High Frequency Statistical Arbitrage Model: Using a band strategy and co-integration to capture alpha in pairs trading

Tyler Coleman, Cedrick Argueta, Vidushi Singhi,
Luisa Bouneder, and Dottie Jones

Stanford University
Spring 2019

Abstract. The goal of this project is to develop a High Frequency Statistical Arbitrage trading approach. Using data from Thesys Technologies, we expanded upon classic strategies in the space to implement a band strategy that leverages co-integration in order to capture alpha. We chose pairs from within the NASDAQ 100 using K-means clustering and determined our time scale to be seconds. We built unique and optimal models to model the residuals for each pair in order to predict and trade on alpha-generating change in difference in price. Our strategy has a lot of potential, though there are several aspects to be further developed. A discussion of the results, strengths, weaknesses, and future steps are included.

Keywords: High frequency trading, Pairs trading, Statistical arbitrage, Band strategy, Co-integration, Model selection

1 Introduction

1.1 Our strategy

Classic Statistical Arbitrage pairs strategy is when you make predictions based on the linear combination of two assets. The underlying assumption is that the two assets co-move and that when they do separate, the separation is temporary and the process will mean-revert. The alpha to be captured is in properly modeling the residuals of the pair to capitalize on the temporary deviations from the mean. There are many different strategies within the Statistical Arbitrage space, and so a lot of room to develop an alpha capture strategy. In classic pairs trading, the strategy consists of going long position in one security and a short position in another security in a predetermined ratio[1]. Expanding upon this classic approach, we leveraged a co-integration factor and a band strategy to determine when and when not to trade.

Co-integration is a type of linear combination of time series variables. The co-integration factor is defined to be

$$\epsilon_t = AS_t^{(StockA)} + BS_t^{(StockB)}$$

where A and B are estimated from the data and Stock A and Stock B are the respective two companies that make up the pair[2]. As described above, the co-integration factor follows a mean-reverting process and there are numerous strategies built around how to capture the alpha in this process. For our strategy, we chose to use bands to determine when to buy and sell. The simplest explanation of the strategy is as follows: place bands one standard deviation above and below the mean-reverting level, which is zero. When the co-integration factor hits the upper band, sell the stock that is overvalued and buy the stock that is undervalued. The same is true when the factor hits the lower band. This idea is rooted in the mean-reverting nature of the co-integration factor. Our strategy builds on this rudimentary idea in order to capture alpha, and we also execute this strategy at the high frequency level. Thus, our model combines High Frequency Trading (HFT) with Statistical Arbitrage strategies. High Frequency Trading (HFT) is a type of algorithmic trading done at very high frequencies, for example down to the microsecond. We

chose to combine a high frequency model with our statistical arbitrage band strategy in hopes to capture as many movements in the co-integration factor as possible. We settled on modeling and trading at the one second frequency.

Figure 1 demonstrates a rough implementation of the strategy described above. The top graph shows the evolution of the price of two companies, VRSK and VIA. The second graph illustrates a thirty-second fragment of information where we predict the difference in price using our model, which corresponds to $t = 10000$ in the first graph. The blue line is the co-integration factor. In this example, we would sell VIA at time zero since it is overvalued and buy VRSK, as it is undervalued. This fact is demonstrated by the blue line falling outside of the green band. We would then close the position for each stock once the blue line reaches the inner band, or after thirty seconds (since that is the prediction window for our strategy).

