Group One Strategy Report: A high-frequency trading strategy on VXX triggered by SPY

Introduction:

In this project, we used the SPY and VXX as our trading symbols, and conducted research on correlations between SPY and VXX, then built our trading strategy based on the correlations.

Background:

SPY is the ticker symbol for SPDR S&P 500 Trust ETF. It is a fund that aims to track Standard & Poor's 500 index which is a market index that keeps track of the performance of the stocks of the 500 large companies in different industries listed on the U.S stock exchange. It is often used as the benchmark of the United States equity market and economy[1] as it's a well-diversified basket of common stocks in multiple industries. ETF is the abbreviation of the exchange-traded fund, which is a fund that can be traded in an exchange, just like stocks.

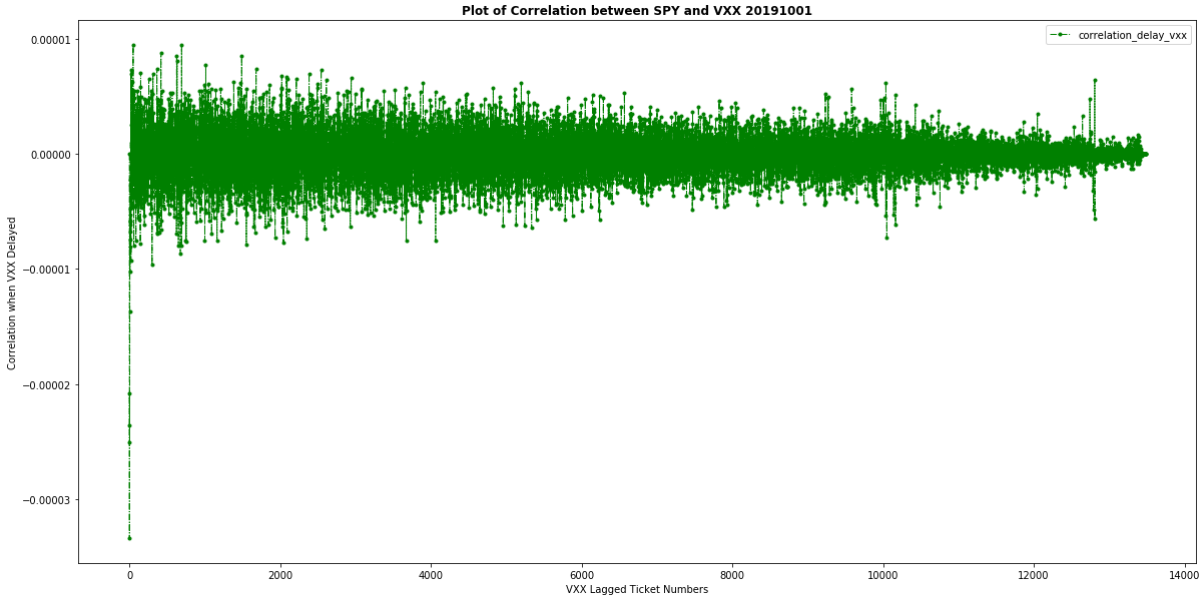
VXX is an abbreviation of the iPath S&P 500 VIX Futures, which is an exchange-traded note (ETN) designed to provide exposure to equity market volatility for investors. ETN is structured as a debt instrument, can be bought, and sold like stock [2]. VXX usually increases when the S&P 500 declines, which means they have a negative correlation in most situations, the hypothesis is supported by the graphs and data analysis in the next section of the report and is used to construct our strategy.

Since SPY and VXX track the performance of a selection of stocks and have less exposure to the risk of individual stocks, we choose VXX and SPY as the assets to study in our project. SPY and VXX are derived from the stock prices of a selection of stocks trading in the market, and they represent the "market performance" rather than performance of certain stocks in a specific industry which could be affected greatly by a single event happening to a specific industry or company.

Pre-implementation Research

At the beginning, we want to find out the leading indicator, SPY or VXX. So we lagged the VXX tickets, and to see the correlation between the price change percent of lagged VXX and non-lagged

SPY. From the following plot we can see it seems to be zero reversion, which means sometime SPY leads, sometime VXX leads.

