FINE 6200S: Investments

Stock-Trak Trading Simulation

Final Report

July 16, 2018

Daniel Fudge	215868904
Gordon Maxeiner	215293202
Maher Khatoun	215308058
Vipul Dudani	207981558

Contents (12 Pages in Main Body)

1 Recap Objectives and Strategy

Our investment objectives were specifically designed to ensure we gain relative experience identifying hypothetical opportunities and establishing a diversified portfolio to achieve an optimal risk to reward ratio. Our goal was to create a portfolio of approximately 20 speculative assets across developed countries with moderate growth and risk. We chose our benchmark to be iShares MSCI ETF (URTH). Our aim was to have a higher return and Sharpe ratio from May 28 through July 13. To reach these objectives our strategy and team responsibilities contained four aspects.

1. Speculative Asset Identification. These assets formed the book from which the portfolio was built.

2. Portfolio Weighting and Trading. Based on the given book, the desired weighting was determined and appropriate trades were executed.

3. Targeted asset selection for risk mitigation. The portfolio was reviewed for specific risks and assets were added to the book to mitigate the specific risks.

4. Currency risk mitigation and derivatives. The portfolio's currency risk exposure was monitored and appropriate action taken if deemed necessary. Derivatives were also evaluated to mitigate specific risks with the portfolio.

2 Strategy Implementation

Each of the following strategy sub-sections was the responsibility of the group member in the associated heading. These sections constitute the one-page description of the member's primary analysis, research and trades defined in the assignment deliverables.

2.1 Speculative Asset Identification (Gordon Maxeiner)

2.1.1 Part I

The first trades were executed on the day we signed up for StockTrak. There was no specific logic in choosing the stocks. We rather wanted to get used to the tool and understand how trades are executed. However, the experience we got from these trades were amongst the most valuable. First, we did not focus on diversifying the portfolio as the execution dates were prior to our StockTrak Proposal Paper. As a result, we were heavily invested in oil during days where the stocks plummeted (e.g. China National Offshore Oil Corporation, Pioneer Natural Resources) and it took us a long time to recover from our initial mistakes. Second, we overbought on the initial stocks, hence we did not have enough capital to invest in a bigger portfolio of 20 stocks or more. Consequently, we had to sell parts of our first bought stocks at a loss.

2.1.2 Part II

The initial picks of these stocks were based on multiple newspaper articles[1], outlining Warren Buffet's current portfolio as published by Berkshire Hathaway as part of the annual shareholder conference. The research also indicated details such as which stocks the investor sold and where he increased his long position through buying additional shares. Our strategy was to hold this diversified portfolio until the end of the investment period. The only changes would come through re-balancing activities. We built our portfolio by acquiring stocks of the following companies: Apple, Teva, Delta Airlines, Monsanto and Visa.

Monsanto contributed specifically to our experience during the StockTrak assignment. The company got acquired by Bayer AG, a deal that has been discussed since mid 2016. When it was finally closed, Monsanto's stock consequently stopped trading. After noticing the details, we contacted StockTrak and they reversed the trade to put the funds back into our account.

In summary, we could have further benefited from this strategy by also engaging in short positions of stocks that Buffet removed from his portfolio, such as IBM and Graham Holdings. This would have further improved the diversification of our portfolio and also increased our returns (both IBM and Graham Holdings show negative returns in the last weeks).

2.1.3 Part III

The following stocks were picked based on a recommendation from Zacks and as a result of the reduced geopolitical tensions in Asia[2]. Two main political events took place that influenced stock markets during the StockTrak simulation period. First, on April 27th 2018, the inter-Korean summit marked a historic event as it was the first meeting between the two countries since 1953. Second, on June 12th 2018, Singapore hosted

[1] For example: Van Doorn (May 7, 2018): Here's how Warren Buffett's stock picks for Berkshire Hathaway have performed in 2018; retrieved from: https://www.marketwatch.com/story/heres-how-warren-buffetts-stock-picks-for-berkshire-hathaway-have-performed-in-2018-2018-05-03

[2] Marwah (May 3, 2018): 4 Winning Asian Stocks to Buy as Geopolitical Tensions Ebb; retrieved from: https://www.zacks.com/stock/news/302109/4-winning-asian-stocks-to-buy-as-geopolitical-tensions-ebb

the meeting between U.S. President Trump and North-Korean's Kim Jong-Un[3]. We decided to follow Zacks' recommendation approximately one week prior to the second event as these stocks showed a positive development over the past weeks and month. All companies are said to benefit from improved relationships with North-Korea, hence our strategy was to validate this positive correlation.

1) Autohome Inc. [NYSE: ATHM]: The company is based in Beijing, China and one of the biggest online automotive advertising firms. While Autohome's stock price development during the last year's has been very positive, the stock price decreased by approximately 7.5% since we acquired shares in the company.
2) Woori Bank Co., Ltd. [KRX: 000030]: As one of the biggest South Korean Banks and part of the Woori Financial Group, the company has also established branches in North Korea, China and other Asian countries. It's stock price has been stagnant over the investment period and after seeing some intermediate gains we closed out the stock with a return of -1.2%.
3) Kingdee International Software Group Company Ltd. [SEHK: 0268]: Also based out of China and listed at the Hong Kong Stock Exchange, the company offers cloud based software solutions and partners with companies such as IBM and Ricoh. Despite the impressive return of over 72% in the 6 months prior to our investment period, our return was negative at -8.8%.

Aside from the pure return numbers, all stocks but Autohome helped us to diversify our currency risk and enabled us to diversify the portfolio. In summary however, the experience showed that neither impressive returns in previous months, nor top rankings in Zacks reports resulted in positive returns for us. This might be, in part, due to our short-term investment horizon. One could also argue, that the impact of the political events mentioned above was already reflected in the stock prices and the markets were therefore efficient.

[3] Rocha et. al. (June 12, 2018): President Trump meets Kim Jong Un; retrieved from: https://www.cnn.com/politics/live-news/trump-kim-jong-un-meeting-summit/index.html

2.2 *Portfolio Weighting and Trading (Daniel Fudge)*

Once a book is generated, the portfolio selection code in Appendix 3 was executed to build the desired portfolio. The input files to this code are listed below.

OpenPosition.csv is the current portfolio downloaded directly from Stocktrak.

override.csv contains hard coded expected monthly return values.

settings.csv contains parameters that allow us to tweak the selection and information such as exchange rates.

stocks/[name].csv, which refers to multiple files containing daily price history for each stock over at least the last 3 years.

With above data, the code first generates the expected returns for the following month. A very crude linear extrapolation on the rolling monthly return was used based on the previous 3 months as shown on next page. Note the user can override these predictions if desired. Figure A1 in the Appendix has a full size image for greater clarity.

Figure 1. Sample stock history plot from portfolio selection code. (Image stacked horizontally for report.)
Data Source: https://ca.finance.yahoo.com/ and http://www.stocktrak.com/account/graphportfolio

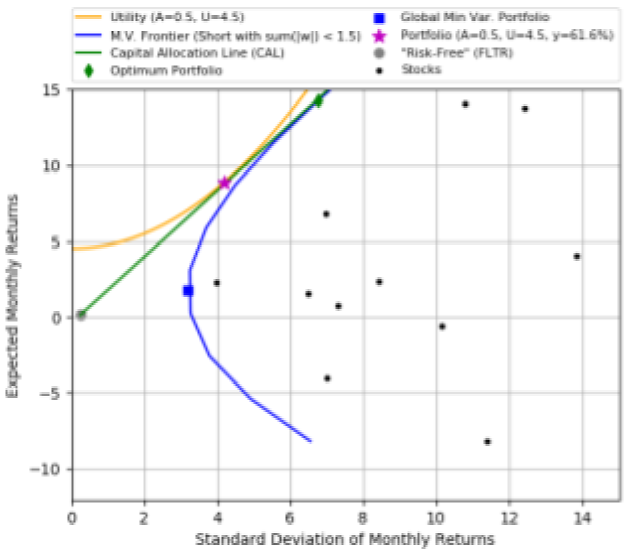
If the exercise was repeated, we would definitely replace the linear extrapolation with a more sophisticated prediction such as a 5 factor Fama-French model.[4]

Combining the covariance matrix of the returns over the previous 3 years with the return predictions, the code uses the Markowitz portfolio optimization model to created the minimum-variance frontier, optimal Capital Allocation Line (CAL) and the optimal

[4] Kenneth R. French, "Current Research Returns," accessed July 14, 2018. http://mba.tuck.dartmouth.edu/pages/faculty/ken.french/data_library.html

complete portfolio to maximize the utility function (u = r - ½Aσ2) as shown in Figure 2 below and a larger version in Figure A2 in the appendix.[5] Shorting was also limited by constraining the sum of the absolute value of the weights to be less than 1.5.

