École Nationale Supérieure d'Informatique pour l'Industrie et l'Entreprise

ENSIIE Évry-Courcouronnes

Academic Research Project: Statistical Arbitrage Strategy using Kalman Filter

Prepared by:
Théo LE MOAL

Professor:
M. PULIDO NINO Sergio

Academic Year : 2022 / 2023

Contents

1 Introduction

Statistical arbitrage is a trading strategy that seeks to profit from discrepancies in the prices of related securities. The approach is based on the idea that the market tends to misprice securities from time to time, and these discrepancies can be exploited by statistical methods. Statistical arbitrage traders analyze historical data and use advanced algorithms to identify patterns in the market that could indicate pricing discrepancies.

It is commonly accepted that statistical arbitrage started with Nunzio Tartaglia who, in the mid-1980s, assembled a team of quantitative analysts at Morgan Stanley to uncover statistical mispricing in equity markets. However, statistical arbitrage came to the fore as a result of Long-Term Capital Management (LTCM), a hedge fund founded in 1994, where Nobel Prize winners Scholes and Merton both worked. The company developed complex statistical arbitrage strategies for fixed income which were initially extremely successful. However, in 1998, as a result of the financial crises in East Asia and Russia, LTCM's arbitrage strategies started producing large losses which endangered global markets and forced the Federal Reserve Bank of New York to organize a bailout in order to avoid a wider financial collapse. Nevertheless, statistical arbitrage continued to grow in popularity with applications progressively expanding to all asset classes. statistical arbitrage has become one of the main investment strategies in investment banks and mostly for hedge funds. In particular, the term statistical arbitrage is used to denote hedge funds that aim to exploit pricing anomalies in equity markets. Technological developments in computational modelling have also facilitated the use of statistical arbitrage in high frequency trading and with the so-called machine learning methods, such as neural networks and genetic algorithms.

Every statistical arbitrage strategy needs to solve the following three fundamental problems: Given a large universe of assets, what are long-short portfolios of similar assets? Given these portfolios, what are time series signals that indicate the presence of temporary price deviations? Last, but not least, given these signals, how should an arbitrageur trade them to optimize a trading objective while considering possible constraints and market frictions? Each of these three questions pose substantial challenges, that prior work has only partly addressed.

In this paper, we investigate pairs trading statistical arbitrage strategies by trading similar assets with respect to some criteria. We present a comprehensive framework for the development of pairs trading statistical arbitrage strategies. The framework consists of several parts:

- Detecting pairs of similar assets.
  First, we will investigate a method to detect similar assets. We are looking for the cointegrated pairs of assets among a set of assets. To carry out this task, we use a machine learning method, clustering using the k-means partitioning algorithm. Then, we wish to test the hypothesis of cointegration among all possible pairs of assets in each cluster: to carry out this task, we use the Augmented Dicked-Fuller test.

- Estimate the relation between these pairs.
  Once we have a list of cointegrated pairs of assets, we want to estimate the relation between the two assets of each pair. As we will explain later, common method to estimate the relation is using an OLS regression. The problem is that this method is not optimal, as the relationship between the asset pairs is not constant, it is more interesting to use a method that allows to follow the variation of this relationship over time: for this, we use the Kalman filter.

- Create and optimize trading signals.
  At this stage, we created a new variable, the spread between the asset pair. We now need to create a signal, which should detect a change in the relationship between the asset pair, which therefore represents an arbitrage opportunity. To achieve this task, we test two different methods: first, we try to build a function based on the standard deviation of the spread. We try to optimize the parameters of the function with Bayesian Search method.

- Backtest the strategy.
  Now that we have trading signals for each of our asset pair, we can then calculate the performance of the strategy using a backtest.

2 Theoretical concepts of the strategy

2.1 Clustering with Kmeans

K-means clustering is a popular unsupervised learning algorithm used for partitioning a given dataset into k distinct non-overlapping clusters. The algorithm aims to minimize the within-cluster variance or the sum of squared distances between data points and their respective cluster centroids.

Here's a step-by-step explanation of the k-means clustering algorithm:

1. Initialization: Randomly initialize k cluster centroids within the feature space or select them based on some heuristics.

2. Assignment: Assign each data point to the nearest centroid based on the Euclidean distance or other distance metrics.

3. Update: Recalculate the centroids of each cluster by taking the mean of all the data points assigned to that cluster.

4. Iteration: Repeat steps 2 and 3 until convergence is achieved. Convergence is typically determined when either the centroids do not change significantly or the maximum number of iterations is reached.