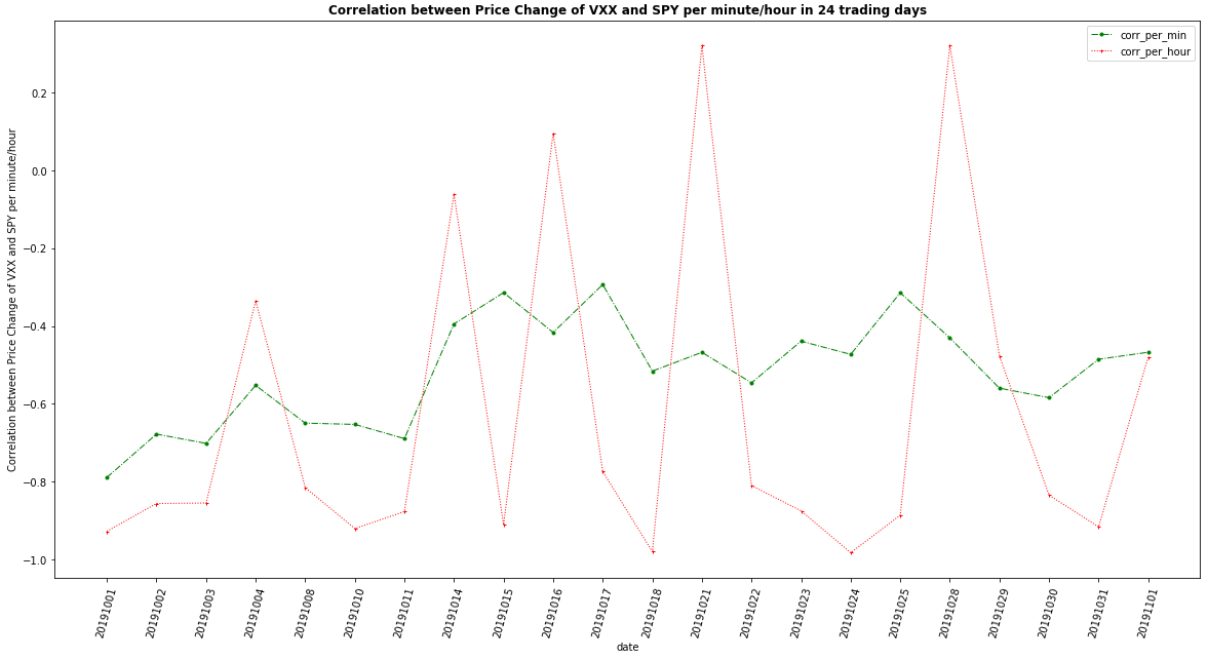


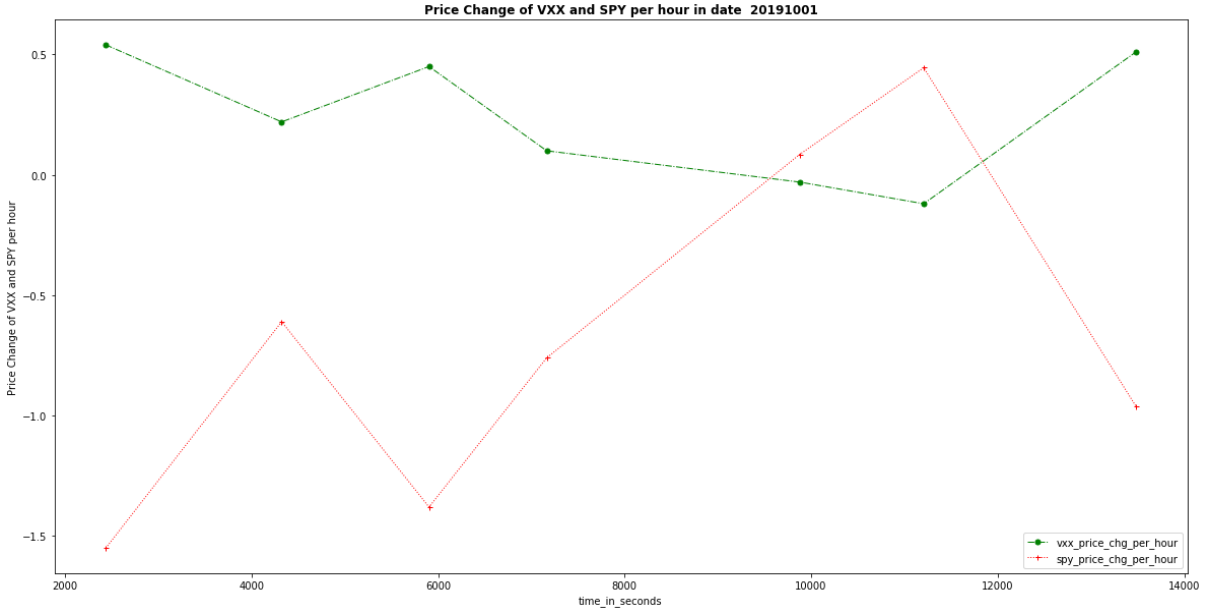
After that, we calculate the correlation correlations between the price change of the VXX and SPY per hour/minute, we found that the price change of VXX and the price change of SPY are negatively correlated.

