# Assignment Week 4. Statistical arbitrage: pairs trading

Data: You will receive a .csv file with market data for 15 fictive stocks, named AA, BB, and so on through to OO. The file contains observations in 5 minute intervals, of bid and offer prices and volumes on a single exchange.

The first column in the file gives the timestamp of each observation, the next four columns, respectively contain the bid price, bid volume, ask price and ask volume of the first stock, the four columns following that give the same values for the second stock, and so on.

Assignment: This assignment consists of two parts. The first part is to run a data analysis to determine which pairs of stocks can be profitably traded. The second part is to implement and backtest a trading strategy to trade these selected pairs.

Results should be extendible to new periods. Do not over-fit to the dataset but attempt to define the analysis and strategy such that they are likely to lead to persistently positive PnLs.

1. Analysis

Make use of the pre-provided python functions implementing the estimation of the long- and short-run cointegration relationships as well as the Engle-Granger test to analyse each possible pair of stocks.

Based on the outputs of these functions, rank the pairs on trading-potential and provide a visual display of the best and worst performers. Describe why these pairs have high or low potential to trade.

1. Strategy

Based on your analysis, make a choice of pairs to trade. More pairs can lead to more profit, but worse pairs might be riskier or less profitable to trade. Make a reasonable trade-off.

Assume you can trade with immediate-or-cancel (IOC) orders, where you can sell against the bid price, and buy from the offer price, up to the volume that is available. Assume a 100% hit rate for your orders.

For each of the selected pairs, for each time step, determine the distance from its long run mean, the value of the “error correction term”. What is your perceived profitability in entering the pair-trade long or short? Depending on the perceived edge and your current position, decide what trade to enter into and what your resulting new position is.

Incorporate a position limit of a maximum of 100 lots long or short per stock.

As before, visualise how your positions and PnL evolve over time.

Be creative in optimizing your strategy. For example, when do you decide to enter a trade? Do you instantly hedge your trade in one instrument in the other, or do you wait for a better price? Attempt to optimize your risk-reward trade-off by backtesting different choices. Describe the process in your report.

Extra assignment: Visualize the Value-at-Risk (VaR) of your position over time by using the variance-covariance method. You can use the entire dataset to calculate standard deviations and correlations. Note that when trading multiple pairs you should calculate the VaR for the entire portfolio, not just per pair.

In addition, choose a single pair of stocks that you have traded and visualize the VaR of the pair and both stocks separately. Do you see a difference between the sum of the VaR of the single stocks and the VaR of the portfolio? Explain.

Extra Materials: Python module with two functions for cointegration analysis

Skeleton Python notebook with starting points for exercise

Worked example Dual Listing arbitrage strategy exercise Week 3

Deliverables: A 3-6 page report describing your analysis and strategy. This needs to be relatively self-contained but does not need to re-explain concepts directly coming from the lectures. The report should contain at least the following, but may contain any other relevant information or visualisations:

A high-level description and/or visualisation of the dataset

Description of the results of the data analysis and choice of pairs

Description of and argumentation for the strategies’ internal logic

A visualisation of the strategy performance, including:

The position taken per stock

The profit and loss over time per stock and in total

An appendix with the python code generating the analysis and strategies

(appendix does not count toward report pages)

Evaluation: Report (20%)

Analysis & Strategy (50%)

Analysis (20%)

Strategy: Potential, Description and Evaluation (20%)

Originality (10%)

Code Quality (30%)

Step-by-step guideline for Python implementation:

1. **Analysis**

First, we will run a cointegration analysis on all potential stock pairs, conclude which pairs to trade, and make this data available for easy use in the trading algorithm.

