Homework 4: Correlated Statistical Arbitrage

Francisco García Flórez, Joris van Lammeren, Wouter Varenkamp

Abstract. In this homework statistical arbitrage is used to extract profits from changes in the market pricing of two stocks. As is common with actual industry sectors several similar stocks behave similarly over time, formally known as correlated stocks. In this case we consider two banks, EUR_CBK and EUR_DBK, and by using statistical arbitrage with extra information from the correlations we will make a certain profit.

1 Trading robot

This trading robot is more complex than the previous one used to benefit from pure arbitrage, since we need to study and understand market fluctuations and guess how is the price going to change at a later point in time. First we will introduce the fundamental concepts on which the robot is based and then briefly explain how it all works together. Finally, results and further improvements will be discussed.

1.1 Trades

In this robot, a **trade** is a set of transactions associated with one ISIN consisting of triplets containing the price, volume and time: (P_i, V_i, t_i). Note that the volume can be negative, meaning we sold that volume. Using this basic unit, our robot will have two modes of operation, namely **adding new trades** and **completing active trades**.

Moreover, each trade will only consist of one transaction initializing the trade, and every subsequent one will reduce the position given by the first. Therefore, the side of a trade is simply defined by the volume of the first transaction: $\text{sign}(V_0)$.

- **New active trade.** We can differentiate between two kind of trades, those which are still active and those that has been completed. Each one is defined by checking if the added volume is zero: $\sum_i V_i = 0$. If it is, then it is considered *completed*, and *active* otherwise.

  Adding a new active trade means finding a right moment in time to buy or sell stock and initialize the trade with that transaction, in the hope that the price will get higher or lower allowing for some profit.

- **Completing active trades.** Once we have created a new trade, we need to check whether or not is profitable to reduce our position in the market, by computing the expected profits of the trade at a certain moment in time. This is done by computing the following value

$$P_{\text{exp}} = P_{\text{prop}} \sum_i V_i - \sum_i V_i P_i \quad ,$$

  where P_{exp} is the expected profit and P_{prop} is the selling or buying price proposed by the market. If this results in a positive amount, we can consider proceeding with the transaction, and also allows us to sort each active trade in terms of expected profits, fulfilling the most profitable ones before the others.

Having defined the main modes of operation, we remark that the number of active trades is not limited, however there are several considerations we need to issue for the program to make any profits. First of all, this model (just by itself) relies on having enough time to fulfill almost every active trade to make profits, since even though our position in the market at some time can be zero, our profits may in the negatives because we are investing in buying stock for later transactions. Later we will discuss several possible ways to reduce this effect.

As a final remark, we can now refer to *completed trades profits* as the profits made only from completed trades, as a measure of how well the program is working excluding other factors like unwinding.

1.2 Detecting price peaks

In the last section we describe the idea of trading at the right moment in time, and for that we need to detect which ones those are. In the first approach, without using any correlations, we can look at the actual market and compute a value for the stock, which will move following what we can consider a Brownian motion. However, this is a mathematical object difficult to work with, so to be able to detect peaks using standard tools like derivatives we need to remove part of this randomness and extract a continuous value.

For this we will use a modified version of the *Holt-Winters algorithm*, also known as triple exponential smoothing. Fundamentally, it works as a high frequency filter on the fluctuations of the market value, resulting in a smooth trajectory that can be differentiated. We use a modified version without seasoning or forecasting, but it also computes the smoothed second derivative of the price, so that we can know if the peak is a maximum or a minimum.

Another benefit of this approach relies on providing us with three parameters, $\alpha, \beta, \gamma \in [0, 1]$, which we can adapt to better approximate the actual price. Values closer to zero correspond to using more past values to compute the current average, while a value of one just returns the real market value. These three values can be further optimized by several approaches, mainly **initial values optimization** and **runtime adaptation**.

In **initial values optimization** we run the algorithm over the feed data several times with different values for these parameters, choosing the ones that maximize the profit. On the other hand, **runtime adaptation** modifies them during the simulation. Each approach can give different results depending on the actual behavior of the market, however in our case we can expect the former to be good enough, since the length of the feeds is not significant. Finally, we can apply this smoothing to any quantity related to the market, but we will mainly use it to determine the ask and bid prices, the total volume of the book and to store the real ask and bid values of the stock. In the program however it has been generalized and can be used for any function using values from the book.

1.3 Stop-Loss algorithm

As we mentioned before, this simple algorithm will fail to make profit because some active trades get "stuck" when we get some position at a value that won't make any profit at a later time. This situation can easily happen when there is a significant dip in the price and we got a negative position at the peak, rendering it unable to make any profits.