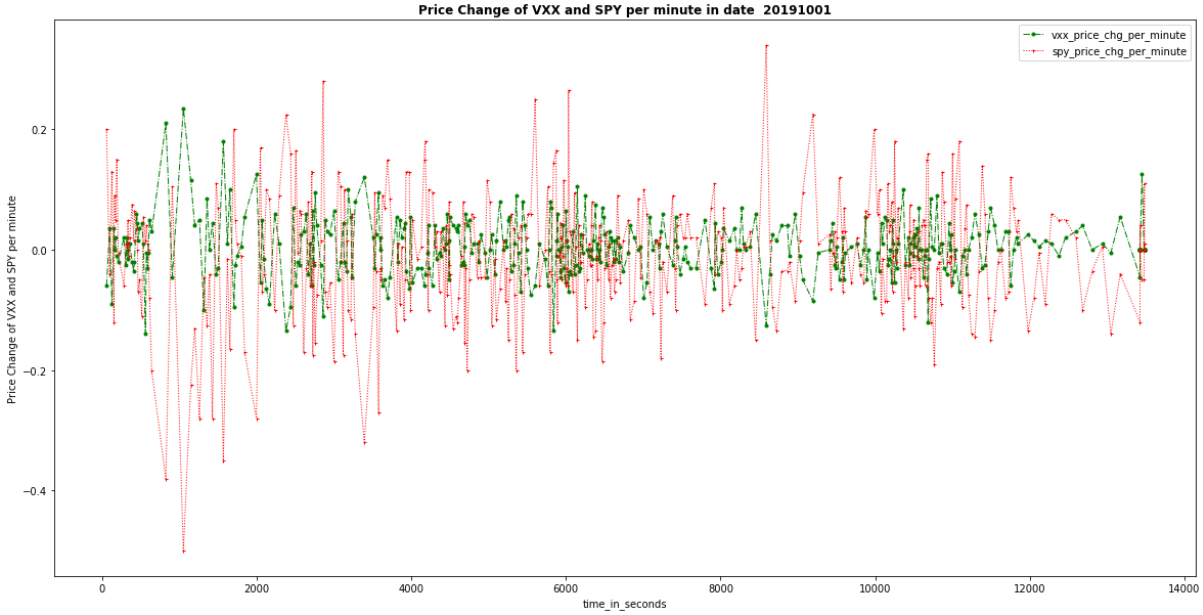
The average correlation of the price change of VXX and SPY per minute within 24 trading days is -0.5189110926715399.

The average correlation of the price change of VXX and SPY per hour within 24 trading days is -0.6290116674372686.