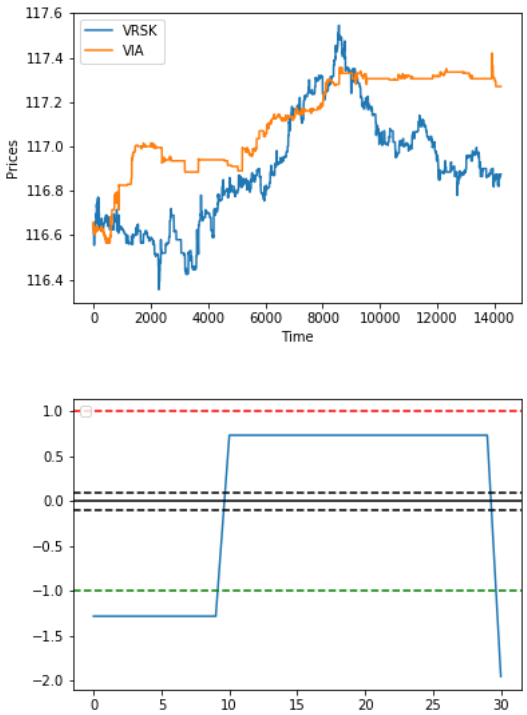

Fig. 1: Sample path of trading using co-integration factor

1.2 Data used

Our data came from Thesys technologies, which is a platform containing high frequency data as well as a built in simulator. Since our timescale is by the second, we had plenty of data to work with. For our data set, we pulled an entire trading day's worth of data to the second (4 am - 8 pm) on January 28th, 2019. In total, we had over 50,000 data points and had we more time and computational power, we would have pulled more. We split this data into train and test sets by keeping the first 75% as train data and the remaining 25% as test data. We also pulled an additional test set from 9:30 am - 12 pm on the following day, January 29th, 2019.

Thesys provided us with the following information: bid price, ask price, the shares quoted at bid/ask price (`b/asize`), the number of orders at bid price (`b/anum`), mid-price, weighted mid-price, volume, the dollars traded until queried time (`notional`), last price, last size, last SRO, and last time. We used all of these features except for last SRO and last time in our full model.

For the pre-market hours (4 am - 9:30 am) and the post-market hours (4 pm - 8 pm) there tended to be more missing data than during the normal trading hours. Despite some missing values, there was still plenty of data to work with in an understandable and usable fashion. Thanks to Thesys, we did not find any problems with the data itself.

We also made use of the Thesys simulator to test how our strategy would work in "real time". For our simulation, we chose to trade on January 29th, 30th, and 31st from 10 am to 12 pm. While we ran into some issues with the simulator, it was helpful in gauging how our model would perform if our trading went live.

1.3 Our approach

To build our strategy, we first determined what our universe was going to be. We chose to look at the NASDAQ 100 companies and make pairs within this index since it contains the largest stocks from various major industry groups. Then, once we had our data from Thesys, we made sure to remove the market effect from our returns by regressing each stock's return on the QQQ (NASDAQ 100 ticker) return. Once we had our pairs, features, and difference in prices, we built unique models for each pairing, using each respective model to predict our target: the difference in mid-price in thirty seconds. We chose to predict thirty seconds in advance for computational reasons. Once we back-tested our models, we were able to calculate the co-integration factor, determine the bands, and subsequently simulate and evaluate our strategy using second by second company data from Thesys.

2 Choosing the Universe

2.1 Selecting the companies

In order to deploy our statistical arbitrage strategy we had to find a set of correlated pairs. As mentioned before we are working with the NASDAQ 100 stock exchange, meaning that we had 103 companies and thus potentially 5253 pairs. Since our goal was to understand the company selection process and eventually apply our strategy on real pairs and see how it performs we decide to select only a few subset of pairs to work with. The goal of the overall model was to predict the difference in return residuals for each pair selected. Thereby it was natural for us to think about correlation between pairs using their return residuals over time.

There can be many ways to find correlation between companies. To find the pairs we decided to reduce the data frame and work with, for each company, a residual vector v_A (for company A) where v_A is a vector composed of all the return residuals over time:

$$v_A = \begin{pmatrix} r_0 & r_1 & ... & r_{n-2} & r_{n-1} \end{pmatrix}$$

where $\forall i \in [0, n-1], r_i$ is the return residual at time i and n is the length of the dataset.

There are many ways to define correlation between companies. One way could be to manually look at every company in the stock exchange and think about them in terms of industry, product, size, business model... This process can give good pairs in term of market type correlation and business correlation however it does not guarantee that the actual stocks of these two companies move together and more importantly this is a long, tedious, non-automated process.
Another naive - but more algorithmic - way of finding pairs is to directly take the residual vectors v presented above and compute a correlation matrix.

