HUDSON & THAMES

hudsonthames.org

THE DEFINITIVE GUIDE TO PAIRS TRADING

V 1.0

HUDSON AND THAMES

WE ARE HUDSON & THAMES

Our mission is to promote the scientific method within investment management by codifying frameworks, algorithms, and best practices to build the world's first central repository of ready to use implementations and intellectual property.

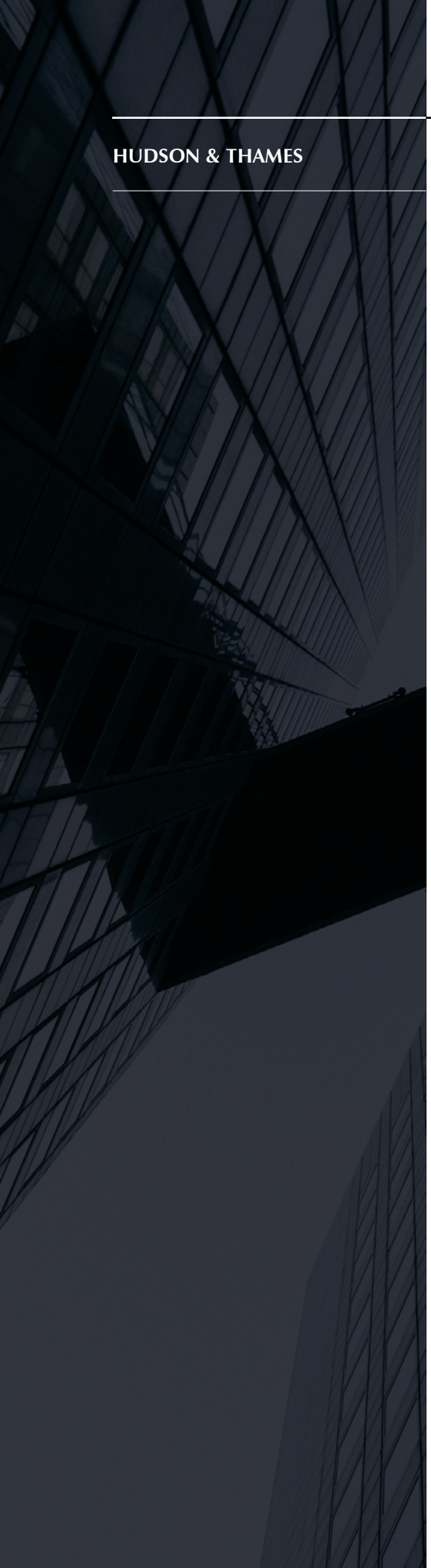
"Rigorous research and efficient algorithms are the cornerstones of modern algorithmic trading. For quants and traders seeking a competitive edge, the ArbitrageLab can empower them to design the optimal mean-reversion trading strategies."

PROFESSOR TIM LEUNG

Boeing Professor of Applied Math & Director of Computational Finance & Risk Management (CFRM) Program at the University of Washington.

1.0

THE DEFINITIVE GUIDE TO PAIRS TRADING

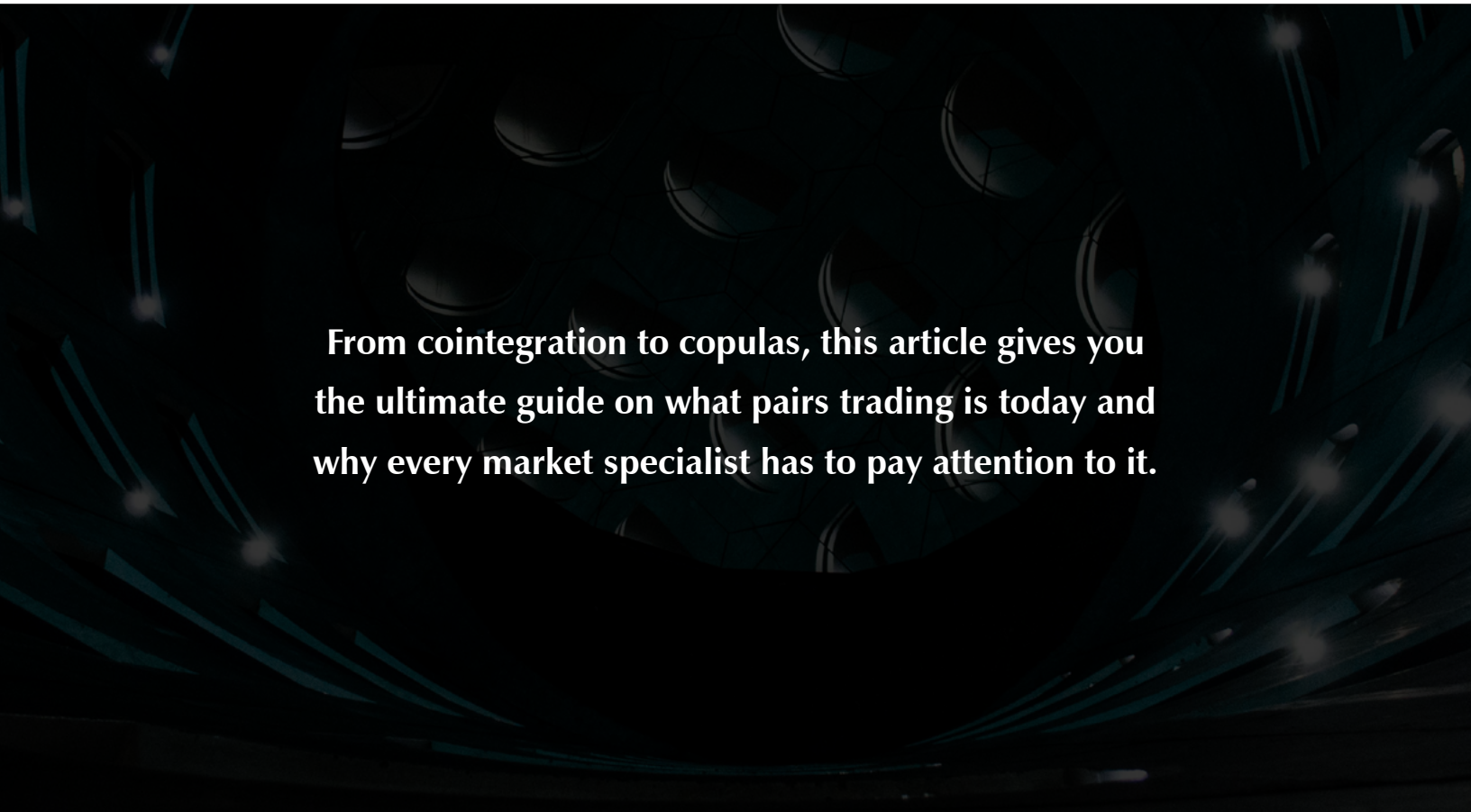


INTRODUCTION

Born at Morgan Stanley in the late 1980s, under the wing of Nunzio Tartaglia and his team, who later split up to start several of the world's best hedge funds, namely **PDT Partners** and **D.E. Shaw** (which then lead to **Two Sigma**). Pairs trading has proven to be a popular and sophisticated trading strategy, often taught in advanced MSc Financial Engineering programs.

With the rapid evolution of markets and advancement in technology, arbitrage opportunities have become scarcer and the margins - slimmer. To adapt to the market, the whole industry needed an upgrade! In the early 2000s, the once universally praised pairs trading strategy entered its "ice age" and after almost ten years, the resurgence of interest in the field brought numerous advanced approaches, and with that - the much-needed change.

The amount of advanced scientific research that has accumulated was exactly what was needed for the strategy to evolve and uncover its full potential as a truly universal and robust approach. The variety of methods range in complexity and choice of assets - this sets an entirely new stage for statistical arbitrage to shine in the current financial world.

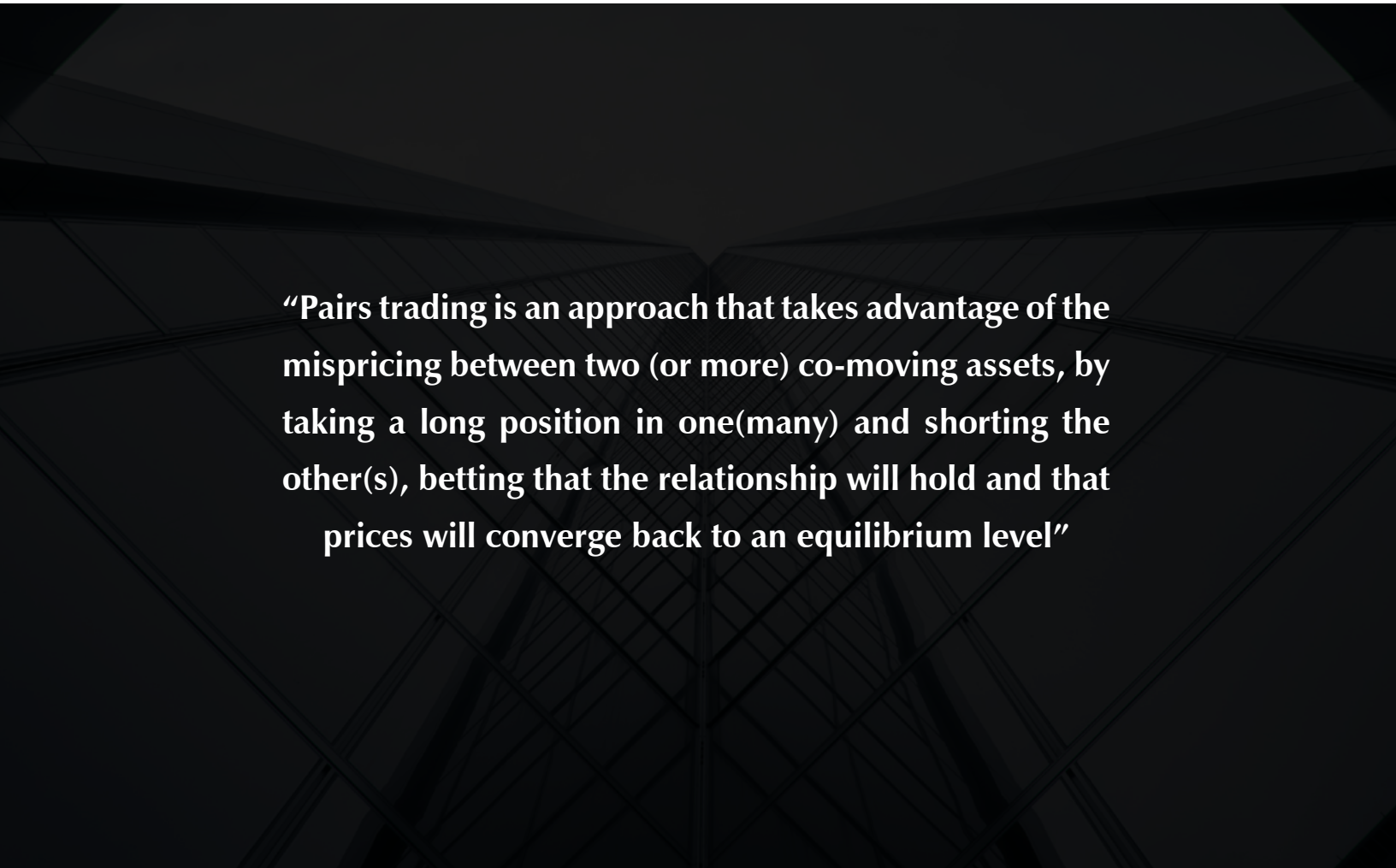


WHAT IS PAIRS TRADING?

First, we need to start with a definition of statistical arbitrage and pairs trading. Often people will use these two terms interchangeably, however, pairs trading is a subset of statistical arbitrage and so we can say that all pairs trading is statistical arbitrage but not all statistical arbitrage is pairs trading.

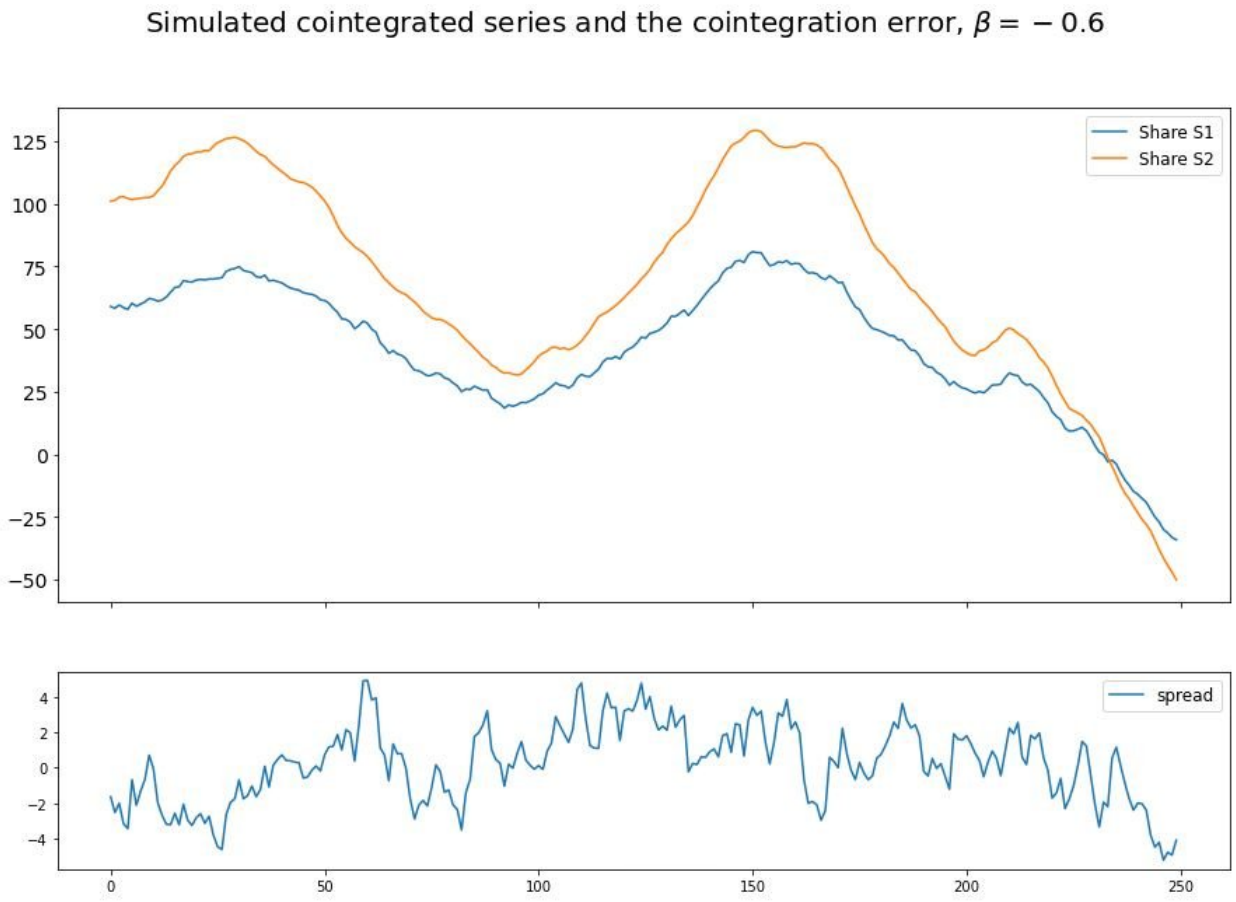
Statistical arbitrage is typically broken down into factor investing and the mean-reverting portfolios of pairs trading. We should add that in its simplest form pairs trading refers to trading only 2 assets but it can be extended to an n-dimensional mean reverting portfolio.

There is no single agreed-upon definition in the literature with each author bringing their own take, so we took it upon ourselves to formalize it as follows:



"Pairs trading is an approach that takes advantage of the mispricing between two (or more) co-moving assets, by taking a long position in one(many) and shorting the other(s), betting that the relationship will hold and that prices will converge back to an equilibrium level"

In the example below, we simulate a pair of co-moving assets using an implementation from **ArbitrageLab**. The first plot shows the co-moving assets whilst the bottom one illustrates how a mean-reverting spread is created by going long the first asset and short 60% in the second. It is this spread that can be traded without needing to make any forecasts about the future.



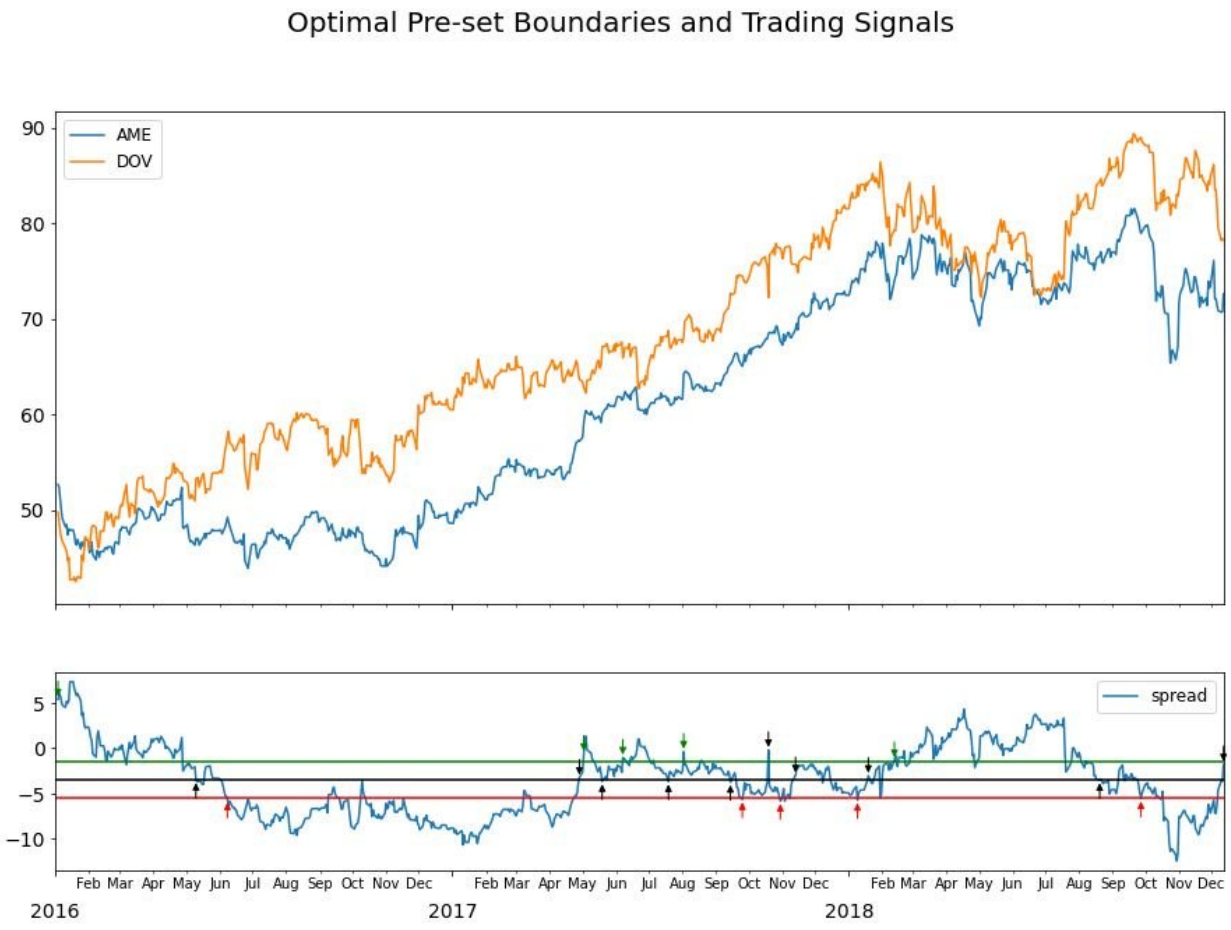
We will simulate the cointegrated pair and build a spread using the cointegration module from Arbitragelab.

The intuition behind pairs trading goes back to the fundamental principle of investing: "buy undervalued - and sell overvalued". However, to determine if the asset is truly over or undervalued, we need to know the intrinsic value, which is at best an approximation and largely what value investing sets out to do.

Statistical arbitrage and pairs trading tries to solve this problem using price relativity. If two assets share the same characteristics and risk exposures, then we can assume that their behavior would be similar as well. This has the benefit of not having to estimate the intrinsic value of an asset but rather just if it is under or overvalued relative to a peer(s). We only have to focus on the relationship between the two, and if the spread happens to widen, it could be that one of the securities is overpriced, the other is underpriced, or the mispricing is a combination of both.

In this case, we are able to take advantage by selling the higher-priced security and buying the lower-priced one, expecting the mispricing to naturally correct itself in the future as prices converge to the equilibrium level.

The example below shows the relationship between AME and DOV, notice how the spread is stationary enough for entry and exit positions to be taken and profits to be made.



An example of cointegration pairs trading strategy.

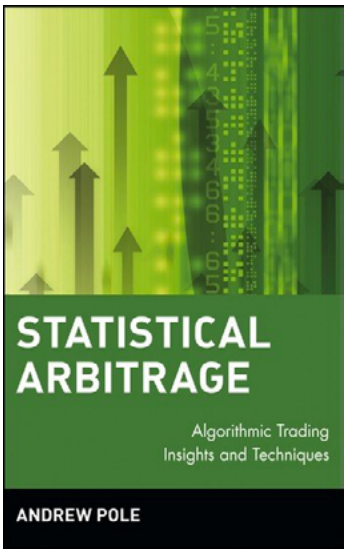
The mutual mispricing between the two assets is represented by the value of the spread. The greater the price difference from 0 and hence the spread, the greater the profit potential. One of the best qualities of pairs trading is **market-neutrality**, as by adjusting the hedge ratio of the spread it can be constructed to have a beta that is negligible, and therefore minimise the exposure to the market.

Pairs trading strategies usually vary in two ways: the way of detecting the "co-moving" pairs and the logic behind the trading rules. We dive into this topic in "A Taxonomy of Pairs Trading Strategies".

