translated to the discrete nature of real-world trading. The most apparent example of this comes in the form of transaction fees. In the real-world, brokerages impose proportional fees for the transfer of financial assets, which can easily overtake profits if the trade sizes/trade frequency is not optimized properly. While an ideal trading algorithm would maintain the most optimal portfolio at all times, the restrictions placed by transaction costs require a more cautious approach where only certain portfolio rebalances are deemed worth the cost.

In this paper, we analyze a given asset price time-series and propose an efficient algorithm to maximize expected returns while also minimizing risk. This algorithm is backtested over the given datasets to determine optimal parameters. The results of this algorithm are then benchmarked against single asset and equal-weight portfolios to determine usefulness.

1.1 Problem Summary

In the chosen problem, we are given an initial balance of \$1000 in cash and asked to develop an optimal trading strategy involving gold and bitcoin. The prices per troy ounce of gold and per bitcoin are given in a dataset spanning from 9/11/2016 to 9/10/2021. While there is no cost to holding an asset, transaction costs of $\alpha_{gold} = 1\%, \alpha_{bitcoin} = 2\%$ are imposed on the system.

We are tasked with developing a trading algorithm that maximizes the returns across the given time-period while minimizing the risk. Our model is also required to be sensitive to transaction fees. After developing a viable model, we must also compare the model against alternatives to demonstrate viability. In this paper, we choose Mean-Variance Optimization approach to the problem.

1.2 Our Model

Our chosen model is Henry Markowitz's Mean-Variance Optimization, the foundational method of portfolio theory. We prepare the data for this application by first aligning the time-series of the bitcoin and gold datasets, and then by designing a function *transact.m* that models transactions over the data. The aligned dataset and the transaction function are used in the implemenation of Mean-Variance Optimization with an inclusion of transaction fees. To optimize the trading frequency, the algorithm is run over the dataset with 300 different rebalancing intervals. We take the maximum of this data to be the optimal rebalancing interval/trading frequency. The algorithm running with the optimal trading frequency is then benchmarked against single asset portfolios of gold and bitcoin to compare returns and risk.

2 Data

Our model takes as an input the historical time series data for both gold and bitcoin from 9/11/2016 to 9/10/2021. The initial complication faced with the given datasets was the misalignment of indices. Bitcoin is traded every day of the week, while gold is only traded during market hours i.e. only on weekdays. To maximize speed as well as vastly minimize the complexity of the algorithm, we aligned the time-series by removing the weekday entries in the bitcoin dataset, resulting in an algorithm that only trades on weekdays. The gold dataset also included 10 empty data points which complicate the analysis, thus these values were also removed from both datasets.

We first built a simple algorithm to check whether the indexes were aligned. The algorithm iterates over the gold dataset and copies the date-corresponding values of the original bitcoin dataset to a new one, while also skipping over missing values in the gold dataset. By completely avoiding estimates for the missing/misaligned values, we were able to run all of our methods on purely the true market data given to us.

Due to the unpredictability of the market and the limited information available during the time of investment about the future, the naive human approach of qualitatively assessing upswings and downswings is not viable (nor efficient enough to warrant such an algorithms existence). Thus, we create a model that maintains an optimal portfolio based on the expected returns and risk of the assets at a given time.

3 Max Sharpe-Ratio Model and Trading Algorithm

The principle behind Mean-Variance Optimization is to maximize the expected return of a portfolio while minimizing the risk. This ratio between expected return and risk of each asset can be quantified by the *Sharpe Ratio* calculated for that asset. For a single asset *i*:

$$\text{Sharpe Ratio}_i = \frac{E(r_i - r_f)}{\sigma_i}$$

where $E(r_i - r_f)$ is the expected risky asset returns minus the expected risk free returns
Since the risk-free asset is cash and has a return of zero, we can simplify the formula to:

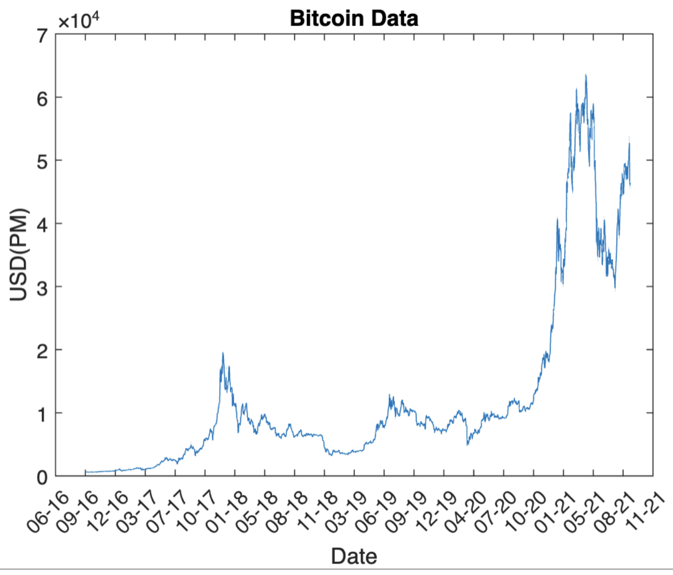

Figure 1: Bitcoin data provided by the problem, plotted against time

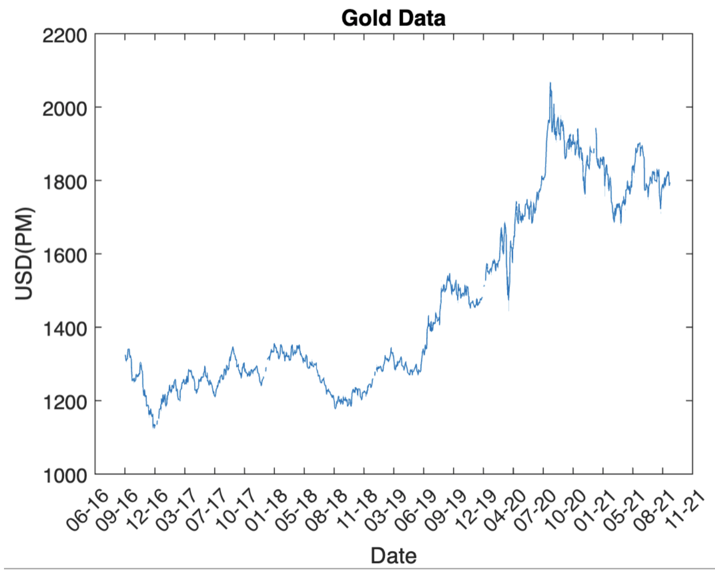

Figure 2: Gold data provided by the problem, plotted against time

$$\text{Sharpe Ratio}_i = \frac{E(r_i)}{\sigma_i}$$

where:

- $\sigma_i = \sqrt{\frac{\sum_{j=1}^{M}(r_j - \bar{r})^2}{M-1}}$ = Variance of stock i given M samples

- $\bar{r}$ = average return of stock i

Given a certain portfolio return r, a number of portfolio weights w_i may be constructed with different volatilities (i.e. variances of the returns). A portfolio attaining the max return possible with a given variance is deemed an *efficient portfolio* with all others under considered *inefficient*. The graph of all efficient portfolios can be used to construct an *efficient frontier graph*:

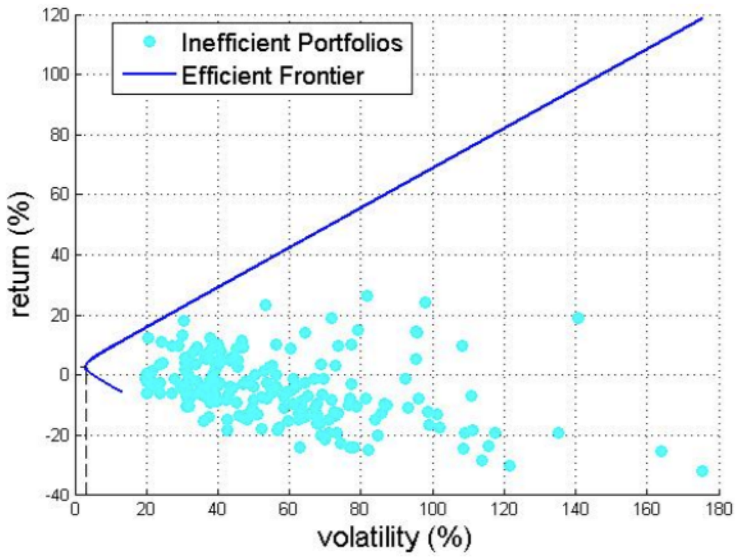

Figure 3: Efficient Frontier example by Martin Haugh (Columbia University)

In our approach to maximizing Sharpe-Ratio, we consider cash to be an asset in and of itself. The efficient frontier of a zero-return risk-free asset thus becomes a straight line. The frontier calculated by our own algorithm at roughly the halfway point demonstrates this.

The optimization of portfolio weights w_i is done by the *estimateMaxSharpeRatio()* function in MATLAB's financial toolbox, which solves the following optimization problem:

Maximize $\frac{\mu^T w}{\sqrt{w^T C w}}$
s.t.