* Using Pandas, load the file with the .csv dataset into a new DataFrame.
* Explore the dataset: print the first 5 rows and display the column names. Make a quick plot of a few of the price- and volume-series.
* Next, prepare a set of market mid prices from the supplied bids and asks. Loop over the list of stock names, and for each stock name:
  + Get the correct columns in a dynamic fashion, first the bid and ask price of AA, then the bid and ask price of BB, and so on. Based on these two columns, calculate the average of them, the mid price, and store it back into the original market data DataFrame, give the new columns the name ‘MidPrice’ and store them under the correct stock name.
  + Visualize the calculated mid prices alongside the bid and ask prices, do they indeed fall in the middle?
* First try running the analysis functions on the mid-prices of a single, randomly chosen pair. Get the outputs, visualize and interpret them. Is the pair you randomly chose a good pair to trade?
* Now repeat this, but instead loop over each possible pair of stocks by creating a nested (double) loop over the stock names. The inner loop will define ‘stock\_name\_2’ and the outer loop will define ‘stock\_name\_1’. Together they will loop over all pairs.
  + For the two stocks, extract the mid prices from the DataFrame and run the cointegration analysis functions on them.
  + To keep the results easily available, store all the outputs from the function calls into a dictionary, indexed by a tuple of the two stock names. For example, if ‘all\_pair\_pvalues’ is a dictionary and ‘pvalue’ is the single obtained p-value for this specific pair, you could use:

all\_pair\_pvalues[(stock\_name\_1, stock\_name\_2)] = pvalue

* + Make sure to also store the error correction term and fitted short-run and long-run relationship parameters in a similar way, as we will use their values later in the trading algorithm.
* Finally, to more easily analyse the results, convert the dictionaries of p-values and Engle-Granger test statistics into DataFrames:
  + To create the DataFrame you can use the DataFrame.from\_dict function, which converts the dictionary to a DataFrame automatically:

pvalues\_df = pd.DataFrame.from\_dict(all\_pair\_pvalues, orient='index')

* + With the results now in a DataFrame, sort the statistics/p-values and visualize them. Show the top 10 least and most cointegrating pairs. Based on these numbers, draw a conclusion on which pairs to trade.

1. **Trading algorithm**

From here, we will proceed in almost the same manner as in the Dual Listing exercise. Use the code from the worked example as a reference. For this trading strategy, you can similarly loop over each time step, make a trading decision based on available volumes, position limits, and the error correction term, store the resulting positions, and calculate PnL afterwards. The two differences are that 1. you do not trade the same stock listed on two different exchanges, but instead trade two different stocks listed on the same exchange, meaning you should normally trade in a non-1:1 ratio and 2. the trading decision is more complex as it depends on the value of the error correction term, rather than on a direct comparison of prices.

* Build up the code for a single stock-pair first. Only after that code runs smoothly, run the code for multiple stock pairs.
* Visualize the problem:
  + For the two stocks in your selected pair, display the bid and offer prices, bid and offer volumes and the error correction term in three separate graphs. If you stored the error correction term as recommended in the Data Analysis section, obtain the Series of error correction term values corresponding to this specific pair from your dictionary first.
  + Is it clear when you would want to do a trade?
* Write the algorithm:
  + For ease of access, add the error correction term output from your data analysis as a new column in your main DataFrame, alongside the original market data. Also obtain the values describing your long-run and short-run relationships, e.g. and .
  + As your algorithm will need to keep track of the position limits per stock, its behaviour will need to depend on the trades it did before. For this reason it will be easiest to implement the algorithm one instant at a time by looping over each of the observations and timestamps in the input data as before.
  + Inside the loop, for each step in time do the following:
    - Obtain the current available prices and volumes for both stocks, as well your previous position in each of them.
    - Obtain the value of the error correction term.
    - Based on these values, make a trading decision. If the ECT is low/high enough, enter a trade (decide reasonable values for low/high enough). How much are you allowed to trade in each stock according to your position limit? How much volume is available on both of the stocks? And what would be your optimally hedged position, in a ratio of ? Combine these factors to calculate your trade and new position in each of the stocks.
* After you have determined your positions in each stock for the whole dataset, use them to calculate your PnL At each instant, the PnL should consist of two parts:
  + The total money you have paid/received for the trades you have done so far
  + The valuation of the position you are holding
* Visualize the trading algorithms position taking alongside the initial dataset, does it do what you expected it to do?
* Finally, collect the positions and PnLs into a DataFrame and write it out to .csv.
* After this runs for a single prespecified pair, automatically loop over different stock pairs and repeat the analysis, storing all results along the way.