2.2 Clustering the companies

We decided to explore different ways of finding correlation between stocks by clustering the residuals (see [3]). We tested two clustering methods. Both of them are based on the K-means

clustering algorithm. This algorithm consists in partitioning on set of n vectors into k distinct clusters. The clusters are formed with the aim of minimizing the within-cluster sum of squares where μ_i is the mean of the points in S_i:

$$\arg\min_{S} \sum_{i=1}^{k} \sum_{v \in S_i} ||v - \mu_i||^2$$

We chose to use this algorithm because it is widely used, classic clustering algorithm that works well and is easily interpreted.

Spherical K-means: our first idea was to run a K-means clustering on the set of residual vectors $(v_c)_{c \in NASDAQ-100}$ and visualize the clusters formed. The "classic" K-means algorithm relies on the Euclidean distance to iteratively produce the final clusters. However each residual vector has a high dimension (more than 20000), so the Euclidean distances are converging and the standard algorithm (Lloyd's algorithm) which is based on updating the centroids by minimizing the Euclidean distance performs poorly. As shown by Figure 2 the algorithm cannot distinguish clusters because of the convergence of the Euclidean distance and ends up putting almost all the companies in the same cluster.

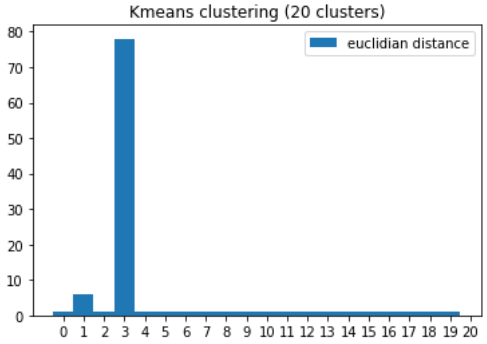

Fig. 2: K-means clustering with $k = 20$ clusters

Of course this was a big issue for us since our goal was to first cluster the companies and then build pairs from that. To overcome this difficulty we decided to use a slightly different update algorithm to deal with this convergence in high-dimension space. We used a Spherical K-means algorithm. This algorithm works similarly to the "classic" Euclidean K-means except that the during the update algorithm the centroids are projected onto the unit sphere in order to be able to compute distances in that space. This enables to change the distance used and bypass the high dimensional problem.

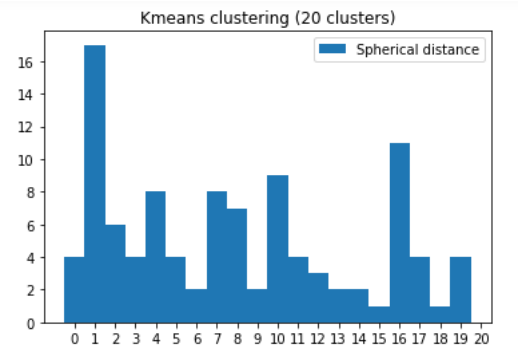

Fig. 3: Spherical K-means clustering with $k = 20$ clusters

We ran the Spherical K-means on our set of residual vectors $(v_c)_{c \in NASDAQ100}$ and found the results shown by Figure 3. We can clearly see that this time the clustering algorithm did work and has successfully divided the vector set into smaller subsets. Once we had the clustering for the NASDAQ 100 companies, we were able to build pairs by associating within-cluster companies.

K-means on small-dimension vectors: as discussed in the previous paragraph, the fact that the residual vector lives in a high-dimension state was a problem for the clustering. Even if we managed to overcome that by modifying the update algorithm we also thought about other ways of defining and finding correlations between companies. We had thus another idea to build pairs. The idea is pretty simple: given the fact that we are struggling with high-dimension vectors, why not reducing the size of the vectors to work in small-dimension space. To reduce the residual vectors we just decided to compute the K-means clustering at every timestamp available. So for each timestamp $i \in [0, n-1]$ we ran a clustering using the residual vector $v_c = (r_i)$ for $c \in NASDAQ100$. Thereby, at the end of this process we end up with a new feature vector K_c for each company c such that:

$$K_c = \begin{bmatrix} k_0 & k_1 & ... & k_{n-2} & k_{n-1} \end{bmatrix}$$

where $\forall i \in [0, n-1], k_i$ is the cluster to which the company c belongs at time i.