- $\sum_j w_j = 1$
- $0 \leq w$

where:

- μ = vector of mean returns for each stock
- C = Covariance matrix= $\begin{bmatrix} \sigma_{r_1}^2 & cov(r_1, r_2) & cov(r_1, r_3) \\ cov(r_2, r_1) & \sigma_{r_2}^2 & cov(r_2, r_3) \\ cov(r_3, r_1) & cov(r_3, r_2) & \sigma_{r_3}^2 \end{bmatrix}$

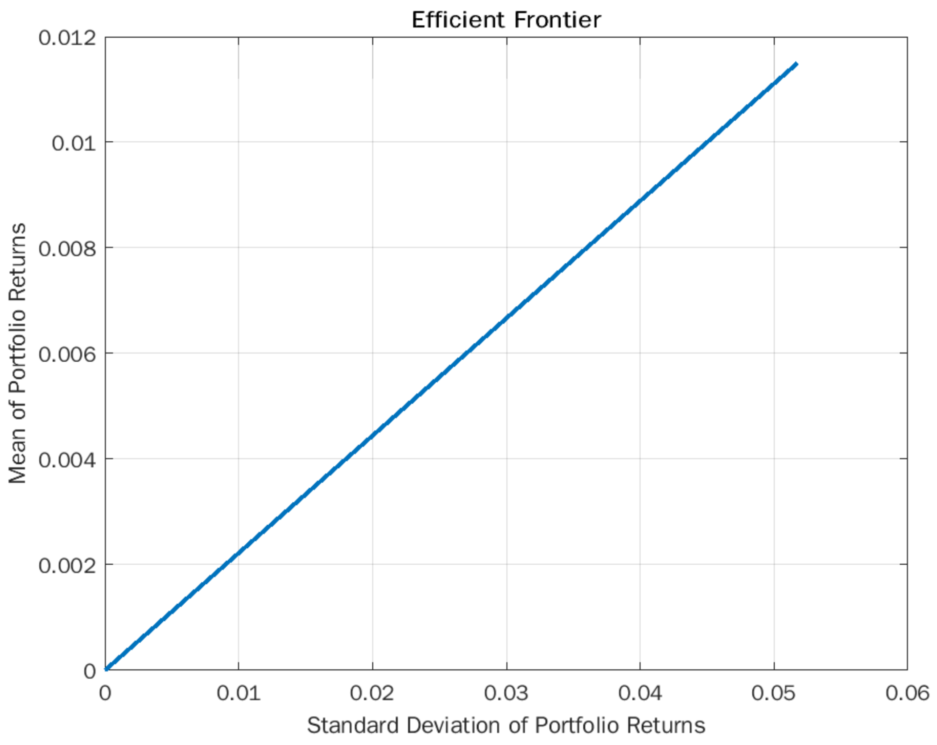

Figure 4: The efficient portfolio frontier at rebalancing interval 4

3.1 Portfolio Rebalancing

At the specified rebalancing interval, the algorithm takes in as an input the gold and bitcoin time-series data up to that day (to simulate live trading) and outputs an optimal weight vector w_i. The w_i and current weighting w_o, as well as the current holdings in cash (as opposed to the native units) are given as an input to the *transact.m* function, in which the rebalancing is performed in a way most analogous to real life. Assets requiring underweighting are *sold* with the value being transferred to the cash, and those requiring overweighting are *bought* with the value removed from the cash. The calculation of fees is incorporated with the assumption that fees slightly reduce the value of asset transaction. In order for the portfolio weighting to remain optimal in spite of fees, the new holdings with fees taken into account is again recomputed to match the optimal weighting. Finally, the holdings are reweighted and passed back to the main function. To prevent unnecessary errors, a threshold of 0.001 is set as the minimum value possible in a holding.

3.2 Transaction Fee Sensitivity

We implemented this maximization function at the specified intervals during the trading period. At each application, previous returns from each asset were computed to calculate the max Sharpe Ratio of the portfolio and give the optimal weight vector w_o Using this weight vector, the portfolio was re-balanced to as closely as possible match the optimal weights. Due to transaction fees, achieving the optimal weights exactly as outputted becomes impossible. Moreover, frequent trading incurred steep transaction fees that offset the returns, while infrequent trading failed to optimize the portfolio weights and missed potential returns. To find the optimal trading frequency given the transaction costs, we iterated through trading frequencies ranging from 1 to 300 days, calculating the final portfolio value after the trading period for each frequency. We took the mode of that data to be the most optimal trading frequency given the transaction costs.

4 Results

The rebalancing interval analysis described in the previous section had a maximum value at 164 days, representing the highest return achieved across all interval lengths. This frequency appears to minimize the cost of transaction fees while still producing near-optimal portfolio rebalancing based on the historical data.