5. Output: Once convergence is reached, the algorithm provides the final k clusters, where each data point belongs to a specific cluster.

It's important to note that the choice of k, the number of clusters, is a crucial parameter that needs to be determined beforehand. Selecting an inappropriate value for k may lead to suboptimal clustering results. Various techniques, such as the elbow method or silhouette analysis, can be employed to estimate the optimal value of k.

The silhouette analysis assigns a silhouette coefficient to each data point, which represents its cohesion within its cluster and separation from other clusters. The silhouette coefficient ranges from -1 to 1, where:

- A coefficient close to +1 indicates that the data point is well-matched to its own cluster, indicating a high level of cohesion and appropriate clustering.

- A coefficient close to 0 indicates that the data point is near the decision boundary between two clusters, suggesting overlapping or ambiguous cluster assignments.

- A coefficient close to -1 indicates that the data point may have been assigned to the wrong cluster, as it would have been better placed in a different cluster.

The average silhouette coefficient can be used to compare different clustering solutions with varying values of k. A higher average silhouette coefficient indicates better-defined and well-separated clusters. However, it's important to note that silhouette analysis should be used as an additional evaluation tool and not the sole criterion for determining the optimal number of clusters or assessing the clustering quality. Other factors, such as domain knowledge and interpretability, should also be taken into account.

2.2 Cointegration test

2.2.1 Stationary process

A stationary process, also known as a stationary time series, is a stochastic process whose statistical properties do not change over time. In simpler terms, it means that the process maintains a constant mean, variance, and covariance structure throughout its entire duration.

Definition 1 (Stationary Process) Let X_t be a time series, where t represents the time index. The process X_t is said to be stationary if, for any positive integers s and t and any set of real numbers c_i (where i ranges from 1 to n), the joint distribution of X_{t+c_1}, X_{t+c_2}, ..., X_{t+c_n} is the same as the joint distribution of X_{t+s+c_1}, X_{t+s+c_2}, ..., X_{t+s+c_n}. In simpler terms, the distribution of the process is invariant to shifts in time.

A classic example of a stationary process is white noise. In this case, each observation is independent and identically distributed with a constant mean and variance. The mean and variance do not change with time, making it a stationary process. An autoregressive process of order 1, denoted as AR(1), can be stationary under certain conditions. For instance, if the absolute value of the autoregressive coefficient is less than 1, the process will exhibit stationary behavior. This means that the process tends to revert to its mean over time, maintaining a constant variance.

2.2.2 Unit Root

A unit root refers to the property of a stochastic process where the series is non-stationary and has a root of 1 in its autoregressive (AR) characteristic equation. In simpler terms, a unit root indicates that the series exhibits a random walk behavior, lacking a stable mean or long-term equilibrium.

Definition 2 (Unit Root) Let X_t be a time series. The series X_t is said to have a unit root if it satisfies the following autoregressive equation: $\Delta X_t = X_t$ - $X_{t-1} = \alpha * X_{t-1} + \epsilon_t$, where α is a coefficient and ϵ_t is the error term. If α= 1, it indicates the presence of a unit root.

To detect the presence of a unit root, several statistical tests can be employed. The most commonly used unit root test is the Augmented Dickey-Fuller (ADF) test.

2.2.3 Augmented Dickey-Fuller Test

The Augmented Dickey-Fuller (ADF) test is a statistical test used to determine the presence of a unit root in a time series. It is widely employed to assess the stationarity or non-stationarity of a series and is an extension of the Dickey-Fuller test. The ADF test allows for the inclusion of lagged differences of the series to account for potential autocorrelation.

Definition 3 (Augmented Dicker-Fuller Test) The ADF test is based on the following autoregressive (AR) model with lagged differences:

$$\Delta Y_t = \alpha + \beta t + \gamma Y_{t-1} + \delta_1 \Delta Y_{t-1} + \delta_2 \Delta Y_{t-2} + ... + \delta_p * \Delta Y_{t-p} + \epsilon_t,$$

where Y_t is the time series, δ represents the differencing operator $\Delta Y_t = Y_t$ - Y_{t-1}, α is a constant term, β is the coefficient on a linear trend, γ is the coefficient on the lagged level of the series, δ_i (for i = 1 to p) are the coefficients on the lagged differences, ϵ_t is the error term, and p is the number of lagged differences included in the model.