Now that we have this new set of feature vectors $(K_c)_{c \in NASDAQ100}$ we compute the correlation matrix to determine the most correlated pairs according to these features. After visualizing different values of the number of clusters k to create when performing the K-means at each timestamp for a few subset of the timestamps we had we decided to use $k = 10$ clusters at each iteration. We found then the correlation matrix presented in Figure 4. We also tried to see whether adding lags before running the clusters was helpful to partition the data. For example with a lag $l = 2$ we would run at each time $i \in [2, n-1]$ the K-means on the vectors:

$$v_c = \begin{bmatrix} r_{i-2} & r_{i-1} & r_i \end{bmatrix}$$

However after visualizing some plots of the clusters as well as the correlation matrix we found that it didn't change the final results and the order of the pairs in terms of correlation values was almost the same (at least for the top of the list). So we chose to not add lag to our residual vector.

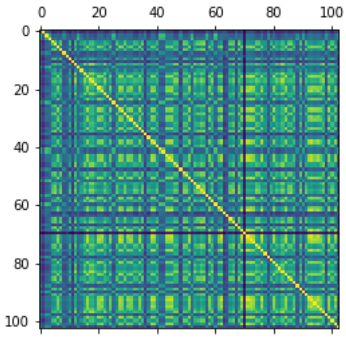

Fig. 4: Correlation matrix for vector of clusters

2.3 Universe selected

We explored two different methods for clustering the return residuals for the companies of NASDAQ 100. They gave us two different types of results:

1. A clustering of the 103 companies for the first method

2. All the possible pairs formed by the 103 companies and their correlation according to the cluster vectors.

We chose to focus on the results of the second method because we were curious to check the accuracy of this method. Since this method is more atypical, we wanted to see whether we can work and find interesting models using the pairs coming from it. We put a threshold on the correlation value, $p = 0.92$ and with that threshold we found 19 pairs above that threshold that we ended up using as our portfolio. We built models for each of these pairs (see next section), but were only able to execute trades on seven of them (see Results section). We acknowledge that not *all* of our pairs are in the same industry, which is a classic approach in choosing pairs for Statistical Arbitrage trading, but we are confident in our numerous checks that these are not spurious correlations. Had we more time, we would expand our portfolio to include more pairs to further diversify our portfolio and increase our chance at making profit. We also would perhaps explore companies within the SP500 rather than just the NASDAQ 100.

(**BKNG**, **NWSA**) (**FOX**, **NWS**) (**HSIC**, **VIA**) (**IDXX**, **NWS**)
(**IDXX**, **VIA**) (**MELI**, **NWS**) (**NTES**, **NWS**) (**NTES**, **VIA**)
(**NWS**, **PDD**) (**NWS**, **ULTA**) (**NWS**, **VIA**) (**NWS**, **VRSK**)
(**PDD**, **VIA**) (**VIA**, **VRSK**)

3 Building the Model for Co-integration

A vital step for running co-integration is properly modeling the residuals. When first exploring the data, we ran a simple regression of the mid-price of company A on the mid-price of B. But once our pairs were determined, we were able to build a more sophisticated and unique model for each pair.

3.1 The full model

The first thing we did was run a univariate regression of all the features Thesys provided us with on the target. We found that `volume` and `wmid` (weighted mid-price) had the strongest univariate predictive power of the target, and so engineering additional features of volume and wmid at lags of 15- and 30-seconds. Thus, our full model contained the full set of features from Thesys for each company in the pair, excluding `last_SRO` and `last_time`, plus four engineered lagged features, totaling thirty-two features. To choose the best and simplest version of the model for each pair, we performed four methods of model selection: LASSO, forward stepwise, backward stepwise, and stepwise in both directions.

3.2 Model selection

For each pair, we ran the four methods mentioned above and calculated the MSE for each chosen model on a validation set. For the two methods that output the smallest MSE, we took their corresponding feature sets and tested on a separate test set to see which performed better. Whichever of those two had the lowest MSE and mean absolute error (MAE) and the highest R^2 score we used as our final model for that pair. If we were to continue building out this strategy, we would train and test these models on more data as well as update them every so often to account for any potentially contributing exogenous factors.