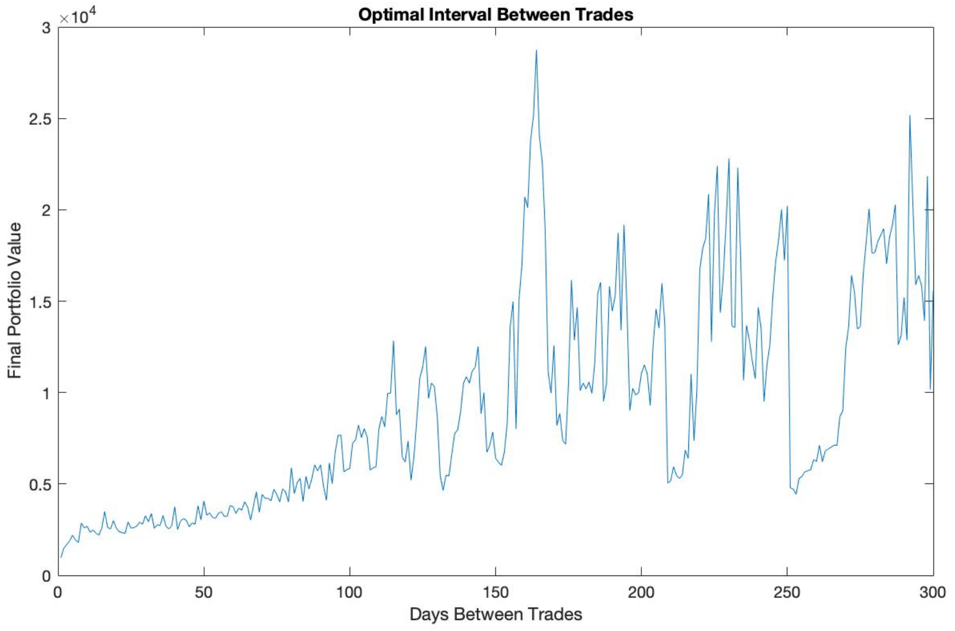

Figure 5: Final Portfolio Value vs. Interval Between Portfolio re-balancing. Portfolio was re-balanced so as to maximize the Sharpe ratio. Maximum at 164 days indicates optimal trading frequency

The optimal trading interval of 164 days returned a final portfolio value of \$28,746, a 2774.6% return from the initial \$1000 dollar investment.

4.1 Benchmark results

The algorithmic returns were benchmarked against two single-asset portfolios i.e. ones containing either \$1000 worth of gold or bitcoin with no rebalancing. While the returns for merely buying and holding bitcoin throughout the time period are indeed greater, such a strategy only seems wise in hindsight. When live-trading, such massive future upswings are often impossible to anticipate and thus minimizing risk is the more wise investment strategy.

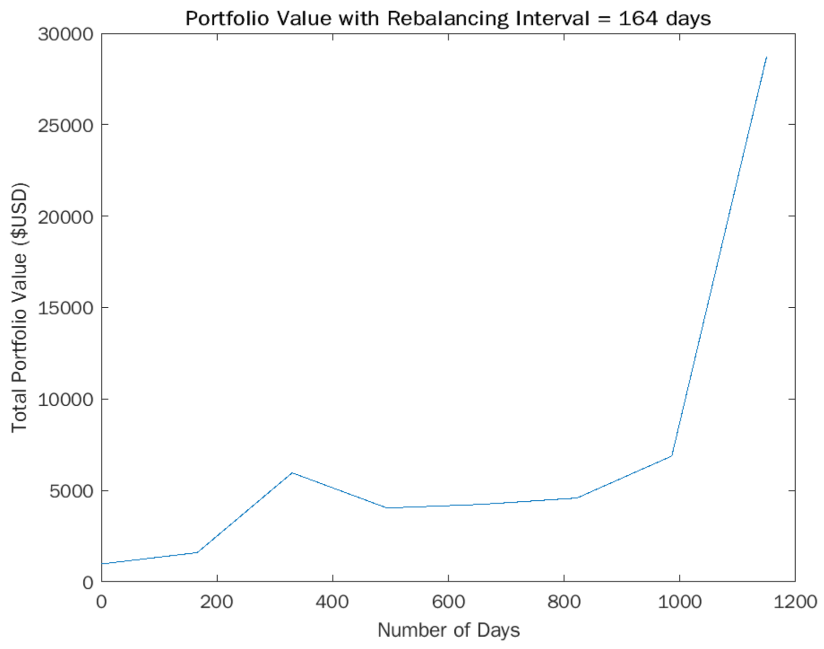

Figure 6: Portfolio growth over 5 year period with re-balancing algorithm applied every 164 days