THE HISTORY OF PAIRS TRADING

It is hard to overestimate the influence that pairs trading has had on the industry. The initial members of the team consisted of mathematicians and physicists under Nunzio Tartaglia, who later branched off to establish such renowned independent practitioners as D.E. Shaw and PDT Partners, just to name a few. Many stories from the golden era of statistical arbitrage, about the business and its practitioners, have been mythologized by the industry.

For example how the SEC came to use algorithms from Morgan Stanley's Black Box to detect atypical price patterns or the gradual embrace of the technique by the independent specialists and later the practice of "bandwagoning" of the weak stock accumulation by the "big house".



As the information spread to a broader audience and the power of low-cost personal computers started to rise rapidly, the number of pairs trading practitioners rose as well. After two decades of prosperity, the swift evolution of the market called for an upgrade in approaches. It caused a significant diminish in returns for pairs trading when compared to the early days. Managers were struggling to adapt the strategies to the new reality and as a result, gradually started to withdraw investments. In his book **"Statistical Arbitrage: Algorithmic Trading Insights and Techniques"**, Andrew Pole called this the "ice age" of statistical arbitrage. Similar to the actual "ice age", it didn't last forever.

"I considered titling the book, The Rise and Fall and Rise? of Statistical Arbitrage, reflecting the history and the possibilities now emerging."

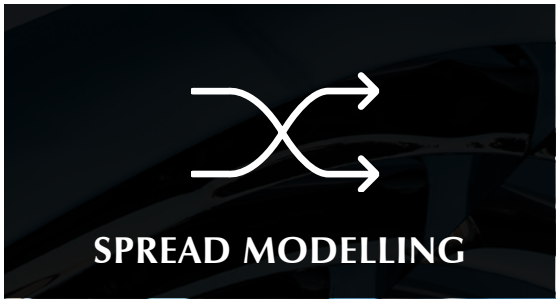
– Andrew Pole

Time was exactly what was needed for the ice to melt and give pairs trading its long-awaited new beginning. Ten years later the interest of the research community sparked yet again turning into the bright fire of technological advancement and continuous research in the field. As research in pairs trading continued to accumulate, it evolved into a truly versatile approach with a variety of frameworks with different complexity and empirical applications across various asset classes.

Numerous advances in techniques are continuously being added to our codebase as new research is released. This makes **ArbitrageLab** a perfect tool for investment managers who want to hit the ground running.

3 STEPS TO BUILDING A PAIRS TRADING STRATEGY

In a pairs selection step, our goal is to identify the co-moving securities. To do so, we utilise such methods as cointegration tests and distance metrics.

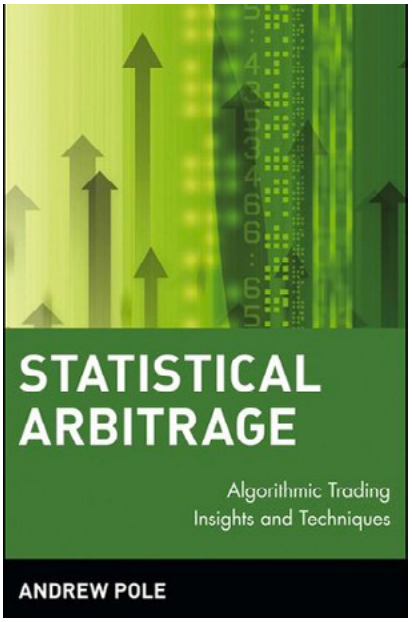


The next step is to model the spread in a way that maximizes mean-reversion and ensures the market neutrality to the strategy.

Lastly, the trading rules are established based on the type of utilised approach and the spread behaviour.

GETTING STARTED WITH THE BASICS

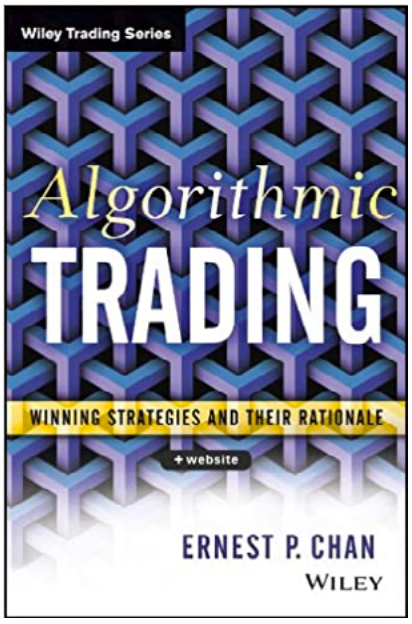
Recommended Literature:



Statistical Arbitrage: Algorithmic Trading Insights and Techniques

By Andrew Pole

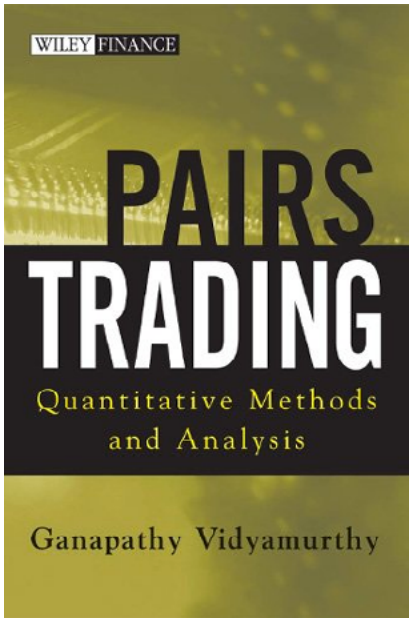
READ NOW



Alogorithmic trading

By Ernest P. Chan

READ NOW

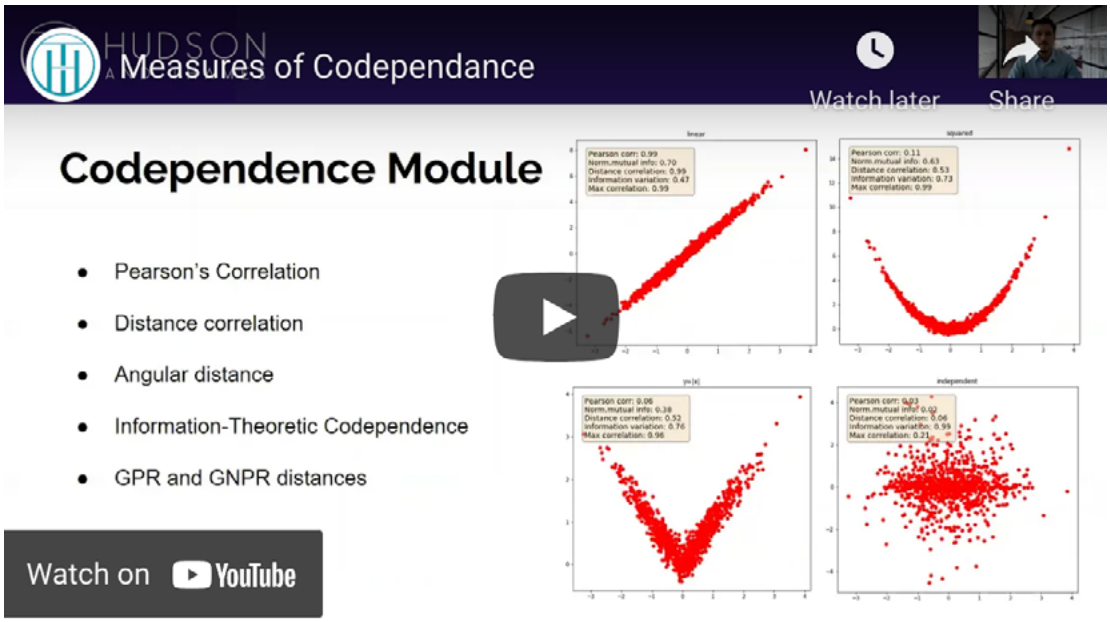


Pairs Trading: Quantitative Methods and Analysis

By Ganapathy Vidyamurthy

READ NOW

Pairs Trading Lectures Series:



PAIRS TRADING IN VARIOUS MARKETS

Previously we've discussed the "what's" and "why's" of pairs trading. Now it's time to move on to the question of "where?". Where do we apply pairs trading strategies? Building a strategy starts with pair selection, so our desire to choose the best market for it is only natural.

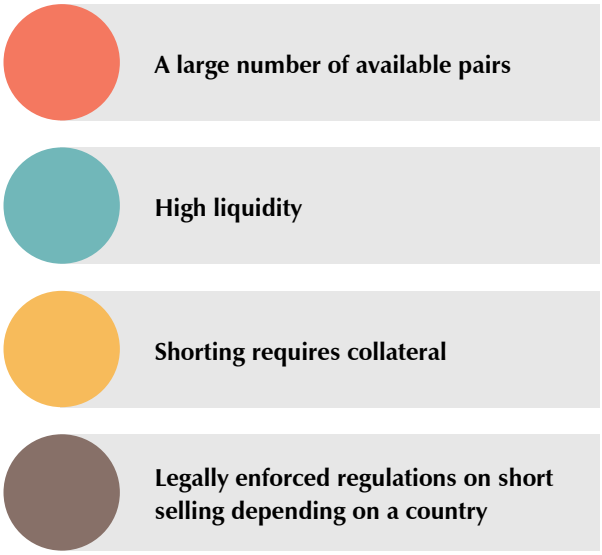
Since pairs trading bears its roots in equity markets, it usually serves as the first pick by most traders, this is largely due to the high number of possible combination pairs. However, asset classes such as commodities, forex, or even crypto have numerous supporting studies on the profitability of statistical arbitrage. Some have been around since 1990, as is the case for **commodities futures**; some are more recent, like **crypto markets**, however, they all support the universality of statistical arbitrage.

The requirement of "price co-movement" and "short-term price inefficiency" can be applied to every conceivable type of asset. Whether it's ETFs or options, if the market exhibits the violation of the "Law of One Price", statistical arbitrage will thrive.

With every market and every asset at our disposal, we need to be aware of the nuances that come with given asset types and how they influence our approach to strategy creation.

EQUITY & ETF

This is the first choice for most traders as it allows for the highest number of combination pairs. The **study by Jacobs & Weber** conducted on 34 international markets has empirically proven that pairs trading is the most profitable in emerging markets, either due to the higher inefficiencies, or markets with a large number of pairs. However, with the benefits of the stock and ETF markets come their limitations. Shorting is vital for statistical arbitrage strategies, so as the investor, we need to be aware of the market's requirements to be able to perform them. Applying statistical arbitrage strategies to equity or ETFs, we need to take into account three main points. First - short selling these types of assets requires collateral. Second - the less popular the stock or the ETF, the harder it is to find a willing lender to make the short possible, and third - legal restrictions on shorting can exist in different countries. An example of the latter could be "The uptick rule." previously used in the US stock market, or the short sale ban that existed in Chinese stock market.



FUTURES

Futures, and commodities futures specifically, are also no stranger to pairs trading. The phenomenon of price co-movement for the raw commodities, essential for statistical arbitrage, was discovered by studies as far as three decades ago. The advantage of choosing futures also has to do with shorting, as no additional requirements have to be fulfilled to perform the short sell. The nuances lie in the technical details connected with strategy backtesting or spread creation. First and foremost, the backtesting data has to be adjusted for the **futures roll** to simulate the correct PnL path. However, to size positions and determine capital consumption, we have to use raw prices. For some pairs trading strategies, it is also important to **account for contract size and implied volatility** to determine the correct hedging ratio.

- **A large number of available pairs**
- **High liquidity**
- **No restructions on shorting**
- **For correct backtesting, asset prices should be adjusted for the future rollover**
- **For some strategies contract size has to be taken into account to calculate hedge ratio**

FOREX

The forex market is not the most popular choice for statistical arbitrage, but still, a fair share of traders utilize pairs trading strategies using currency pairs. To create the desired spread, one would use a technique called “synthetic pairs”. The goal is to avoid a situation where directly trading the currency is impossible due to low liquidity, as provided by the broker. To do so, the institutional trader places two separate offsetting trades of target currencies with one highly trading currency(usually USD) acting as the intermediary(example: AUD/USD and USD/CAD). Another advantage of the forex market is the fact that shorting is as easy as selling the corresponding currency pair.

- **A large number of available pairs**
- **No restrictions on shorting**
- **Hard to create a truly market-neutral strategy**
- **To avoid liquidity issues, “synthetic pairs” have to be used**

CRYPTO

Despite being the youngest and having less research than previously mentioned assets, the cryptocurrency market is very attractive for statistical arbitrage, from a theoretical standpoint. Al-Yahyaee et al. found that Bitcoin's inefficiency is higher than stocks or forex markets, indicating that trading might have an even greater potential to be profitable here than elsewhere. However, the main limitation for crypto traders today is that only a few of the top traded coins are available for short-selling, and **not all exchanges** provide you with the option to do so. Additionally, transaction costs from brokers in the form of commissions and spreads can be larger than in more established markets such as equities.

High market inefficiency

Only a few cryptocurrencies are available for short selling

Not all exchanges support margin trade

A TAXONOMY OF PAIRS TRADING STRATEGIES

The pairs trading literature significantly grew its conceptual framework and empirical application base when the interest of the research community was sparked once again in the late 2010-s and continues to fuel the research till this day. As a large amount of literature on statistical arbitrage strategies accumulated - the necessity for classification arose.

Krauss (2015) recognizes 5 types of approaches:

- **Distance approach**
- **Cointegration approach**
- **Time series approach**
- **Stochastic control approach**
- **Other:** consisting of PCA, copula, and Machine Learning approaches

Let's take a deeper look into the taxonomy of pairs trading strategies.

PAIRS TRADING STRATEGIES

01

DISTANCE APPROACH

02

COINTEGRATION APPROACH

03

TIME SERIES APPROACH

04

STOCHASTIC CONTROL APPROACH

05

OTHER APPROACHES

- **PCA APPROACH**
- **COPULA APPROACH**
- **MACHINE LEARNING APPROACH**

01 | DISTANCE APPROACH

Popularized by **Gatev in 2006**, this approach holds the position of the most cited pairs trading strategy. The simplicity and transparency of this method make it the first choice when it comes to large empirical research.

Main idea:

During the pair selection process**, various distance metrics like pearson's correlation, distance correlation, angular distance, and so on**, are leveraged to identify the co-moving securities. In the trading period, simple nonparametric threshold rules are used to trigger the trading signals.

02 | COINTEGRATION APPROACH

Another extremely popular approach outlined by Vidyamurthy 2004, is the cointegration approach. Pairs selected in this method possess econometrically more reliable equilibrium relationships.

Main idea:

The pairs selection process is conducted by **applying a cointegration tests** to identify co-moving assets. Trading signals are generated by using simple rules, mostly generated by (Gatev et al 2006) threshold rules.

03 | TIME SERIES APPROACH

To improve the trading rules of a given strategy, we turn to time series modeling of a mean-reverting process, other than cointegration.

Main idea:

During the pairs selection step our goal is to **create a truly mean-reverting portfolio/spread**. The mean-reverting process of choice is fitted to the spread, in order to determine the optimal rules for the trading strategy. One of the most popular mean-reverting processes used in the literature is the **Ornstein-Uhlenbeck process**.

04 | STOCHASTIC CONTROL APPROACH

By using stochastic processes we can determine the optimal trading rules for a given mean-reverting strategy, without the need to forecast how the spread will move in the next period or needing a formation period.

Main idea:

This is an advanced pairs trading strategy that relies on using **stochastic processes** to generate the optimal trading rules and policies for mean reverting portfolios.

05 | OTHER APPROACHES

Although grouped as "other", the approaches contained in this bucket are one of the most advanced in the field, we will look at them one by one and expand on their differences.

COPULA

The Copula approach allows you to study the deeper relationship between the different legs of the trade and enables you to analyze the dependency structure from multiple random variables. This novel approach also allows you to work with multiple assets rather than only working with a single pair (2 assets). **The trading rules selection process** usually relies on the concept of conditional probability. **The data fitting process for copulas** is usually divided into two parts: translating marginal data into quantiles and fitting a copula with the following quantiles.

PCA

The approach proposed by Avellaneda and Lee (2010) uses PCA to create a seemingly mean-reverted spread which in turn is modeled by an OU-process. The trading rules are generated similarly to the time series approach, and the main focus is on the type of principal component analysis method utilized - asymptotic, regular, or a multivariate cointegration model. This model, at the time of writing, is considered by many to be the cutting edge of mean reversion trading.

MACHINE LEARNING

The machine learning approach is characterized by utilizing the techniques used in different approaches in combination with machine learning algorithms and applying them to strategy creation process.

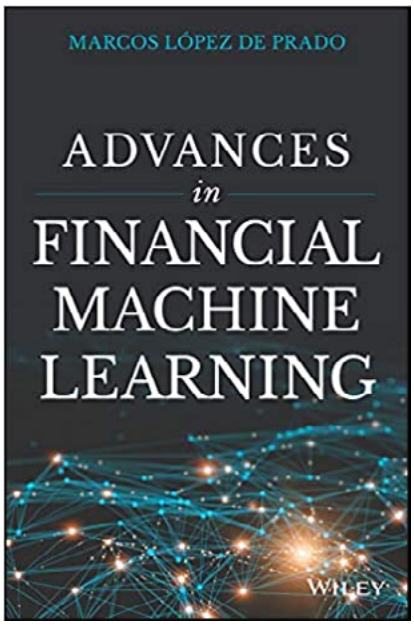
Whilst forecasting time series is a hard problem, and non-linear models show a lot of promise, we instead choose to focus on machine learning for pairs selection. The book "A Machine Learning based Pairs Trading Investment Strategy" by Sarmento and Horta provides a sophisticated and well thought out process to determine high-quality pairs. You can read all about it here: **Employing Machine Learning for Pairs Selection**.

BACKTESTING THE STRATEGY

"Backtesting is not a research tool. Feature importance is."

By Marcos Lopez De Prado

In his book Advances in Financial Machine Learning, Prof. Marcos Lopez de Prado defines the process of backtesting as a **"Historical simulation of how a strategy would have performed should it have been run over a past period of time"**. It means that backtesting doesn't provide us with the tools to deduce a precise cause-effect relationship akin to what **feature engineering** does. We can't guarantee that the Sharpe ratio achieved can be replicated if we travel back in time.



READ NOW

3 KEY BACKTESTING METHODS

The Walk-Forward Method

The walk-forward method is the most commonly used method in the industry. In this approach, the testing moves forward in time with the goal of preventing leakage from look-ahead information and can simply be thought of as a historical simulation / historical backtesting. This method assesses the performance of an investment algorithm under the assumption that history repeats itself exactly. However, it doesn't account for the randomness of the data-generating process. It only accounts for a single historical path.

The Resampling Method

The resampling method addresses the weakness of the walk-forward technique, by introducing the assumption that future paths can be simulated by resampling of past observations. The resampling methods can vary from deterministic (jackknife, cross-validation) to random (subsampling, bootstrap). The ability to produce different paths in return allows us to do things like bootstrapping the Sharpe ratio distribution, which was impossible with a single path simulation. In addition, it is more difficult to overfit a resampling backtest than the walk-forward one. And yet there is still a possibility that resampling from a finite historical sample may not be representative of the future.

The Monte-Carlo Method

This approach improves on what the previous two were lacking - a deeper understanding of the data generation process. It assesses the algorithm's performance under the assumption that future paths can be simulated via Monte Carlo and relies on knowledge derived from the statistical analysis of the observations or theory (market microstructure, institutional processes, economic links, etc.). An example from Lopes de Prado: "For instance, economic theory may suggest that two variables are cointegrated, and empirical studies may indicate the range of values that characterize the cointegration vector. Accordingly, researchers can simulate millions of years of data, where the cointegration vector takes many different values within the estimated range.This is a much richer analysis than merely resampling observations from a finite (and likely unrepresentative) set of observations (see Advances in Financial Machine Learning, chapter 13)"

NUANCES OF BACKTESTING

Backtesting is important for every strategy, however, it usually leads to overfitting and false investment discovery. The hope is that if done properly, it provides us with necessary insight to validate or discard the results of our research. Hence, it comes without saying that you have to be attentive to the nuances that come with the type of strategy you are using. Overlooking such details can render the whole procedure useless, even if all our assumptions are flawless and we account for every possible bias.

It is for this reason that the best researchers and practitioners advocate for the use of performance metrics not derived from a backtest. A good example of this is the ROC, and confusion matrices generated from classification models.

One particular problem that comes up when trying to backtest strategies that use futures contracts is adjusting for the roll and another is that the typical vectorized backtest methodology doesn't work when backtesting a strategy that has multiple long and short positions, as it is in mean reverting portfolios (pairs trading). A good example of this is our informative article on **The Correct Vectorized Backtest Methodology for Pairs Trading**. In the article, we highlight common pitfalls and how to correctly generate an equity curve / PnL plot.

CONCLUSION

Pairs trading is a truly versatile class of trading strategies that harbours enormous potential. **ArbitrageLab** is a collection of the landmark and most cutting edge implementations all packaged into a python library, making it perfect for investors who want to hit the ground running. The Definitive Guide to Pairs Trading was created to answer the questions arising whilst building these strategies.

Starting from the basics, we covered the history of pairs trading, potential benefits and nuances of different asset classes to trade, the taxonomy of a Pairs Trading Strategy, and finally notes on backtesting and a walk-forward backtesting framework **to help you in your journey to construct the ultimate pairs trading strategy**.

To dive into our latest research on constructing mean-reverting portfolios click **here**.