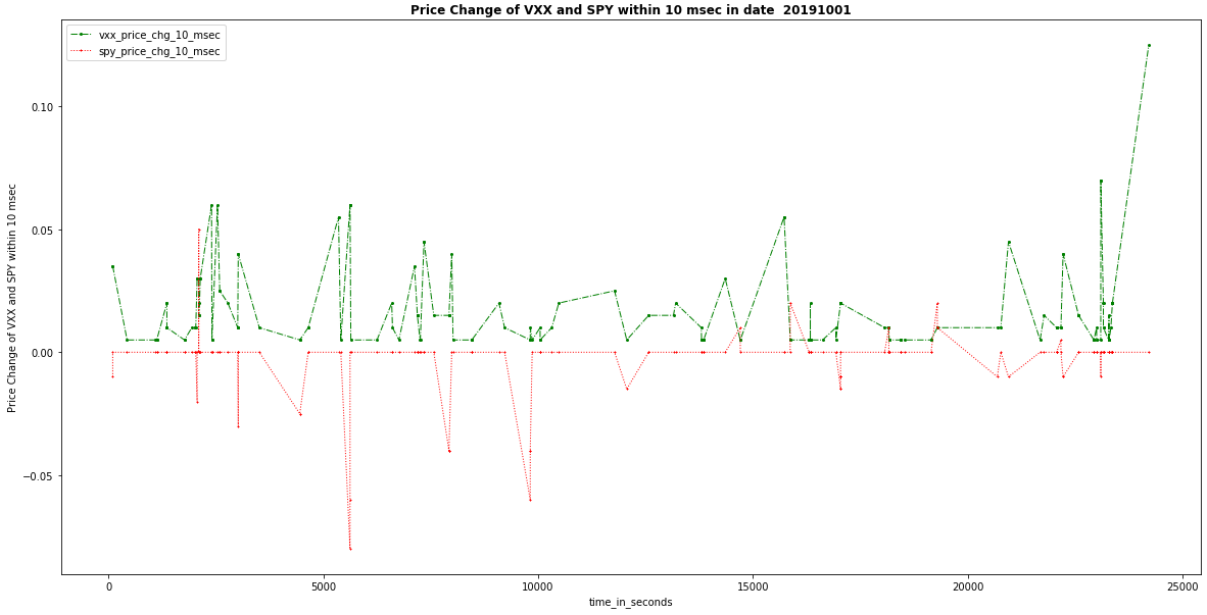
We can also see it from the following plots.


Correlation between Price Change of VXX and SPY per minute/hour in 24 trading days
corr_per_min
corr_per_hour
Correlation between Price Change of VXX and SPY per minute/hour
0.2
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
20191001
20191002
20191003
20191004
20191008
20191010
20191011
20191014
20191015
20191016
20191017
20191018
20191021
20191022
20191023
20191024
20191025
20191028
20191029
20191030
20191031
20191101
date


Price Change of VXX and SPY per hour in date 20191001
Price Change of VXX and SPY per hour
0.5
0.0
−0.5
−1.0
−1.5
2000
4000
6000
8000
10000
12000
14000
time_in_seconds
vxx_price_chg_per_hour
spy_price_chg_per_hour



The following plot is the correlation between price change of VXX and SPY within 10 milliseconds. Here we dropped all the ticks without price change within 10 milliseconds, because in many situations, SPY and VXX did not change, and it would lower the actual correlation.



So we build the strategy based on the negative correlations between the price change of VXX and SPY. When SPY price goes up and we do not have VXX we will short VXX, or clear VXX position if we have VXX positions; If SPY price goes down, we will buy VXX.

C++ implementation (non-optimized version)