The code then reads the desired weights from the generated portfolio, the given desired portfolio value and the current portfolio positions to generated the required trade information as shown in Figure 2 below.



ticker	percentage	value_cad	price_cad	desired_number	current_number	buy
AAPL	0%	$0.00	$247.71	0	589	(589)
AC.TO	(5%)	($260,083.43)	$20.84	(12,480)	(10,625)	(1,855)
AMZN	18%	$897,384.46	$2,347.31	382	367	15
ATHM	0%	$0.00	$129.84	0	1,922	(1,922)
BIDU	(7%)	($342,258.70)	$345.93	(989)	(132)	(857)
CEO	0%	$0.00	$214.38	0	2,299	(2,299)
DDS	14%	$731,456.20	$110.79	6,602	1,894	4,708
FLTR	38%	$1,961,696.81	$32.73	59,936	5,700	54,236
NFLX	17%	$870,521.76	$512.44	1,698	672	1,026
PXD	0%	$0.00	$238.92	0	1,899	(1,899)
RIG	0%	$0.00	$17.39	0	7,192	(7,192)
SPR	(4%)	($189,282.62)	$112.06	(1,689)	(2,254)	564
TEVA	14%	$696,247.75	$30.15	23,090	14,821	8,269
V	15%	$749,002.57	$180.51	4,149	500	3,649

Figure 2. Mean-Variance plot and sample trade information generated by selection code.
Data Source: https://ca.finance.yahoo.com/ and http://www.stocktrak.com/account/graphportfolio

2.2.2 Executing Trades

We deviated from the ideal weights generated by the selection code in several manners. First, the selection routine was only executed once a week so price movement would alter the ideal weights between executions. Second, to reduce transaction fees, trades were only submitted when the its value was greater than ~5% of the portfolio. Third, trades were executed with limit orders (often staggered) if we believed the market would move in our favor. And finally, due to delays in order execution due to StockTrak and limit orders, we would often wait for trade proceeds before buying or short selling. This reduced the amount of volatility and interest fees due to leverage.

---

[5] Bodie, Kane, Marcus, Perrakis, Ryan. *Investments*. 8th Canadian Edition, Toronto, McGraw-Hill Ryerson 2015, p. 205-213.

2.3 *Targeted Asset Selection for Risk Mitigation (Vipul Dudani)*

The review for specific risk in the portfolio was done once every two weeks and a team meeting was held to decide if negatively correlated stocks should be added to mitigate risks. The two biggest holdings which were discussed the most was Baidu Inc. (BIDU) and Pioneer Natural Resources (PXD). Initially, we shorted 3622 shares of BIDU and bought approximately 5000 shares of PXD. The Bloomberg terminal was used to find negatively correlated stocks for both companies, as shown in Figure 3 below.

BIDU US Equity 1) Edit 2) Save to CORR 3) Launch Chart Grid 4) Settings Peer Correlation
06/20/2017 - 06/20/2018 Daily Calculation Correlation Local CCY
Peer Source Equity Index SPX Closest 500 By Market Cap
<Filter> Correlation Matrix (500 Rows x 10 Columns)

Security	BIDU	CCMP	IXK	NDX	NYFANG	QNET	MXAPEX	NYFANG	HXC	OGIGXT
11) DPS	-0.089	0.024	-0.001	0.024	0.042	0.011	-0.039	0.042	-0.046	-0.017
12) DTE	-0.060	0.096	0.083	0.107	0.030	0.019	0.047	0.030	-0.013	0.047
13) PCG	-0.054	0.115	0.100	0.119	0.040	0.052	0.096	0.041	-0.062	0.037
14) AEP	-0.028	0.094	0.088	0.112	0.026	0.022	0.020	0.026	-0.024	0.041
15) EIX	-0.024	0.126	0.110	0.126	0.051	0.071	0.127	0.052	0.044	0.094
16) XEL	-0.016	0.105	0.105	0.126	0.038	0.025	0.026	0.038	-0.013	0.049
17) MO	-0.011	0.291	0.238	0.273	0.127	0.151	0.055	0.127	0.079	0.135
18) LNT	-0.009	0.143	0.150	0.160	0.089	0.056	-0.016	0.089	-0.009	0.088
19) DUK	-0.003	0.100	0.108	0.127	0.061	0.028	-0.013	0.062	-0.051	0.021
20) SCG	-0.002	0.085	0.060	0.088	0.013	0.035	0.014	0.013	-0.008	0.026
21) PNW	0.000	0.173	0.166	0.186	0.084	0.079	-0.031	0.085	0.024	0.097
22) AEE	0.003	0.145	0.155	0.158	0.112	0.077	0.024	0.113	0.033	0.109
23) ES	0.005	0.170	0.156	0.183	0.100	0.106	0.052	0.100	0.110	0.135
24) SBUX	0.006	0.356	0.317	0.357	0.219	0.232	0.124	0.219	0.134	0.272
25) NKTR	0.007	0.236	0.194	0.207	0.142	0.202	0.014	0.142	0.200	0.197
26) WEC	0.016	0.176	0.181	0.192	0.117	0.095	0.011	0.118	0.025	0.111
27) SO	0.016	0.111	0.139	0.132	0.077	0.064	-0.066	0.077	-0.045	0.042
28) CMS	0.017	0.145	0.141	0.157	0.083	0.071	0.030	0.084	0.022	0.089
29) FE	0.017	0.164	0.165	0.177	0.136	0.114	0.053	0.136	0.035	0.123
30) SRE	0.019	0.097	0.102	0.109	0.085	0.062	0.038	0.085	0.018	0.074

(Color Bands Based on Statistical Significance: Most Significant , Significant , Less Significant , Least Significant)

PXD US Equity 1) Edit 2) Save to CORR 3) Launch Chart Grid 4) Settings Peer Correlation
06/20/2017 - 06/20/2018 Daily Calculation Correlation Local CCY
Peer Source Equity Index SPX Closest 500 By Market Cap
<Filter> Correlation Matrix (500 Rows x 10 Columns)

Security	PXD	SPX	S5ENRS	RIY	RAY	RLV	SVX	SPR	IXE	BWORLD
11) SO	-0.156	0.133	0.002	0.126	0.120	0.104	0.111	0.127	-0.005	0.070
12) DUK	-0.134	0.142	0.008	0.135	0.127	0.124	0.126	0.134	0.001	0.074
13) XEL	-0.113	0.161	0.019	0.155	0.147	0.150	0.152	0.154	0.013	0.119
14) AEP	-0.108	0.144	-0.027	0.139	0.132	0.131	0.131	0.139	-0.033	0.085
15) WEC	-0.105	0.217	0.034	0.214	0.208	0.190	0.190	0.213	0.028	0.144
16) AEE	-0.100	0.197	0.026	0.195	0.189	0.183	0.181	0.194	0.022	0.126
17) ED	-0.086	0.203	0.034	0.198	0.190	0.187	0.187	0.197	0.029	0.143
18) NI	-0.076	0.213	0.070	0.210	0.204	0.179	0.178	0.208	0.065	0.163
19) DTE	-0.075	0.162	0.040	0.158	0.152	0.167	0.163	0.158	0.033	0.134
20) WELL	-0.075	0.272	0.022	0.273	0.272	0.243	0.234	0.272	0.020	0.186
21) D	-0.071	0.178	0.035	0.173	0.167	0.176	0.182	0.172	0.032	0.118
22) COL	-0.068	0.298	0.145	0.298	0.296	0.274	0.272	0.296	0.141	0.273
23) ETR	-0.065	0.222	0.081	0.220	0.215	0.201	0.198	0.219	0.076	0.130
24) VTR	-0.065	0.212	0.003	0.213	0.213	0.186	0.179	0.212	0.001	0.152
25) K	-0.062	0.277	0.070	0.277	0.276	0.289	0.296	0.280	0.066	0.193
26) EVHC	-0.060	0.135	-0.000	0.140	0.148	0.137	0.138	0.143	-0.002	0.065
27) LNT	-0.059	0.187	0.039	0.184	0.178	0.170	0.166	0.183	0.033	0.125
28) HCP	-0.059	0.185	0.032	0.187	0.186	0.179	0.169	0.185	0.031	0.117
29) PNW	-0.059	0.236	0.066	0.232	0.225	0.224	0.226	0.231	0.060	0.147
30) CMS	-0.057	0.202	0.035	0.199	0.194	0.197	0.196	0.199	0.029	0.130

Figure 3. Screen capture of Bloomberg terminal showing stocks negatively correlated to PXD and BIDU.

As a team, we decided that instead of adding negatively correlated stocks, we would reduce our exposure in these companies. We started buying back shares of BIDU periodically. We also bought a protective put against PXD to mitigate the risk.