REFERENCES

1. Al-Yahyaee, K.H., Mensi, W. & Yoon, S.-M., 2018. Efficiency, multifractality, and the long-memory property of the Bitcoin market: A comparative analysis with stock, currency, and gold markets. Finance Research Letters, 27, pp.228–234.
2. Anon, 2015. Arbitrage and Pairs Trading. The Handbook of Pairs Trading, pp.89–95.
3. Chan, E., 2013. Algorithmic Trading: Winning Strategies and Their Rationale, John Wiley & Sons.
4. Ehrman, D.S., 2006. The handbook of pairs trading: strategies using equities, options, and futures, John Wiley & Sons.
5. Elliott, R.J., *, J.V.D.H. & Malcolm, W.P., 2005. Pairs trading. Quantitative Finance, 5(3), pp.271–276.
6. Endres, S. & Stübinger, J., 2019. Optimal trading strategies for Lévy-driven Ornstein–Uhlenbeck processes. Applied Economics, 51(29), pp.3153–3169.
7. Fil, M. & Kristoufek, L., 2020. Pairs Trading in Cryptocurrency Markets. IEEE Access, 8, pp.172644–172651.
8. Gatev, E., Goetzmann, W.N. & Rouwenhorst, K.G., 2006. Pairs Trading: Performance of a Relative-Value Arbitrage Rule. Review of Financial Studies, 19(3), pp.797–827.
9. Hagen, J.V., 1989. Relative Commodity Prices and Cointegration. Journal of Business & Economic Statistics, 7(4), pp.497–503.
10. Jacobs, H. & Weber, M., 2015. On the determinants of pairs trading profitability. Journal of Financial Markets, 23, pp.75–97.
11. Krauss, C., 2016. Statistical Arbitrage Pairs Trading Strategies: Review And Outlook. Journal of Economic Surveys, 31(2), pp.513–545.
12. Liew, R.Q. & Wu, Y., 2013. Pairs trading: A copula approach. Journal of Derivatives & Hedge Funds, 19(1), pp.12–30.
13. Mavrakis, E. & Alexakis, C., 2018. Statistical Arbitrage Strategies under Different Market Conditions: The Case of the Greek Banking Sector. Journal of Emerging Market Finance, 17(2), pp.159–185.
14. Pole, A., 2007. Statistical arbitrage: algorithmic trading insights and techniques, John Wiley & Sons.
15. Prado, M.L.de, 2018. Advances in Financial Machine Learning, Wiley.
16. Prado, M.L.D., 2019. Tactical Investment Algorithms. SSRN Electronic Journal.
17. Ramos-Requena, J.P., Trinidad-Segovia, J.E. & Sánchez-Granero, M.Á., 2020. Some Notes on the Formation of a Pair in Pairs Trading. Mathematics, 8(3), p.348.
18. Vidyamurthy, G., 2011. Pairs Trading: Quantitative Methods and Analysis, John Wiley & Sons, Inc.

2.0

AN INTRODUCTION TO COINTEGRATION FOR PAIRS TRADING

INTRODUCTION

Cointegration, a concept that helped Clive W.J. Granger win the Nobel Prize in Economics in 2003 (see Footnote 1), is a cornerstone of pairs and multi-asset trading strategies. Anecdotally, forty years have passed since Granger coined the term "cointegration" in his seminal paper "Some properties of time series data and their use in econometric model specification" (Granger, 1981), yet one still cannot find the term in Merriam-Webster, and some spell checkers will draw a wavy line without hesitation beneath its every occurrence.

Indeed, the concept of cointegration is not immediately apparent from its name. Therefore, in this article, I will attempt to answer the following questions:

1. **What does "integration" in the word "cointegration" refer to?**
2. **What are some intuitive interpretations of cointegrated time series?**
3. **How to construct a stationary spread from two non-stationary price series?**
4. **How to simulate a cointegrated asset pair from scratch?**

Hopefully, after reading this article, you will understand better why cointegration techniques, which were initially intended to avoid spurious regression results when using non-stationary regressors in macroeconomic time series analysis (McDermott, 1990), became an indispensable member of the statistical arbitrage arsenal. I will try my best to cut down the amount of hypnotizing math formulae as intuition is more important. Let's dive in!

WHAT IS COINTEGRATION?

Is it $\int f(x)\,dx$? I have to admit that every time I read about cointegration, the integral symbol would always pop into my head. But no, cointegration has nothing to do with the integral symbol.

The word "integration" refers to an **integrated time series of order *d***, denoted by $I(d)$. According to Alexander *et al.* (Alexander, 2002), price, rate, and yield data can be assumed as $I(1)$ series, while returns (obtained by differencing the price) can be assumed as $I(0)$ series (see Footnote 2). The most important property of the $I(0)$ series that is relevant to statistical arbitrage is the following:

$I(0)$ **series are weak-sense stationary**

FOOTNOTES

1. https://www.nobelprize.org/prizes/economic-sciences/2003/summary/

2. Astute readers might have noticed that I did not give a rigorous definition of $I(0)$ series. In fact, the definition of $I(0)$ is not clear-cut. See **When is a Time Series** $I(0)$? for a discussion about the definition of $I(0)$ series.

Weak-sense stationarity implies that the mean and the variance of the time series are finite and do not change with time. Mathematically, a stricter definition of stationary, or strict-sense stationary exists, but for financial applications, weak-sense stationarity is sufficient. Therefore, in the remaining part of this article, I will simply use "stationary" to refer to weak-sense stationary series.

This is great news. We found a time-invariant property, which is the mean of the time series. This implies the behavior of the time series has become more predictable, for if the time series wanders too far away from the mean, the time-invariant property will "drag" the series back to make sure the mean does not change. Sounds like we just described a mean-reversion strategy.

But wait, the $I(0)$ series is the returns: we cannot trade the returns! Only the price is tradable, yet the price is an $I(1)$ series, which are not stationary. We cannot make use of the stationary property of the $I(0)$ series by trading one asset.

What about two assets? According to the definition of cointegration (Alexander, 2002):

x_t and y_t are cointegrated, if x_t and y_t are $I(1)$ series and $\exists\, \beta$ such that $z_t = x_t - \beta y_t$ is an $I(0)$ series

Voilà! Cointegration allows us to construct a stationary time series from two asset price series, if only we can find the magic weight, or more formally, the cointegration coefficient β. Then we can apply a mean-reversion strategy to trade both assets at the same time weighted by β. There is no guarantee that such β always exists, and you should look for other asset pairs if no such β can be found.

INTUITIVE INTERPRETATION OF COINTEGRATION

Looks like our work is done if we figure out how to find the cointegration coefficient. But before we get gung-ho about β, it is helpful to establish an intuitive understanding of the definition of cointegration.

Going back to the definition, we have two $I(1)$ series, x_t and y_t, and they are cointegrated. Now we decompose x_t and y_t into a nonstationary component ν and a stationary component ε as follows (Vidyamurthy, 2004):

$$xt = \nu_{x_t} + \varepsilon_{x_t}$$

$$yt = \nu_{y_t} + \varepsilon_{y_t}$$

Now we construct the cointegrated series z_t, or the spread between x_t and y_t, with the cointegration coefficient β:

$$z_t = x_t - \beta y_t = (\nu_{x_t} - \beta \nu_{y_t}) + (\varepsilon_{x_t} - \varepsilon_{y_t})$$

The definition made clear that z_t is an $I(0)$ process, which indicates $\nu_{x_t} = \beta \nu_{y_t}$, as there can be no nonstationary component in a stationary time series.

Two takeaways from this derivation:

1. $\nu_{x_t} = \beta \nu_{y_t}$

The time series x_t and y_t share common nonstationary components, which may include trend, seasonal, and stochastic parts (Huck, 2015). This is an important revelation. When two assets are cointegrated, the underpinning factors that made their price non-stationary should be similar; or in financial terms, the two assets should have similar risk exposure so that their prices move together. For example, good candidates for cointegrated pairs could be:

- Stocks that belong to the same sector.
- WTI crude oil and Brent crude oil.
- AUD/USD and NZD/USD.
- Yield curves and futures calendar spreads (Alexander, 2002).

In other words, cointegration can be viewed as a similarity measure between two assets.

2. z_t is Stationary

We have discussed the properties of a stationary series previously. Here, the stationary series is the spread z_t. The time-invariant mean of z_t has two implications. For one thing, the mean of z_t is insensitive to time. This suggests that cointegration is a long-run relationship between two assets (Alexander, 2002; Galenko, 2012). For another, the invariability of the mean of the spread will keep the prices of the two assets tethered (Galenko, 2012), i.e. one asset cannot get excessively overpriced (or underpriced) against the other if cointegration holds.

The above derivation showcased the common trends model proposed by Stock and Watson (Stock, 1988). When two assets are cointegrated, they share "common stochastic trends". While the model is an oversimplified description of cointegrated financial time series, it did provide us with insight into what cointegration exactly means. To summarize:

- Cointegration describes a long-term relationship between two (or more) asset prices.
- Cointegration can be viewed as a measure of similarity of assets in terms of risk exposure profiles.
- The prices of cointegrated assets are tethered due to the stationarity of the spread.

A BRIEF DIGRESSION: CORRELATION VS COINTEGRATION

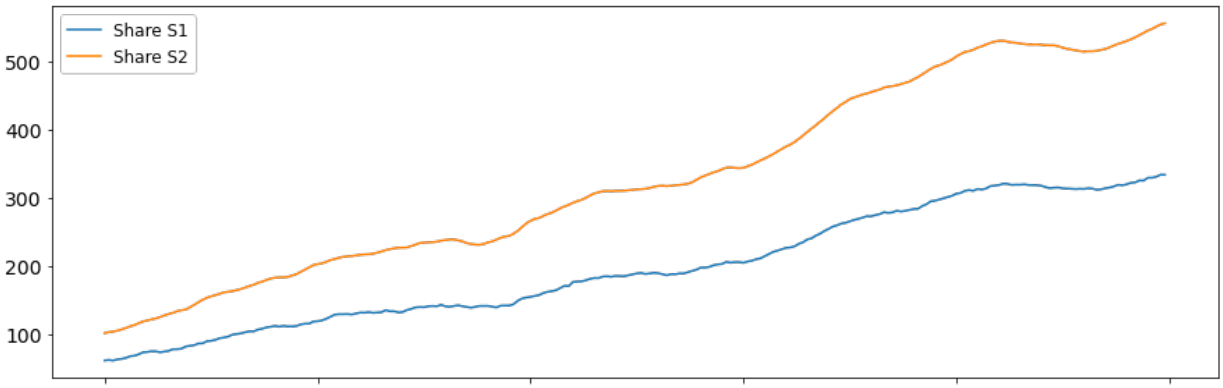
Since the topic of this article is cointegration, I would give away the conclusion first.

- **Correlation** has no well-defined relationship with cointegration. Cointegrated series might have low correlation, and highly correlated series might not be cointegrated at all.
- **Correlation** describes a short-term relationship between the returns.
- **Cointegration** describes a long-term relationship between the prices.

When we say two assets are correlated, the fact that the correlation is between the returns was implied. As we have discussed previously, asset prices are $I(1)$ series and returns are $I(0)$ series. When calculating the correlation coefficient of two assets, we are effectively performing a linear regression between the returns of asset A and asset B because the returns are stationary. Spurious correlation will occur if non-stationary price series are used in the regression. I will give two examples below and refer interested readers to (McDermott, 1990) and (Alexander, 2002) for more about this topic.

Figure 1 demonstrated an example of cointegrated series and the variation in its rolling 20-day correlation. Although the two asset price series are cointegrated and moving together, there exists a time period when the correlation between the returns of the two assets are negative. Figure 2 demonstrated an example of two highly correlated but not cointegrated series.

Simulated cointegrated series and the cointegration error, $\beta = -0.6$



(Figure 1: Continued on next page)

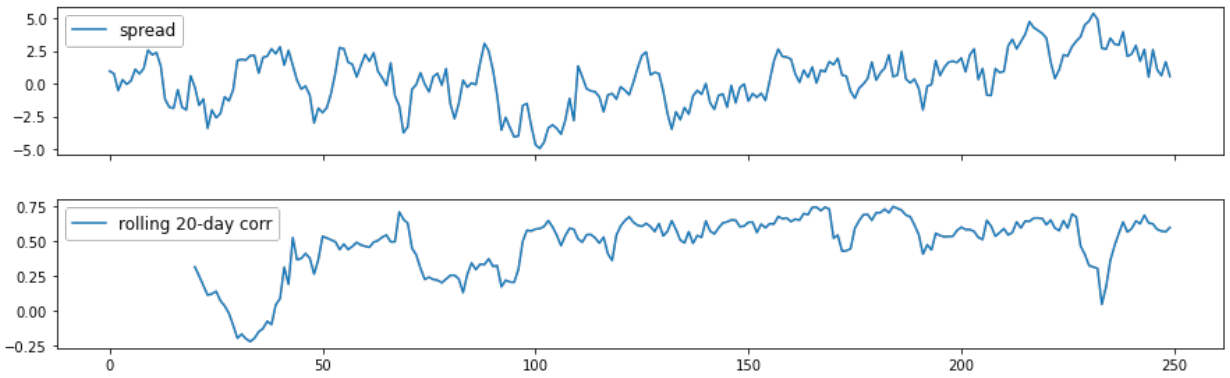


Figure 1. Cointegrated series can sometimes show negative correlation in returns.

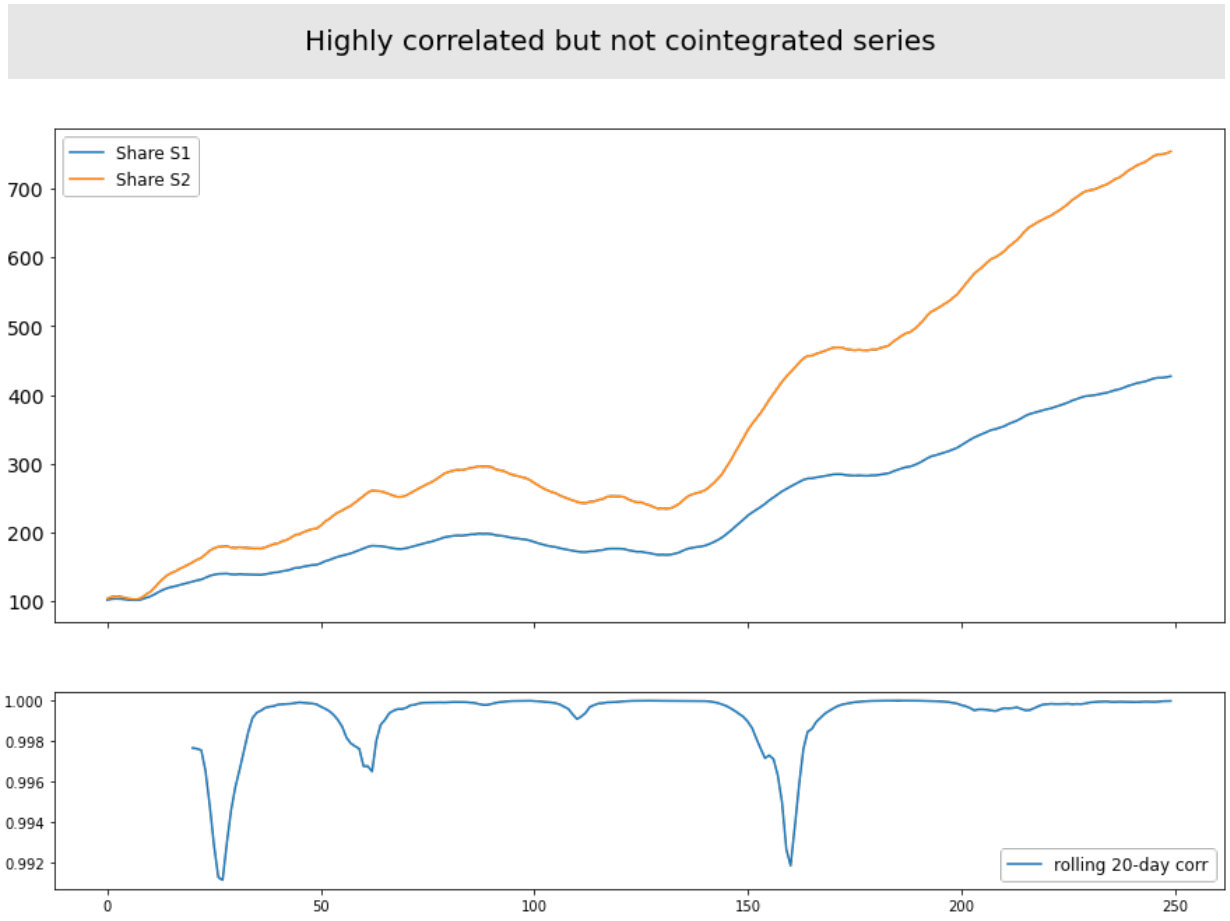


Figure 2. Two highly correlated price series but not cointegrated at all. Engle-Granger test returned an ADF-statistics value of 0.41, which cannot reject the null hypothesis (two series not cointegrated) even at 90% significance.

SIMULATION OF COINTEGRATED SERIES

Simulation is a powerful tool to understand a financial concept and cointegration is not an exception. The simulation of two cointegrated price series is now straightforward using the concepts introduced in the previous sections (Lin, 2006).

1. **Returns are stationary. So we start with the simulation the returns of one asset using a stationary AR(1) process.**
2. **Retrieve the price by summing up the returns.**
3. **The spread between the two assets is stationary. Again, simulating this spread with a stationary AR(1) process.**
4. **Derive the price of the other asset using the cointegration relation.**

This is exactly how ArbitrageLab helps you simulate the prices of a pair of cointegrated assets. Figure 3 demonstrates the result of such a simulation.

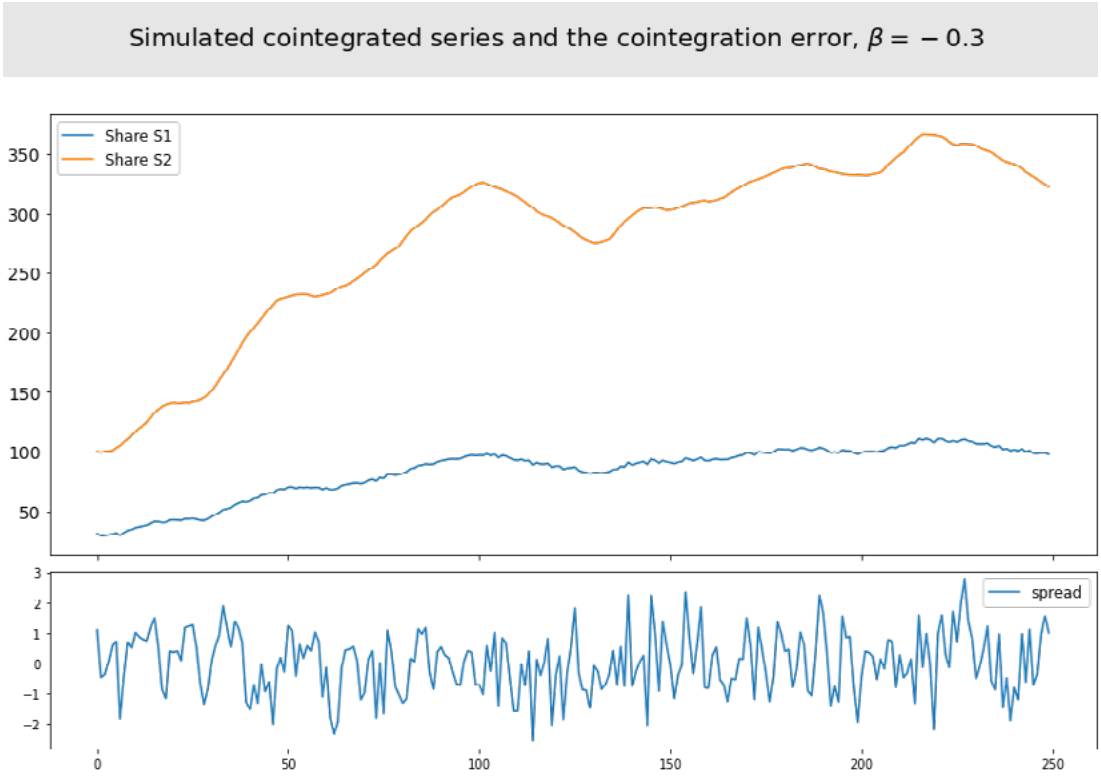


Figure 3. Cointegrated series simulation results from ArbitrageLab. The stationarity of the spread is demonstrated.

DERIVATION OF THE COINTEGRATION COEFFICIENT β

The two workhorses of finding the cointegration coefficient β (or cointegration vector when there are more than 2 assets) are the Engle-Granger test (Engle, 1987) and the Johansen test. I will focus on the comparison of the two methods in terms of application rather than the mathematical underpinnings to prevent this article from getting into the quagmire of endless linear algebra. I will refer curious readers to (Johansen, 1988) and (Chan, 2013) for a more comprehensive introduction to Johansen test and Vector Error Correction Model (VECM).

Engle-Granger Test

The idea of Engle-Granger test is simple. We perform a linear regression between the two asset prices and check if the residual is stationary using the Augmented Dick-Fuller (ADF) test. If the residual is stationary, then the two asset prices are cointegrated. The cointegration coefficient is obtained as the coefficient of the regressor.

An immediate problem is in front of us. Which asset should we choose as the dependent variable? A feasible heuristic is that we run the linear regression twice using each asset as the dependent variable, respectively. The final \beta would be the combination that yields a more significant ADF test.

But what if we have more than two assets? If we still apply the abovementioned heuristic, we will have to run multiple linear regressions, which is rather cumbersome. This is where Johansen test could come in handy.

Johansen Test

Johansen test uses the VECM to find the cointegration coefficient/vector β. The most important improvement of Johansen Test compared to Engle-Granger test is that it treats every asset as an independent variable. Johansen test also provides two test statistics, eigenvalue statistics and trace statistics, to determine if the asset prices are statistically significantly cointegrated.

In conclusion, Johansen test is a more versatile method of finding the cointegration coefficient/vector β than the Engle-Granger test. Both tests are implemented in ArbitrageLab, so finding the cointegration coefficient is no longer a problem.

A caveat: Raw prices or log prices?

Should we use raw prices or log prices in the cointegration tests?

One key theme in this article is that prices are $I(1)$ series (which includes log-prices), and returns are $I(0)$ series. $I(0)$ series can be obtained by differencing the $I(1)$ series. So it comes naturally that log prices fit this description better, for the difference of log prices is directly log returns, but the difference of raw prices is not percentage returns yet. However, according to (Alexander, 2002), "Since it is normally the case that log prices will be cointegrated when the actual prices are cointegrated, it is standard, but not necessary, to perform the cointegration analysis on log prices." So it is OK to analyze raw prices, but log prices are preferable.