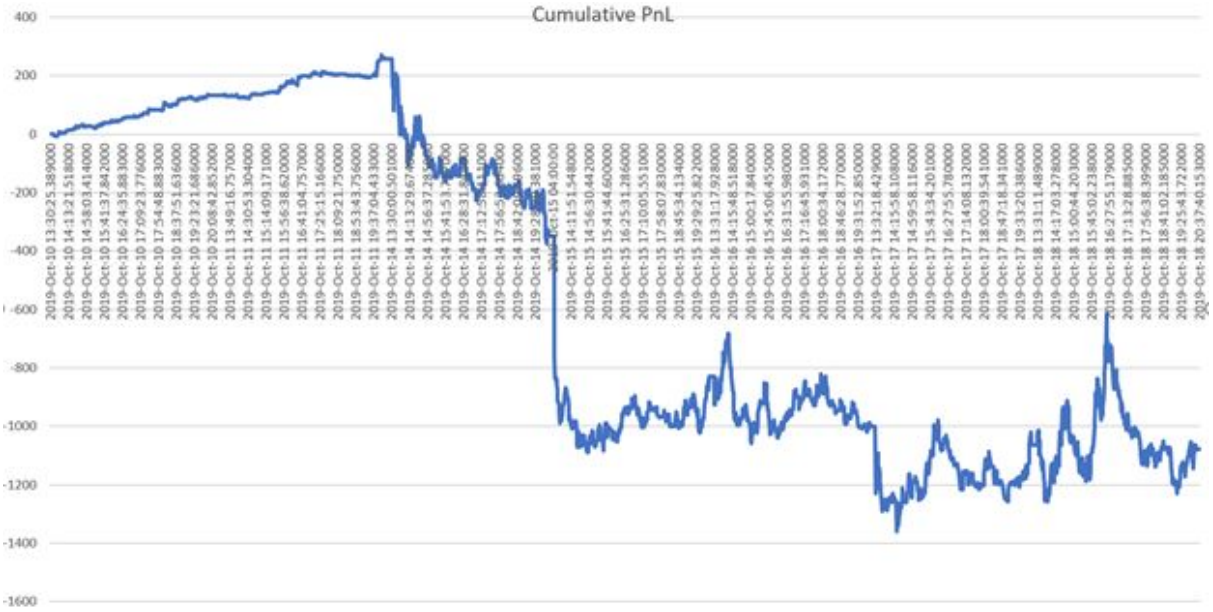
In order for this strategy to work, we need to use two double-ended queues: "current_50_trades" and "lagged_50_trades" to store the trade prices to compute the moving average. Our group decided to use a double-ended queue because it can be expanded or contracted on both its front and back so that we can freely add or remove the trade prices. First, we assign "SPY" to "m_instrumentX" and "VXX" to "m_instrumentY". In order to compute the moving average of current 50 trades prices, we need to remove the oldest trade price from the "current_50_trades" queue and add the least trade price to the queue. Adding or removing trade prices dynamically enables us to always use the latest trade prices to compute the moving average of the current 50 trades. The "lagged_50_trades" queue stores the lagged 50 trade prices. In other words, it does not contain the most current trade prices. Now, the trade prices stored in "current_50_trades" and "lagged_50_trades" are transferred to the variables "ma_first" and "ma_second", respectively" to compute the moving averages. Importantly, we need to wait until we have at least 51 trades to begin our strategy; otherwise, it is impossible for us to compute the average of current 50 trade prices or last 50 trade prices. Furthermore, we need a threshold for our strategy to work, which is determined by using the median of the absolute values of the last 50 SPY ticks moving average price changes among the beginning 10 trade days. The threshold value itself obtained via our statistical analysis is 0.00025. Finally, our strategy is simply defined in the following illustration:

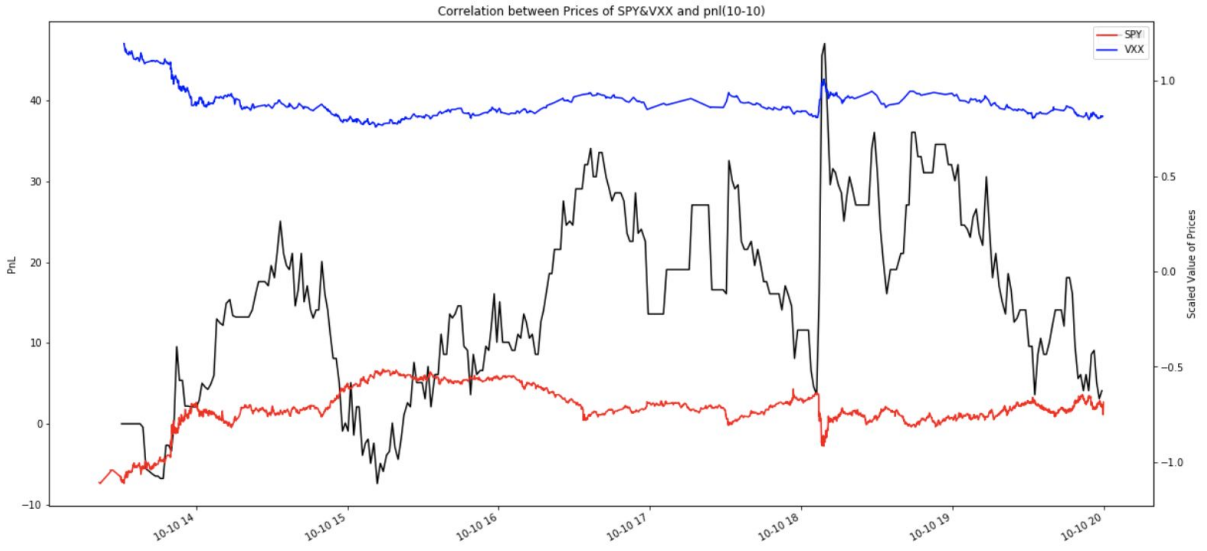
$$\begin{cases} buy\ 100\ VXX & if\ (ma\_first - ma\_second) \geq 0.00025\ and\ hold\_position = 0 \\ sell\ 100\ VXX & if\ (ma\_first - ma\_second) < -0.00025\ and\ hold\_position = 1 \end{cases},$$

where hold position is a boolean which tracks if we have current position.