The procedure to apply an ADF Test is:

1. Specify the null and alternative hypotheses:
   - Null hypothesis (H0): The series has a unit root (non-stationary).
   - Alternative hypothesis (HA): The series is stationary.
2. Estimate the autoregressive model by selecting an appropriate lag order (p) and accounting for trend components if necessary.
3. Calculate the test statistic, typically the ADF statistic, which measures the significance of the coefficient γ.
4. Compare the test statistic with critical values from the ADF distribution (obtained from statistical tables or software) to determine whether the null hypothesis can be rejected.
5. If the test statistic is lower than the critical value, the null hypothesis of a unit root is rejected, indicating the presence of stationarity in the series.

The critical value for the ADF test is typically 5%.

2.2.4 Cointegration Test

A cointegration test[9] is a statistical procedure used to determine whether two or more non-stationary time series are integrated of the same order and have a long-term equilibrium relationship. It helps to identify whether a linear combination of the series is stationary, indicating a stable relationship between them.

Definition 4 (Cointegration Test) Let X_t and Y_t be two non-stationary time series. The series are said to be cointegrated if there exists a linear combination $Z_t = aX_t + bY_t$ such that the resulting series Z_t is stationary. In other words, if the order of integration of X_t and Y_t is d, and a linear combination Z_t with order of integration less than d results in a stationary series, then X_t and Y_t are said to be cointegrated.

To test for cointegration, various statistical tests can be employed. The most commonly used test is the Engle-Granger test, which involves estimating a regression model on the two series and examining the stationarity of the residuals. The steps involved in the Engle-Granger test are as follows:

- Check for the individual stationarity of the series using unit root tests like the Augmented Dickey-Fuller (ADF) or Phillips-Perron (PP) test.
- If both series are non-stationary, regress one series on the other to obtain the residuals.
- Conduct unit root tests on the residuals to determine if they are stationary.

- If the residuals are found to be stationary, it indicates cointegration between the original series.

It is important to note that cointegration does not imply causality. While the presence of a cointegrating relationship suggests a long-term equilibrium connection between the series, it does not provide information about the direction of causality or the short-term dynamics between the variables.

Cointegration analysis is widely used in time series econometrics and finance to identify and model long-term relationships, especially in the context of pairs trading, asset pricing, and macroeconomic modeling.

2.3 Estimate the spread using Kalman Filter

A common quant trading technique involves taking two assets that form a cointegrating relationship and utilizing a mean-reverting approach to construct a trading strategy. This can be carried out by performing a linear regression between the two assets and using this to determine how much of each asset to long and short at particular thresholds.

One of the major concerns with such a strategy is that any parameters introduced via this structural relationship, such as the hedging ratio between the two assets, are likely to be time-varying. That is, they are not fixed throughout the period of the strategy. In order to improve profitability, it would be useful if we could determine a mechanism for adjusting the hedging ratio over time. A sophisticated approach is to utilize a state space model that treats the "true" hedge ratio as an unobserved hidden variable and attempts to estimate it with "noisy" observations. In our case this means the pricing data of each asset. The Kalman Filter performs exactly this task.

In its simplest form, we model the relationship between a pair of securities in the following way:

$$\beta(t) = \beta(t-1) + w,$$

$\beta(t)$ the unobserved state variable, that follows a random walk. We have: $w \sim N(0, Q)$, meaning w is gaussian noise with zero mean and covariance Q, Q the covariance matrix of the process noise.

$$Y(t) = \beta(t)X(t) + v,$$

the observed processes of asset prices Y(t) and X(t). We have: $v \sim N(0, R)$ meaning v is gaussian noise with covariance R, R the covariance matrix of the observation noise.

So, this is just like the usual pairs' relationship $Y_t = \beta_t X_t + v$, where the typical approach is to estimate beta using least squares regression, or some kind of rolling regression (to try to take account of the fact that beta may change over time). In this traditional framework, beta is static, or slowly changing. In the Kalman framework, beta is itself a random process that evolves continuously over time, as a random walk. Because it is random and contaminated by noise we cannot observe beta directly, but must infer its (changing) value from the observable asset prices X and Y.