CONCLUSION

Hopefully, this article helped you understand cointegration better. The key takeaways of this introductory are:

- **Cointegration describes a long-term relationship between asset prices.**
- **Cointegration can be seen as a measure of similarity of assets in terms of risk exposure profiles.**
- **The prices of cointegrated assets are tethered due to the stationarity of their spread.**
- **Correlation and cointegration are two different concepts. Correlation is a short-term relationship between returns, while cointegration is a l long-term relationship to prices. Do not use correlation for prices!**
- **Engle-Granger and Johansen test can help find the cointegration coefficient/vector β such that a cointegrated spread can be constructed.**
- **ArbitrageLab can help with the simulation of cointegrated series and the calculation of β.**

Now that we know how to construct a stationary spread from a pair or even a group of assets, it is time to design a trading strategy that can take advantage of the stationarity of the spread. In Part 2 of this series, I will demonstrate two mean reversion pair/multi-asset trading strategies that can guarantee a minimum profit. Keep tuned.

REFERENCES

1. Alexander, C., Giblin, I. and Weddington, W., 2002. Cointegration and asset allocation: A new active hedge fund strategy. Research in International Business and Finance, 16 (5), pp.65-90.
2. Chan, E., 2013. Algorithmic trading: winning strategies and their rationale (Vol. 625). John Wiley & Sons.
3. Engle, R.F. and Granger, C.W., 1987. Co-integration and error correction: representation, estimation, and testing. Econometrica: Journal of the Econometric Society, pp.251-276.
4. Galenko, A., Popova, E. and Popova, I., 2012. Trading in the presence of cointegration. The Journal of Alternative Investments, 15 (1), pp.85-97.
5. Granger, C.W., 1981. Some properties of time series data and their use in econometric model specification. Journal of Econometrics, 16 (1), pp.121-130.
6. Huck, N. and Afawubo, K., 2015. Pairs trading and selection methods: is cointegration superior?. Applied Economics, 47 (6), pp.599-613.
7. Johansen, S., 1988. Statistical analysis of cointegration vectors. Journal of economic dynamics and control, 12 (2-3), pp.231-254.
8. Lin, Y.X., McCrae, M. and Gulati, C., 2006. Loss protection in pairs trading through minimum profit bounds: A cointegration approach. Advances in Decision Sciences, 2006.
9. McDermott, C.J., 1990. Cointegration: origins and significance for economists. New Zealand Economic Papers, 24 (1), pp.1-23.
10. Stock, J.H. and Watson, M.W., 1988. Testing for common trends. Journal of the American statistical Association, 83 (404), pp.1097-1107.
11. Vidyamurthy, G., 2004. Pairs Trading: quantitative methods and analysis (Vol. 217). John Wiley & Sons.

3.0

SPARSE MEAN-REVERTING PORTFOLIO SELECTION

INTRODUCTION

"Buy low, sell high." One cannot find a more succinct summary of a mean-reversion trading strategy; however, single assets that show stable mean-reversion over a significant period of time such that a mean-reversion trading strategy can readily become profitable are rare to find in markets today. Even if such gems were found, celebrating the discovery of the gateway to easy money could prove premature:

- The asset may exhibit low volatility and the gap between the extreme prices of the asset and its long-term mean is too small. In this case, you can either try trading large size to capture enough profit with your risk desk's consent (good luck with that) or do not bother with it at all.
- The asset is hard to short-sell. Either due to regulatory restrictions or unexpected events, not being able to short-sell at the most opportune moment can significantly affect the profitability of the strategy.
- The mean-reversion occurs at a time scale of months or even years. In this case, you might as well try a momentum strategy on a shorter time scale instead of carrying out the mean-reversion strategy to a T.

Looks like searching for a magic asset that possesses perfect mean-reversion is impractical. Instead, building tradable mean-reverting long-short portfolios whose value is the weighted aggregate price of a basket of assets has become the focus of statistical arbitrageurs. The simplest form of this type of portfolios is an asset pair, and I have discussed in my previous article how to find such asset pairs using the concept of cointegration in detail. This article, on the other hand, extends the discussion into the territory of multi-asset (reads more than two assets) sparse mean-reverting portfolio selection.

"Why do I even need this new methodology? I can assemble multiple cointegrated pairs into a multi-asset mean-reverting portfolio anyway."

This is definitely feasible thanks to **machine learning**, but it could become confusing to build the portfolio if the pairs have overlapping legs. Say your favorite pair selection black box spits out a list of candidate mean-reverting asset duos whose spread value is stationary. One of these pairs longs asset A and shorts asset B, while another one longs asset C and shorts asset A. Now the ambiguity kicks in. Since both these pairs are stationary, any linear combination of them will stay stationary as well. There are theoretically infinite possibilities and it would become a nightmare scenario for decision-making when the number of overlapping pairs increases. Therefore, a multi-asset oriented methodology that can give more definitive solutions is preferred.
"OK. Sounds fair. So how many assets are adequate for such a mean-reverting portfolio?"

FOOTNOTES

1. The data used are daily data of 45 international equity ETFs starting from Jan 1st, 2016 to Jan 27th, 2021. The details of these ETFs can be downloaded here as an Excel spreadsheet.
2. $\binom{505}{5} = 268{,}318{,}178{,}226$ just in case you are wondering.

Now we're rolling! Let's see an example first (Footnote 1).

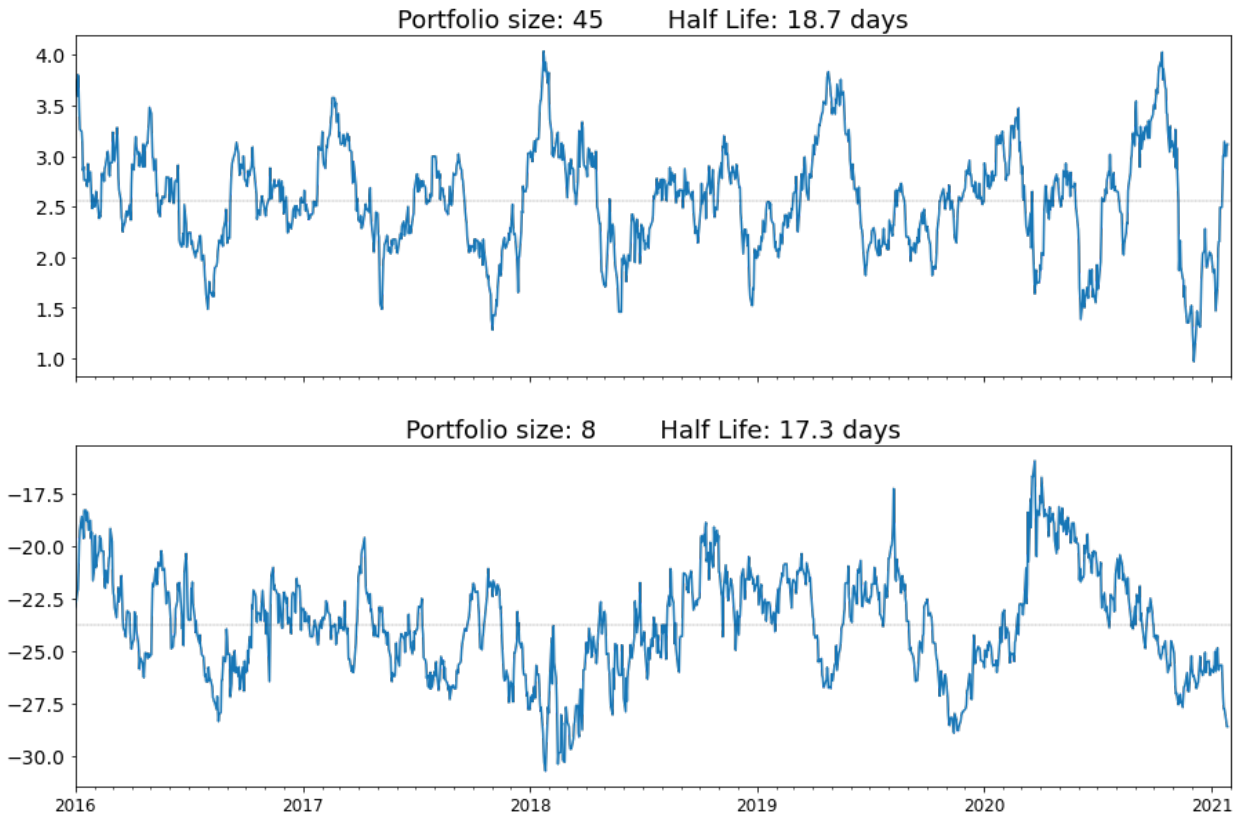
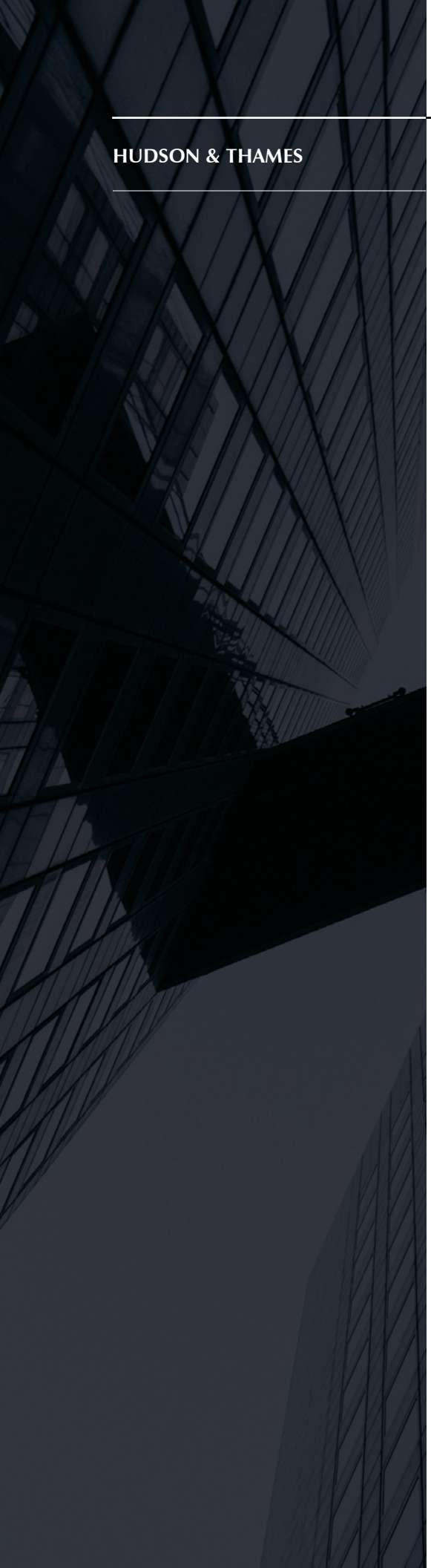


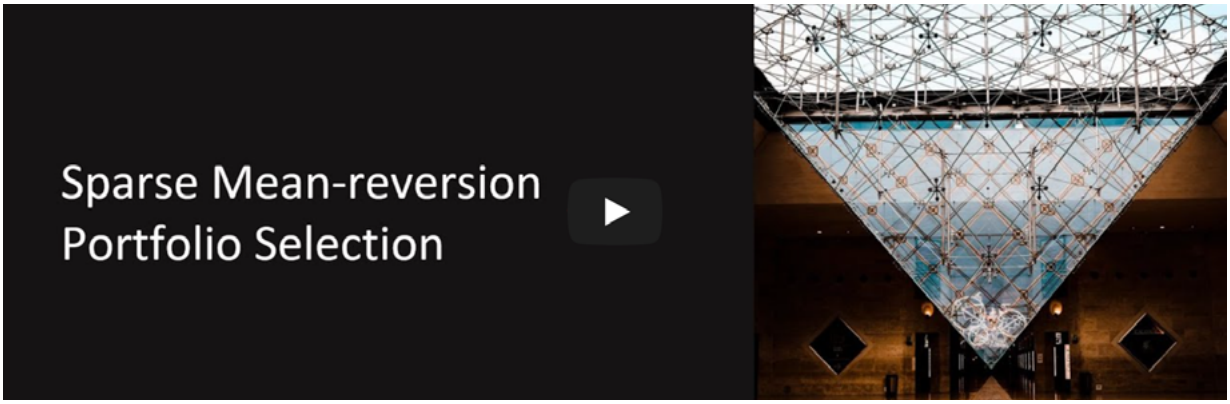
Figure 1. Two portfolios with similar mean-reversion properties but trading different number of assets.

Obviously, these two portfolios have similar mean-reversion properties. However, the first portfolio requires us to trade 45 assets, while the second one only requires eight! I would choose the second portfolio without a doubt as it is easier to manage, its P&L is more straightforward to interpret, and fewer assets means lower transaction costs. Sounds like winning! Therefore, the answer to the previous question is: while there are no strict rules for the exact number of assets to include in the portfolio, it is preferable to have as few assets as possible.

So how should we build the ideal multi-asset mean-reverting portfolio? As you are grabbing the snacks and water for the treasure hunt, I'll present you the treasure map:



The article follows very closely to the work of d'Aspremont (2011) and Cuturi and d'Aspremont (2015), so make sure you download these two papers from the reference section at the end of this article as well. **If you prefer watching a video instead of reading an article, I've got you covered as well.**



Let's now get to it!

YOU NEED TO QUANTIFY WHAT YOU WANT TO OPTIMIZE

Let's recapitulate the objective:

The portfolio, i.e. the linear combination of the assets, should be strongly mean-reverting.
The portfolio should be sparse, i.e. having more than two assets but not too many in the meantime.
Computers will not be able to understand qualitative descriptions such as "strongly mean-reverting" and "not too many assets", but these two quantities are the crux of finding a solution to this portfolio selection problem.

The "not too many assets" condition is straightforward. Suppose there are n assets in total in the investment universe, and we can simply set the number of assets to an integer k, where k << n; for example, we can set $k = \lfloor 0.1n \rfloor$. Then what about "strongly mean-reverting"? How should we measure strength of mean-reversion of a time series?

Metrics to measure mean reversion strength

Assuming the value of the mean-reversion portfolio follows an Ornstein-Uhlenbeck (OU) process (for more details on Ornstein-Uhlenbeck processes, check out our previous article or the wonderful book **Optimal Mean reversion Trading: Mathematical Analysis and Practical Applications** by Prof. Tim Leung and Xin Li (2015)):

$$d\Pi_t = \lambda(\bar{\Pi} - \Pi_t)dt + \sigma dZ_t$$
$$\Pi_t = \sum \binom{n}{i=1} x_i S_{ti}$$
$$\lambda > 0$$

where Π_t is the value of the portfolio at time t , $\bar{\Pi}$ is the long term mean level of the portfolio value, S_{ti} is the value of an asset S_i at time t, x_i is the weight for asset S_i, σ is the instantaneous volatility, Z_t is a standard Brownian motion, and the parameter λ is the mean reversion speed.

"Woah, slow down with the math!" You started rubbing your eyes. "Wait… Isn't this mean reversion speed parameter λ the metric we are looking for? The faster the mean reversion speed, the stronger the mean reversion. Problem solved!"

Indeed, with the introduction of λ, we can now formulate our sparse portfolio selection problem in more precise mathematical language:

$$\text{maximize} \quad \lambda$$
$$\text{subject to} \quad ||x||_0 = k$$

where $\|x\|_0$ is the cardinality, or the l_0 norm (both reads the number of non-zero elements) of the asset weighting vector x.

Say we find a weighting vector y that formed a portfolio with a reasonably high λ value. How do we know y is the optimal solution? If y is not the optimal solution, in which direction should we search for a better weighting? It is hard to answer these questions without a well-defined function between λ and x.

Other proxies of mean reversion strength

The moment of joy was evanescent. The problem still looks difficult to solve. But not all hopes are lost. The OU mean reversion speed parameter λ is not the only mean reversion strength metric available. Cuturi and d'Aspremont (2015) introduced three mean reversion strength proxies that make the above optimization problem easier.

Predictability and Box-Tiao canonical decomposition

Suppose that the portfolio value Π follows the recurrence below:

$$\Pi_t = \hat{\Pi}_{t-1} + \varepsilon_t$$

where $\hat{\Pi}_{t-1}$ is the predicted value of Π_t based on all the past portfolio values $\Pi_0, \Pi_1, \ldots \Pi_{t-1}$, and $\varepsilon_t \sim N(0, \Sigma)$ is a vector of i.i.d Gaussian noise independent of all past portfolio values up to $t-1$.

Now let $\mathbf{var}(\Pi_t) = \sigma^2, \mathbf{var}(\hat{\Pi}_{t-1}) = \hat{\sigma}^2$, and by definition we know $\mathbf{var}(\varepsilon_t) = \Sigma$.

Therefore,

$$\mathbf{var}(\Pi_t) = \mathbf{var}(\hat{\Pi}_{t-1} + \varepsilon_t) = \mathbf{var}(\hat{\Pi}_{t-1}) + \mathbf{var}(\varepsilon_t)$$

The second equal sign stands due to the independence between $\hat{\Pi}_{t-1}$ and ε_t.

Substitute in the variance, we get:

$$\sigma^2 = \hat{\sigma}^2 + \Sigma$$

$$1 = \frac{\hat{\sigma}^2}{\sigma^2} + \frac{\Sigma}{\sigma^2}$$

The predictability is then defined as:

$$\nu := \frac{\hat{\sigma}^2}{\sigma^2}$$

The intuition behind this variance ratio is straightforward. If ν is small, then this means the Gaussian noise is the dominant force and the portfolio value will look like pure noise, which is desirable as the noise is stationary and highly mean-reverting. However, when ν is large, the portfolio value is almost perfectly predictable based on its past information and this usually leads to a momentum portfolio. Therefore, we want to minimize ν to obtain a mean-reverting portfolio.

The remaining question is to derive the estimate $\hat{\Pi}_{t-1}$ and its variance $\hat{\sigma}^2$. Assumptions have to be made in order to get a tractable solution of the estimate. Remember that the portfolio is a linear combination of assets, i.e., $\Pi_t = x^T S_t$, according to the notation in the previous section. Without loss of generality, we will assume that each asset price S_{ti} is centered to zero mean before the analysis.

Box and Tiao (1977) proposed that the asset prices S_t could be assumed to follow a vector autoregressive model of order one – a VAR(1) model. Under this assumption, the predictability can be represented by covariance matrix of S_t, denoted by Γ_0, and the least square estimate of the VAR(1) coefficient matrix A:

$$\nu = \frac{x^T A \Gamma_0 A^T x}{x^T \Gamma_0 x}$$

Curious readers can refer to d'Aspremont (2011) and Cuturi and d'Aspremont (2015) to jostle your brain.

Let's see how the optimization problem looks now that we have an expression of ν.

$$\text{minimize } \nu = \frac{x^T A \Gamma_0 A^T x}{x^T \Gamma_0 x}$$

$$\text{subject to } \|x\|_0 = k$$

This is looking much nicer. In fact, this minimization problem without the cardinality constraint is a generalized eigenvalue problem (Ghojogh, 2019) that can be solved with ease with SciPy, and the minimum is achieved when x is the eigenvector corresponding to the smallest eigenvalue. Let's see an example of a portfolio of low predictability and one of high predictability.

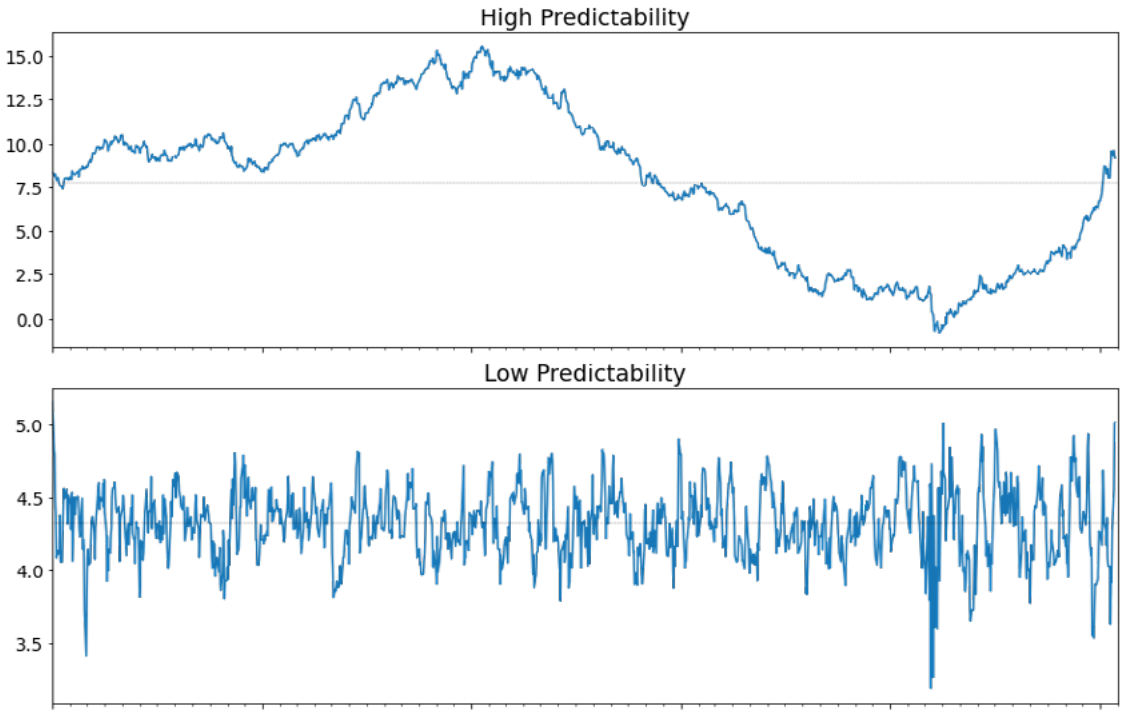


Figure 2. A comparison of a highly predictable portfolio to a barely predictable portfolio.

It is apparent that the more predictable portfolio seems a great candidate for momentum strategies, while the less predictable portfolio looks very much like noise and highly mean-reverting (and perhaps too noisy to trade). This suggests that substituting the OU-model mean-reversion speed with predictability is a move towards the right direction.

Portmanteau and crossing statistics

Before we move onto the next part, two other proxies for measuring mean-reversion strength are worth noting.