We update “trade_count” every time and transfer the data stored in “current_50_trades” to “lagged_50_trades” after we execute every order (buy/sell 100 VXX) so that we have the latest information to continue our strategy. In addition, we also set an integer *trade_num* which keeps track of how many trades we make every day and an integer *max_trade_number* to limit the maximum trade number we can execute every day. This is designed mainly for debugging. During the backtest, “max_trade_number” is set to a very large integer to remove this limitation.

Backtesting Results Analysis & Optimization

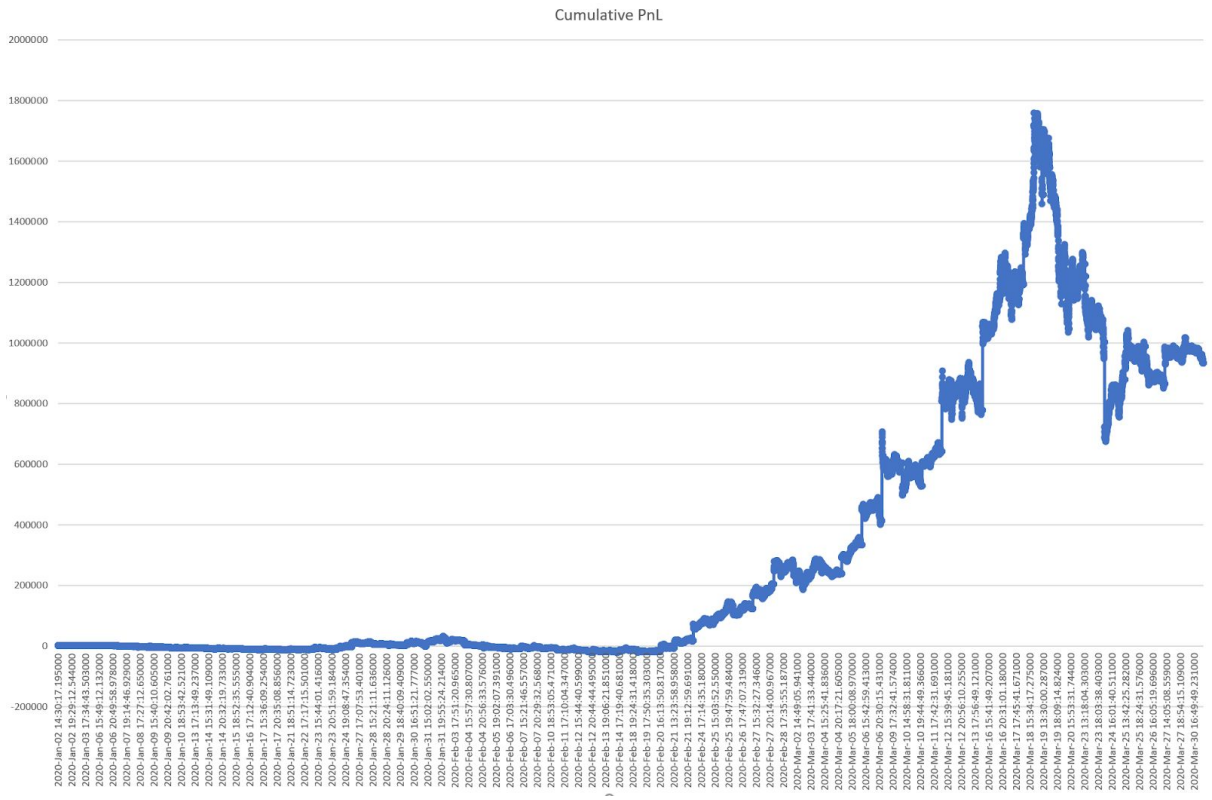




Examining the backtesting results of our initial strategy, we realized that our threshold is too small and therefore we increased our threshold from 0.00025 to 0.00040. Indeed, we observed better results after the increase of the threshold. furthermore, we found that our strategy starts losing money on Oct 14 and there is an abnormal vertical decline on our PNL graph. By looking further into the event that caused us to lose money, we discovered a bug in our initial strategy. At the beginning of certain trading days, there are a number of SPY trade updates coming in, while there are no VXX trades. This caused our strategy to keep submitting buy orders at the first minute of the trading day. All the orders get filled later and are not closed until the end of our backtesting period. To solve this problem, we added a piece of code that prevents our strategy from sending orders in the first minutes of each trading day. We also realized that there is a significant intra-day risk and we should not hold our position overnight. As a result, we added a block of code that stops our trading activity and close our position when we are in the last minutes of each trading day. Additionally, we utilized OnResetStrategyState() to reset our parameters at the beginning of each trading day.