2.4 Currency Risk Mitigation and Derivatives (Maher Khatoun)

2.4.1 Interest Rate and Currency

Historically, the bank of Canada raises interests with a lag of 6 months to that of the United States Federal Reserve and given that we have now entered a rising interest rate environment, the US interest rate will generally be higher for at least 6 months for every hike, and thus would promise a devaluation of the loonie as the funds outflow from Canada to the US.

As per the below figure 4, representing our portfolio's open position as at June 18, 2018, 88% of our portfolio's exposure to currency was to the USD dollar, followed by 5% to the EUR, for a total of 93% for the 2 major currencies.

Given the Above, we have decided in our meeting on June 18 that the currency mitigation strategy would be non-interventionist, and that no specific transactions would be necessary as we repatriate the value of the portfolio back to CAD at devalued levels at the end of the 3 months stk trak exercise.

Currency	Exposure	Percentage of Total
USD	7,821,730.19	88.01%
EUR	445,551.66	5.01%
CAD	244,481.25	2.75%
KRW	143,293.01	1.61%
GBP	128,684.44	1.45%
HKD	103,858.64	1.17%
	8,887,599.19	**100.00%**

Figure 4. Currency Exposure of our portfolio as at June 18, 2018
Data Source: own calculation and http://www.stocktrak.com/account/openpositions

2.4.2 Use of Derivatives:

As part of our risk mitigation, we have utilized options as protective puts to hedge against a fall of the price of the equities we hold. The options in Figure 5, are put options on the stocks of CNOOC (China National Offshore Oil Corporation) and Pioneer Natural Resources Company, both natural resource equities. The derivatives offer us the option to sell if the equities we hold devalue as oil prices fall, given their relationship to oil prices.

Order Date	Order	Symbol	Qty	Order Price	Trade Price	Type	Currency	Status
5/27/2018	MARKET - BUY	CEO1815R160	22	MKT	$4.40	OPTIONS	USD	FILLED 5/29/2018 3:55 PM
5/27/2018	MARKET - BUY	PXD1820S185	25	MKT	$3.90	OPTIONS	USD	FILLED 5/30/2018 11:07 AM
6/16/2018	MARKET - SELL	CEO1815R160	-22	MKT	$0.00	OPTIONS	USD	FILLED 6/16/2018 6:02 PM

Figure 5. Protective put options used
Data Source: stock track transaction history - http://www.stocktrak.com/account/transactionhistory

3 Evaluation Against Benchmark

Our proposal stated that we would benchmarking our portfolio against the iShares MSCI World ETF (URTH).[6] In particular our goal was to have a higher return (cumulative) and Sharpe ratio over the period from May 28 through July 13 2018. The summary code in the Appendix generates Figure 6 below, which clearly illustrates that we achieved our objective. Our cumulative excess return was 3.1% higher than the benchmark and the Sharpe ratio was also 0.04 greater.

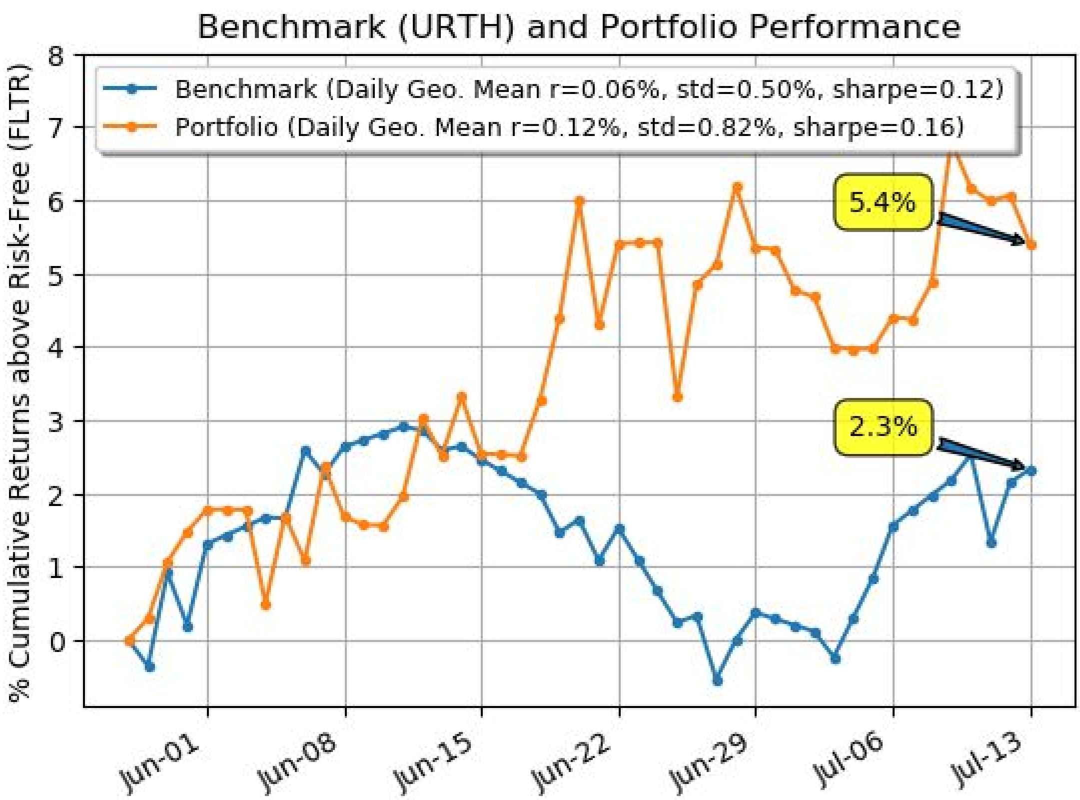


Figure 6. Sample trade information generated by selection code (see appendix).
Data Source: https://ca.finance.yahoo.com/ and http://www.stocktrak.com/account/graphportfolio

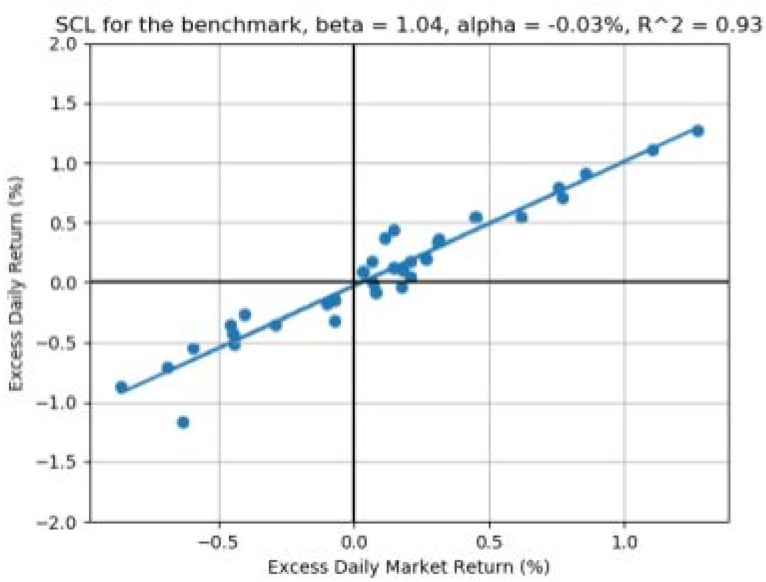
[6] "Ishares MSCI World Index Fund (URTH)", Quotes, Yahoo Finance, accessed July 14, 2018. https://finance.yahoo.com/quote/URTH?p=URTH

4 Detailed Performance Analysis

Figures 7 and 8 below summarize the performance of the portfolio relatively to the market and the benchmark from May 28 through July 13 2018. The market was represented by the S&P 500 index (^GSPC) and the benchmark was the iShares MSCI World ETF (URTH). Appendix 4 contains the code used to generate these performance metrics from the daily Yahoo Finance price history. The portfolio performance was derived directly from the daily portfolio values downloaded from StockTrak.