There are several strategies we can follow to reduce the effect of this situation on our profits, but we can categorize them in two groups: **volume limiting** and **stop loss**, however both can work in parallel. The former relies in trying to estimate the right amount of stock we should trade in the beginning so that the volume is not high enough for the market to absorb if we want to get rid of our position. The latter, on the other hand, estimates how much losses each active trade is having and tries to fulfill it before they are too large.

In our robot, both are implemented in a simple way. The position gained is limited by the volume contained in the first entries of the book, while the stop loss checks for active trades with a relative loss larger than a set value and tries to complete them.

Another relevant source of losses comes from the end of the feed, since our position at the end has to be zero. If we have a significant position when the feed ends we risk two problems. First, we don't know if the market has that volume available so we can remove our position, so we can get stuck with some stock. Second, even if the market can take it, we can incur in significant losses because of a dip in price. To reduce this, we have to be aware of how long the feed is and try to reduce our position gradually, meaning that near the end of the feed we will be making less profits but we can get rid of our temporal position without significantly affecting the rest.

2 Results

In Figures 1, 2 and 3 we can see the actions of the robot, where the red continuous line is the ask price level (computed by the smoothing algorithm), discontinuous red line is the market ask price (first entry of the book), and similarly for blue lines, being the bid price in this case. Dots, red and blue, represent buying and selling respectively, so we can know when exactly the program is detecting a peak.

Since in this case we have limited the amount of stock we buy, we can see several intervals in which nothing seems to happen, and is in this periods when the stop-loss algorithm should kick in and fulfill previous trades to keep new ones coming. We haven't included correlations in the algorithm yet, but the most significant contribution from them would be that instead of buying or selling not exactly at the peak, we can actually predict where the peak is going to be, and then stop loosing a small amount on each trade.

We can also clearly see in this plots one of the problems we have to work with: the bid volume is on average almost an order of magnitude smaller than the ask volume. This means that we need to be very careful buying stock, since it is quite easy to get stuck with it.

Now, since there are still a few problems with the algorithm, by looking at the completed trades profits:

```
Completed trades profits:

  DBK_EUR:     486.96
  CBK_EUR:      67.55
  Total:       554.51
```

We can see that completed trades are actually making profits, however the final profits are `-126.92` (assuming we unwind successfully), mostly because we have bought stock to compensate the total position by opening active trades that didn't get fulfilled.