Portmanteau statistic

The portmanteau statistic of Ljung and Box (1978) is used to test whether a process is white noise. For our portfolio process Π_t, denote the lag-k autocovariance matrix of the asset prices S_t by Γ_k, the portmanteau statistic can be written as:

$$\hat{\phi}_p(x) = \frac{1}{p} \sum \binom{p}{i=1} \left(\frac{x^T \Gamma_i x}{x^T \Gamma_0 x} \right)^2$$

The smaller the portmanteau statistic, the stronger the mean-reversion.

Crossing statistic

Kedem and Yakowitz (1994) define the zero crossing rate of a univariate process $\boldsymbol{s_t}$, i.e. the expected number of crosses around 0 per unit of time, as

$$\gamma(x) = E\left[\frac{\sum_{t=2}^{T} \mathbf{1}_{\{s_t s_{t-1} \leq 0\}}}{T-1} \right]$$

The larger the crossing statistic, the stronger the mean reversion.

Directly optimizing this statistic is difficult. However, if we assume that $\boldsymbol{s_t}$ follows a stationary AR(1) process, then the zero crossing rate can be simplified using the cosine formula:

$$\gamma(x) = \frac{\arccos(a)}{\pi}$$

where a is the AR(1) coefficient, or the first-order autocorrelation. The stationarity implies $|a| < 1$, therefore, the smaller the first-order autocorrelation, the larger the crossing statistic. This result can be readily extended to the multivariate case, where we minimize the first-order autocorrelation and keep all absolute higher order autocorrelations small.

SPARSE IS NP-HARD

We have figured out the objective function to optimize, which is predictability. Now it is time to focus on the sparsity constraint.

"How hard can that be?" You murmured to yourself. "As the old saying goes, 'code first, optimize later'. Let's try brute force search first."

For the sake of demonstration, let's say the goal is to build a 5-asset portfolio from 30 candidate assets.

"Fine. 30 choose 5… is 142,506. Wait, this is going to take a while to enumerate every one of them."

Now suppose we want to do the same thing but with all 505 S&P 500 constituents.

"505 choose 5… This is impossible!" (Footnote 2)

Indeed, once the cardinality constraints has been added, the easy-to-solve generalized eigenvalue problem suddenly became a hard combinatorial problem, and Natarajan (1995) has shown that it is equivalent to subset selection, which is NP-hard. This means there is no algorithm with polynomial running time that can yield the optimal solution.

However, financial practitioners are usually not perfectionists. A sub-optimal sparse portfolio that has decent mean-reversion strength is preferable if it costs much less computing power. The remainder of this article will focus on a few different approaches to getting good approximate solutions that ensure the sparsity of the portfolio.

Narrow down the investing universe

Let's get back on the challenge of building a 5-asset portfolio from 30 candidate assets. Suppose that after analyzing the intrinsic structure of the asset price processes, we managed to find three clusters of assets, each of which contains 8 assets; and 6 isolated assets with their idiosyncrasies. Instead of examining all possible 5-asset combinations from 30 assets, we only consider the combinations within the clusters. Time to do some counting:

$$\binom{30}{5} = 142,506$$

$$3 \times \binom{8}{5} = 168$$

$$\frac{142,506}{168} = \mathbf{848.25}$$

A computation reduction by a factor of 848.25! This is now even manageable for a brute-force search algorithm.

The immediate advantage of adding a preprocessing step has naturally led to the question: "how should we do this?" While the instinctive reaction would be **machine learning**, I would like to introduce something rather vanilla in flavor – the family of Least Absolute Shrinkage and Selection Operators (Lasso) and their application in covariance selection and structured VAR model estimation.

Covariance selection

The inverse covariance matrix, Γ_0^{-1}, is a wealth of information on the dependencies of the asset price data. Under the assumption of a Gaussian model, zeroes in Γ_0^{-1} are indicators of conditional independence between variables, which reflects "independence between the idiosyncratic components of asset price dynamics." (d'Aspremont, 2011) Therefore, if we are able to set a reasonable portion of elements in Γ_0^{-1} to zeroes, the remaining elements can represent the underlying structure of the asset price dependencies.

"Structure? What structure? You can't possibly expect me to see some fascinating structure just by staring at a matrix as if I am in the Matrix!"

Don't worry, the visualization is on its way. Note that covariance matrix Γ_0 is a symmetric matrix, hence its inverse Γ_0^{-1} is also symmetric. This means we can use an undirected graph to represent the underlying structure of asset price dynamics. Each asset can be represented as a vertex. The non-zero element indicates an edge exists between two nodes depending on its position. The sparse estimate was obtained by graphical Lasso model, which solves a penalized maximum likelihood estimation problem:

$$\max_X \ln \det X - \mathbf{Tr}(\Sigma X) - \alpha \|X\|_1$$

$\alpha > 0$ is a parameter which determines how many zero elements there will be in the final sparse estimate. The larger α is, the more elements will be penalized to zero, hence the sparser the estimate.

Let's now compare the graph obtained from a dense covariance estimate and a sparse estimate. By gradually increasing α from 0.6 to 0.9, the edges are quickly eliminated and the final estimate are quite sparse.

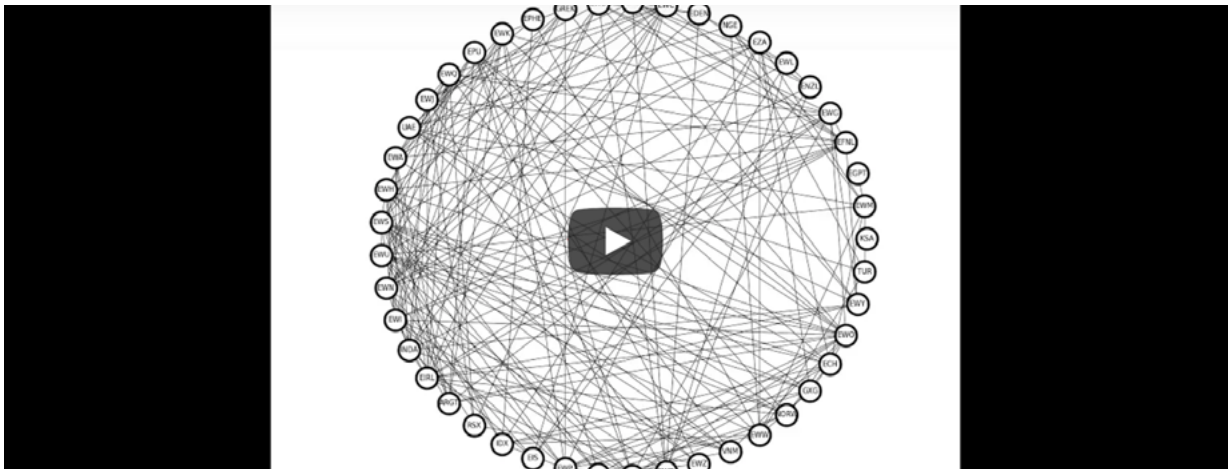


Figure 3. Covariance selection at work. The graph is a representation of the inverse covariance matrix Γ_0^{-1}.

What about the covariance matrix itself?

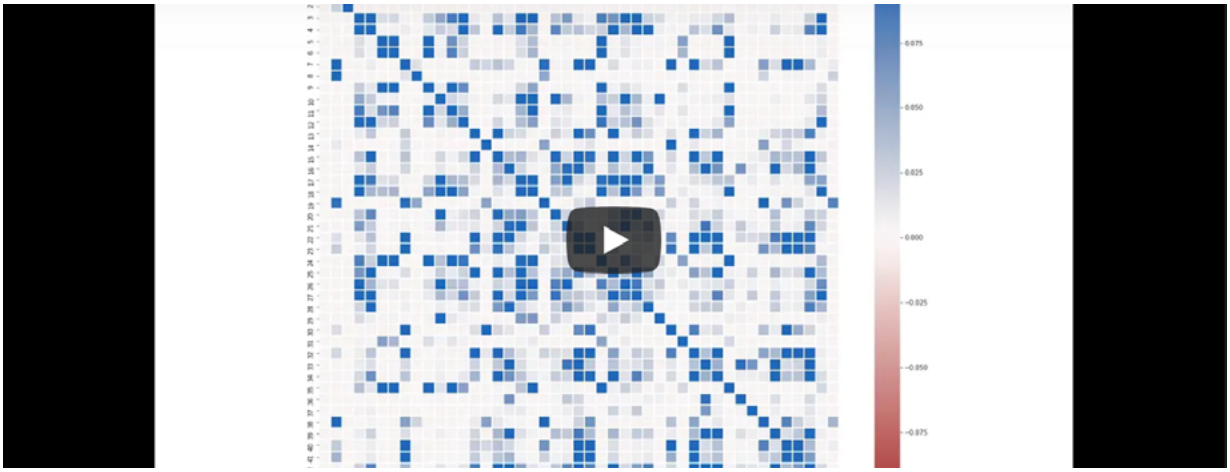


Figure 4. Covariance matrix obtained from the covariance selection procedure. Most matrix elements have been reduced to 0 as α increases.

It's super effective! However, to obtain the smaller clusters, we still need another puzzle piece, which is a structured VAR model estimate.

Structured VAR estimate
Recall the assumption that the asset prices S_t follow a VAR(1) model:

$$S_t = S_{t-1}A + Z_t$$

where S_{t-1} is the lag-one process of S_t, and $Z_t \sim N(0, \Sigma)$ is an i.i.d Gaussian noise independent of S_{t-1}. The dense least square estimate of A is then obtained by minimizing the objective function:

$$\arg\min_A \|S_t - S_{t-1}A\|^2$$

Adding an $\ell_1 - norm$ penalty to the objective function will make A sparse. The power of Lasso lies in that we can append different forms of $\ell_1 - norm$ to exert control on the final structure of the sparse estimate of A. We can either apply a univariate Lasso column-wise and optimize the following objective function n times:

$$\arg\min_a \|S_{it} - S_{t-1}a\|^2 + \lambda\|a\|_1 \text{ for all } i$$

Or we can apply multi-task Lasso and get the sparse estimate of A in one go by optimizing the following objective function:

$$\arg\min_A \|S_t - S_{t-1}A\|^2 + \lambda \sum_i \sqrt{\sum_j a_{ij}^2}$$

where a_{ij} is the elements in matrix A.

Figure 5 and 6 demonstrated how the structure of A will evolve as we increase the regularization parameter λ under two different $\ell 1$ - norm penalties. As you can see, the multi-task Lasso will suppress the coefficients in an entire column to zero, but the solution will become less robust as λ increases. The univariate Lasso applied column-wise, on the other hand, will lead to a much more scattered structure but more stable.

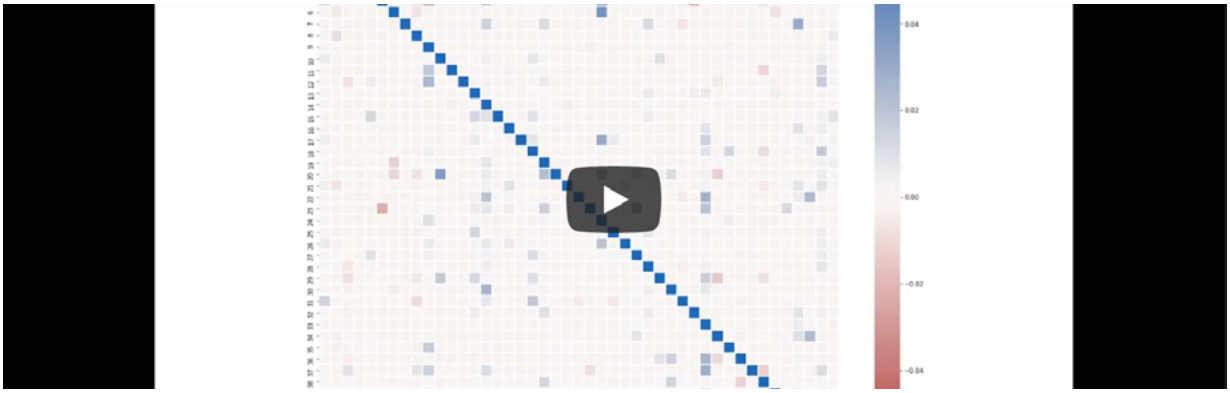


Figure 5. Univariate Lasso applied column-wise for structured VAR(1) estimate.

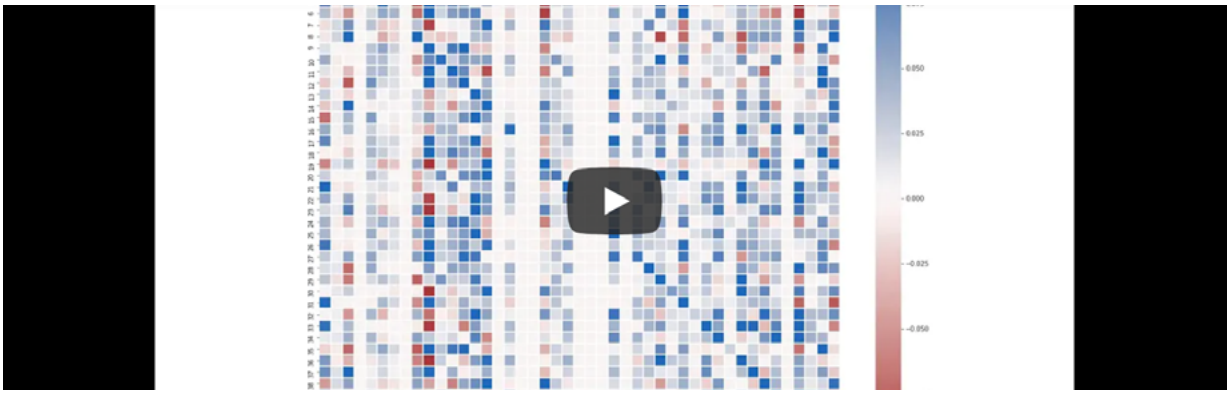


Figure 6. Multi-task Lasso applied column-wise for structured VAR(1) estimate.

LOCK IN THE CLUSTERS USING BOTH SPARSE ESTIMATES

Both the inverse covariance matrix Γ_0^{-1} and the VAR(1) coefficient matrix A contains the conditional dependence structure of the asset price dynamics. If we can combine the information from both matrices, we can further pinpoint the clusters from which we can build our sparse mean-reverting portfolio.

While Γ_0^{-1} has a readily available undirected graph representation, A does not have one due to its asymmetry. If we can find a way to construct an undirected graph representation for A, the intersection between the graph of Γ_0^{-1} and A will highlight the asset clusters we are looking for.

Gilbert (1994) has shown that if the covariance matrix of the i.i.d. Gaussian noise Z_t in the VAR(1) model is diagonal, then the graph of Γ_0^{-1} and $A^T A$ will share the identical block-diagonal structure. The important takeaway from this result is that we can use $A^T A$, which is perfectly symmetric, as the graph representation for conditional dependence structure.

Now we can wrap up our preprocessing procedure as all necessary information has been gathered.

1. **Use graphical Lasso to get a sparse estimate of the covariance matrix Γ_0. Apply a sufficiently large regularization parameter to break down the graph of Γ_0^{-1} into small clusters.**
2. **Use column-wise univariate Lasso or multi-lask Lasso to get a sparse estimate of the VAR(1) coefficient matrix A. Calculate the graph representation $A^T A$.**
3. **Retrieve the intersection graph of Γ_0^{-1} and $A^T A$.**
4. **The connected components of the intersection graph are the clusters we are looking for.**

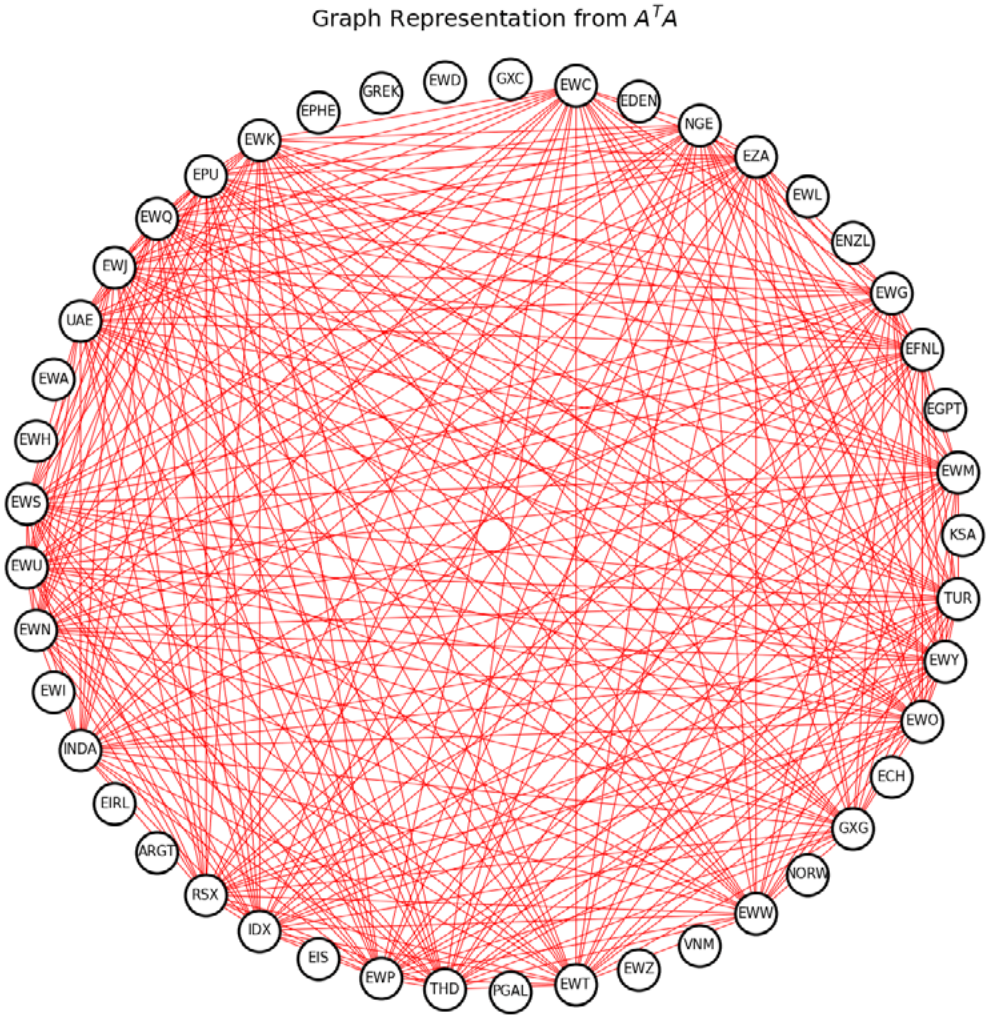


Figure 7. The clustering process.

In practice, successful clustering is highly dependent on covariance selection rather than the structured VAR(1) estimate. It is thus recommended to set a large regularization parameter for the graphical Lasso model for a highly sparse covariance matrix estimate (but of course you don't want to eliminate all the edges in the graph!)

Construct a sparse solution

Recall that our portfolio selection problem has been formulated as:

$$\text{minimize } \nu = \frac{x^T A \Gamma_0 A^T x}{x^T \Gamma_0 x}$$

$$\text{subject to } \|x\|_0 = k$$

A quick-and-dirty sub-optimal solution can be built from scratch with a greedy algorithm. Let's watch an animated example instead of resorting to math again.

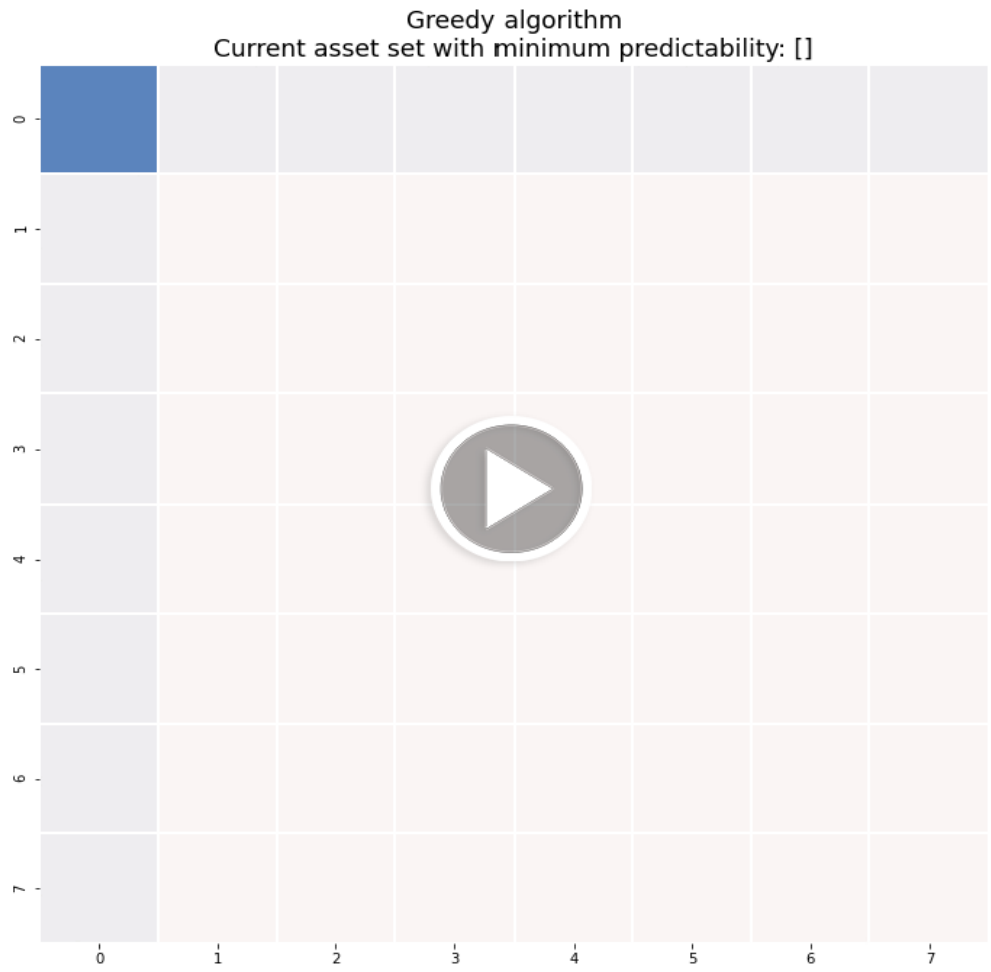


Figure 8. An animated explanation of the greedy algorithm.

