Assignment 3: Statistical Arbitrage

M.M. Dekker R. Erdem B.E. Zijlstra

April 30, 2016

1 Introduction

In this assignment we will arbitrage between two stocks: Commerzbank (`CBK_EUR`) and Deutsche Bank (`DBK_EUR`). Both Commerzbank (CBK) and Deutsche Bank (DBK) are listed on the Frankfurt Stock Exchange, so we hypothesize there is a correlation between both stocks. Finding a significant correlation can give important information about the behavior of these stocks. When these two stocks do not behave in this way, we can assume this is a temporary mispricing. Accordingly, this temporary mispricing can be used to make money by selling the assumed overpriced stock and buy the temporary underpriced stock. Since statistical arbitrage (SA) is not deterministic like dual listing arbitrage, implementing the correct analysis and trading parameters are very important.

2 Methods

2.1 Analysis of the feeds

In order to perform SA, we need to implement statistical methods to quantify the behavior of CBK and DBK. For this we use the `AnalysisRobot`, which is a class with two methods: `HandleDepthUpdate` and `PlotMarket`, with the use of the function `Valuate`.

2.1.1 HandleDepthUpdate

In the `HandleDepthUpdate` function, the robot searches for the best bids and offers. This is obviously done for all feeds. These best bid and best offer prices give an indication of what the best (lowest) possible price is for which you can acquire a share, and of what the best (highest) possible price is for which you can sell a share. With this information we can compute a valuation function. How exactly, is described in the next section Valuate.

2.1.2 Valuate

For the `Valuate` function, the mere average between the bid- and askprice is a naive valuation. We think taking volumes into account is needed. For example, when a lot of assets are bid for in the market (i.e. the bid volume is high), we can expect that the price of this stock will rise as well. In general, the lower the highest layer of a book in volume, the more the actual value of the stock will be attracted to this (bid or ask). To implement this, the valuation function will do the following operation:

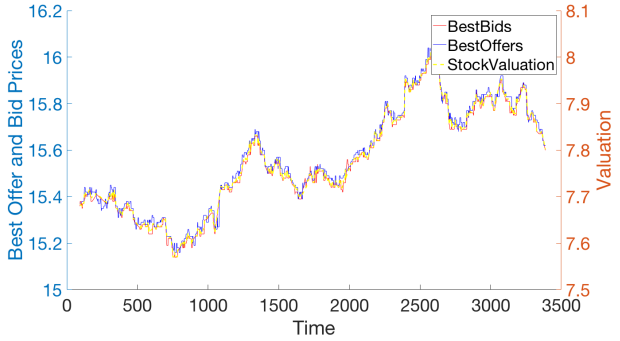
$$\texttt{Value} = \frac{1}{2}\left(\frac{\texttt{aBid}}{\texttt{aAsk} + \texttt{aBid}} \cdot \texttt{aAskp} + \frac{\texttt{aAsk}}{\texttt{aAsk} + \texttt{aBid}} \cdot \texttt{aBidp}\right) \tag{1}$$

With `aBid` and `aAsk` the bid and ask volumes, and `aBidp` and `aAskp` the bid and ask prices of the stock at the first layer of the book at the time at which we want to know the value of the stock. There is factor $\frac{1}{2}$ added, which means that the value that is gained is not an actual *price*. To calculate the actual fair price, multiply by two again. Using this, the actual value of a stock can be analyzed and used to determine mispricing.

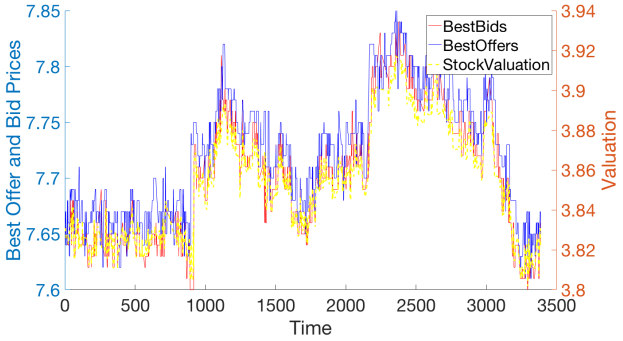
2.1.3 PlotMarket

Another method in the `AnalysisRobot` is the `PlotMarket`. This function analyzes the best bids and offers per share (which are stored in properties like `myTradingRobot.BestOffersDBK` by the `HandleDepthUpdate` in the `AnalysisRobot`). Using the `Valuate` function, the exact evolution of the values of the two stocks are gained. Between these two lines, a correlation and linear regression function is made.