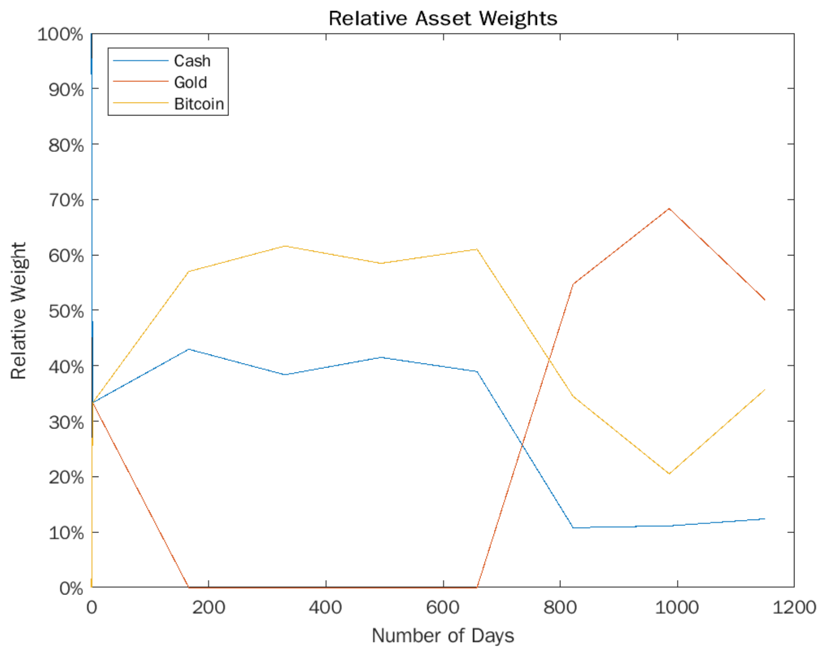

Figure 7: Relative asset weights over the time period with interval = 164 days

The benchmark tests demonstrate the relatively high returns of our algorithmic strategy while maintaining significantly less risk than the bitcoin single-asset portfolio.