After we fixed the problem, our strategy can profit during the backtest period (2019-10-10 to 2019-10-30). Both SPY and VXX fluctuated during this period, but our strategy is able to generate stable profit. By the end of the backtest period, our strategy made 2383 trades and generated 912.76 USD profit. The plot of the first trading day is included in the report while the rest of the PNL graph is included in the gitlab repo.

2020 Backtest Result



In order to test the profitability of our project during an abnormal period, namely during the 2020 pandemic, we backtested our strategy in a period starting from Jan 1, 2020 to Mar 30, 2020. Our strategy behaved normally until late February. It generates some profit at the beginning of January and then starts losing money later in that month. The reason for that is probably at the end of January the market starts rallying, with S&P 500 going up to its historical highest point. As a result, VXX remained low during that period while SPY went up rapidly and we got hurt by the SPY/VXX level change discrepancy. However, when it came to the end of February, the market started to drop due to COVID-19 concerns and our profit skyrocketed. From the end of February to the end of March, the market dropped rapidly, with VXX rallying. The threshold of our strategy is fixed, but during this period the volatility of the market increased by a lot. Consequently, we are making a lot more trades every day. Furthermore, our strategy only allows us to take a long position, so we took full advantage of the VXX price increase.

ToubleShooting:

Throughout the project, we experienced following problems:

1. We sent orders to wrong exchanges. During the development of our strategy, we sent orders to COBE and NASDAQ instead of IEX. Consequently, none of the order got executed.
2. We did not update the compiled strategy into the correct path.
3. We changed the username to our group name instead of dlariviere, so the backtesting engine will not take in our strategy because the author is not in the database
4. At the beginning of certain trading days, there are a number of SPY trade updates coming in, while there are no VXX trades. This caused our strategy to keep submitting buy orders at the first minute of the trading day.

Individual Contributions / Lessons Learned and Next Steps:

Guancheng Guo:

(28 commits and one closed issue) In this project, I first constructed a simple pair trading strategy and run back testing on the strategy. Since this strategy was rejected later, I discussed with our teammates and came up together with the current strategy implemented by our group. The basic structure of our C++ implementation was completed by my teammate Lingyu He. I brainstormed the data structure we needed to use in our strategy in order to compute the moving average: two double-ended queues. After the majority of our C++ codes were written, I debugged the code and corrected several mistakes so that the back testing can be run without issues. I added comments into the codes of our C++ implementation, and updated our strategy description in the "GroupOneStrategy" folder. Furthermore, I also maintained our repository and updated key files including back testing reports and readme file. In addition, I cooperated with Lingyu He to optimize our C++ code implementation. Finally, I made significant contributions to the completion of our group report and the readme file.

This project is really interesting to me. I learned the importance of SPY and VXX in the world of finance. In fact, both of them are exchange traded funds (EFTs) that are actively traded in the financial market. However, to my surprise, based on our correlation analysis, it is not always the case that SPY leads VXX or VXX leads SPY because such relationships change frequently, especially

during the period of the pandemic. Furthermore, I also have great fun building a workable strategy and examining the P&L at a result of running the back testing on the strategy. In the future, I hope that I can build a strategy that is robust enough to generate appearing P&L diagrams not matter what the actual market situations are.

For the next steps, I proposed the following:

- Performed more correlation analyses to determine the changing relationship between SPY and VXX in a real time analysis
- Dynamically adjust the threshold based on the relationship between SPY and VXX
- Perform stress tests under different kinds of scenarios to furtherly examine our model performances.