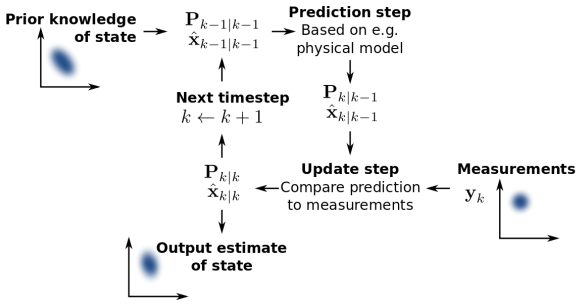

Figure 1: Explanation of the basic steps of Kalman filtering: prediction and update.

2.4 Create and optimize trading signals

In a mean-reverting strategy, long positions are taken in under-performing stocks and short positions in stocks that have recently outperformed: based on the spread created, we need to create a signal to enter a position.

In the literature, in general people use the standard deviation of the spread as a trading signal:

- If $Spread(t) > \sigma_{spread}$, then enter position: short asset 1, long asset 2. We exit the position when $Spread(t)$ goes back to zero.
- If $Spread(t) < -\sigma_{spread}$, then enter position: long asset 1, short asset 2. We exit the position when $Spread(t)$ goes back to zero.

Here, we will use a signal based on this equation:

$$Signal(t) = \alpha + \delta * \sigma_{spread(t)},$$

with α a constant which will be optimized, δ a parameter that is multiplied by the rolling volatility of the spread, which will be calculating according a specific period.

To choose the optimal parameters of our signal function, we are going to use a Bayesian Search: the method will look for the optimal parameters to minimize a function, according to a search space defined beforehand for each parameter.

The Bayesian Search method uses the previous iterations to guide the following iterations. It consists of building a function distribution (Gaussian process) that best describes the function to be optimised. In this case, the function to be optimised is the one that, given the parameters, returns the performance of the strategy. After each step, this function distribution is updated and the algorithm detects which regions of the parameter space are most interesting to explore and which are not. After a defined number of iterations, the algorithm stops and returns the optimal tuple. Bayesian optimisation is a very efficient method to quickly explore the set of possible parameters.

3 Implementation of the strategy

3.1 Data retrieval

The first step is to collect data: we want to collect the evolution of the market price of a large number of assets, to detect the peers of similar assets among this set. We decided to retrieve the data of all the assets that make up the CAC40. We will also provide an example on commodities assets, especially a list of five energy products. To retrieve data, we used pandas_datareader: this package is very useful to extract data from a wide range of internet sources into a pandas DataFrame.

3.2 Find Cointegrated pairs

3.2.1 Clustering

Once we have the data, we can start the clustering. We create a new dataframe, in which we stock the mean return and the volatility of each assets along the period. We train our Kmeans model on this new dataframe.

By using the silhouette method, we are searching for the optimum number of clusters, to maximize the performance and to avoid underfitting and overfitting.

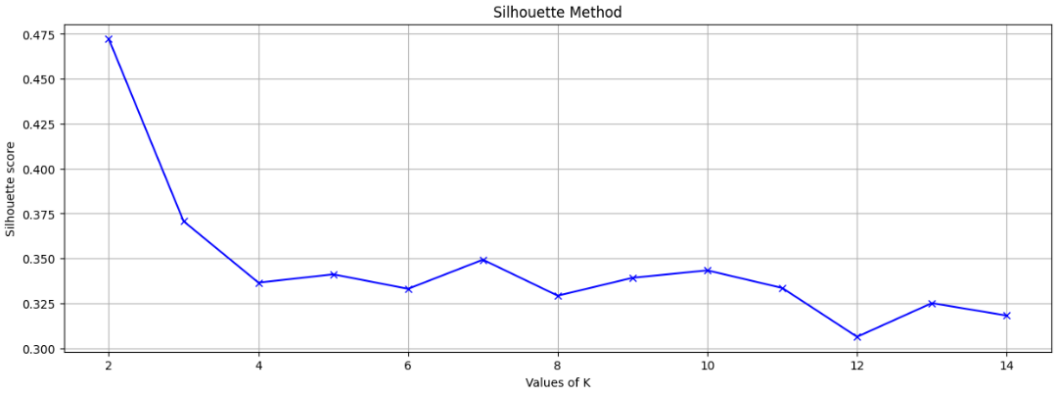

Figure 2: Silhouette method

As we can see, the silhouette method suggests that k=4 is the optimal value.

Now we can apply the Kmeans algorithm. In the following graph, we can visualize how the algorithm partitioned the data.