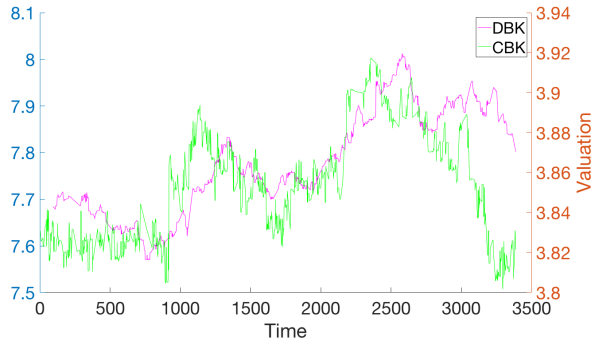
Lastly, in order to actually view the concepts mentioned above, for each feed there are four plots are made. In figure 1 til 4, four plots of analysis of the first feed (CBKDBK1) are shown. Figure 1 shows the best bids and offers, along with the stock valuation of the DBK share. Figure 2 shows the best bids and offers, along with the stock valuation of the CBK share. Figure 3 shows both stock valuations of the DBK and CBK shares. It also shows the correlation between these two stock valuations (in the upper left corner). Figure 4 shows the regression and its 95 % percent confidence intervals, along with the valuation for the DBK share.



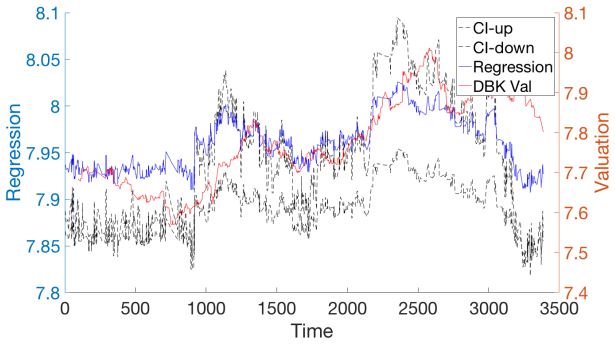
(a) DBK best bids/offers



(b) CBK best bids/offers



(c) Stock valuations



(d) Regression with 95% confidence interval

Figure 1: Various figures showing results of the regression and valuation for feed 1.

2.2 The Δ parameter - significance and confidence intervals

2.2.1 Idea of the Δ parameter

It is important to note that the (average) correlation between the two valuation functions is still relatively uncertain. In the results section (Tab. 1), the correlation coefficients are shown, and it is visible that they are not always close to one. This should be reflected back in our analysis, to ensure that we only trade stocks of which we are quite certain that they are mispriced. This is where our Δ comes in. Δ is a value that acts like a confidence interval around the regression between the two valuated stock functions. We define the stocks to be mispriced if the price of, say, the stock of DBK exceeds the value what we would predict using the regression and the value of CBK *plus* Δ (or if the stock becomes lower than the regression *minus* Δ). In terms of equations:

$$\begin{aligned} \texttt{DBK\_value} &> a \cdot \texttt{CBK\_value} + b + \Delta, \text{ or:} \\ \texttt{DBK\_value} &< a \cdot \texttt{CBK\_value} + b - \Delta \end{aligned}$$

where a and b are gained from the regressions we got using the `AnalysisRobot` (as earlier described). A first approximation of Δ would be to take the 95% confidence interval, calculated from the regressions itself (using `polyfit()` in Matlab). However, Δ should not be a constant. In the end of the feed, we do not want to have any position, because that will be corrected using very bad prices. There are two considerations concerning this to take into account using a variable Δ. One of these is that in the first part of the feed, we do not have to be too picky on which trades we want to choose: we can afford to buy more assets for less profit per asset because the amount of assets does not matter yet. This, however, changes in the second part of the feed, where we should be more careful with buying assets. This results in a time dependence of Δ:

$$\Delta = \Delta_0 \cdot f(t) \tag{2}$$

with Δ_0 (`myDelta` in Matlab) marks the δ-parameter of the 95% confidence intervals, and $f(t)$ a function of time to be determined. The second consideration concerning the zero-position problem is that in the end of the feed, we want to be less picky (low Δ) with trades that make us lose position, while we should be very picky (high Δ) with trades that create additional position. This creates a position dependence of Δ:

$$\Delta = \Delta_0 \cdot f(t) \cdot g(Q) \tag{3}$$

with $g(Q)$ a function of position Q to be determined.

2.2.2 Quantification of the Δ

The goal now is to determine the function $f(t)$ and $g(Q)$. First, it is important to note that $f(t)$ is not really dependent on time, but on the fraction of time with respect to the total amount of time. Therefore, we created a property `Time`, which creates an additional element (all equal to `1`) each time the `TryArbitrage` is called. This way, from earlier runs we could see that the total amount of time was about 3417 (i.e. this number is not gained while the `TradingRobot` is running, so we had to specify this explicitly). The next step is to make a distinction between the first half of the feed, and the second half. In the early half, the position Q should not play a major role because in this period, we want the robot to trade with maximum profits: we have enough time to reduce our profits in the later half. This means that in the first half, we set $g(Q) = 1$. Now, to start with a $\Delta > 0$ at $t = 0$ and to gradually increase this value to Δ_0, we use the following function for Δ (in the first half of the feed, unto $t = \frac{3417}{2} \approx 1708$):

$$\Delta = 0.25 \cdot \Delta_0 + 0.75 \cdot \Delta_0 \cdot \sin(\frac{\pi}{3417} t) \text{ for t} < 1708$$

To be clear: only a quarter sine will be present in this part of the feed, so this time dependence will make it gradually approach Δ_0. Now, the second half of the feed should contain a $g(Q) \neq 0$. Also, from now on we need to make a distinction between cases of trades that potentially attribute to our position (e.g. selling when you already have a negative position) and cases of trades that potentially reduce our position (e.g. selling when you have a positive position). In the first case, your Δ should be higher, because the negative effect of having extra position will likely be stronger than the positive effect of making an extra good-regression based trade. In the second case, *vice versa*: we need a lower Δ to get rid of position more easily. This means we need to make a distinction between which Δ to use; one that is extra high (Δ_h), and one that is extra low (Δ_l):

$$\Delta_h = \Delta_0 + \Delta_0 \cdot (1 + \frac{|t - \frac{3417}{2}|}{3417})(1 + \frac{|Q|}{50}) \text{ for t} \geq 1708,\ sgn(Q_t) = sgn(Q)$$

$$\Delta_l = \Delta_0 - \Delta_0 \cdot (1 + \frac{|t - \frac{3417}{2}|}{3417})(1 + \frac{|Q|}{50}) \text{ for t} \geq 1708,\ sgn(Q_t) \neq sgn(Q)$$

for Q_t the volume of the specific trade (< 0 when selling, > 0 when buying). As can be seen, both the function start with Δ_0 and diverge from that on. The more time has passed (approaching the end of the feed, $t = 3417$) and the higher the position is, the further the Δ_l and Δ_h come from Δ.

2.2.3 Implementation of Δ

After calculating Δ_0 (described above), the Δ values per stock per kind of trade (sell/buy) are defined (e.g. called `myLowDeltaCBK`). After this, four `if`-loops check whether to define sell-trades as Δ_l or Δ_h and the same with buy-trades (e.g. called `myDeltaBuyCBK`). Of course, only one of these per stock be needed (if there is überhaupt a viable trade), but for clarity purposes all are defined. These new Δ's are used in the core of the trading robot where it determines whether lines are empty or not, calculates the valuations and makes the trades.