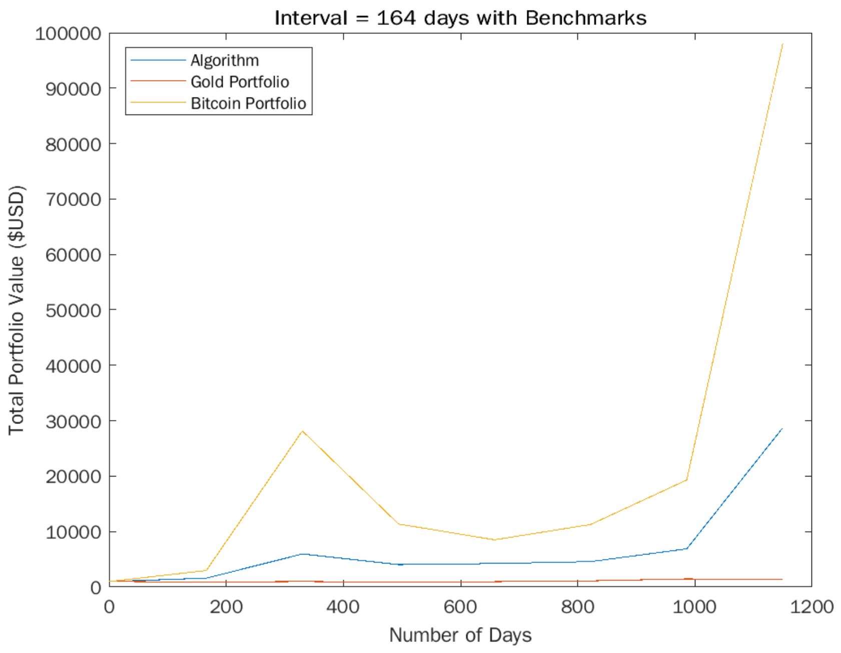

Figure 8: Total portfolio value alongside single-asset portfolios

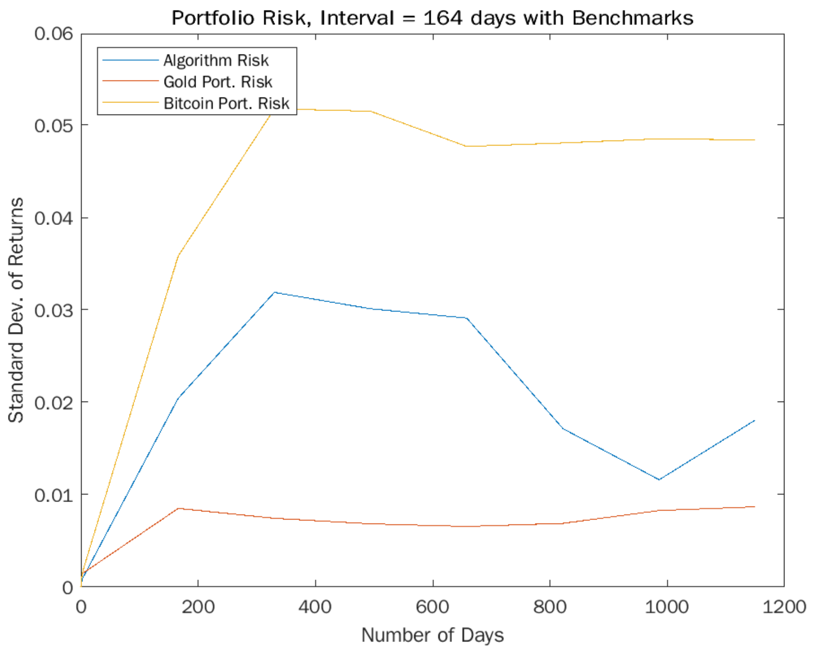

Figure 9: Risk calculated as standard deviation of returns

5 Memorandum

In trading risky assets, a number of different strategies are available to be used. Here we propose a relatively conservative approach to trading gold and bitcoin that guarantees favorable returns by minimizing the risk of investment.

The Sharpe Ratio is a widely used means of measuring investment risk relative to expected returns to generate a score evaluating the relative strength of a stock or portfolio. By calculating the maximum Sharpe ratio for a given portfolio we can find the optimal distribution of assets, or *weights* of each asset. This optimal distribution describes a portfolio on the *efficient frontier*, insofar as the portfolio minimizes the likelihood of losing value while maximizing the potential returns given the risk minimization constraints. Although the optimal asset distribution changes daily, as new price data updates the measurements of expected returns and volatility, our model found it very inefficient to trade daily. Due to the transaction fees involved in re-balancing the portfolio to match the optimal asset distribution, it is more efficient to trade at a slower frequency. We tested every trading frequency from 1 to 300 days and found the maximum returns at a trading frequency of 164 days. This frequency appears to minimize transaction costs while still re-balancing frequently enough to capture important data about the risky assets and re-balance accordingly.

Using only price data up to the given day, our model was able to grow an initial investment of \$1,000 in September 2016 to \$28,746 in September of 2021 by trading every 164 days.

When benchmarking against single-asset portfolios of gold and bitcoin, we find that our algorithmic approach achieves remarkable high returns while only incurring fairly moderate risk. While the risk involved in merely buying and holding gold is negligible over the given time period, the returns are miniscule in comparison to our algorithm. On the other hand, buying and holding bitcoin would have resulted in massive returns over our model, but is a strategy that only seems wise in hindsight. Without foresight, the Mean-Variance approach of maximizing potential returns while minimizing risk seems the most prudent, and our algorithm demonstrates significantly less risk than the bitcoin single-asset portfolio, ensuring its viability in live trading.