The key idea here is to construct the portfolio recursively. At the start of the algorithm, we haven't selected any asset into our portfolio and thus the portfolio has zero predictability. At k-th iteration, we look to add the asset that leads to the smallest increase in the portfolio predictability by solving the generalized eigenvalue problem $n - k$ times for each possible asset choice as shown in Figure 8.

Why is greedy algorithm much faster? Because we ignored many candidate portfolios. Let's say we found an m-asset ($m < k$) portfolio $\mathbf{\Pi}_a$ with minimal predictability. Now if we want to add one more asset, we will not calculate the predictability for all possible portfolios with $m + 1$ assets. Rather, we will build the portfolio based on $\mathbf{\Pi}_a$. This effectively reduced the number of portfolios under examination from $\binom{n}{m+1}$ to $n - m$, which is a huge improvement.

The efficiency comes with a tradeoff. Since so many candidate portfolios are not even considered, the solution given by the greedy algorithm might be far-from-optimal. Following the example in Figure 8, once the greedy algorithm selected Asset #2, it will keep building the portfolio based on this first selection. However, the optimal portfolio might not contain Asset #2 at all if we comb through every possible portfolio. This may sound like a serious disadvantage, but Fogarasi (2012) tested with simulated data and found that brute force search was able to produce a stronger mean-reversion portfolio than the greedy algorithm only 59.3% of the time. In other words, brute force search guarantees the optimal solution while greedy algorithm does not; but approximately four out of ten times, the greedy algorithm will generate the exact solution as brute force search while saving huge amount of computing time. Greed is good indeed!

Relax the constraints

The cardinality constraint $\|x\|_0 = k$ is the culprit for all the headaches so far because it made the portfolio selection problem non-convex. Here, the concept "convex" means "when an optimal solution is found, then it is guaranteed to be the best solution", which is quite different from the fearful sense related to notorious derivatives trading debacles.

There are already a handful of tools for solving convex problems, but they are not directly portable to non-convex ones that have potentially many local optima or saddle points. Disappointing, innit? Not exactly. An eureka moment is right around the corner.

What if we loosen the non-convex cardinality constraint to a convex counterpart? Of course, the optimization problem will have changed drastically when we change the constraints, and the optimal solution for the new problem is not necessarily the optimal solution for the original one; but again, we will happily offer optimality for computational speed.

You may have wondered why I have been constantly referring to the cardinality as ℓ_0-norm in this article, and here is the reason:

ℓ_1-norm is the convex counterpart of the ℓ_0-norm constraint.

This is an oversimplification of the relaxation technique, but helpful for building intuition nonetheless. So how does our optimization problem look like after the convex relaxation?

Instead of working with the weighting vector $\boldsymbol{x}$, we will now work with the semidefinite matrix $\boldsymbol{X} = \boldsymbol{x}\boldsymbol{x}^T$ (Lovász and Schrijver, 1991). The portfolio selection problem can be rewritten as:

$$\text{minimize} \quad \mathbf{Tr}(A\Gamma_0 A^T X)/\mathbf{Tr}(\Gamma_0 X)$$

$$\text{subject to} \quad \|X\|_1 \leq k$$

$$\mathbf{Tr}(X) = 1$$

$$X \succeq 0$$

The l_1 norm $\|X\|_1$ is simply the sum of the absolute value of all elements in the symmetric matrix $\boldsymbol{X}$. The extra constraint $\mathbf{Tr}(X) = 1$ is equivalent to normalizing the weighting vector $\boldsymbol{x}$ to a unit vector. This could help with numerical stability. While the constraints are convex, the objective function is only quasi-convex because it is a ratio of two matrix traces, and thus cannot be readily solved by convex optimization methods.

There are two ways to get around this issue. One approach is to do a clever change of variables, and the above optimization problem can be solved via semidefinite programming (SDP). I will leave out the mathematical legerdemain and refer you to d'Aspremont (2011) for details, for this method has not been able to generate consistent results in experience despite it retained the flavor of the original optimization problem. Specifically, the cardinality parameter $\boldsymbol{k}$ did not bear its supposed meaning because the optimal X^* was almost never a rank-one matrix in real application, and the leading eigenvector of X^* in most cases had higher cardinality than expected. We could derive the sparse leading eigenvector of X^* with exactly k non-zero weights using the truncated power method (Yuan and Zhang, 2013), but the resulting portfolios would take months to mean-revert, which is undesirable.

The other approach is to rewrite the optimization problem with a ℓ_1-norm penalty following Cuturi and d'Aspremont (2015):

$$\text{minimize} \quad \mathbf{Tr}(A\Gamma_0 A^T X) + \rho\|X\|_1$$

$$\text{subject to} \quad \mathbf{Tr}(\Gamma_0 X) \geq V$$

$$\mathbf{Tr}(X) = 1$$

$$X \succeq 0$$

Comparing to the original objective function $\mathbf{Tr}(A\Gamma_0 A^T X)/\mathbf{Tr}\ (\Gamma_0 X)$, this formulation focuses on minimizing the numerator of the ratio while keeping the denominator above a threshold. The addition of the ℓ_1-norm penalty introduces sparsity. Everything makes sense mathematically, but are there any applicable financial interpretations?

The answer is a resounding yes. Note that the term $\mathbf{Tr}(\Gamma_0 X)$ represents the variance of the portfolio and the constraint guarantees that the variance exceeds V. This optimization problem is de facto minimizing portfolio predictability while ensuring the portfolio value has sufficient volatility! We also made sure that not too many assets are included in the portfolio with the ℓ_1-norm penalty. But wait, there's more: this formulation can be used to optimize portmanteau and crossing statistic as well!

The only drawback of this approach is that the cardinality is no longer clearly defined as it is determined by the ℓ_1-regularization parameter ρ, which means a bit more tweaks are required to achieve the desired cardinality, but we can't really complain as the flexibility and interpretability we have gained via this formulation far outweighs the minor inconvenience.

Compared to the cut-to-the-chase greedy algorithm, the convex relaxation approach is more complicated in terms of theory and time complexity, but its versatility allows more control over the behavior of the portfolio. Curious readers can refer to Cuturi and d'Aspremont (2015) for a deep-dive into the application of convex relaxation approach to solving the sparse mean-reverting portfolio problem.

TALK IS CHEAP, SHOW ME THE RESULTS!

I am extremely impressed if you are still with me from the start, as I have already fatigued myself from explaining all the theories. Now let's see how ArbitrageLab can simplify the sparse mean-reverting portfolio construction process with a few line of codes. Supposedly, the data have already been properly processed and all missing values have been imputed. Since the ETF data was used for demonstration (Footnote 1), I will assume the price data has already been saved in the variable etf_df.

Asset Clustering

In the previous section, I discussed how to use sparse estimates of covariance matrix and VAR(1) coefficient matrix to find smaller asset clusters. Box-Tiao canonical decomposition can be then applied to the small cluster to get a sparse mean-reverting portfolio. This can be readily done with ArbitrageLab.

```
from arbitragelab.cointegration_approach.sparse_mr_portfolio
import SparseMeanReversionPortfolio
import networkx as nx
etf_sparse_portf = SparseMeanReversionPortfolio(etf_df)
# Calculate the penalized estimates
sparse_var_est = etf_sparse_portf.LASSO_VAR_fit(1.4, threshold=7, multi_task_lasso=True,
                                                use_standardized=True)
_, sparse_prec_est = etf_sparse_portf.covar_sparse_fit(0.89, use_standardized=True)
# Generate the clusters
multi_LASSO_cluster_graph = etf_sparse_portf.find_clusters(sparse_prec_est, sparse_var_est)
# Initialize a new class instance with the smaller cluster
small_cluster = list(sorted(nx.connected_components(multi_LASSO_cluster_graph),
                            key=len, reverse=True))[1]
small_etf_portf = SparseMeanReversionPortfolio(etf_df[small_cluster])
# Perform Box-Tiao decomposition
bt_weights = small_etf_portf.box_tiao()
# Retrieve the portfolio weights
most_mr_weights = bt_weights[:, -1]
medium_wr_weights = bt_weights[:, 4]
least_mr_weights = bt_weights[:, 0]
```

The clustering results have shown that a 15-asset cluster and an 8-asset cluster have been formed with this data set. To demonstrate Box-Tiao canonical decomposition, I have chosen the 8-asset cluster due to its smaller size. Three portfolios have been built according to the Box-Tiao weights as shown in Figure 9 ranging from the least mean-reverting to the most mean-reverting.

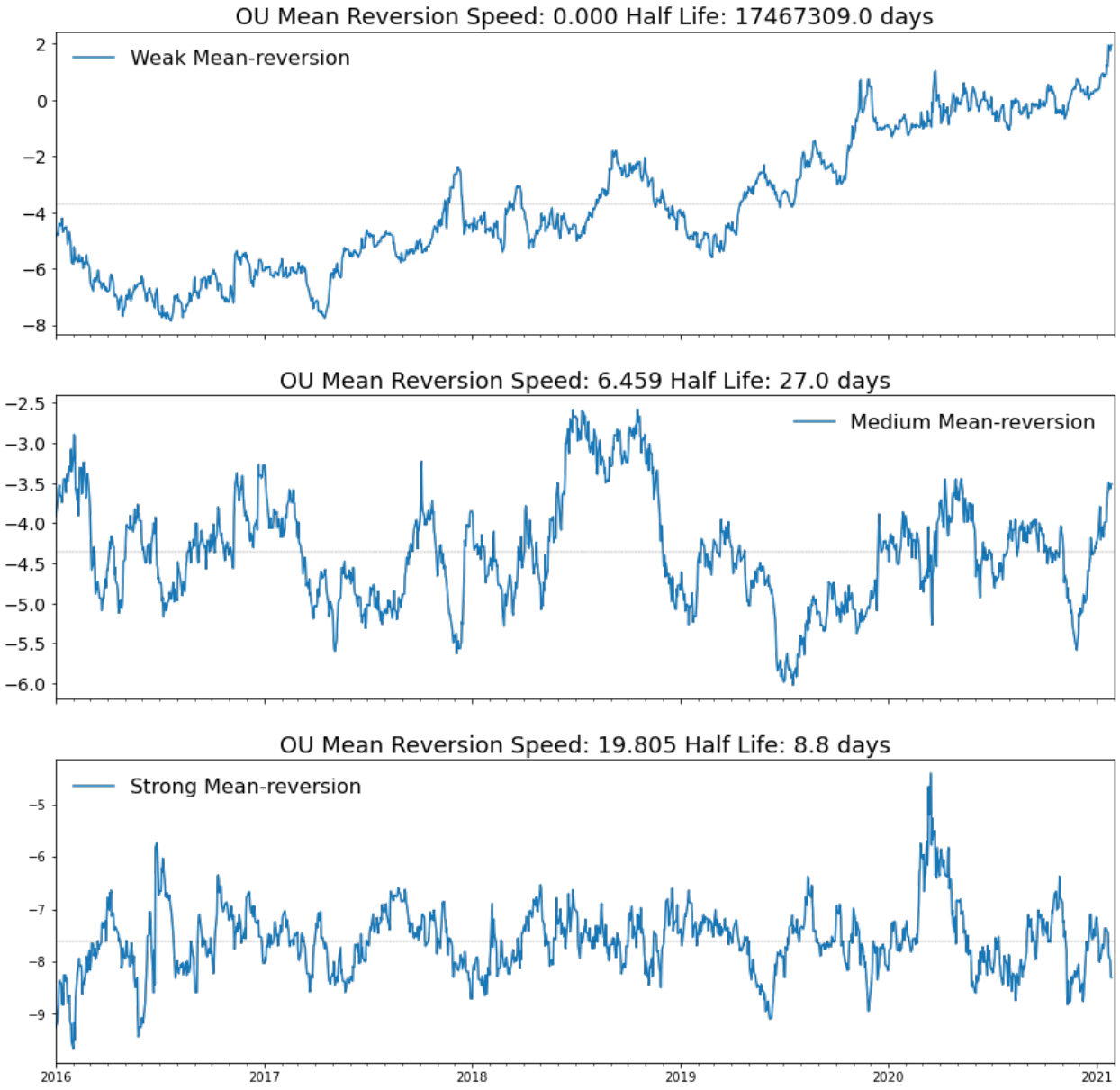


Figure 9. Box-Tiao canonical decomposition results.

The results have shown that the least mean-reverting portfolio have shown a persistent upward drift and thus might be suitable for a momentum strategy. This is further corroborated by the fact that the portfolio value process cannot be fit by an OU model according to its abnormally long mean-reversion half-life. The most mean-reverting portfolio, on the other hand, looks noisy; it would take approximately two weeks for the portfolio value to come back half way toward its long-term mean. Depending on the desired trading frequency, a statistical arbitrageur could either choose the most mean-reverting portfolio for more action, or he/she could use a portfolio with moderate predictability for a less noisy portfolio to avoid chopping.

Let's take a look at the weights for these three portfolios.

Asset	⠿	Weights	⠿
⠿ ⠿	Least mean-reverting	Moderate mean-reverting	Most mean-reverting
ECH	-0.300885	0.070991	0.104442
EWP	-0.263137	-0.441379	-0.152039
EWU	-0.006681	0.462643	-0.640490
EZA	-0.027801	-0.020616	-0.105541
THD	0.046753	-0.042529	-0.109448
GXG	-0.185164	0.016147	0.259847
EWS	0.128065	-0.617560	0.666222
EWO	0.886869	0.449702	0.145805

The weights of the least mean-reverting portfolio have shown that it is heavily overweight Austria equities (EWO). The behavior of the portfolio was dominated by this single asset and it is rather unsurprising that the portfolio value has shown non-stationarity, or momentum-like behavior. The most mean-reverting portfolio in this specific example happens to be able to get grouped into four pairs, i.e. Singapore (EWS) – UK (EWU), Austria (EWO) – Spain (EWP), Chile (ECH) – South Africa (EZA), Colombia (GXG) – Thailand (THD). The moderate mean-reverting portfolio, however, has four weights that are considerably smaller than the others; so it can be approximately regarded as a four-asset portfolio that focuses on European equities (EWP, EWU, EWO). The mean-reverting portfolios were constructed based only on price data, but the results made sense financially. For example, the equities from the same region were grouped together as in the moderate mean-reverting portfolio; or the equities from the commodity exporters were paired with each other (Chile is a

major copper miner and South Africa is a major platinum and palladium miner) as in the most mean-reverting portfolio. This suggests the sparse estimate of covariance matrix and VAR(1) coefficients can successfully isolate the conditional dependencies between the assets and make use of this information to form mean-reverting portfolios.

A few tips:

- Lasso methods are sensitive to the data scales. Therefore, the data have to be standardized before model fitting. This is achieved by setting the option use_standardized=True for the fitting functions.
- The sparsity of the covariance matrix is the determining factor of the size of the clusters. It is recommended to set a high regularization parameter for the graphical Lasso model to make sure the graph of the inverse covariance matrix Γ_0^{-1} is sufficiently disconnected.
- Box-Tiao method is not the only method to use after narrowing down the investing universe. You can still use the greedy algorithm and the convex relaxation approach on the smaller asset clusters.

Greedy Algorithm

The most straightforward approach needs minimal introduction. Let's directly jump into the codes.

```
# Calculate least-square VAR(1) estimate and sample covariance
full_var_est = etf_sparse_portf.least_square_VAR_fit(use_standardized=False)
full_cov_est = etf_sparse_portf.autocov(0, use_standardized=False)
# Use greedy algorithm to calculate portfolio weights
greedy_weight = etf_sparse_portf.greedy_search(8, full_var_est,
                                               full_cov_est, maximize=False)
```

Three lines of codes are all you need to get a greedy algorithm running using ArbitrageLab. To demonstrate the power of the greedy algorithm, I did not do clustering and took all 45 assets as the investing universe; the covariance estimate and the VAR(1) coefficient matrix estimate were also dense. To make sure we are not comparing apples to oranges, the target cardinality of the portfolio was set to 8.

Let's take a look at the portfolio formed by this approach.

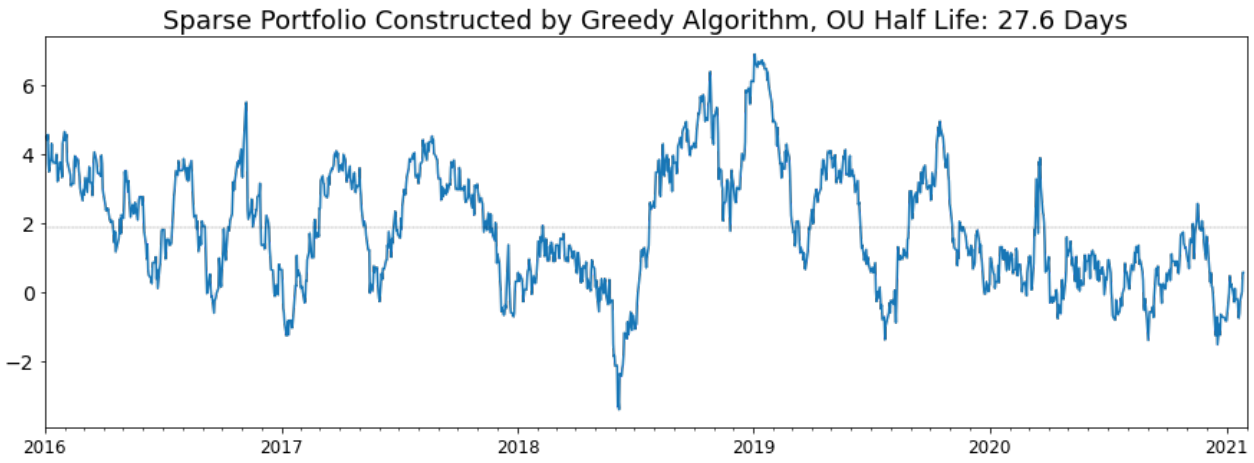


Figure 10. Sparse mean-reverting portfolio built by greedy algorithm.

Asset	Weights
EGPT	-0.035084
ARGT	-0.332868
EWZ	0.206248
EWW	0.504653
GXG	-0.680563
ECH	0.315425
EWY	-0.026486
TUR	-0.165515

The mean-reverting portfolio selected by greedy algorithm looks great for a strategy of lower frequency as the OU half-life of the portfolio is about a month. While the asset clustering method focused on the European region, the greedy algorithm preferred the Latin America region by selecting Argentina (ARGT), Brazil (EWZ),

Mexico (EWW), Colombia (GXG), and Chile (ECH) equities in the portfolio. The contribution from Egypt (EGPT), South Korea (EWY), and Turkey (TUR) equities were relatively small. Both the mean-reversion strength and the interpretability of the portfolio built by the greedy algorithm looks promising. Especially, the portfolio did not fall on its face during the market turbulence in early 2020.

Convex Relaxation

Although the convex relaxation looked complicated in theory, in practice it is rather straightforward with the help of ArbitrageLab. Let's see the code!

Similarly to the setup in the greedy algorithm demonstration, the SDP approach was directly applied to all 45 assets in the investing universe. To set a portfolio variance threshold, a fraction of the median variance of all the asset prices were used, which could be readily calculated with the diagonal of the covariance matrix. Again, the cardinality of the portfolio was set to 8 for a fair comparison.

```
# Retrieve the median variance of all the asset prices
cov = etf_sparse_portf.autocov(0, use_standardized=False)
var_median = np.median(np.diag(cov))  # Median variance is 16.1


# Use c30% of the median variance as the portfolio variance lower bound, which is 5
sdp_pred_vol_result = etf_sparse_portf.sdp_predictability_vol(rho=0.001, variance=5,
                                                              max_iter=5000,
                                                              use_standardized=False,
                                                              verbose=False)
# Deflate the optimization result into a weight vector
sdp_pred_vol_weights = etf_sparse_portf.sparse_eigen_deflate(sdp_pred_vol_result,
                                                             8, verbose=False)
```

What did the mean-reverting portfolio look like?

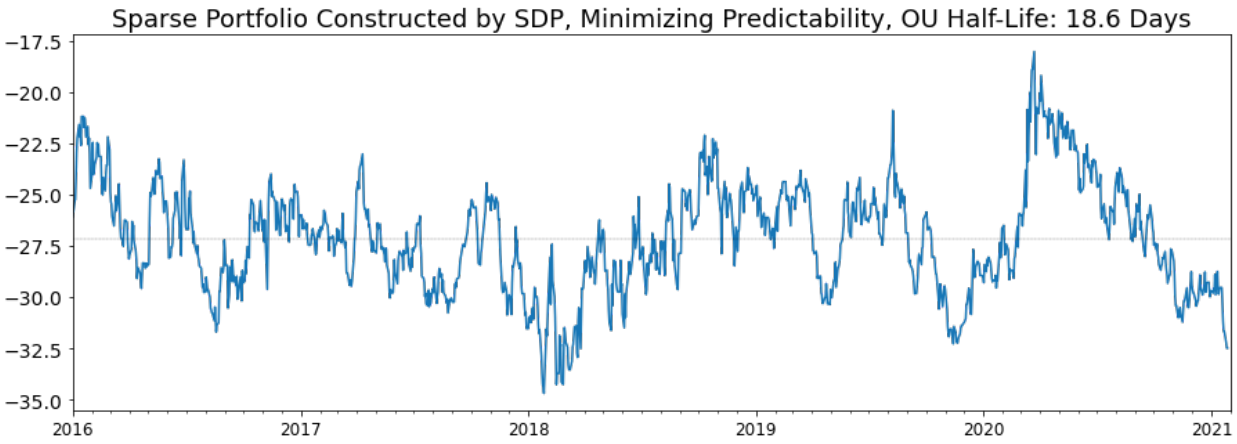


Figure 11. Sparse mean-reverting portfolio built by convex relaxation approach. Predictability is minimized to obtain this portfolio.

Asset	Weights
EGPT	0.114472
ENZL	-0.171927
EZA	-0.754660
GXC	-0.089197
EPU	0.112914
ARGT	0.503596
EIS	-0.081626
ECH	0.326876

Compared to the portfolio constructed by greedy algorithm, this one built by the convex relaxation approach has stronger mean reversion but is slightly noisier. The portfolio focuses on commodity exporters such as New Zealand (ENZL), South Africa (EZA), Peru (EPU), Argentina (ARGT), and Chile (ECH) equities, but its interpretability is worse than the greedy algorithm portfolio as China (GXC), Egypt (EGPT), and Israel equities (EIS) are also included in the portfolio with considerable weights. The volatility of the portfolio is also greater than the greedy algorithm one thanks to the volatility constraint.