3 Conclusion

4 Appendix

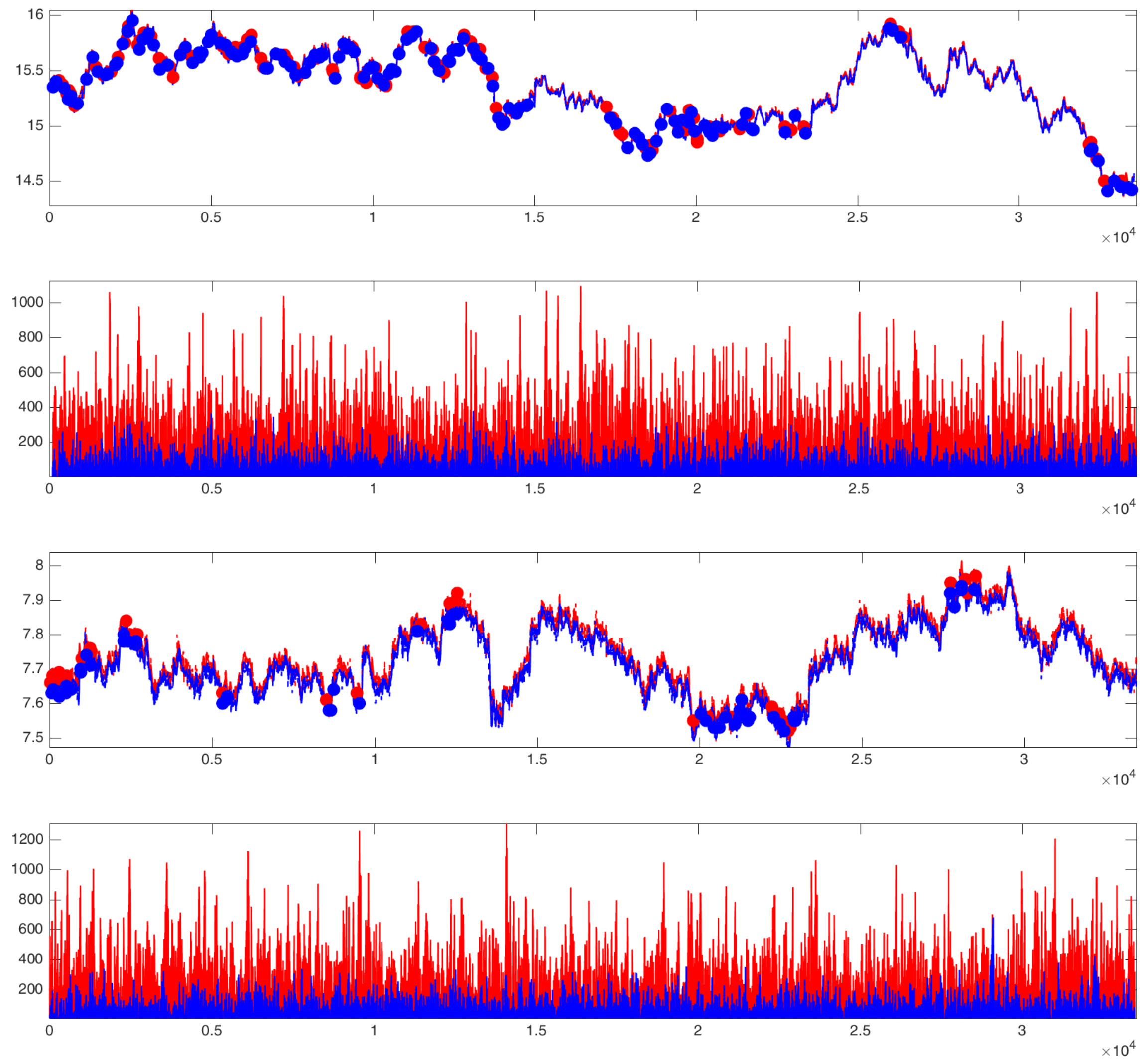

Figure 1: Results after running the program for the first feed. The first and second plots correspond to the ISIN DBK_EUR, ploting price value and volume respectively, and the last two correspond to CBK_EUR.