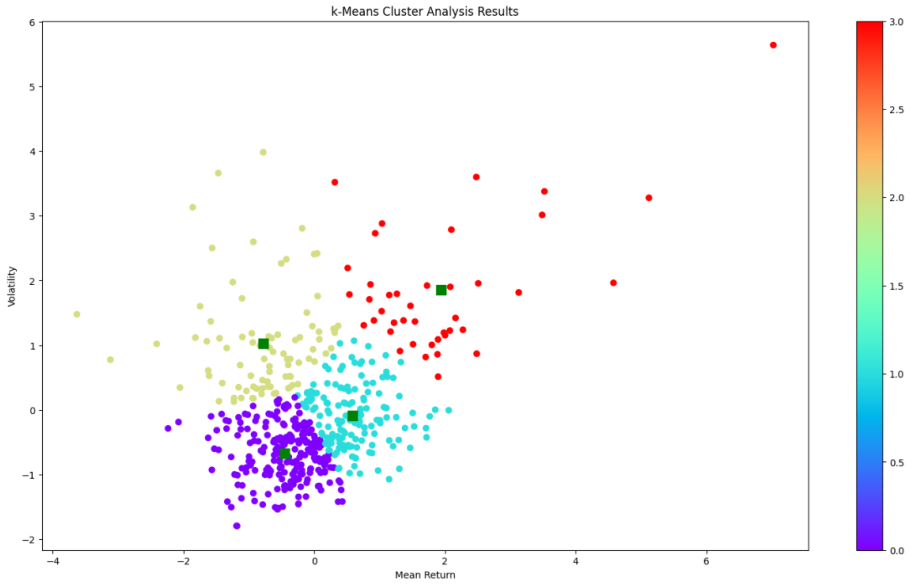

Figure 3: Cluster analysis

3.2.2 Cointegration Test

As we have seen before, we now have 4 clusters, composed of several assets. For each cluster, we will apply a cointegration test to all the combinations of pairs of assets that we can create within this cluster.

For each pair,We fit an OLS regression on the two assets, then we calculate the residuals: $\epsilon = Y - \hat{Y}$. We compute the ADF test on the residuals, and we add the pair to our final list of cointegrated pairs if the p_value of our test statistic is lower than the critical value, which is 5% here.

It allows us to obtain the pairs with the lowest p_value: we will converse only these pairs afterwards.

We obtain this list of pairs :

```
Number of clusters:  2
Number of cointegrated pairs:  25
Pairs with lowest p-value among all the clusters:

(['RI.PA', 'YUM'],
 ['EMR', 'LH'],
 ['ACN', 'PKI'],
 ['CS.PA', 'RE'],
 ['CB', 'CS.PA'],
 ['ACN', 'HD'],
 ['MMC', 'RSG'],
 ['LH', 'TEL'],
 ['DGX', 'LR.PA'],
 ['AMCR', 'CL'],
 ['HOLX', 'PCAR'],
 ['ROP', 'WMT'],
 ['APD', 'WMT'],
 ['AMGN', 'LNT'],
 ['MNST', 'PCAR'],
 ['CS.PA', 'HIG'],
 ['NKE', 'TYL'],
 ['ED', 'ENGI.PA'],
 ['DUK', 'JNJ'],
 ['CAP.PA', 'JBHT'],
 ['HOLX', 'MNST'],
 ['KO', 'SRE'],
 ['CMCSA', 'MDT'],
 ['CA.PA', 'MO'],
 ['AMGN', 'XEL'])
```

Figure 4: Optimal pairs of assets

3.3 Estimate dynamic beta using Kalman Filter

An OLS regression estimates the beta of the relationship between the two assets of the pair in a static way, which is not optimal, as we want the beta to be estimated dynamically. We are going to use the Kalman Filter to achieve this task.

To implement the Kalman filter, we use the pykalman package. First of all, we have to initialise the filter: we have to initialise the different parameters of the filter, such as the list of mean states and the matrix of covariance states. Once the filter is initialized, we can then update it at each iteration, i.e. as soon as we receive a new market price.

Now, we have a beta relation that depends on the time: we can calculate the spread between the two assets : $Spread_t = Price_{asset1} - \beta_t * Price_{asset2}$.

We centre and reduce the variable, choosing an optimal look back window: we chose to centre-reduce the variable with respect to the mean and standard deviation of the last 50 values of the spread. This variable will be used throughout the trading strategy: when this signal reaches a certain value, there is an arbitrage opportunity between the two assets.