As illustrated by Figure 5, the model does an excellent job of accounting for the variance in the residuals, in this case demonstrated by the pairing of Booking Holdings Inc. and News Corp Class A. The MSE and MAE are small and the R^2 is very high, and the autocorrelation plot in Figure 6 demonstrates the efficiency of the model in capturing noise since the autocorrelation of the residuals resembles that of white noise.

```
BKNG NWSA
Mean square error: 1.143
R2: 0.963
Mean abs error: 0.706
```

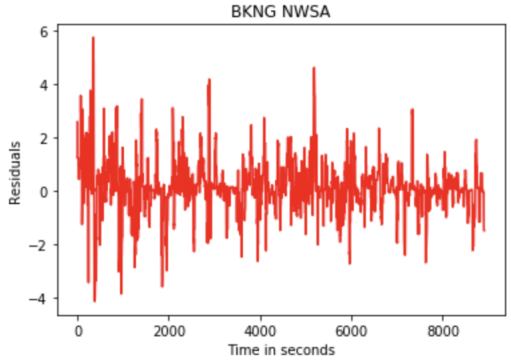

Fig. 5: Test results for BKNG-NWSA residuals model

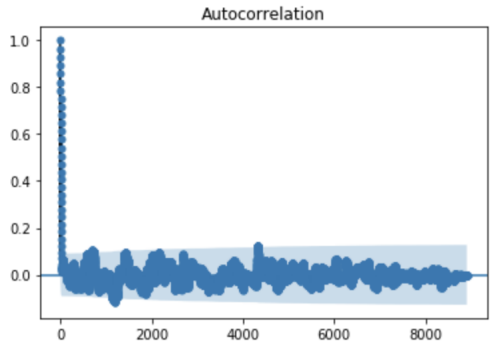

Fig. 6: Autocorrelation of BKNG-NWSA residuals

The statistical model used is a multivariate linear regression and we got similar results for the majority of our pairs. We considered experimenting with other model techniques, but felt there was no need to over-complicate what was working, and our results demonstrated that a linear fit worked well. These results are very promising since the more accurately the movement in the residuals is predicted, the more accurately we can track movements of the co-integration factor for trading. For each pair, the method that produced the optimal model varied; for some LASSO chose the optimal model and for others, one of the stepwise methods. We felt that customizing the feature set to each pair was essential to avoid under- or over-fitting the data for any of the pairs.

The size of the models also varied. For example, some pairs had as few as six features and others had as many as eighteen features. However, we always chose the simplest model that provided the best results to minimize the chance of overfitting. Another interesting observation is that for certain pairs, one company tended to dominate the model. For example, in the BKNG-NWSA example featured, seven of the eight features were from BKNG and only one - weighted mid-price lagged at 15 seconds - was used from NWSA. One reason for this observation could be that there is such a high correlation in the movement between the two companies that using BKNG features is enough to predict changes.

Once the models were built and cross-validated on unseen data, we used them to build the co-integration factor using the features selected and second by second company data within Thesys for our prediction of residuals. The next section describes this implementation process and the simulation of our strategy.

4 Implementing the Strategy

4.1 Thesys Simulator

Implementation of the strategy depended on the Thesys Technologies Education Simulator. The simulator is a modular, robust system that allows very fine control over a trading strategy. To run a simulation, a trading strategy is implemented and passed to the simulator, which runs through a predetermined time period with the trading strategy. The strategy may execute actions at various times or events through the use of callbacks, i.e., an order may be placed at the end of every second, when trading volume for a certain stock reaches a certain point, etc. The simulator's primary interface is through a Jupyter notebook hosted on Thesys' internal servers. Because of this, it depends on a constant connection to the Thesys servers.

Unfortunately, there were several challenges when using the simulator. The primary challenge was the lack of documentation for the functions that the simulator provided. Because of this, we were unable to take full advantage of the simulator because of its black box internals. Only through trial and error were we able to hack together a strategy that would be used by the simulator. Because of the power of Python's live object introspection, we were able to determine which functions were useful to our strategy.

An additional challenge was the poor up-time of the Thesys servers. More precisely, the servers hosting the simulator often failed silently when resources were requested, leaving us with access to the Jupyter notebooks with our code but not the ability to run the code. We are unsure of the cause of the issue, and multiple emails to Thesys education support only resulted in restarting the servers. This issue was only prevalent in the few days leading up to our final presentation, severely impacting our ability to simulate our strategy.