	Based on Daily Returns			Portfolio Comparison	
	Market	Benchmark	Portfolio	Market	Benchmark
G. Mean Return	0.08%	0.06%	0.12%	0.04%	0.06%
Standard Dev.	0.46%	0.50%	0.82%	0.35%	0.32%
Sharpe Ratio	0.178	0.107	0.145	(0.032)	0.039
CAPM Beta	1.000	1.041	0.401	(0.599)	(0.640)
CAPM Alpha	0.00%	(0.03%)	0.09%	0.09%	0.12%
Treynor Measure	0.077	0.048	0.285	0.208	0.237
Information Ratio	0.000	(0.060)	0.106	0.106	0.165
M2	0.00%	(0.03%)	(0.01%)	(0.01%)	0.02%

Figure 7. Performance analysis results by summary code (see appendix).
Data Source: https://ca.finance.yahoo.com/ and http://www.stocktrak.com/account/graphportfolio



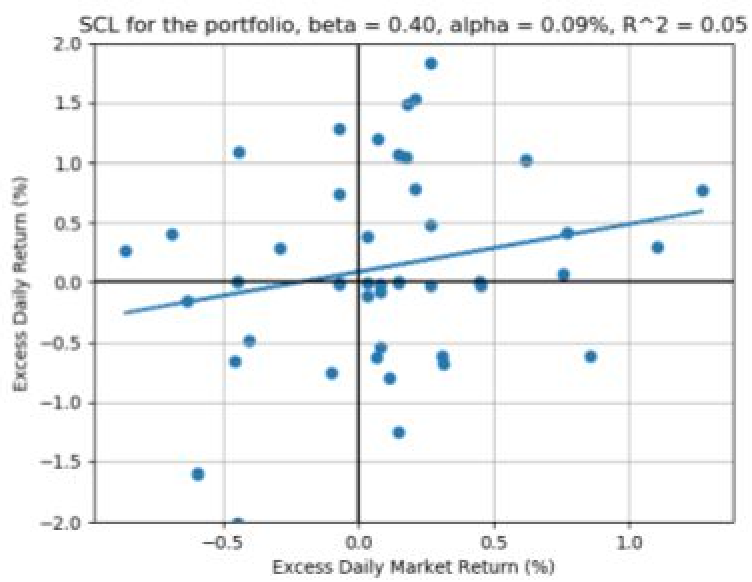


Figure 8. Security Characteristic Line (SCL) for benchmark & portfolio by summary code (see appendix).
Data Source: https://ca.finance.yahoo.com/ and http://www.stocktrak.com/account/graphportfolio

As shown by the colouring in Figure 7, our portfolio bet the benchmark in all metrics except the standard deviation, however the higher return allowed us to achieve a higher Sharpe ratio. We also bet the market in all measures except for the Sharpe ratio due to the much higher standard deviation.

Figure 8 captures the Security Characteristic Line (SCL) for both the benchmark and the portfolio. The y-intercept is Jensen's CAPM alpha and the slope is the Beta. The figure also illustrates the portfolio's much lower R^2 value, which indicates that the portfolio's alpha and beta estimates are very poor resulting in noisy Treynor measures and information ratio values. This is most likely due to the shifting portfolio weighting and possibly there was insufficient diversity to eliminate the non-systematic risk.

Our M^2 measure was also negative but still greater than the benchmark. This is illustrated in Figure 9 below. Since the slope of the CAL (Sharpe ratio) for both the benchmark and the portfolio was less than the market, the returns interpolated at the market standard deviation ("*") were both less than the market return.

This illustrates that for the same level of total risk over the given time period, our portfolio bet the benchmark but the market bet both our portfolio and the benchmark.

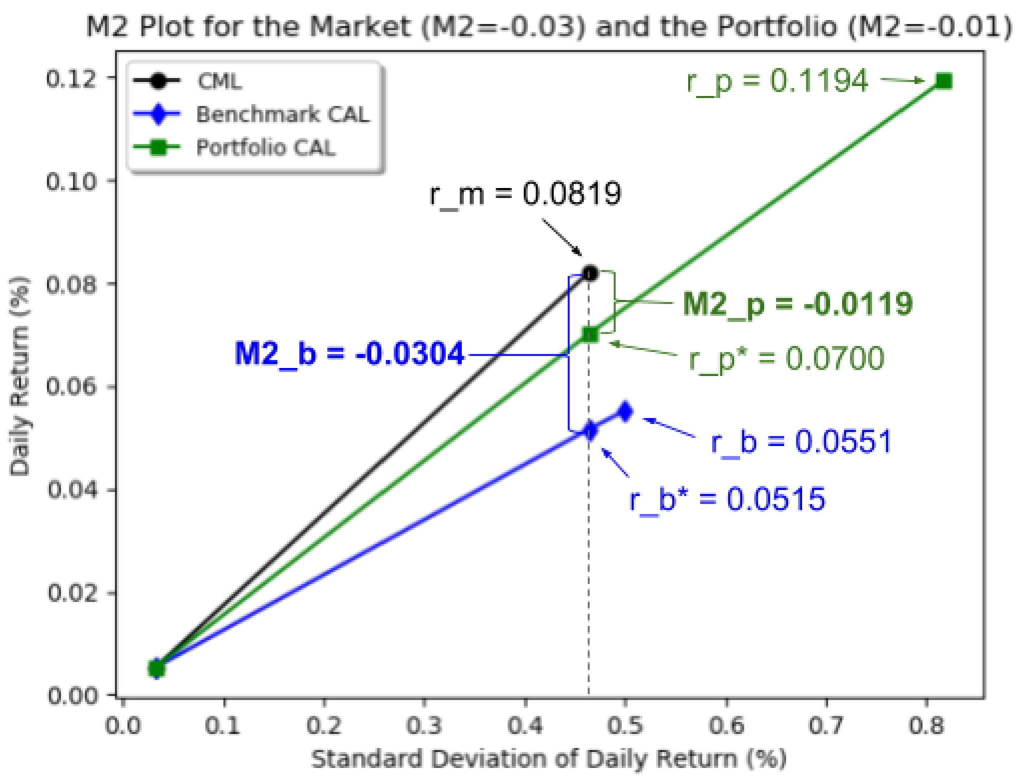


Figure 9. M^2 plot of the benchmark (b) & portfolio (p) generated by summary code (see appendix).
Data Source: https://ca.finance.yahoo.com/ and *and http://www.stocktrak.com/account/graphportfolio*

5 *Reflection of Experience*

The main takeaway from this exercise is the importance of diversification, calmly evaluating possible tactics and strategies and planning logical rules for entering and exiting positions. As was described in section 2.1.1, we initially entered very large positions without properly diversifying the portfolio. This was a major learning experience. We should have planned a fully diversified portfolio, including targeted risk mitigation against large positions, currency and interest rate risk before making the first trade. We should have also discussed contingencies for best and worst case scenarios such as oil dropping 4.5% in a day.

Once the plan was agreed, we should have slowly stepped into the positions. This would have allowed us to maintain a diversified portfolio as we were getting established. Not all trades execute immediately, this is especially true when using limit orders. If, for instance, the desired portfolio has a 10% weighting on a stock, we should not place an order for the full 10%. If we did and the order went through before the other stocks that add diversification, we would be exposed to non-systematic risk. Instead each position should be entered in smaller equal portions, to maintain diversification and minimize non-systematic risk.

We also learned to fill the system with stop-loss orders to protect against very bad days and limit orders to capture gains on very good days. Limit orders were also used to step into or out of positions in a methodical manner to maximize gains. This was required because we could not monitor the markets everyday. It also forces you to think through your strategy in a calm, non-rushed manner. Instead of in a panic as the "analysts" claim the sky is falling or a stock is skyrocketing. This was especially true at the opening of trading when wild price swings were very unnerving.

On a much more tactical note, we also agreed that the simplistic linear extrapolation of rolling monthly returns should have been replaced with a more sophisticated prediction algorithm. We don't expect to be able to predict the return perfectly, but a more advanced model such as a 5 factor Fama-French model would have increased the accuracy. A more accurate expected return value would generate portfolio weights that result in a higher portfolio return for the given level of systematic risk.

We also gained a great deal of skepticism regarding return prediction based previous impressive performance, top analyst rankings such as Zacks or following the "news". This might be due to our short-term investment horizon or simply the market efficiency resulting in us always being too late to benefit.

Appendix 1 - Bibliography

Bodie, Zvi, Kane, Alex, Marcus, Alan J., Perrakis, Stylianos, Ryan, Peter J., Switzer, Lorne N.. *Investments*, 8th Canadian Edition, Whitby: McGraw-Hill Ryerson, 2015.