For example, if at time t, the value of the spread is greater than 1.5, then this means that the price of asset 1 is diverging upwards from the price of asset 2. The strategy then suggests selling one unit of asset 1, and buying beta units of asset 2. We reverse our two positions when the spread value is zero: we buy one unit of asset 1 and sell beta units of asset 2.

If, on the other hand, the value of the spread is less than -1.5, then this means that the price of asset 1 is diverging downwards with respect to the price of asset 2. The strategy then suggests buying one unit of asset 1, and selling beta units of asset 2. We reverse our two positions when the spread is zero: sell one unit of asset 1 and buy beta units of asset 2.

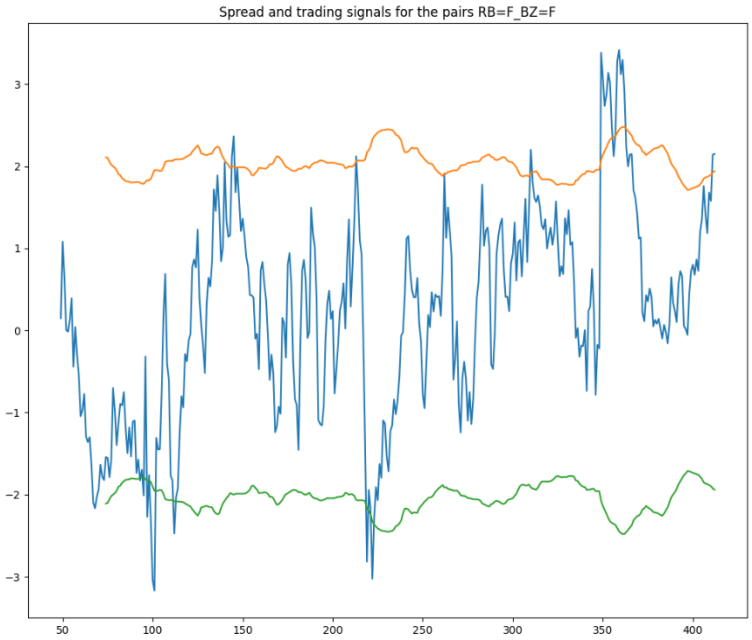

Figure 5: An exemple of the evolution of the spread and the trading signals for a pair of assets

3.4 Optimize trading signals

As we explained above, we are trying to optimize a signal based on this equation : $Signal(t) = \alpha + \delta * \sigma_{spread(t)}$. To optimize each of these parameters, and thus find the trading signal that maximizes the cumulative returns of the strategy, we use Bayesian Search.

To optimize the parameters using Bayesian Search, we need to define a space region for each of the parameters. The algorithms will then look to the best combination that maximize the cumulative returns of the strategy. We optimize these parameters on a train dataset, and then we take the optimal trading signal and apply it to a test dataset.

4 Analysis of the results

4.1 Backtest on CAC40 index assets

We have a dataset containing the price evolution of the 40 assets that make up the CAC40 index. We obtain 22 cointegrated pairs: for each of these pairs, we look for the values of the parameters of the trading signals function that maximize the cumulative returns. This signal can then be tested using a backtest.

Once we have backtested the strategy of each asset pair, we simply create an equiprobable portfolio for the strategy of each asset pair: we have 22 asset pairs, so we create a portfolio in which the proportion of capital allocated to the strategy of each asset pair is 1/22.

To judge the quality of our strategy, we need a metric. Generally, in finance, we use a common asset and compare the performance of the strategy with the performance of that asset. As in our case we selected our asset peers from the shares that make up the CAC40 index, we chose to take the CAC40 index prices as the reference metric. If the strategy achieves a better performance than a long position on the CAC40 index over the period, then we can consider that the strategy is interesting, in the opposite case it is not interesting.

First, we define a train period, from may 2018 up to November 2021, where we optimize the trading signals of our strategy. Then, we test our strategy from November 2021 up to June 2023.