Like real trading data, the simulator often was missing data for certain fields. This could be circumvented through the use of some type of interpolation in the corresponding field, but in the interest of saving time we chose our stocks according to the ones that we could run our full tests on without interpolating data.

4.2 Our strategy in code

Our strategy depends on robust models of residual returns generated as above. The algorithm is described in 1.

Algorithm 1: Pairs Trading with a Band Strategy

```
for each pair of stocks do
    Begin sim with $10,000
    for every sec in duration of simulation do
        if 60 seconds elapsed then
            continue
        end
        Train model online
        Predict residual for next 30 sec
        if residual lies outside band then
            Execute trade
        end
    end
end
```

We perform trades every second according to this strategy. Notably, we wait 60 seconds before performing trades to allow our model to accumulate enough data for training. The model is

training online, i.e., as more data is received, the model is retrained. The band is the factor defining the sensitivity of our model to deviations in residual returns. It was empirically chosen to be 4 standard deviations from the mean residual return.

Several limitations are in place in our trading strategy. The largest is that we are currently limited to trades with a quantity of 1, largely due to the time constraints placed on us by the downtime of the Thesys servers. A simple extension would be to trade in quantities of equivalent price, i.e., trading 10 of a $10 stock with 1 of a $100 stock. More complex strategies have not been explored at this time. We are additionally limited by buying and selling at mid prices as opposed to buy and ask prices. With enough time we would have implemented our strategy in line with buy and ask prices, but again downtime on our simulator's servers prevented this from happening. Beginning with $10,000 dollars for each pair of stocks is not a hard limit, and we may modify the starting value for each pair at will.

4.3 Implementation details

The strategy was implemented in Python, with all mathematical modeling and visualization done with the SciPy stack. The features dictated by the tests above are imported into the simulator and used to define the linear regression model's features for every pair of stocks. Buy and sell orders are tracked manually as opposed to through the simulator, as it allows finer control over values and circumvents issues with the lack of simulator documentation. The simulator itself only allows for trades within one day, so a wrapper for the simulator was created that instantiates new simulator instances for each day that we wish to run simulations. Some features are not available directly from Thesys ticker data, and must be computed at run time. For example, we maintain a buffer of previous mid-price values so that we may compute VWAP as needed. Lags of features were created in the same way, with buffers logging previous values. This is why we must wait 60 seconds for data to accumulate, as we have some features that are lagged 30 seconds and are predicting 30 seconds in the future.

5 Results and Discussion

Pair of Stocks from NASDAQ 100	Profit (In $)
BKNG, NWSA	82.79
NTES, NWS	-26.61
NWS, PDD	1.34
NWS, VIA	2.96
PDD, VIA	2.40
HSIC, VIA	-2.60
IDXX, VIA	-1.97

As shown in the table above, we did end up making money on some of the pairs and our overall profit over the test period is $58.31. That being said, these numbers do not account for transaction costs and are currently based on buying and selling on the mid-price rather than on ask and bid due to complications we ran into with the simulator. Moreover, we tried to simulate all nineteen pairs, but due to missing data only were able to results for the seven shown.

The table shows that the amount of money we make also varies across the pairs we have chosen and is not completely consistent, but this could be due to the time frame we were simulating on. However, overall results for the trades executed over the 10am-12pm time span on the three days we tested are promising and indicate that the models do have some predictability.

5.1 Strengths

There are several strengths of our strategy. First, it is intuitive and easily understandable. Having a model that makes sense is an advantage not only in actually building the model, but also when trying to attract investors. Statistical Arbitrage and pairs trading is also one of the older trading strategies, and one that has been shown to have great Sharpe ratios and annual returns. Despite time and computational restraints, we see that even a rudimentary form of our algorithm makes money, even though there are various adjustments to make. This means that there *is* alpha and we successfully capture some of it in the small deviations of the highly correlated stock pairs.