French, Kenneth R.. “Kenneth R. French”. Accessed July 7, 2018. http://mba.tuck.dartmouth.edu/pages/faculty/ken.french/index.html

Kahout, Tania. “How a US interest rate hike affects Canadians, oil and the loonie.” *Global News*, December 14, 2016. https://globalnews.ca/news/3126521/us-fed-rate-hike-canada-oil-loonie-interest-rates/

Marwah, Nitish.“4 Winning Asian Stocks to Buy as Geopolitical Tensions Ebb.” *Zacks*, May 3, 2018. https://www.zacks.com/stock/news/302109/4-winning-asian-stocks-to-buy-as-geopolitical-tensions-ebb

Rocha, Veronica, Ries, Brian, Westcott, Ben, Wagner, Meg, Wills, Amanda. “President Trump meets Kim Jong Un.” *CNN*, June 12, 2018. https://www.cnn.com/politics/live-news/trump-kim-jong-un-meeting-summit/index.html

van Doorn, Philip. “Here’s how Warren Buffett’s stock picks for Berkshire Hathaway have performed in 2018”, *Market Watch*, May 7, 2018. https://www.marketwatch.com/story/heres-how-warren-buffetts-stock-picks-for-berkshire-hathaway-have-performed-in-2018-2018-05-03

Yahoo. “Finance.” Last accessed July 14, 2018. https://ca.finance.yahoo.com/

Appendix 2 - Illustrations

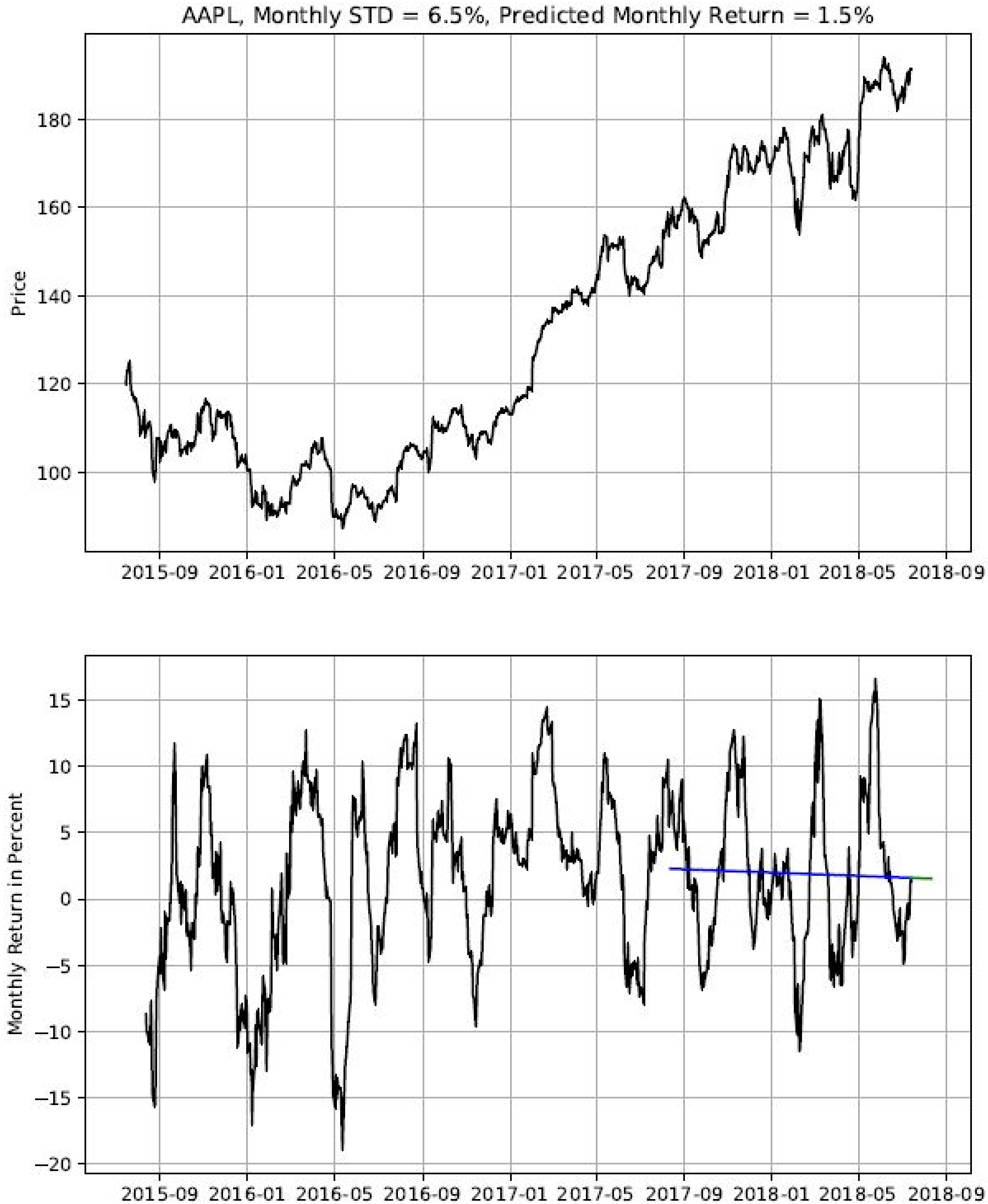


Figure A1. Sample stock history plot from portfolio selection code.

Data Source: https://ca.finance.yahoo.com/

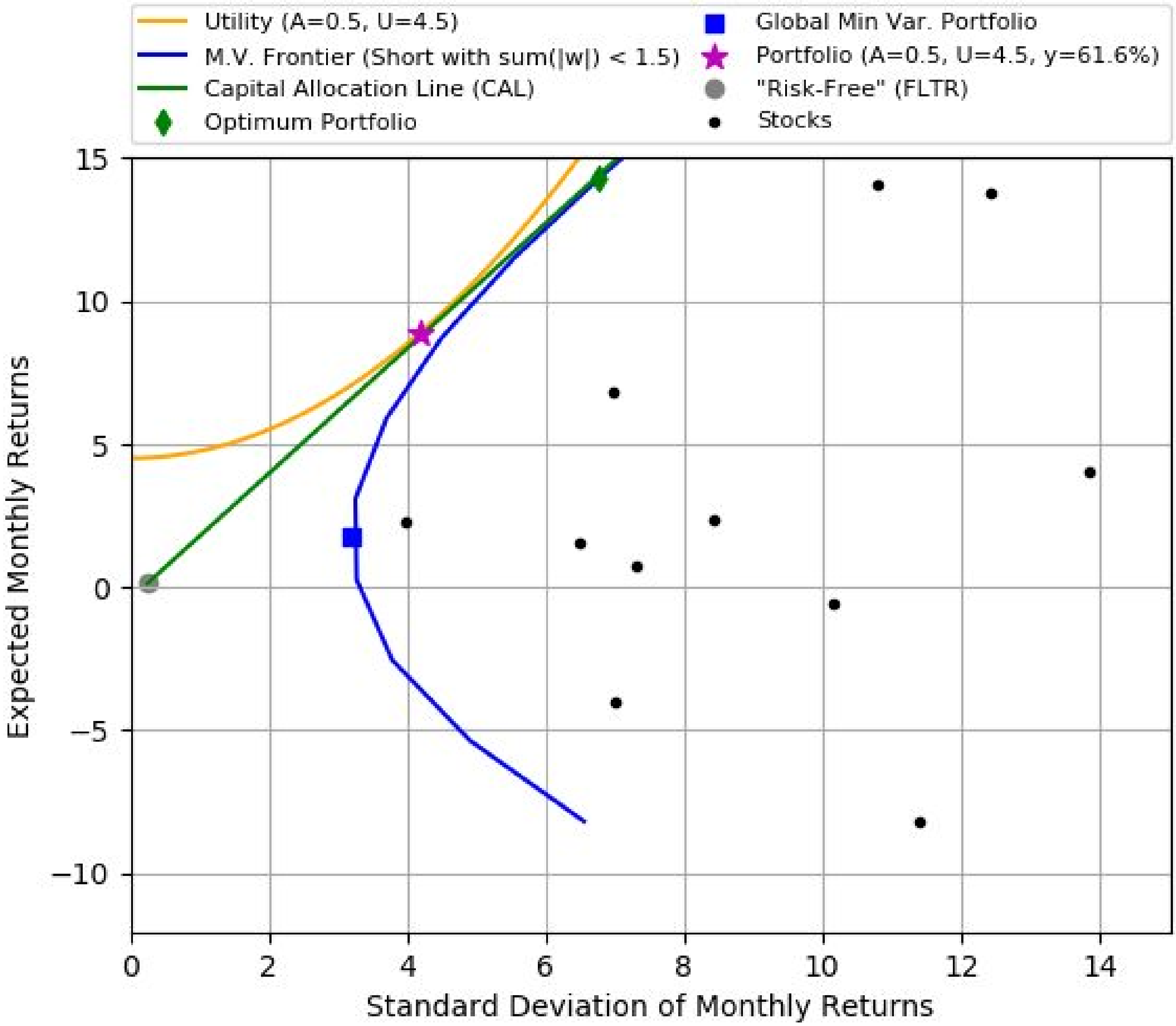


Figure A2. Markowitz portfolio optimization generated by selection code.