Again, a few tips:

- Since the SDP result is almost never a rank-one matrix, it is necessary to run sparse_eigen_deflate function to retrieve the leading sparse eigenvector as the portfolio weight.

- How do we know if the regularization parameter ρ was set properly? Check the leading sparse eigenvector. Let's take the portfolio we just obtained as an example. The cardinality of the portfolio was set to 8. If a few weights are orders of magnitude smaller than the others, then a smaller ρ should be used.

- You may have noticed that in both greedy algorithm and convex relaxation approach, the data have only been centered to zero mean. Greedy algorithm is more robust and cares less about the scale of the data, but convex relaxation approach may sometimes run into numerical issues, especially for data that are significantly different in the scale. For example, if you would like to build a mean-r everting portfolio out of a group of emerging-market (EM) FX rate against USD, having USDIDR in your portfolio might cause problems (1 USD $\approx$ 14,000 IDR, but 1 USD $\approx$ 7 TRY, for example). In this case, it might be worthwhile to try standardized data by setting use_standardized=True and see if the resulting mean-reverting portfolio is tradable.

CONCLUSIONS

This article has covered the sparse mean-reverting portfolio selection problem in depth. Choosing the correct markets to trade are crucial for any trading strategies. Here the focus is to introduce tools for statistical arbitrageurs to build a multi-asset basket from a large number of candidates for mean-reversion trading strategies, but concepts like predictability can be helpful for selecting markets for momentum strategies as well.

Key Takeaways

- The mean-reverting portfolios should have as few assets as possible, or sparse in short. Sparse mean-reverting portfolios have better P&L interpretability, use less leverage, and incur fewer transaction costs.
- The mean-reversion speed of the Ornstein-Uhlenbeck model directly measures mean-reversion strength but is hard to optimize with respect to portfolio weights. Therefore, mean-reversion proxies like predictability, portmanteau statistic, and crossing statistic are used in the sparse mean-reverting portfolio selection problem.
- Stronger mean-reversion is equivalent to less predictability, smaller portmanteau statistic, and higher crossing statistic.
- Momentum strategies can be applied to portfolios with high predictability.
- Greedy algorithm is a robust method to build sparse mean-reverting portfolios.
- Convex relaxation approach offers more control over the sparse mean-reverting portfolio at the cost of model simplicity.
- Covariance selection and structured VAR(1) estimate with Lasso models can help find smaller asset clusters. This can be regarded as a preprocessing step, for Box-Tiao canonical decomposition, greedy algorithm, and convex relaxation approach can all be used on the these clusters to construct a sparse mean-reverting portfolio.

REFERENCES

1. Box, G.E. and Tiao, G.C., 1977. A canonical analysis of multiple time series. Biometrika, 64(2), pp.355-365.
2. d'Aspremont, A., 2011. Identifying small mean-reverting portfolios. Quantitative Finance, 11(3), pp.351-364.
3. Fogarasi, N. and Levendovszky, J., 2012. Improved parameter estimation and simple trading algorithm for sparse, mean-reverting portfolios. In Annales Univ. Sci. Budapest., Sect. Comp, 37, pp. 121-144.
4. Ghojogh, B., Karray, F. and Crowley, M., 2019. Eigenvalue and generalized eigenvalue problems: Tutorial. arXiv preprint arXiv:1903.11240.
5. Gilbert, J.R., 1994. Predicting structure in sparse matrix computations. SIAM Journal on Matrix Analysis and Applications, 15(1), pp.62-79.
6. Kedem, B. and Yakowitz, S., 1994. Time series analysis by higher order crossings (pp. 115-143). New York: IEEE press.
7. Ljung, G.M. and Box, G.E., 1978. On a measure of lack of fit in time series models. Biometrika, 65(2), pp.297-303.
8. Natarajan, B.K., 1995. Sparse approximate solutions to linear systems. SIAM journal on computing, 24(2), pp.227-234.
9. Yuan, X.T. and Zhang, T., 2013. Truncated Power Method for Sparse Eigenvalue Problems. Journal of Machine Learning Research, 14(4).

4.0

MINIMUM PROFIT OPTIMIZATION: OPTIMAL BOUNDARIES FOR MEAN-REVERSION TRADING

INTRODUCTION

In my previous articles, I introduced how to construct long-short asset pairs according to **the concept of cointegration** and how to build a **sparse mean-reverting multi-asset portfolio**. Now that we are able to answer the question "what to trade" with confidence, it is time to get down to the nitty-gritty of the implementation of a mean-reversion strategy.

The crux of implementing a mean-reversion trading strategy is to pinpoint the trade location. Apparently, we want to initiate a trade when the spread value has deviated considerably from its long-term mean. However, "a considerable deviation" is a rather vague description and needs to be quantified when it comes to trade execution. For the sake of convenience and clarity, I will use "boundary" to refer to the trade location and "spread" to both the spread of the long-short asset pairs and the value of the multi-asset portfolio in the remainder of this article.

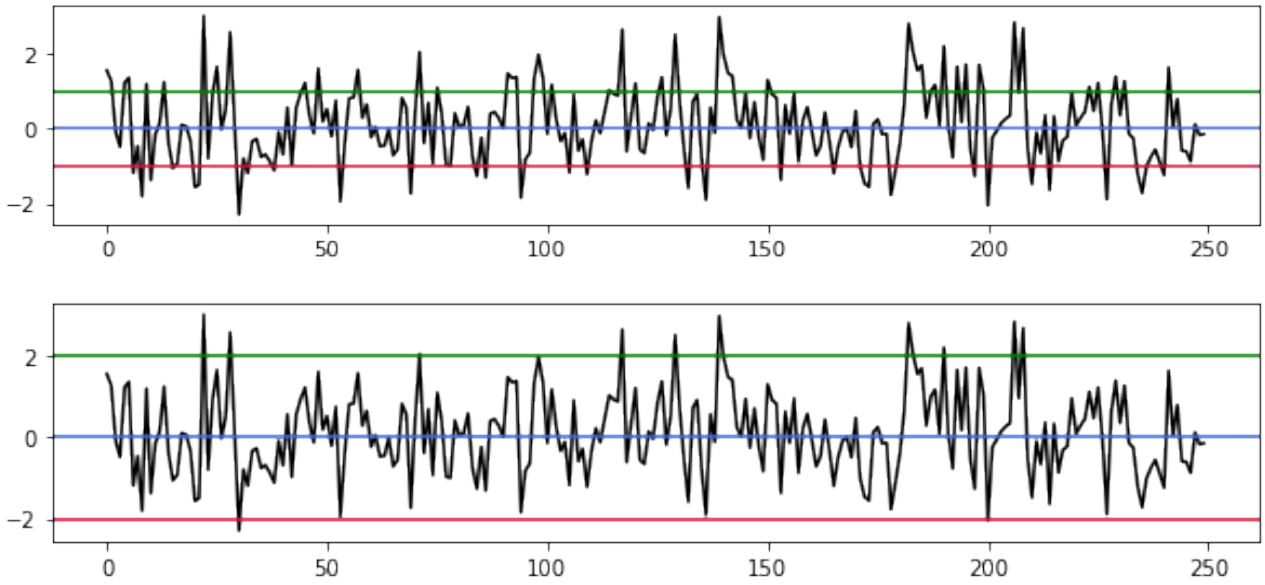


Figure 1: Two mean-reversion strategies with different trade locations on the same simulated stationary AR(1) series.

Let's use a toy example to illustrate the importance of the boundaries. The trading strategy is rather simple: we initiate a short position when the spread value is above the upper boundary (green line) and a long position when the spread value is below the lower boundary (red line). The exit condition is the same: when the spread value returns to the mean (blue line), the position is closed.

The only difference between the two strategies is the location of the boundaries. The first strategy set the boundaries at ± 1. The boundaries are tight, which make the strategy trade frequently, but each trade yields a smaller minimum profit. The second strategy set a wider boundaries at ± 2. Now the trades are far less frequent, but each trade yields a greater minimum profit. The boundaries effectively governs the tradeoff between trade frequency versus the minimum profit per trade.

A tradeoff implies that an optimization opportunity is just around the corner. "Sweet spots" for both boundaries should exist such that the minimum total profit over a defined time period, which is just the minimum profit per trade multiplied by the number of trades, is maximized. This article will demonstrate how to formulate and solve this optimization problem using the concept of mean first-passage time of a stationary AR(1) process based on the work of Puspaningrum, Lin, and Gulati (2009).

MEAN FIRST-PASSAGE TIME OF A STATIONARY AR(1) PROCESS

What exactly is the first-passage time of a process? Its rigorous mathematical definition looks rather daunting, but let's take a peek anyway. The first-passage time of a process Z_t passing through a lower boundary a is defined as follows:

$$\mathcal{T}_a(z_0) = \mathcal{T}_{a,\infty}(z_0) = \inf\{t : Z_t < a | Z_0 = z_0 \geq a\}$$

Similarly, the first-passage time of the same process passing through a upper boundary b is defined as follows:

$$\mathcal{T}_b(z_0) = \mathcal{T}_{-\infty,b}(z_0) = \inf\{t : Z_t > b | Z_0 = z_0 \leq b\}$$

While looking rather intricate, these equations can still be translated into plain English. Let's take the first equation as an example. The first-passage time of Z_t through the lower boundary a is only defined when Z_t has an initial value greater than a. When this condition is satisfied, we can simply take the earliest time when the value of Z_t goes below a. The first-passage time through the upper boundary b can be explained likewise.

Still confused? Don't worry, I have prepared a visualized explanation.

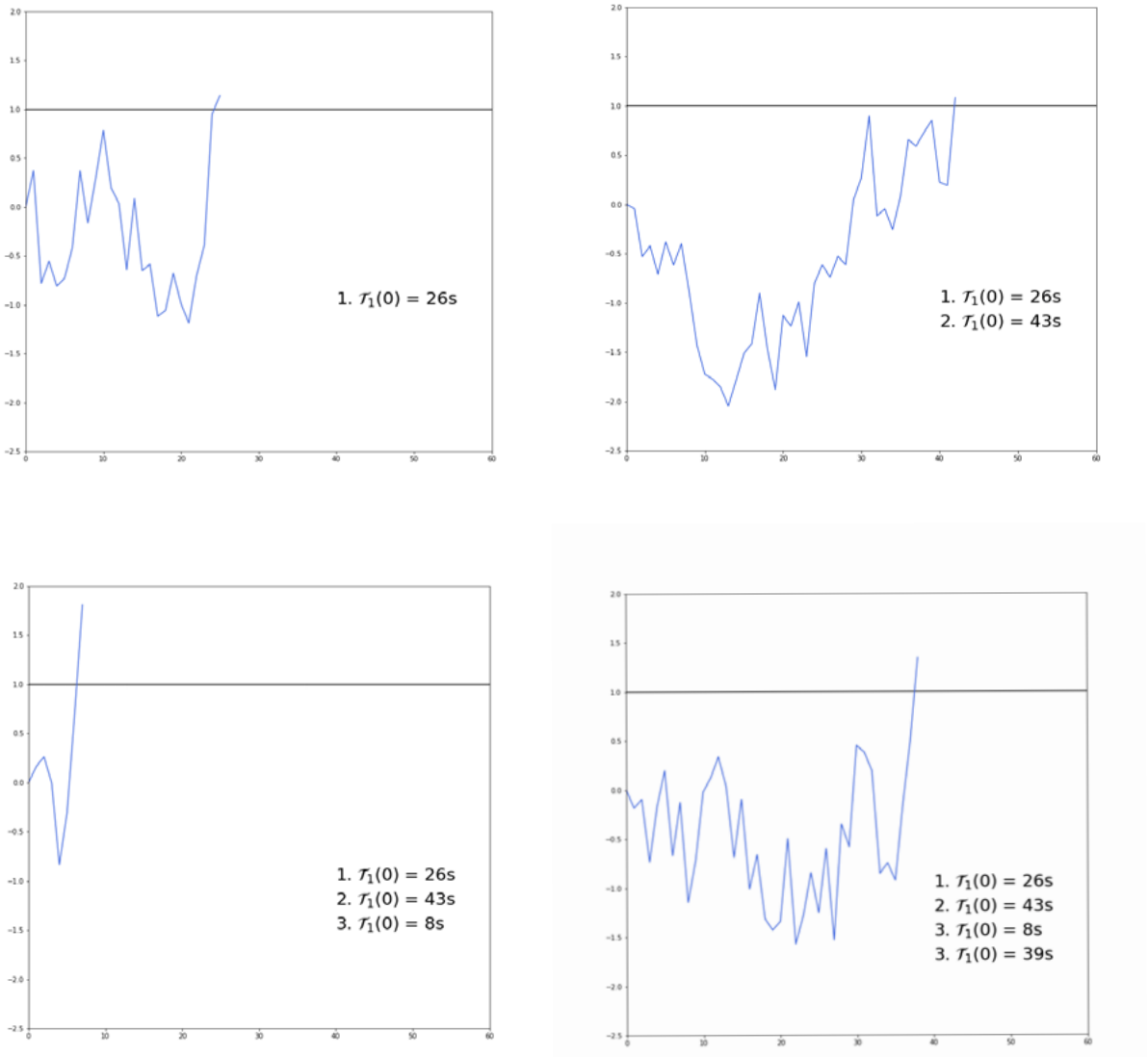


Figure 2: An animated demonstration of first-passage time. All four time series are simulated with a stationary AR(1) process (The AR(1) coefficient is 0.9) and starts at $z_0 = 0$. The time elapsed after they first pass through the upper boundary, which is $b = 1.0$ in this example, has been recorded. The corresponding notation is thus $\mathcal{T}_1(0)$.

Hope this animated example has helped you understand the concept of first-passage time. So why is first-passage time important?

The first-passage time can help us determine the trade frequency. Let's say we initiated a trade exactly at the upper boundary. As soon as the spread crosses over the mean for the first time, we will close the trade without hesitation. This matches the definition of first-passage time perfectly. If we can get a reliable estimate of how long this crossover would happen on average, we know the expected trade duration. Similarly, we can also estimate the expected duration of the spread that starts at the long-term mean crossing over the upper boundary. This corresponds to how long we need to wait to put on a position, i.e. the expected inter-trade interval. With both the expected trade duration and inter-trade interval known, we can easily calculate the expected number of trades within a certain time period.

We can see from Figure 2 that the first-passage time is highly variable. The shortest first-passage time was only 8 seconds, while the longest first-passage time was 43 seconds, a 5.4 times difference. When it comes to strategy execution, we need to obtain an estimate of the expected first-passage time on average as we cannot rely on previous observations of first-time passages. The first-passage time is influenced by a few factors:

1. The property of the time series itself, e.g. stationarity.

2. The starting location of the time series.

3. The location of the boundary.

The first factor is crucial. The first-passage time might be intractable if the time series is ill-behaved. Fortunately, since our focus is on mean-reversion trading strategies, it is thus safe to assume stationarity of the spread, which is guaranteed by construction via either cointegration tests or sparse mean-reverting portfolio selection methods. Furthermore, we assume the spread follows an AR(1) process as its mean first-passage time is tractable and can be numerically estimated.

I will directly give the formula of the mean first-passage time of a stationary AR(1) process.

$$E(\mathcal{T}_{a,b}(z_0)) = \frac{1}{\sqrt{2\pi}\sigma} \int_a^b E(\mathcal{T}_{a,b}(u)) \exp\left(-\frac{(u-\phi z_0)^2}{2\sigma^2} \right) du + 1$$

where ϕ and σ are defined by the stationary AR(1) process:

$$Z_t = \phi Z_{t-1} + \xi_t$$

$$|\phi| < 1$$

$$\xi_t \sim N(0, \sigma^2) \text{ i.i.d}$$

Curious readers can find the derivation in Basak and Ho (2004).

CALCULATING MEAN FIRST-PASSAGE TIME OF A STATIONARY AR(1)

I will briefly go over the numerical methods to obtain a reasonable estimate for $E(\mathcal{T}_{a,b}(z_0))$ for completeness. If you are more interested in the application of this method rather than the technical details, you can skip this part.

Notice that there is a similar term $E(\mathcal{T}_{a,b}(u))$ in the integral on the other side of the formula. Therefore, the first step is to use trapezoidal rule to convert the integral into a summation.

$$\begin{aligned} E(\mathcal{T}_{a,b}(z_0)) &= \frac{1}{\sqrt{2\pi}\sigma} \int_a^b E(\mathcal{T}_{a,b}(u)) \exp\Big(-\frac{(u-\phi z_0)^2}{2\sigma^2} \Big) du + 1 \\ &= \frac{h}{2\sqrt{2\pi}\sigma} \sum_{j=0}^{n} w_j E(\mathcal{T}_{a,b}(u_j)) \exp\Big(-\frac{(u_j-\phi z_0)^2}{2\sigma^2} \Big) + 1 \end{aligned}$$

where $h = \frac{b-a}{n}$, n is the number of partitions in $[a, b]$ and h is the length of each partition. The higher the n, the finer each slice of the partition $[a, b]$ is, and the more exact the final estimation will be. Moreover, according to trapezoidal rule, the values of w_j are as follows:

$$w_j = \begin{cases} 1 & \text{for } j = 0 \text{ or } j = n \\ 2 & \text{otherwise} \end{cases}$$

There are $n+1$ grid points u_j for n intervals, the idea here is thus to evaluate the summation at each grid point to get a linear system of $n+1$ equations with respect to $n+1$ variables, which

Denote

$$K(u_i, u_j) = \frac{h}{2\sqrt{2\pi}\sigma} w_j \exp\Big(-\frac{(u_j-\phi u_i)^2}{2\sigma^2} \Big)$$

Then the linear system can be written as

$$\begin{pmatrix} 1-K(u_0,u_0) & -K(u_0,u_1) & \dots & -K(u_0,u_n) \\ -K(u_1,u_0) & 1-K(u_1,u_1) & \dots & -K(u_1,u_n) \\ \vdots & \vdots & \vdots & \vdots \\ -K(u_n,u_0) & -K(u_n,u_1) & \dots & 1-K(u_n,u_n) \end{pmatrix} \begin{pmatrix} E_n(\mathcal{T}_{a,b}(u_0)) \\ E_n(\mathcal{T}_{a,b}(u_1)) \\ \vdots \\ E_n(\mathcal{T}_{a,b}(u_n)) \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}$$

This can be readily solved in $O(n^3)$ time.

GETTING THE ESTIMATES OF TRADE DURATION AND INTER-TRADE

Now that we have a working numerical algorithm to calculate mean first-passage time of a stationary AR(1) series, we can obtain estimates of trade duration and inter-trade interval. Recall the definition of trade duration and inter-trade interval based on mean first-passage time:

- Trade duration ($TD(U)$): Average time elapsed between the time series starting at the upper boundary U and crossing over its mean.
- Inter-trade interval ($I(U)$): Average time elapsed between the time series starting at its mean and crossing over the upper boundary U.

The case for lower boundary is the same due to symmetry, for the mean of the spread will be always centered to zero before any calculation.

Substituting the upper boundary U and the mean (which is zero) into the mean first-passage time equation and we can obtain:

$$TD(U) = E(\mathcal{T}_{0,\infty}(U)) = \lim_{b\to\infty} \frac{1}{\sqrt{2\pi\sigma}} \int_0^b E(\mathcal{T}_{0,b}(s)) \exp\Big(-\frac{(s-\phi U)^2}{2\sigma^2} \Big) ds + 1$$

$$I(U) = E(\mathcal{T}_{-\infty,U}(0)) = \lim_{-b\to-\infty} \frac{1}{\sqrt{2\pi\sigma}} \int_{-b}^U E(\mathcal{T}_{-b,U}(s)) \exp\Big(-\frac{s^2}{2\sigma^2} \Big) ds + 1$$

Over a certain time period T, we can expect the number of trades for a pre-set boundary U to be at least:

$$N(U) = \frac{T}{TD(U) + I(U)} - 1$$

I will try to explain this result in an intuitive way instead of resorting to mathematical derivations. Let's say we have programmed a trading bot that executes this strategy accurately. The bot should work in cycles, and each cycle contains two phases: it is either in a trade, which on average takes $TD(U)$ time; or it is waiting to initiate the next trade, which on average takes $I(U)$ time.

The bot would do exactly one trade in each cycle, and thus the number of cycles is equal to the number of trades. What is the average length of each cycle? It's $TD(U) + I(U)$. So over a time period T, there are $\frac{T}{TD(U)+I(U)}$ cycles, which is also the number of trades. Note that T is not guaranteed to be a multiple of $TD(U) + I(U)$, so we need to a minor adjustment to not overestimate the number of trades.

The minimum profit per trade is just U. Since a trade would only be initiated when the spread rises over U, so in the worst-case scenario, the (short) entry price is at least U and the exit price is zero, which is the mean of the spread. The minimum total profit over a time period T is thus:

$$MTP(U) = U \cdot N(U) = \left(\frac{T}{TD(U)+I(U)} - 1\right)U$$

This value will be maximized by a grid search over a set of pre-defined values.

You might have noticed that in both the trade duration and the inter-trade interval formulae one of the integration limits is infinity. A numerical algorithm cannot accept infinity as an input and thus an approximation of infinity is required.

Puspaningrum, Lin, and Gulati (2009) proposed to use five standard deviation of the spread as an approximation. Thanks to the stationarity of the spread, the probability of the absolute value of the spread exceeding this threshold is close to zero. A larger threshold could certainly be used to get a more accurate estimate, but the

APPLYING THE MINIMUM PROFIT OPTIMIZATION ALGORITHM WITH

We can now summarize the minimum profit optimization algorithm. The input is the time series of the spread \epsilon_t, which is your favorite cointegrated spread or mean-reverting multi-asset portfolio.

1. Build the grid of U. Set the leftmost grid point to 0 and the rightmost grid point to $|5\sigma_{\epsilon_t}|$. The granularity of the grid is empirically determined and set to 0.01. The grid of U is thus $\{0, 0.01, 0.02, \ldots, |5\sigma_{\epsilon_t}|\}$.