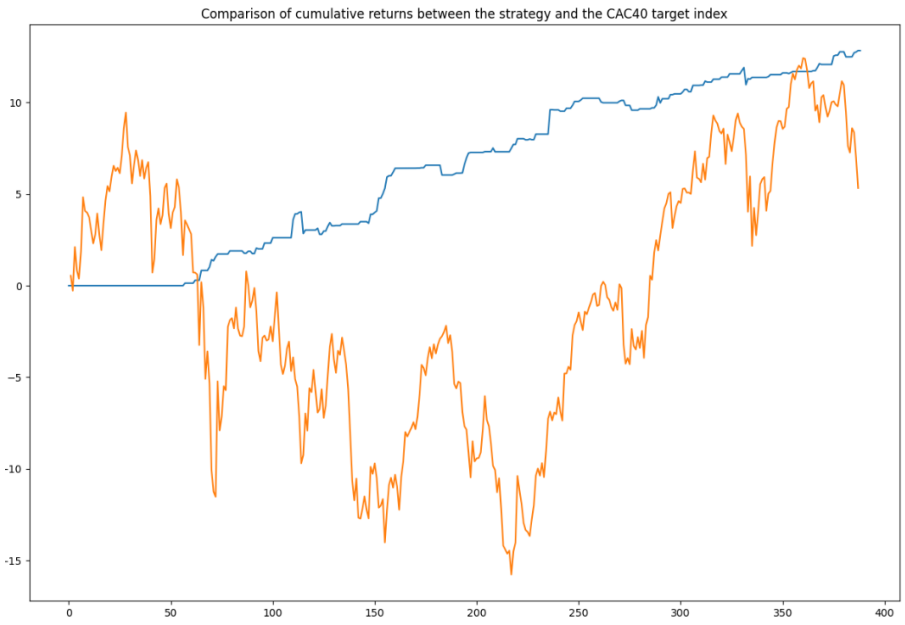

Figure 6: Cumulatives returns of our strategy and the CAC40 index, from November 2021 up to June 2023

We can see that our strategy outperform the CAC40 index. The cumulative returns of the strategy and the CAC40 are sometimes close, with the CAC40 index sometimes outperforming our strategy. The striking difference is that our strategy achieve much less volatile returns than

the CAC40 index. This is a very good thing, because market finance players are particularly fond of low-volatility assets with constant returns. The standard deviation of our strategy is about 4.5%, whereas the standard deviation of the cumulative return of the CAC40 index is about 7.1%.

When we test the strategy over a longer period, we notice that the results are less good, which means that the strategy needs to calibrate the trading signal parameters regularly to obtain interesting results.

If we look at the Profit and Loss achieved for each pair in our strategy, we can see that the results are very widely dispersed. There are several extreme values: some strategies achieve very poor or very good results. This highlights the importance of creating a portfolio of several pairs and not focusing on a single pair, at the risk of unpleasant surprises.

HO.PA_TTE.PA	-27.583791
BNP.PA_DSY.PA	-9.777590
AI.PA_OR.PA	-8.633376
EL.PA_RI.PA	1.840564
ERF.PA_TEP.PA	3.438327
CS.PA_RI.PA	6.032885
DSY.PA_SGO.PA	6.546360
MC.PA_STMPA.PA	7.335794
CA.PA_RI.PA	7.962544
CAP.PA_SGO.PA	8.955087
CA.PA_EN.PA	9.229672
LR.PA_RI.PA	11.821476
EN.PA_ML.PA	16.348973
ENGI.PA_RI.PA	17.204368
STMPA.PA_SU.PA	17.305267
BNP.PA_CAP.PA	17.690886
EL.PA_LR.PA	18.051876
CS.PA_OR.PA	19.502969
CS.PA_EL.PA	23.502793
ACA.PA_AIR.PA	24.821636
DSY.PA_VIE.PA	32.269642
ALO.PA_VIV.PA	78.437774

Figure 7: Profit and Loss (in %) of each pair of our portfolio

4.2 Backtest on oil products

The second example focuses on a list of energy products. This list of assets includes Brent, WTI, Heating Oil, natural gas and Gasoline Futures.

As we did previously, we select the pairs of cointegrated assets: we then obtain a list of 9 cointegrated pairs. We define a train period, from January 2019 up to November 2021, where we optimize the trading signals of our strategy. Then, we test our strategy from November 2021 up to June 2023. The strategy performed well, with a profit of 22.1%.

This time too, when we test the strategy over a longer period, we notice that the results are less good, which means that the strategy needs to calibrate the trading signal parameters regularly to obtain interesting results.