Data Source: https://ca.finance.yahoo.com/

Appendix 3 - Portfolio Selection Code

```
#!/usr/bin/env python
"""
This script performs an optimal portfolio selection based on the Sharpe ratio and utility maximization.

Notes:
   [CC] is the Currency Code such as "gbp" for Great Britain Pound.
   [SE] is the Stock Extension used in the Yahoo Finance name on the stock price history files, such as "to" for the
       Toronto Stock Exchange.  The [SE] is used to make to the associated [CC].

Input files required in current working directory:
   OpenPosition.csv - Current StockTrak portfolio positions.
   override.csv - Contains hard coded expected monthly return values.
   settings.csv - Contains the following settings.
       risk_free_name (str):  The risk free asset name.
       utility_a (float):  The utility factor use to make asset allocation.
       shorting (str):  The shorting method, which can be:
           n = No shorting.
           y = Full shorting.
           limited = The absolute value of the weights can't be greater than 1.5.
       portfolio_value (float):  The desired portfolio value used to determine number of stock to purchase / sell.
       [CC]_cad (float):  The CC to Canadian dollar conversion rate.
       [SE] (str):  The [CC] associated with the given [SE].
   stocks/[name].csv - The [name] refers to the stock ticker.  Each csv file needs to contains the "Date" and
       "Adj Close" columns.  Daily frequency is required.

Output files generated in current working directory:
   portfolio.csv - Detailed summary of desired portfolio and the required trades.
   portfolio.png - Mean-variance plot of the desired portfolio.
"""

from datetime import date, timedelta
from glob import glob
import numpy as np
from os import chdir, getcwd, remove
from os.path import isfile
import pandas as pd
from scipy.optimize import minimize
from shutil import move

# ************************************************************************************
# Set version number
# ************************************************************************************
__version__ = '1.0'


# ************************************************************************************
def make_portfolio():
   """Create the optimal portfolio.

   Input files required in current working directory:
       settings.csv - Contains the following settings under the "name" and "value" columns.
           risk_free_rate (float):  The risk free rate in percentage, i.e. 10% as 10.0.
           utility_a (float):  The "A" constant in the quadratic utility function; u = e - 1/2 * A * std^2
   """

   import matplotlib
   matplotlib.use('Agg')
   import matplotlib.pyplot as plt

   a = settings["utility_a"]
```

```python
# Set the weight bounds and constraints
# ------------------------------------------------------------------------------------
weights = np.ones(book.shape[0]) / book.shape[0]

weight_con = ({'type': 'eq', 'fun': lambda x: x.sum() - 1.0},)
weight_bnd = weights.shape[0] * [(0.0, 1.0)]
shorting = "n"
if "shorting" in settings.index:
    if settings['shorting'].lower() in ['t', 'true', 'y', 'yes']:
        weight_bnd = weights.shape[0] * [(-1.0, 2.0)]
        shorting = "y"
    elif settings['shorting'].lower() == 'limited':
        weight_con += ({'type': 'ineq', 'fun': lambda x: 1.5 - np.abs(x).sum()},)
        weight_bnd = weights.shape[0] * [(-1.0, 1.0)]
        shorting = "L"

# Determine the optimal portfolio
# ------------------------------------------------------------------------------------
print("Generating optimal portfolio.")
res = minimize(negative_slope, x0=weights, bounds=weight_bnd, constraints=weight_con)
opt_weights = res.x
opt_r = (opt_weights * book.loc[:, 'return']).sum()
opt_std = get_std(opt_weights)

# Determine the minimum variance portfolio
# ------------------------------------------------------------------------------------
print("Generating minimum variance portfolio.")
res = minimize(get_std, x0=opt_weights, bounds=weight_bnd, constraints=weight_con)
min_var_r = (res.x * book.loc[:, 'return']).sum()
min_var_std = get_std(res.x)

# Determine complete portfolio point
# ------------------------------------------------------------------------------------
y = (opt_r - risk_free["return"]) / (a * opt_std ** 2)
if y > 1.0:
    print("Optimal y is {:.1f}%, but limiting to 100.0%".format(100.0 * y))
    y = 1.0
complete_r = y * opt_r + (1.0 - y) * risk_free["return"]
complete_std = y * opt_std
u = complete_r - 0.5 * a * complete_std ** 2

# Compile desired portfolio
# ------------------------------------------------------------------------------------
print("Saving portfolio.csv")
portfolio = pd.DataFrame(columns=["percentage"], index=book.index)
portfolio.loc[:, 'percentage'] = opt_weights
portfolio = portfolio.loc[portfolio.percentage.round(2) != 0.00, :]
portfolio['percentage'] = y * portfolio['percentage'] / portfolio['percentage'].sum()
portfolio.loc[settings["risk_free_name"], 'percentage'] = 1.0 - y
portfolio['value_cad'] = portfolio['percentage'] * settings['portfolio_value']

for name in portfolio.index:
    tmp_a = name.lower().split(".")
    if len(tmp_a) == 1:
        ex = settings['usd_cad']
    else:
        ex = settings['{}_cad'.format(settings[tmp_a[1]])]

    if name == settings['risk_free_name']:
        portfolio.loc[settings['risk_free_name'], 'price_cad'] = risk_free["close"] * ex
    else:
        portfolio.loc[name, 'price_cad'] = book.loc[name, 'close'] * ex

portfolio['desired_number'] = portfolio['value_cad'] / portfolio['price_cad']
```

```
# Read current portfolio and calculate the necessary trades
# ---------------------------------------------------------------------------------
current_portfolio = pd.read_csv("OpenPosition.csv")
current_portfolio.rename(columns={"Quantity": "current_number"}, inplace=True)
for i in current_portfolio.index:
    currency = current_portfolio.loc[i, "Currency"].lower()
    if currency != "usd":
        market = settings[settings == currency].index[0].upper()
        current_portfolio.loc[i, "Symbol"] = "{}.{}".format(current_portfolio.loc[i, "Symbol"], market)
current_portfolio.set_index("Symbol", inplace=True)

portfolio = pd.concat([portfolio, current_portfolio["current_number"]], axis=1)
for name in ["current_number", "percentage", "value_cad", "desired_number"]:
    portfolio[name] = portfolio[name].fillna(0.0)
missing = portfolio.index[portfolio.price_cad.isnull()].tolist()

for name in missing:
    tmp_a = name.lower().split(".")
    if len(tmp_a) == 1:
        ex = settings['usd_cad']
    else:
        ex = settings['{}_cad'.format(settings[tmp_a[1]])]
    portfolio.loc[name, 'price_cad'] = current_portfolio.loc[name, "LastPrice"] * ex

portfolio["buy"] = portfolio['desired_number'] - portfolio['current_number']

# Save portfolio
# ---------------------------------------------------------------------------------
portfolio = portfolio.astype({n: int for n in ["desired_number", "current_number", "buy"]})
portfolio.to_csv("portfolio.csv", index_label="ticker", float_format='%.2f')

# Create the utility function for plotting
# ---------------------------------------------------------------------------------
utility_std = np.linspace(0, max(opt_std, complete_std, book.loc[:, 'std'].max()) * 1.1, num=100)
utility_r = u + 0.5 * a * utility_std ** 2

# Create the capital allocation line (CAL)
# ---------------------------------------------------------------------------------
cal_std = [risk_free["std"], utility_std[-1]]
d_r = (utility_std[-1] - risk_free["return"]) * (opt_r - risk_free["return"]) / (opt_std - risk_free["std"])
cal_r = [risk_free["return"], risk_free["return"] + d_r]

# Create minimum variance frontier
# ---------------------------------------------------------------------------------
print("Generating minimum variance frontier.")
frontier_r = np.linspace(min(min_var_r, book.loc[:, 'return'].min()),
                         max(opt_r, book.loc[:, 'return'].max()), num=10)
frontier_std = np.empty_like(frontier_r)
weights = opt_weights.copy()
for i in range(frontier_r.shape[0]):

    r_con = weight_con + ({'type': 'eq', 'fun': lambda x: (x * book.loc[:, 'return']).sum() - frontier_r[i]},)
    res = minimize(get_std, x0=weights, bounds=weight_bnd, constraints=r_con)
    weights = res.x
    frontier_std[i] = get_std(res.x)
```

```
    # Make a plot
    # ---------------------------------------------------------------------------------
    print("Making portfolio.png.")