Second, the pairs chosen by K-means does a good job of choosing highly correlated pairs. Even if not all of these pairs are in the same industries, they still show great results when used for co-integration. We also think that not following intuition (e.g. choosing companies in the same industry) but using data and statistical analysis to choose companies may provide us with pairs other people are not trading. This could be a potential advantage of our strategy.

Third, we can also see that the model does a good job of roughly identifying and predicting the deviations in the residuals for the pairs of stocks. We did not over-fit our data, which is the downfall of many algorithmic strategies. This was likely due to our caution in testing on multiple test sets and ensuring there was no look-ahead bias in our training.

Finally, there is a lot of room to expand and build upon our model, from fitting on additional targets to increasing the number of pairs in our portfolio to building strategies at the minute and/or micro-second. Our strategy can be easily scaled, which opens the door to numerous other opportunities to capture alpha.

5.2 Weaknesses

Despite the strengths of the model, there are several areas where it can improve. First, a weakness of our model is the amount of data we trained, tested, and simulated on. Given our high frequency data and the fact we had access to tons of data from Thesys, we *could* have used far more data to test our model - be it weeks or even months. That being said, we were inhibited by time and computational power to actually do so. But, the fact we were training on a specific day and simulating on only a few could have contributed to our less impressive results.

Second, as mentioned, we were not able to get an accurate read of the profit our model would make due to problems with the simulator and having run out of time to implement parts of our strategy. We were not able to get results based on buying and selling at bid and ask and they do not account for transaction costs, and so could not compute relevant and necessary statistics, such as Sharpe ratio and annualized return. We also did not specify the quantity of the stocks that we are trading or have enough time to optimize the placement of our bands. Consequently, we are likely not capturing all of the alpha available and also have difficulty accurately evaluating our strategy.

Finally, our strategy is susceptible to a few risks. First, because some of our pairs are not in the same industry, certain macro-economic factors (e.g. the industry of one in the pair entering a downturn) could affect our model. Since we are trading at the second, however, we should have enough time if this were to happen to properly adjust. Second and more pertinently are intraday risks. Though we leveraged company selection, model selection, and placed restrictions on execution of trades to minimize as much risk as possible, the exposure to these trades on a longer timescale does not account for high volatility periods, liquidity constraints, and if the market is entering a downturn. Because our strategy is rooted in mean-reversion, if the difference in price does not revert, our book would suffer.

6 Conclusion and future steps

Overall, we are proud of the model we built. While basic, it shows a lot of potential to make money if we had more time to address our weaknesses. Namely, there are three things we would do to both improve and better evaluate our strategy: first, we would determine optimal quantities to trade on. If we could determine how much of the stocks quantities we should trade on, we would better capitalize on the amount of profit we could make. Second, we would want to build out an optimal band model for each pair. Right now, we are somewhat arbitrarily placing the bands. While this basic strategy still works, if we developed a model for optimal band placement (as outlined in the book), we would also capture more alpha. Finally, we would build our own simulator. We are very grateful to Thesys for providing us with so much data and a simulator, but we ran into numerous issues with not only understanding *how* to use the simulator and read the documentation, but with it crashing when we were trying to do things such as buy on bid and ask and calculate a Sharpe ratio. With time, we could develop our own simulator to test the strategy and this would also give us a better idea as to how our strategy would be executed in the market. All in all, our preliminary results are promising and the opportunities to grow are strategy are endless. We learned a lot, especially in tackling both HFT and Statistical Arbitrage at once, and believe we are truly onto something profitable.

References

[1] Robert Elliott, John Van der Hoek, and William Malcolm. "Pairs Trading". In: *Quantitative Finance, 5(3) 271:276* (Apr. 2005). URL: http://stat.wharton.upenn.edu/~steele/Courses/434/434Context/PairsTrading/PairsTradingQFin05.pdf.

[2] Álvaro Cartea, Sebastian Jaimungal, and Jose Peñalva. *Algorithmic and High-Frequency Trading*. Cambridge University Press, 2019.

[3] Charu Sharma, Amber Habib, and Sunil Bowry. "Cluster analysis of stocks using price movements of high frequency data from National Stock Exchange". In: *arXiv e-prints* (Mar. 2018). URL: https://ui.adsabs.harvard.edu/abs/2018arXiv180309514S/abstract.