6 Appendix

```
table = readtable('BCHAIN-MKPRU.csv');
bValue = table2array(table(:,2));
b_date = datenum(table2array(table(:,1)));
mat = [b_date bValue];
index = isnan(mat(:,2));
mat(index,:) = []; %remove nan values

g_table = readtable('LBMA-GOLD.csv');
g_value = table2array(g_table(:,2));

g_dates = datenum(table2array(g_table(:,1)));

gMat = [g_dates g_value];
idx = isnan(gMat(:,2));
gMatOld = gMat;
gMat(idx,:) = []; % remove nan values from data

s = size(b_date); s = s(1);
j = 1; k = 1;
sG = size(gMat); aux = zeros(sG(1), sG(2));
sg2 = size(gMatOld);
aux2 = zeros(sg2(1), sg2(2));

%Match sizes of Bitcoin and Gold timeseries
%by simply removing weekend data from Bitcoin
for i = 1:s
    if mat(i,1) == gMat(j,1)
        aux(j,:) = mat(i,:); %copy entire row into aux array
        j = j+1;
    else
      %increment i
    end
    if mat(i,1) == gMatOld(k,1)
        aux2(k,:) = mat(i,:);
        k= k+1;
    else
    end
end

symbol=["Cash";"Gold";"Bitcoin"];
x = size(aux,1);
cash = ones(x,1);
holdings = [1000, 0, 0];
w = [1,0,0];
interval = 1;

newHoldings = zeros(ceil(sG(1)/interval), 3);
newHoldings(1,:) = holdings;

weights = zeros(ceil(sG(1)/interval), 3);
```

```
weights(1,:) = w;

B_usdP = aux(1,2);
G_usdP = gMat(1,2);

feeMinimizer = zeros(300,1);
feeTot = zeros(sG(1), 1);

%pure gold benchmark
pureG = zeros(x, 1);
Gamount = 1000 / gMat(1,2);
pureG(1,1) = 1000;
gweight = [0, 1, 0];

%pure bitcoin benchmark
pureB = zeros(x,1);
Bamount = 1000 / aux(1,2);
pureB(1,1) = 1000;
bweight = [0, 0, 1];

%for interval = 1:300
    weights = zeros(ceil(sG(1)/interval), 3);
    weights(1,:) = w;
    holdings = [1000, 0, 0];
    newHoldings = zeros(ceil(sG(1)/interval), 3);
    newHoldings(1,:) = holdings;
    w = [1,0,0];
    j=2;

    %keep track of days
    days = zeros(ceil(sG(1)/interval)+1, 1);
    inter = 2;
    days(1,1) = 0;

    %keep track of benchmarks
    gbench = zeros(ceil(sG(1)/interval)+1, 1);
    bbench = zeros(ceil(sG(1)/interval)+1, 1);
    gbench(1,1) = 1000;
    bbench(1,1) = 1000;

    %keep track of risk
    algoRiskMat = zeros(ceil(sG(1)/interval)+1, 1);
    gRiskMat = zeros(ceil(sG(1)/interval)+1, 1);
    bRiskMat = zeros(ceil(sG(1)/interval)+1, 1);

    for i = 2:interval:sG(1)
        days(inter, 1) = i;

        %every day
        returnB= tick2ret(aux(1:i,2));
        returnG= tick2ret(gMat(1:i,2));
        returnC= tick2ret(cash(1:i,1));
        returnTot =[returnC returnG returnB];

        B_usd = aux(i,2); %current B_usd value
        G_usd = gMat(i,2); %current gold usd value

        %enter benchmark vals
        pureG(i,1) = Gamount * G_usd;
        pureB(i,1) = Bamount * B_usd;

        %track benchmark vals
        gbench(inter, 1) = pureG(i,1);
        bbench(inter, 1) = pureB(i,1);

        %track days
        days(inter, 1) = i;

        scalar(j,:) = [1, B_usd/B_usdP, G_usd/G_usdP]; %change in value ...
           since last transaction
        holdings = scalar(j,:) .* holdings;

        %...Portfolio workflow
        p = Portfolio('AssetList',symbol,'RiskFreeRate',0.00/252);
        [lb, ub, isbounded] = estimateBounds(p);
        p= estimateAssetMoments(p, returnTot);
        p=setDefaultConstraints(p);

        %Find optimal weight
        w1=estimateMaxSharpeRatio(p);
        w1 = w1';

        %Track weights
        weights(j,:) = w1;

        %Calculate moments for algo and benchmarks
        [algoRisk, algoRet] = estimatePortMoments(p,w1');
        [gRisk, gRet] = estimatePortMoments(p,gweight');
        [bRisk, bRet] = estimatePortMoments(p,bweight');

        %Track Risks
        algoRiskMat(inter,1) = algoRisk;
        gRiskMat(inter,1) = gRisk;
        bRiskMat(inter,1) = bRisk;

        %Plot efficient frontier at roughly halfway point
        if inter == ceil(sG(1)/(2*interval))
            disp('Plotting frontier at interval = ' + num2str(inter))
            plotFrontier(p)
        end

        %Complete transaction
        [holdings, w1, fee] = transact(w, w1, holdings);
        feeTot(interval) = feeTot(interval) + fee;
        newHoldings(j,:) = holdings;

        %Prepare for next loop
        w = w1; %update weights
        j=j+1;
        inter = inter + 1;
        G_usdP = G_usd;
        B_usdP = B_usd;
    end

%Construct total returns matrix
f = newHoldings(:,1) + newHoldings(:,2) + newHoldings(:,3);

%...plot total returns x days
%plot(days, f)
%ax = gca;
%ax.YAxis.Exponent = 0;
%xlabel('Number of Days')
%ylabel('Total Portfolio Value ($USD)')
%title(['Portfolio Value with Rebalancing ' ...
%    'Interval = 164 days'])

%...plot total returns with benchmark
%g = [f gbench bbench];
%plot(days, g)
%ax = gca;
%ax.YAxis.Exponent = 0;
%xlabel('Number of Days')
%ylabel('Total Portfolio Value ($USD)')
%title(['Interval = 164 days with Benchmarks'])
%legend({'Algorithm', 'Gold Portfolio', 'Bitcoin Portfolio'}, ...
%    'Location', 'northwest')

%...plot risk with benchmarks
%r = [algoRiskMat gRiskMat bRiskMat];
%plot(days, r)
%ax = gca;
%ax.YAxis.Exponent = 0;
%xlabel('Number of Days')
%ylabel('Standard Dev. of Returns')
%title(['Portfolio Risk, Interval = 164 days with Benchmarks'])
%legend({'Algorithm Risk', 'Gold Port. Risk', 'Bitcoin Port. Risk'}, ...
%    'Location', 'northwest')

%...plot relative weights
%perWeights = weights * 100
%plot(days, perWeights)
%ax = gca;
%ax.YAxis.TickLabelFormat = '%g%%';
%legend({'Cash', 'Gold', 'Bitcoin'}, 'Location', 'northwest')
%title('Relative Asset Weights', 'Fontweight', 'bold')
%xlabel('Number of Days')
%ylabel('Relative Weight')


feeMinimizer(interval) = f(end); %end return value

end
```

```
function [newHoldings, newWeights, fees] = transact(currentWeights, ...
    optimalWeights, currentHoldings)

    %take in two 3 element weight vectors and a vector with asset values,
    %return new Portfolio
    %currentHoldings measures assets in USD
    alphaG = 0.01;
    alphaB = 0.02;
    change = optimalWeights - currentWeights;
    newWeights = currentWeights;
    theoreticalCash = sum(currentHoldings);
    changeAssets = (change * theoreticalCash); % change in cash value for ...
        every asset

    fees = [0, alphaG, alphaB] .*abs(changeAssets);
    %fees = [fees(1,3) + fees(1,2), 0, 0];
    newHoldings = currentHoldings;

    for i = 2:3
        if change(i) < 0 %if stock is to be sold, move stock to cash and ...
            take out the fee
            %soldAmt = newHoldings(i) - changeAssets(i);
            newHoldings(i) = newHoldings(i) - abs(changeAssets(i)); ...
                %subtract change in USD from stock
            newHoldings(1) = newHoldings(1) + abs(changeAssets(i)) - ...
                fees(i); %add the cash but minus the fees!
                    end
    end
    %recompute!!!
    newWeights = newHoldings ./ sum(newHoldings);
    change = optimalWeights - newWeights;
    theoreticalCash = sum(newHoldings);
    changeAssets = (change * theoreticalCash); % change in cash value for ...
        every asset
    fees = [0, alphaG, alphaB] .*abs(changeAssets);
    for i = 2:3
        if change(i) > 0 %if stock is to be BOUGHT
            %buyAmt = changeAssets(i); %amount to remove from cash = change!
            newHoldings(1) = newHoldings(1) - abs(changeAssets(i));  ...
                %subtract from cash
            newHoldings(i) = newHoldings(i) + changeAssets(i) - fees(i); ...
                %only add 99% of stock purchased!
        end
    end

    newWeights = newHoldings ./ sum(newHoldings);

    for i = 1:3
        if newHoldings(i) < 0.001
            newHoldings(i) = 0;
        end
    end

    fees = sum(fees);
```