    fig, ax1 = plt.subplots(nrows=1, ncols=1)
    ax1 = plt.subplot(111)
    ax1.set_xlabel("Standard Deviation of Monthly Returns")
    ax1.set_ylabel("Expected Monthly Returns")
    ax1.grid()

    ax1.plot(utility_std, utility_r, color='orange', label='Utility (A={}, U={:.1f})'.format(a, u))
    if shorting == 'y':
        ax1.plot(frontier_std, frontier_r, color='b', label='Min Var. Frontier (Shorting)')
    elif shorting == 'n':
        ax1.plot(frontier_std, frontier_r, color='b', label='Min Var. Frontier (No Shorting)')
    else:
        ax1.plot(frontier_std, frontier_r, color='b', label='M.V. Frontier (Short with sum(|w|) < 1.5)')
    ax1.plot(cal_std, cal_r, color='g', label='Capital Allocation Line (CAL)')
    ax1.scatter(opt_std, opt_r, color='g', marker='d', label='Optimum Portfolio', zorder=8)
    ax1.scatter(min_var_std, min_var_r, color='b', marker='s', label='Global Min Var. Portfolio')
    ax1.scatter(complete_std, complete_r, color='m', marker='*',
                label='Portfolio (A={}, U={:.1f}, y={:.1f}%)'.format(a, u, 100.0*y), zorder=9, s=80)
    ax1.scatter(risk_free["std"], risk_free["return"], color='grey', marker='o',
                label='"Risk-Free" ({})'.format(settings["risk_free_name"]))
    ax1.scatter(book.loc[:, 'std'], book.loc[:, 'return'], color='k', marker='.', label='Stocks', zorder=10)
    ax1.legend(bbox_to_anchor=(0., 1.02, 1., .102), loc=3, ncol=2, mode="expand", borderaxespad=0., fontsize=8)
    max_axis = 1.05 * max(opt_r, complete_r, book.loc[:, 'std'].max())
    ax1.set_ylim(top=max_axis)
    ax1.set_xlim(left=0, right=max_axis)

    fig.savefig("portfolio.png", orientation='landscape', bbox_inches="tight")
    plt.close(fig)


# ***************************************************************************************
def negative_slope(weights):
    """Calculates the negative slope from the given weighted portfolio to the risk-free asset."""
    r = (weights * book.loc[:, 'return']).sum()
    std = get_std(weights)

    return (risk_free["return"] - r) / (std - risk_free["std"])


# ***************************************************************************************
def get_std(weights):
    """Calculates the portfolio standard deviation from the given weights."""

    std = 0.0
    for i in range(book.shape[0]):
        std += (weights * book_cov.iloc[:, i]).sum() * weights[i]
    std = np.sqrt(std)

    return std
```

```python
# ************************************************************************************
def override():
    """Override trended return values with manually supplied values.

    Input files required in current working directory:
        override.csv - Monthly return values to override trend values.
"""

    if isfile("override.csv"):
        manual_returns = pd.read_csv("override.csv")
        manual_returns.set_index("name", inplace=True)
        tmp_str = "{} has predicted value of {:.1f}% that is overridden by {:.1f}%."
        for name in manual_returns.index.intersection(book.index).tolist():
            print(tmp_str.format(name, book.loc[name, "return"], manual_returns.loc[name, "return"]))
            book.loc[name, "return"] = manual_returns.loc[name, "return"]


# ************************************************************************************
def trend():
    """Analyze daily stock prices to determine standard deviation and expected monthly return.

    Input files required in current working directory:
        stocks/[name].csv - The [name] refers to the stock ticker.  Each csv file needs to contains the "Date" and
            "Adj Close" columns.  Daily frequency is required.

    Returns:
        pd.Series:  [std, return, close]  The price information for the given risk-free asset.
        pd.DataFrame:  [name, [std, return, close]]  Each row is a stock by the name and columns are the standard
            deviation, expected monthly return and today's adjusted close value.
        pd.DataFrame:  [name, name]  The covariance matrix of the monthly stock returns.
"""

    import matplotlib
    matplotlib.use('Agg')
    import matplotlib.pyplot as plt
    from matplotlib.backends.backend_pdf import PdfPages

    # Create DataFrame with index starting 36 months ago
    # ----------------------------------------------------------------------------------
    now = date.today()
    start = now - timedelta(days=3*365.25)
    prices = pd.DataFrame(index=pd.date_range(start, now))

    # Read stock prices
    # ----------------------------------------------------------------------------------
    chdir("stocks")
    for file_name in glob("*.csv"):
        csv = pd.read_csv(file_name, usecols=["Date", "Adj Close"])
        name = file_name[:-4]
        csv.rename(columns={'Adj Close': name}, inplace=True)
        csv[name] = pd.to_numeric(csv[name], errors='coerce', downcast='float')
        csv['Date'] = pd.to_datetime(csv['Date'])
        csv.set_index('Date', inplace=True)
        prices = pd.concat([prices, csv], axis=1)
    prices = prices.resample('D').interpolate().bfill().ffill()
    prices = prices.loc[prices.index >= start.strftime("%Y-%m-%d"), :]
    returns = 100.0 * (prices.iloc[28:, :] - prices.iloc[:-28, :].values) / prices.iloc[:-28, :].values
    stocks = pd.DataFrame(index=prices.columns.tolist(), columns=["std", "return", "close"])
    chdir(cwd)
```

```
# Fit trend to rates, predict one month and calculate standard deviation
# ---------------------------------------------------------------------------------------
prediction_duration = 12*28
x = np.arange(prediction_duration)
fits = {}
for name in returns.columns:
    fits[name] = np.poly1d(np.polyfit(x, returns[name].values[-prediction_duration:], 1))
    stocks.loc[name, "return"] = fits[name](prediction_duration + 28)
    stocks.loc[name, "std"] = returns.loc[:, name].std()
    stocks.loc[name, "close"] = prices.loc[now, name]

# Create pdf report
# ---------------------------------------------------------------------------------------
pdf_name = 'stocks.pdf'
print("Making {}.".format(pdf_name))
tmp_pdf_name = '.tmp.pdf'
for name in [n for n in [pdf_name, tmp_pdf_name] if isfile(n)]:
    remove(name)
pdf = PdfPages(tmp_pdf_name)

t = [now - timedelta(days=prediction_duration), now, now + timedelta(days=28)]
title_text = "{}, Monthly STD = {:.1f}%, Predicted Monthly Return = {:.1f}%"
for name in prices.columns:
    fig = plt.figure(figsize=(8.5, 11))
    ax1 = plt.subplot(211)
    plt.title(title_text.format(name, stocks.loc[name, "std"], stocks.loc[name, "return"]))
    ax1.plot(prices.index, prices[name], color='k', marker=',')
    ax1.set_ylabel("Price")
    ax1.grid()

    ax2 = plt.subplot(212, sharex=ax1)
    ax2.plot(returns.index, returns[name], color='k', marker=',')
    ax2.set_ylabel("Monthly Return in Percent")
    ax2.grid()

    r = [fits[name](0), fits[name](prediction_duration), stocks.loc[name, "return"]]
    ax2.plot(t[:2], r[:2], color='b')
    ax2.plot(t[1:], r[1:], color='g')

    pdf.savefig(fig, papertype='letter', orientation='landscape', pad_inches=0.25)
    plt.close(fig)

pdf.close()
move(tmp_pdf_name, pdf_name)

# Remove "risk-free" asset
# -----------------------------------------------------------------------------------
rf = stocks.loc[settings["risk_free_name"], :]
stocks.drop(settings["risk_free_name"], inplace=True, axis=0)
returns.drop(settings["risk_free_name"], inplace=True, axis=1)

# Calculate the covariance
# -----------------------------------------------------------------------------------
cov = returns.cov()

return rf, stocks, cov
```

```
# %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
# Main body of Code
# %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

cwd = getcwd()

# Read the settings
# ---------------------------------------------------------------------------------
settings = pd.read_csv("settings.csv", index_col="name", squeeze=True)
settings = settings.str.strip()
settings = pd.to_numeric(settings, errors='coerce').fillna(settings)
settings['cad_cad'] = 1.0

# Initialize the book with historical trend and calculate the monthly return covariance
# -------------------------------------------------------------------------------------
risk_free, book, book_cov = trend()

# Updated book with manually entered values
# -------------------------------------------------------------------------------------
override()

# Create portfolio
# -------------------------------------------------------------------------------------
make_portfolio()
```