Lingyu He

Contribution: (23 commits and 4 closed issues) In this final project, I mainly wrote the C++ code for our strategy and run back testing on the Virtual Machine. I created the "GroupOneStrategy" directory to store all the code files for our strategy and "backtesting-cra-exports" to store the exported csv files as a result of running the backtesting. Furthermore, after reviewing our backtesting results, I optimized our C++ code implementation. For example, I increased our threshold from 0.00025 to 0.0004 so that our strategy could have better performance. I also modified the code to make sure that we close out our position prior to the end of the trading date to eliminate potential risk during market closure. In addition, I added comments to explain the updates I made to the code. Importantly, during this project, I also collaborated with my teammate Guancheng Guo to fix errors and bugs in our C++ code implementation.

Lessons Learned and next steps:

From this project, I learned how to design, implement and back test a high-frequency trading strategy with the help of industrial-grade software. Although our trading strategy is very naive, it helps me to understand the dynamics of a trading strategy. The back test and analysis of our strategy forced me to

look at the nitty-gritty of the market microstructure, which is completely different from retail trading at a large time scale. The way that the order book is constructed, how individual orders are sent and processed becomes very important and can cause unexpected bugs and a significant loss of money. I also learned how to work with a team to maintain an online repo and this is very helpful to me.

As for the next steps, I would try to implement more strategies, as I found out that a good way to learn is by doing. Also, the analysis part of our strategy is relatively weak, and it is something that we want to improve. If we had more time, we would choose a solid hypothesis and support it with robust graphs.

Jialing Zhu

Contribution: (10 commits and 4 closed issues). In this project, I am mainly responsible for studying the correlation between SPY and VXX.

First, I cleaned the dirty data. Because the number of SPY ticks is much larger than the VXX, so I merged these two datasets first and filled the missing value with the most recent price. And transformed the 'date' from string type to integer. After that, we could calculate the price percent change for VXX and SPY per hour/minute.

Secondly, I used python to calculate the correlation between the price change of SPY and price change of VXX, and proved the negative correlation between them. I worked with Jiaqi Su to try to find out which is the leading factor, SPY or VXX. The conclusion is that sometimes SPY leads, sometimes VXX leads.

What's more, I analyzed the backtest results with all team members together to see why it profited or lost. And thought about how to optimize the codes.

Also, I was in charge of the presentation, and Pre-implementation analysis part in the final report.

Lessons Learned and next steps:

I learned various types of trading ticks and different trading strategies, and know about how to make an order book. And I also learned about how market news affects the securities.

What's more, it is a great experience to use the RCM trading system to build a trading strategy, do the backtests, and conduct the post-backtesting analysis.

For the next steps, I think we should use dynamic threshold instead of a fixed threshold. Also, we could add more securities into our strategy as indicators. It is not robust to use only one security as the indicator.

Jiaqi Su:

Contribution: (3 commits and 2 closed issues)During this project, my main contributions lie in analysis on correlation between SPY and VXX prices. To be more specific, firstly I import data from txt files to a python data frame. Secondly, I did data preprocessing work including transferring time information into datetime type values, scaling prices of SPY and VXX, and generating feature 'percentage of changes' which is further used to capture large movement of SPY and VXX prices. Thirdly, I draw plots of SPY and VXX prices movement after large movements of VXX prices (some of the samples can be seen in gitlab). Finally, depending on pnl information obtained by Lingyu He, I further draw plots of scaled VXX & SPY prices and pnl on each day to show their correlations (plots have been uploaded in the 'plot' folder in gitlab).

From my cooperation with Jialing, I learned that VXX and SPY have really high correlation. However, both VXX and SPY do not always lead the other. In other words, VXX sometimes have price changes after the price of SPY changes. However, in some circumstances, the price of SPY will change first.

For the next steps, we can try more statistical methods to find their correlation. For example, we can make use of knowledge about cointegration. Besides, from what we have observed from plots, it is worthy to find correlations between SPY and VXX at each day separately.

From the FIN566 lesson, I learned a lot about how to design trading strategies and how to do backtesting to measure its performance. It is during both course time and the period of finishing this project with all my teammates. Besides, I learned how to use virtual machine and a lot about linux. All knowledge is really valuable.

References and Citations:

1. https://www.investopedia.com/articles/investing/122215/spy-spdr-sp-500-trust-etf.asp

2.https://www.investopedia.com/articles/investing/080715/etf-analysis-ipath-sp-500-vix-futures-vxx.asp

3. https://www.etf.com/sections/features-and-news/why-spy-king-liquidity?nopaging=1

4. https://en.wikipedia.org/wiki/VIX