3 Results

In order to run the `TradingRobot` on the test feeds, first the `AnalysisRobot` needs to calculate the variables mentioned in 2.1.These findings are presented in Tab. 1.

Table 1: Results Analysis Robot

	Correlation	**Delta**	**Slope**	**Crossing**
CBKDBK1	0.59377	0.16725	2.3960	-1.4621
CBKDBK2	0.31892	0.13478	0.65321	5.2221
CBKDBK3	0.8616	0.13629	2.9922	-3.7514
CBKDBK4	0.81649	0.15749	3.4283	-5.3519

Table 2 shows the profit of `TradingRobot` on each feed. Each feed has a net positive cash, except in feed 3 we have to buy back 15 DBK stocks which will result in a very small loss. Apart from a small loss, the `TradingRobot` shows to be robust otherwise. Since some of the assumptions that were made about the parameters were a little bit arbitrarily, lots of runs were tried among the test feeds to measure differences between these values.

In table 3 we measured the amount of assets that were traded in each test feed. An important note here is the fact that `CBKDBK3` has a significant amount of fewer trades and `CBKDBK2` has the largest amount of trades. This is good news for our robot, as it shows that the performance of the robot becomes less volatile after more trades.

4 Discussion and Reflection

We are making a net profit, which is a lot better than a few of the alternative methods we mention below. The correlations and profits per feed differ quite strongly. It is interesting, though expecting, to see that in the feed where the valuation functions have the lowest correlations (feed 2; $r \approx 0.32$), the profit is highest.

4.1 Optimization possibilities

There are a number of possibilities where one can tune this way of trading to optimize profit. First, we could have chosen the boundary between the two parts (the first is independent of position, the second dependent) to be at another position than exactly in the middle of the feed. For example, if you know that in general there are more asset volumes in the books, one can afford to neglect position for a longer time than just the first half.

4.2 Pros, Cons and Alternative methods

It is helpful to reflect on the various consequences of our method. There are some notes to be given, however:

- An advantage of our method is that it directly uses the regressions gained from the example feeds and also weights the different parameters using the correlation coefficients of each specific feed. This way, low correlations are weighted weaker than others. However, this approach assumes a correlation between `CBK_EUR` and `DBK_EUR`, which in reality is not always the case.

Table 2: Results TradingRobot

	Cash	**CBK**	**DBK**
CBKDBK1	220.42	19	0
CBKDBK2	989.94	0	0
CBKDBK3	222.42	0	-15
CBKDBK4	819.37	0	0
Total	2252.15	19	-15

Table 3: Traded assets

	CBK	DBK	Average CBK	Average DBK
CBKDBK1	5160	3752	47,34	39,91
CBKDBK2	12610	13783	42,75	42,94
CBKDBK3	1808	2005	46,36	48,90
CBKDBK4	3555	2970	50,07	38,57

- An advantage of using the Δ function is that it takes into account the balance between wanting to lose position in the end and taking risk to have a chance on profit. The different factors can be quantified very sharply and tuned in various ways. This is also directly a disadvantage. Straying too far off regular linear equations to quantify $f(t)$ and $g(t)$ can become rather ambiguous. It also becomes a rather slow process with probably high uncertainty to quantify specific parameters inside these functions, for example the denominator in the quotient that determines the influence of $|Q|$ (which is now 50).
- A disadvantage of the use of these `myHighDeltaCBK`-like variables is that it is very discrete. If there is a negative position, the Δ for selling becomes enormous, while one time step ago we had a positive position and the Δ for the exact same trade would have been much lower. Of course this is only the case for higher positions and later in the feed, but still this is something that could have been programmed to be more elegant.

An obvious alternative method would have been to leave Δ constant, or even out of the game. However, this would have lead to very high positions in the end of the feed, or very low profit.

5 Conclusions

The findings of our robots show that there is indeed high correlations between the valuation functions as we hypothesized. When assessing a price of the stocks at each certain time that matches the market price, we see that the behavior of DBK and CBK stocks are similar. In the long run, using this knowledge should result in profit when it is implemented correctly. After using our `TradingRobot` on four test feeds, we are confident that our robot is implemented in a logically way that is tested empirically.