Appendix 4 - Evaluation Code

```
#!/usr/bin/env python
"""
This script generates a summary of the Stock-Trak performance.

The standard deviation is computed with daily returns and the Sharpe Ratio uses the geometric average daily returns.
The date range used in the above calculations is May 28, 2018 until the current data.

Input files required in current working directory:
   settings.csv - Contains the following settings.
       risk_free_name (str):  The risk free asset name.
       benchmark_name (str):  The benchmark asset name.
       market_name (str):  The market index name.
   stocks/[risk_free_name].csv - Yahoo formatted historical performance of risk free asset.
   [benchmark_name].csv - Yahoo Finance formatted historical performance of benchmark asset.
   [market_name].csv - Yahoo Finance formatted historical performance of market index.
   OpenPosition.csv - Contains the current Stock-Trak portfolio.
   portfolio_history.csv - Contains the historical performance of the Stock-Trak account.

Output files generated in current working directory:
   currency.png - Pie chart of the currency distribution in the portfolio.
   m2.png - Capital Allocation Line (CAL) for the market, benchmark and portfolio showing M2 calculation.
   scl_benchmark.png - The Security Characteristic Line (SCL) for the benchmark.
   scl_market.png - The Security Characteristic Line (SCL) for the market.
   scl_portfolio.png - The Security Characteristic Line (SCL) for the portfolio.
   stats.csv - Compilation of summary statistics for the market, benchmark and portfolio.
"""

from datetime import datetime
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
from os.path import join
import pandas as pd
from scipy.stats.mstats import gmean
from sklearn import linear_model
from sklearn.metrics import r2_score

# ******************************************************************************************
# Set version number
# ******************************************************************************************
__version__ = '1.0'


# %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
# Main body of Code
# %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

start_date = datetime(2018, 5, 28)

# Read the settings
# --------------------------------------------------------------------------------------
settings = pd.read_csv("settings.csv", index_col="name", squeeze=True)
settings = settings.str.strip()

# Read Risk-Free performance
# --------------------------------------------------------------------------------------
name = join("stocks", settings["risk_free_name"] + ".csv")
risk_free = pd.read_csv(name, parse_dates=["Date"], usecols=["Date", "Adj Close"])
risk_free.set_index("Date", inplace=True)
risk_free = risk_free.resample('D').interpolate()
start_price = risk_free.loc[start_date, "Adj Close"]
risk_free['Return'] = 100.0 * (risk_free['Adj Close'] - start_price) / start_price
risk_free['Daily_Return'] = risk_free['Adj Close'].pct_change()
risk_free = risk_free.loc[risk_free.index > start_date, :]
```

```
r_f = 100.0 * (gmean(risk_free['Daily_Return'] + 1.0) - 1.0)
risk_free['Daily_Return'] = risk_free['Daily_Return'] * 100.0
std_f = risk_free['Daily_Return'].std()

# Read benchmark, market and portfolio data
# ----------------------------------------------------------------------------------
total_r = pd.DataFrame(0.0, index=risk_free.index, columns=["market", "benchmark", "portfolio"])
daily_r = pd.DataFrame(0.0, index=risk_free.index, columns=["market", "benchmark", "portfolio"])
stats = pd.DataFrame()
for name in total_r.columns:
   if name == "portfolio":
       data = pd.read_csv("portfolio_history.csv", parse_dates=["Date"], usecols=["Date", "Value"])
   else:
       data = pd.read_csv(settings["{}_name".format(name)] + ".csv", parse_dates=["Date"],
                          usecols=["Date", "Adj Close"])
       data.rename(index=str, columns={"Adj Close": "Value"}, inplace=True)

   data.set_index("Date", inplace=True)
   data = data.resample('D').interpolate()
   data = data.loc[data.index >= start_date]
   start_price = data.loc[start_date, "Value"]
   total_r[name] = 100.0 * (data['Value'] - start_price) / start_price - risk_free['Return']
   daily_r[name] = data['Value'].pct_change()
   stats.loc["mean_r", name] = 100.0 * (gmean(daily_r[name].loc[daily_r[name].index > start_date] + 1.0) - 1.0)
   daily_r[name] = 100.0 * daily_r[name]
   stats.loc["std", name] = daily_r[name].std()
   stats.loc["Sharpe", name] = (stats.loc["mean_r", name] - r_f) / (stats.loc["std", name] - std_f)

# Determine CAPM beta and Security Characteristic Line (SCL)
# ----------------------------------------------------------------------------------
r_m = daily_r["market"] - risk_free['Daily_Return']
for name in total_r.columns:
   r = daily_r[name] - risk_free['Daily_Return']
   model = linear_model.LinearRegression()
   model.fit(r_m.values.reshape(-1, 1), r.values.reshape(-1, 1))
   r_predict = model.predict(r_m.values.reshape(-1, 1))
   stats.loc["beta", name] = model.coef_[0, 0]
   stats.loc["alpha", name] = model.intercept_[0]
   stats.loc["Treynor", name] = (stats.loc["mean_r", name] - r_f) / stats.loc["beta", name]
   stats.loc["info", name] = stats.loc["alpha", name] / stats.loc["std", name]
   excess_r = stats.loc["mean_r", name] - r_f
   stats.loc["p_star", name] = stats.loc["std", "market"] / stats.loc["std", name] * excess_r + r_f
   stats.loc["M2", name] = stats.loc["p_star", name] - stats.loc["mean_r", "market"]

   # Plot Security Characteristic Line (SCL)
   fig, ax = plt.subplots()
   ax.scatter(r_m, r)
   ax.plot(r_m, r_predict)
   tmp_str = "SCL for the {}, beta = {:.2f}, alpha = {:.2f}%, R^2 = {:.2f}"
   ax.set_title(tmp_str.format(name, model.coef_[0, 0], model.intercept_[0], r2_score(r, r_predict)))
   plt.ylabel("Excess Daily Return (%)")
   plt.xlabel("Excess Daily Market Return (%)")
   ax.axhline(y=0.0, c='black')
   ax.axvline(x=0.0, c='black')
   ax.set_ylim(-2, 2)
   plt.grid(True)
   fig.savefig('scl_{}.png'.format(name))
   plt.close()
```

```
# Plot M2
# --------------------------------------------------------------------------------
fig, ax = plt.subplots()

# Characteristic market Line (CML)
ax.plot([std_f,  stats.loc["std",  "market"]],  [r_f,  stats.loc["mean_r",  "market"]],  label="CML",  marker="o",
c='black')

# Benchmark Characteristic Asset Line (CAL)
label = "Benchmark CAL"
ax.plot([std_f, stats.loc["std", "market"], stats.loc["std", "benchmark"]],
       [r_f, stats.loc["p_star", "benchmark"], stats.loc["mean_r", "benchmark"]], label=label, marker="d", c="blue")

# Portfolio Characteristic Asset Line (CAL)
label = "Portfolio CAL"
ax.plot([std_f, stats.loc["std", "market"], stats.loc["std", "portfolio"]],
             [r_f,  stats.loc["p_star",  "portfolio"],  stats.loc["mean_r",  "portfolio"]],  label=label,  marker="s",
c="green")

ax.set_title("M2 Plot for the Market (M2={:.2f}) and the Portfolio (M2={:.2f})".format(stats.loc["M2", "benchmark"],
                                                                                        stats.loc["M2", "portfolio"]))
plt.ylabel("Daily Return (%)")
plt.xlabel("Standard Deviation of Daily Return (%)")
lgd = ax.legend(loc='upper left', fancybox=True, shadow=True, ncol=1, fontsize=9)
fig.savefig('m2.png', bbox_extra_artists=(lgd,), bbox_inches='tight')
plt.close()

# Print stats to screen for reference and save to stats.csv
# --------------------------------------------------------------------------------
print(stats)
stats.to_csv("stats.csv")

# Plot the returns
# --------------------------------------------------------------------------------
fig, ax = plt.subplots()
label = "Benchmark (Daily Geo. Mean r={:.2f}%, std={:.2f}%, sharpe={:.2f})".format(stats.loc["mean_r", "benchmark"],
                                                                                    stats.loc["std", "benchmark"],
                                                                                    stats.loc["Sharpe", "benchmark"])
ax.plot(total_r.benchmark, label=label, marker=".")
label = "Portfolio (Daily Geo. Mean r={:.2f}%, std={:.2f}%, sharpe={:.2f})".format(stats.loc["mean_r", "portfolio"],
                                                                                    stats.loc["std", "portfolio"],
                                                                                    stats.loc["Sharpe", "portfolio"])
ax.plot(total_r.portfolio, label=label, marker=".")
ax.set_title("Benchmark ({}) and Portfolio Performance".format(settings["benchmark_name"]))
plt.ylabel("% Cumulative Returns above Risk-Free ({})".format(settings["risk_free_name"]))

x = total_r.index.tolist()[-1]
for y in [total_r.portfolio.iloc[-1], total_r.benchmark.iloc[-1]]:
      plt.annotate("{:.1f}%".format(y),  xy=(x,  y),  xytext=(-40,  10),  textcoords='offset  points',  ha='right',
va='bottom',
                bbox=dict(boxstyle='round,pad=0.5', fc='yellow', alpha=0.8),
                arrowprops=dict(arrowstyle='fancy', connectionstyle='arc3,rad=0'))

ax.set_ylim(top=8.0)
lgd = ax.legend(loc='upper left', fancybox=True, shadow=True, ncol=1, fontsize=9)
plt.grid(True)
ax.xaxis.set_major_formatter(mdates.DateFormatter("%b-%d"))
fig.autofmt_xdate()
fig.savefig('summary.png', bbox_extra_artists=(lgd,), bbox_inches='tight')
plt.close()

# Read current portfolio
# --------------------------------------------------------------------------------
current_portfolio = pd.read_csv("OpenPosition.csv", usecols=["Currency", "MarketValue"])
currency = current_portfolio.groupby("Currency").sum()
currency["Percent"] = 100.0 * currency["MarketValue"] / currency["MarketValue"].sum()
```

```
# Plot currency distribution
# ---------------------------------------------------------------------------------
ax = currency.plot.pie(y="MarketValue", title="Currency Exposures", figsize=(5, 5))
ax.legend(labels=["{}, {:.1f}%".format(n, currency.loc[n, "Percent"]) for n in currency.index])
ax.set_ylabel("")
fig = ax.get_figure()
fig.savefig('currency.png')
plt.close()
```