References

[1] Haugh, M. (n.d.). Mean-variance optimization and the CAPM - Columbia University. Retrieved February 21, 2022, from http://www.columbia.edu/ mh2078/FoundationsFE/MeanVariance-CAPM.pdf

[2] Brunnermeier, M. K. (n.d.). Lecture 05: Mean-variance analysis - Princeton University. Eco 525: Financial Economics I. Retrieved February 21, 2022, from https://www.princeton.edu/ markus/teaching/Eco525/05

[3] Kontosakos, V. (2020). Fast quadratic programming for mean-variance portfolio optimization. SSRN Electronic Journal. https://doi.org/10.2139/ssrn.3586789

[4] Maximizing the sharpe ratio - University of South Carolina. (n.d.). Retrieved February 21, 2022, from https://people.stat.sc.edu/sshen/events/backtesting/reference/maximizing

[5] Mean-variance portfolio optimization. Mean-Variance Portfolio Optimization - MATLAB amp; Simulink. (n.d.). Retrieved February 21, 2022, from https://www.mathworks.com/help/finance/mean-variance-portfolio-optimization.html

[6] Mean-variance portfolio theory. CFA, FRM, and Actuarial Exams Study Notes. (2021, July 14). Retrieved February 21, 2022, from https://analystprep.com/study-notes/actuarial-exams/soa/ifm-investment-and-financial-markets/mean-variance-portfolio-theory/

[7] Portfolio analysis. Portfolio Analysis - MATLAB amp; Simulink. (n.d.). Retrieved February 21, 2022, from https://www.mathworks.com/help/finance/portfolio-analysis.html

[8] Portfolio optimization and asset allocation. Portfolio Optimization and Asset Allocation - MATLAB amp; Simulink. (n.d.). Retrieved February 21, 2022, from https://www.mathworks.com/help/finance/asset-allocation-and-portfolio-optimization.html

[9] The Sharpe Ratio. MoneyChimp. (n.d.). Retrieved February 21, 2022, from http://www.moneychimp.com/articles/risk/sharpe$_ratio.htm$