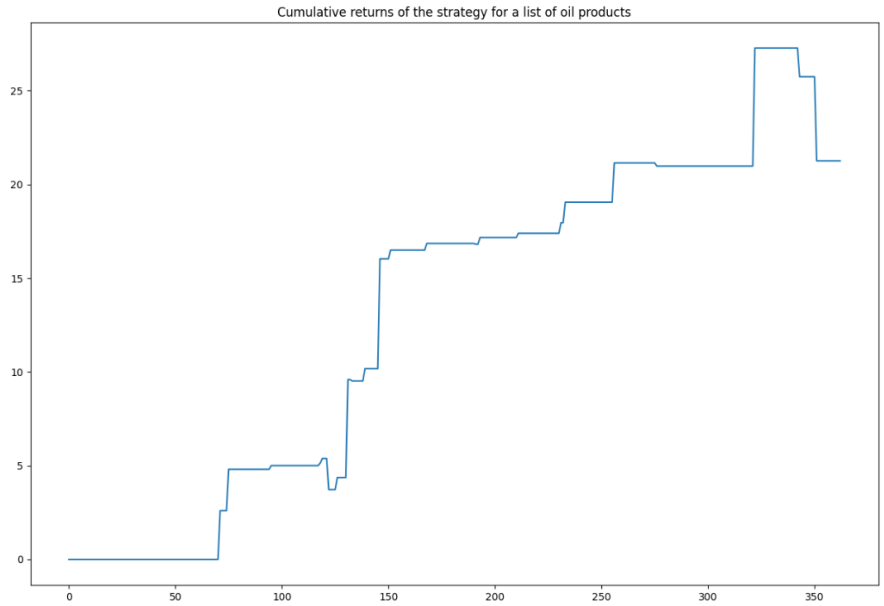

Figure 8: Cumulative returns of our strategy from November 2021 up to June 2023

5 Future improvements

A method for implementing a statistical arbitrage strategy using the Kalman filter to dynamically calculate the beta of the relationship between two cointegrated assets has been proposed. However, this method could be improved and could be further developed.

First, during this project we had access to poor quality data, with a daily frequency, which is not ideal: in fact, if we had access to data with an hourly frequency, we could obtain much better results. Unfortunately, we haven't found any APIs that provide access to a substantial history of hourly data.

Secondly, I think the clustering part could be developed further to detect asset pairs, in particular by adding data.

Then, we did not find an optimal solution for choosing the function that best defines the trading signals, so as to maximize the profit obtained from the strategy. Several methods were compared, but none of them stood out in terms of performance. We are convinced that an optimal method can be found to define trading signals.

Finally, at the end of the strategy, we simply create an equiprobable portfolio for the strategy of each asset pair: we have 22 asset peers, so we create a portfolio in which the proportion of capital allocated to the strategy of each asset peer is 1/22. This method is not optimal, as it does not give a higher point to the peers of assets that perform best. To solve this problem, we could implement a portfolio optimization algorithm, with a mean-variance approach for example.

6 Conclusion

In this research project, we worked on statistical arbitrage strategies, strategies widely deployed in the financial industry by proprietary traders, looking for discrepancies to take advantage of.

It has been observed that there are common methods to create a trading strategy of peers of cointegrated assets. These methods have many limitations, so new methods have been proposed that have recently been used in the field of quantitative finance research to overcome these limitations.

A method of selecting the peers of cointegrated active ingredients was first implemented, using clustering methods and a cointegration test. This allowed us to identify 22 peers of cointegrated assets among the 40 assets that make up the CAC40 index, the benchmark of the largest French companies.

Then, we estimated the relationship between these asset peers dynamically, which represents an evolution of the classical model that calculates this relationship statically.

Finally, we explored different methods of creating and optimizing trading signals, without

succeeding in finding an extremely convincing method. We finally show that our strategy outperforms the CAC40 index over a specific time period.

Even if we have proposed an improvement of the conventional method, our strategy can still be improved, as explained in the future improvements section. This could include finding new methods to define trading signals, and implementing final portfolio optimization.

List of Figures

References

[1] DBSCAN Clustering in Machine Learning

[2] Developing statistical arbitrage strategies using cointegration

[3] Kalman Filter based pair trading strategy in QSTrader

[4] State space models and the Kalman Filter

[5] ETF pairs trading with Kalman Filter

[6] Mean-Reverting Statistical Arbitrage in Crude Oil Markets, Viviana Fanelli, 10 April 2017

[7] The arbitrage strategy in the crude oil futures market of shanghai international energy exchange, Jing Niu, Chao Ma and Chun-Ping Chang, 14 December 2022

[8] Evaluation of Dynamic Cointegration-Based Pairs Trading Strategy in the Cryptocurrency Market, Masood Tadi, Irina Kortchmeski, 22 September 2021

[9] Computing Cointegration and Augmented Dickey Fuller test

[10] DBSCAN Pair Selection