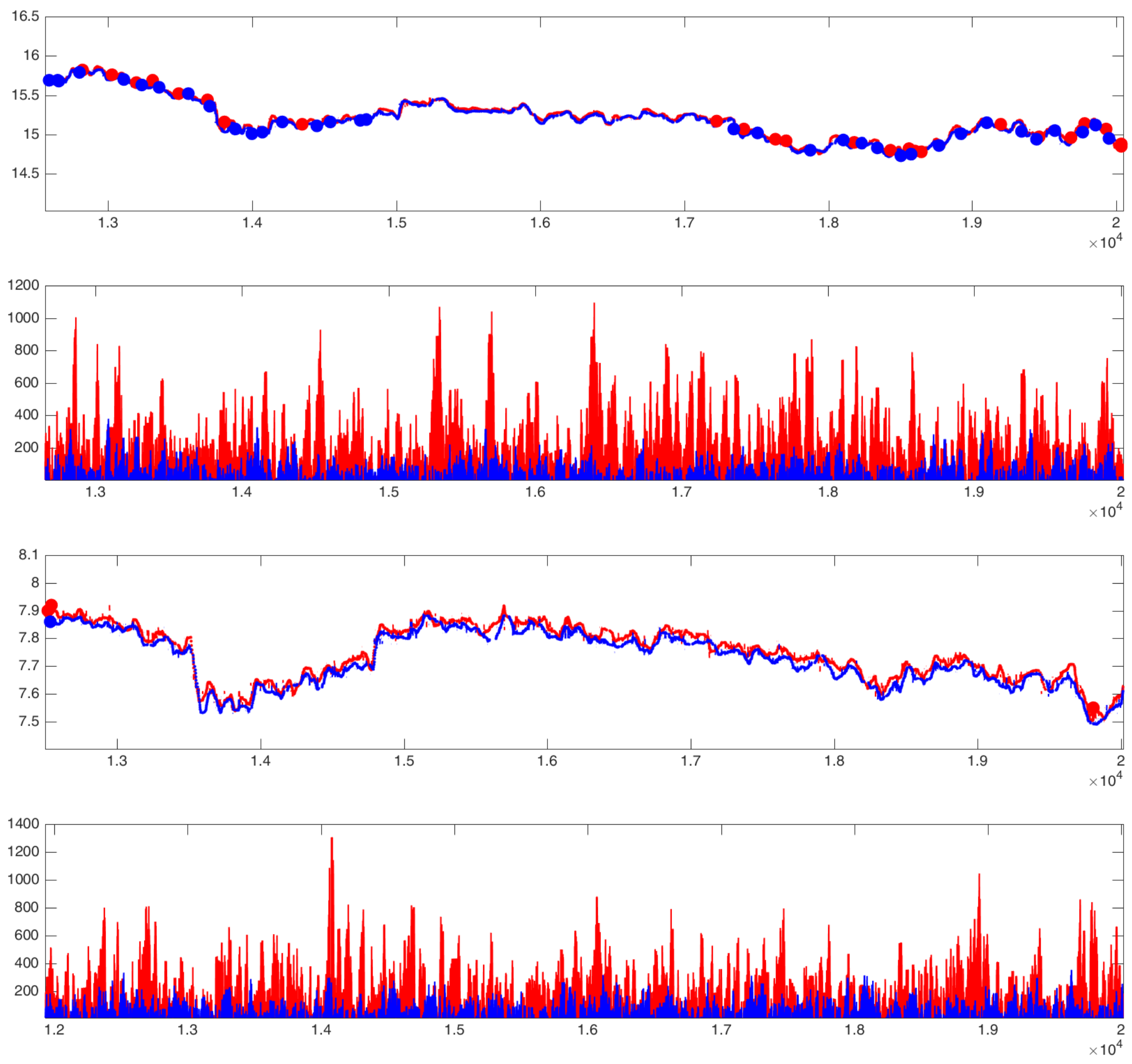

Figure 2: Results after running the program for the first feed, closer look.

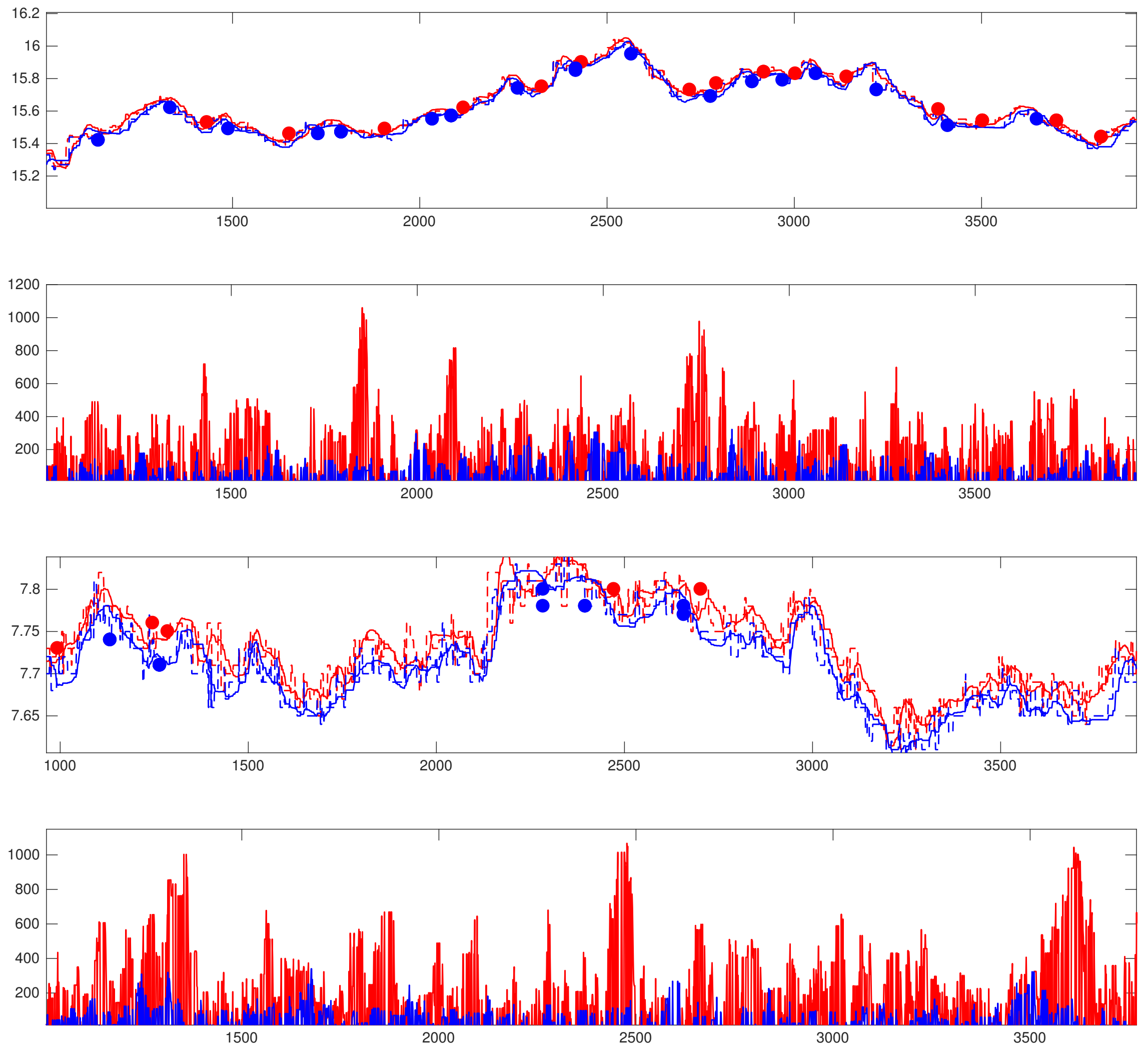

Figure 3: Results after running the program for the first feed, another closer.

References

[1] P. Wilmott et al, *The Mathematics of Financial Derivatives*, 1995.