2. Fit the spread ϵ_t to an AR(1) model, retrieve the AR(1) coefficient ϕ, and calculate the standard deviation of the fitted residual σ, which corresponds to the ϕ and σ in the mean first-passage time equations, respectively.

3. For each grid point U_i,

    1. Calculate the trade duration $TD(U_i)$.

    2. Calculate the inter-trade interval $I(U_i)$.

    3. Calculate the minimum total profit over the backtest period T, $MTP(U_i)$.

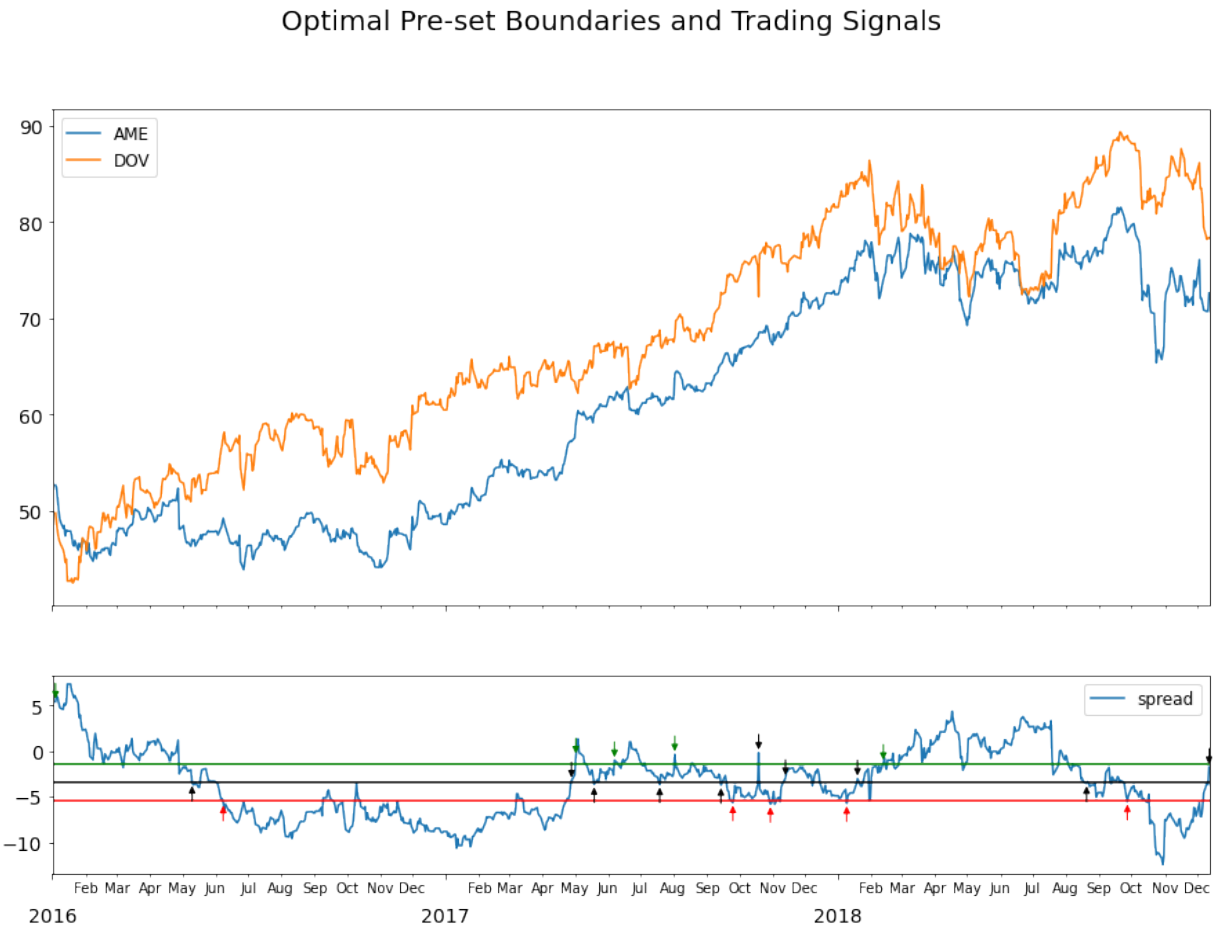
4. Return the optimal upper boundary U^* such that $MTP(U^*)$ is maximized.

In the remainder of this section, I will demonstrate how to quickly set up a mean-reversion trading strategy based on minimum profit optimization with ArbitrageLab. The assets involved are two S&P 500 stocks, Ametek Inc. (AME), and Dover Corp. (DOV). The price data range from Jan 4th, 2016 to Nov 23th, 2020. The first three years of data were used for boundary optimization and the rest of the data were used for out-of-sample testing.

```
from arbitragelab.cointegration_approach.minimum_profit import MinimumProfit
# Assume you already have the price of the two stocks stored in a Pandas dataframe "data".
# Initialize the minimum profit optimizer.
optimizer = MinimumProfit(data)

# Split the entire price history into training and trading period.
train_df, trade_df = optimizer.train_test_split(date_cutoff=pd.Timestamp(2019, 1, 1))

# Determine the cointegration coefficient and fit the cointegrated spread with AR(1)
beta_eg, epsilon_t_eg, ar_coeff_eg, ar_resid_eg = optimizer.fit(use_johansen=False,
                                                                sig_level="95%")

# Optimize the trade boundaries.
optimal_ub, _, _,
optimal_mtp, optimal_num_of_trades = optimizer.optimize(ar_coeff_eg, epsilon_t_eg,
                                                        ar_resid_eg, len(train_df))

# Generate trade signals for both in-sample data and out-of-sample data
trade_signal_is, num_of_shares_is,
cond_value_is = optimizer.trade_signal(optimal_ub, optimal_ub,
                                       beta_eg, epsilon_t_eg, insample=True)
trade_signal_oos, num_of_shares_oos,
cond_value_oos = optimizer.trade_signal(optimal_ub, optimal_ub,
                                        beta_eg, epsilon_t_eg, insample=False)
```

With seven lines of code, a mean-reversion trading strategy has been set up with optimal boundaries. The trade signal generated by this algorithm can be then used for backtesting.

How did the strategy perform? Let's take a look at in-sample result first.



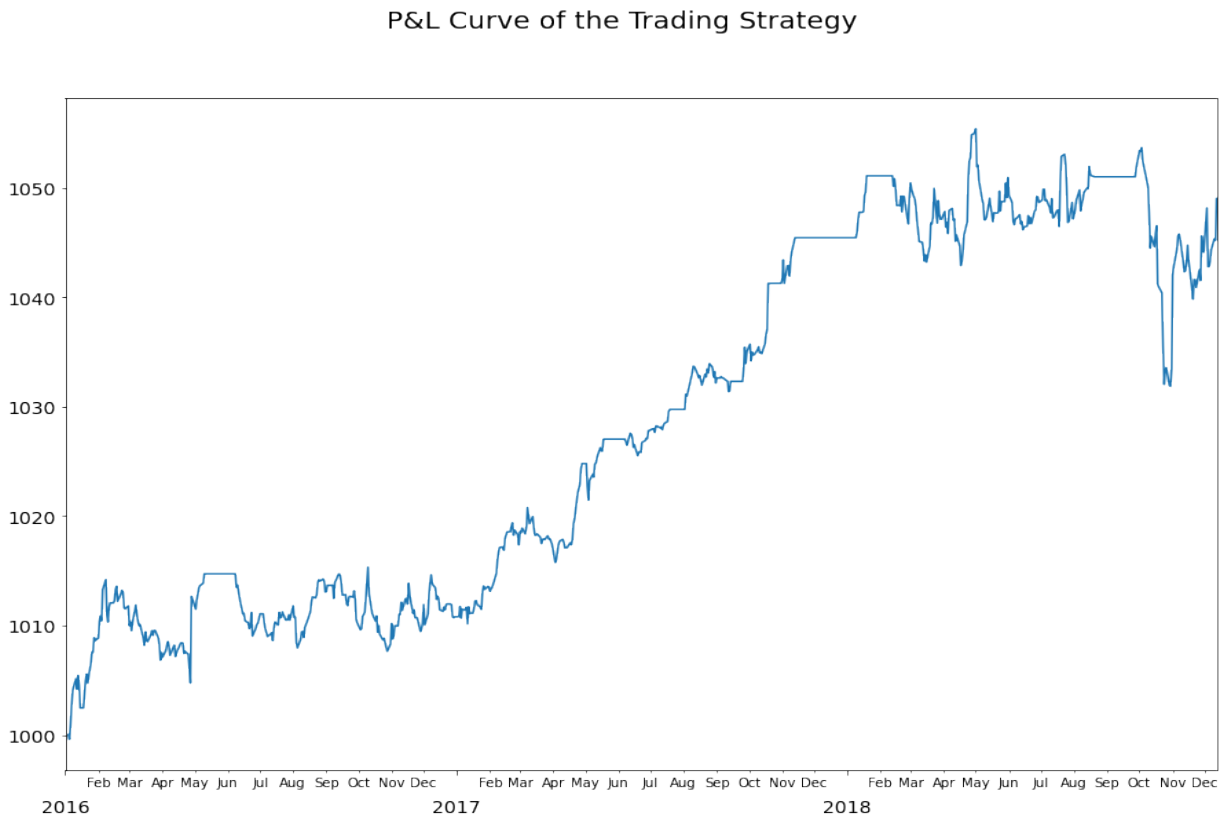


Figure 3: The in-sample performance of the trading strategy based on minimum profit optimization methods. The optimal upper boundary corresponds to the green line in the trading signal chart. The symmetric lower boundary corresponds to the red line in the trading signal chart. The strategy was able to generate a profit in the end of the trading period.

What about out-of-sample performance?

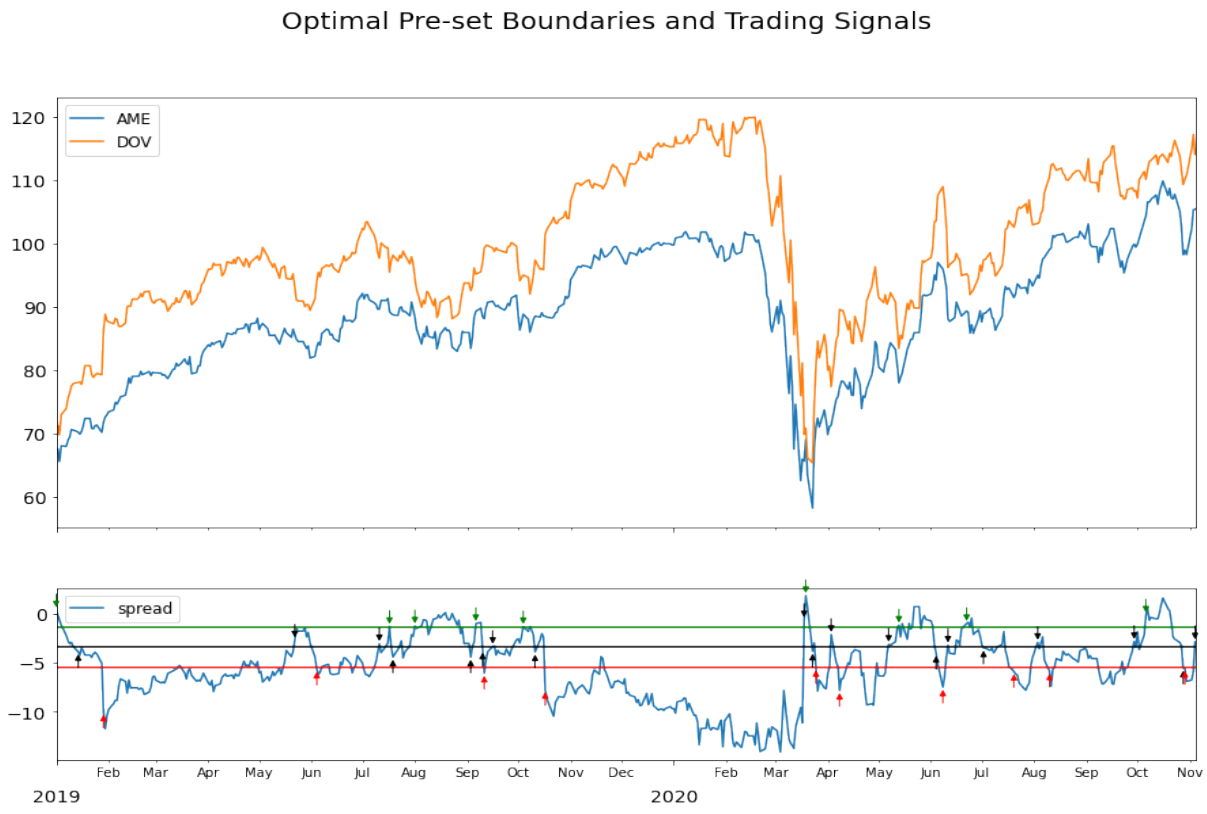


Figure 4: The out-of-sample performance of the trading strategy based on minimum profit optimization methods. The optimal upper boundary corresponds to the green line in the trading signal chart. The symmetric lower boundary corresponds to the red line in the trading signal chart. The strategy was able to generate a profit in the end of the trading

The optimal boundary was 2.05 away from the mean for this cointegrated stock pair, meaning that the minimum profit optimization strategy is able to generate a $2.05 profit trading one unit of the pair. To increase the trade frequency, a symmetric lower boundary at -2.05 away from the mean was used for trade initiation as well so that the strategy can both "buy low" and "sell high". The in-sample results have shown that the average profit per trade is $4.90, which is higher than the minimum profit per trade $2.05. The out-of-sample results did not decay significantly, which generated $4.74 per trade on average.

This result was promising as the out-of-sample testing period included the coronavirus market crash, yet the performance was not severely affected. But the minimum profit optimization method is certainly not the holy grail and has its limitations.

We have assumed that the spread follows a stationary AR(1) process. However, the real underlying process of the spread may not be AR(1). As we can see from the trading signal charts, the spread spent a considerable time outside the boundaries. This suggests that the mean first-passage time of an AR(1) process might not yield the best estimates for trade duration and inter-trade intervals. However, a numerical estimation scheme may become unavailable as a result of using a more complicated time series model.

A possible fix to this issue could be try using sliding window instead of cumulative window to optimize the boundary. For example, the minimum profit optimization should be carried out on 6-month rolling price data so that the optimal boundary will not stay fixed throughout the trading period. This may increase the ability of the strategy to adapt to different market regimes.

CONCLUSION

This article has introduced the minimum profit optimization method on mean-reverting asset pairs and multi-asset portfolios and how it can be used to set up a mean-reversion trading strategy.

Key Takeaways

- The optimal boundary can be determined by optimizing trade duration and inter-trade interval.
- Trade duration and inter-trade interval can be estimated with mean first-passage time.
- If the spread follows a stationary AR(1) process, a numerical algorithm can be applied to estimate trade duration and inter-trade interval in $O(n^3)$ time.
- The optimal boundary maximizes the total minimum profit over a trading period.

REFERENCES

1. Basak, G.K. and Ho, K.−W.R. (2004). Level-crossing Probabilities and First-passage Times for Linear Processes. Advance Applied Probability, 36, 643-666.

2. Puspaningrum, H., Lin, Y.-X., and Gulati, C. M. (2010). Finding the optimal pre-set boundaries for pairs trading strategy based on cointegration technique. Journal of Statistical Theory and Practice, 4(3):391–419.

5.0

OPTIMAL STOPPING IN PAIRS TRADING: ORNSTEIN-UHLENBECK MODEL

INTRODUCTION

Nothing makes a situation better like good timing. Whether it's getting a promotion, catching the last train after a night out, meeting the love of your life, or joining a quant community – it is many of small consequential gambles of stopping decisions that get us to that triumphant "Yes!" moment. When outputs may vary, the basic setup of all these problems is the same: we observe some process that involves randomness in it evolving in time. Based only on the knowledge available the person has to decide how to maximize reward or minimize the cost. There is no requirement on how much information has to be available, whether it is scarce or abundant the decision-maker has to make the best choice. Luckily for us, knowing the powers of probability allows us to tilt odds in our favor.

Let's look at some more mundane problems that can be solved with the little help of optimal-stopping theory.

The first example is the problem of finding a suitable partner, also known as the secretary problem, dowry, or best-choice problem. Imagine, you decide to marry, and to find your perfect other half you conduct an interview with 100 applicants. The decision about an applicant is made immediately. You can not marry candidates who you already rejected, and if you are not married after interviewing candidate 99 – you have to marry candidate 100. Of course, you can choose the first applicant and get a 1% chance of getting the best spouse. But there is a way to marry the absolute best candidate more than one-third of the time.

The rule is to reject $n/e = 100/2.718 = 37$ potential spouses, and then choose the first applicant who is better than every applicant interviewed so far.

The various problems similar to the 'secretary problem' create so-called 'search theory', that has especially

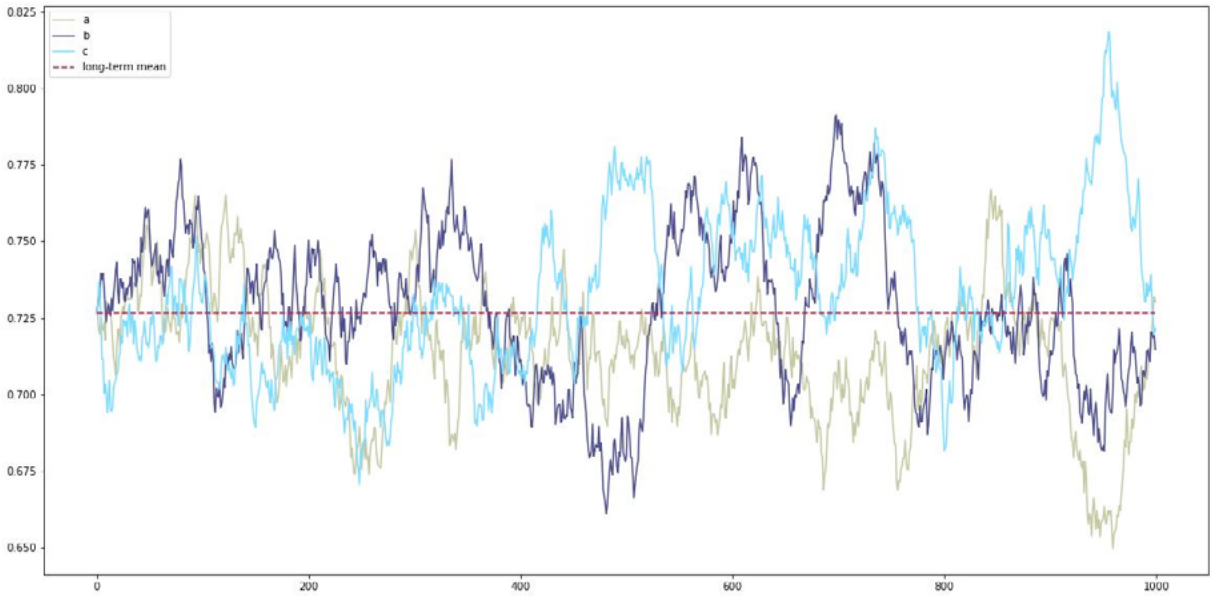
focused on a worker's search for the highest-paying job or a consumer's search for the best deal on wanted goods.

Optimal stopping is also encountered in house selling. For example, if you wish to sell a house. Each day you are offered X_n for your house, and pay k to continue advertising it. If you sell your house on day n, you will earn y_n, where $y_n = (X_n - nk)$. You maximize the amount you earn by choosing the best stopping rule.

The field of optimal stopping problems is broad and vast, it ranges from option pricing with the Black-Scholes model to choosing the best parking spot when going to buy groceries. But what if I tell you that there is a simple and elegant way to use it for the statistical arbitrage?

Welcome to Part 1 of the series of blog posts on optimal stopping problems for statistical arbitrage. And today we are going to talk about the Ornstein-Uhlenbeck model application to optimal stopping problems in pairs trading.

MEAN-REVERSION



But before that, we need to start with something that will allow the statistical arbitrage in the first place – mean-reversion.

Mean-reversion is a financial term for the assumption that a stock's price will tend to return to the average price over time. Some asset prices are naturally mean-reverted: commodities, foreign exchange rates, volatility indices, equities – all of those can be modeled with mean-reverting processes, alongside with interest rate and default risk.

However, there is a much more lucrative approach that also allows for greater freedom of choice for the investor – pairs-trading. Facilitated by many hedge-fund managers, creation of the mean-reverting portfolios gave the ability to both pick almost any asset you'd want to trade and simultaneously rely on rigid mathematical concepts in the decision-making process. The only requirement to be is that the pair chosen has to be correlated or co-moving. Technically speaking, a portfolio is created by longing an $\alpha = \frac{A}{S_0^{(1)}}$ amount of one asset and shorting $\beta = \frac{B}{S_0^{(2)}}$ amount of a second asset.

$$X_t^{\alpha,\beta} = \alpha S^{(1)} - \beta S^{(2)}, t \geq 0$$

It is no surprise that such spreads are widely used for statistical arbitrage.

Looking deeper, one very important commonly faced problem still stands – how to determine when to open and close the position. Should an investor wait or close the position immediately? When is it the optimal time to enter the market? All these questions bring us to a brilliant idea: what if we could look at it as an optimal stopping problem? Can we create a procedure that will give us the means to predict the best time to enter or liquidate the position?

For an answer to this question, we turn to a book by Professor Tim Leung and Xin Li: "**Optimal Mean reversion Trading: Mathematical Analysis and Practical Applications**". We will showcase how the widely known Ornstein-Uhlenbeck process can be used to create your optimal mean-reverted portfolio and to also find the solution for the optimal timing of trades problem.

THE WORK OF PROFESSOR TIM LEUNG

Ornstein-Uhlenbeck model is established by the following SDE:

$$X_t^{\alpha,\beta} = \alpha S^{(1)} - \beta S^{(2)}, t > 0$$
$$\mu, \sigma > 0,$$
$$\theta \in \mathbb{R},$$
$$B - \text{a standard Brownian motion}$$

Where:

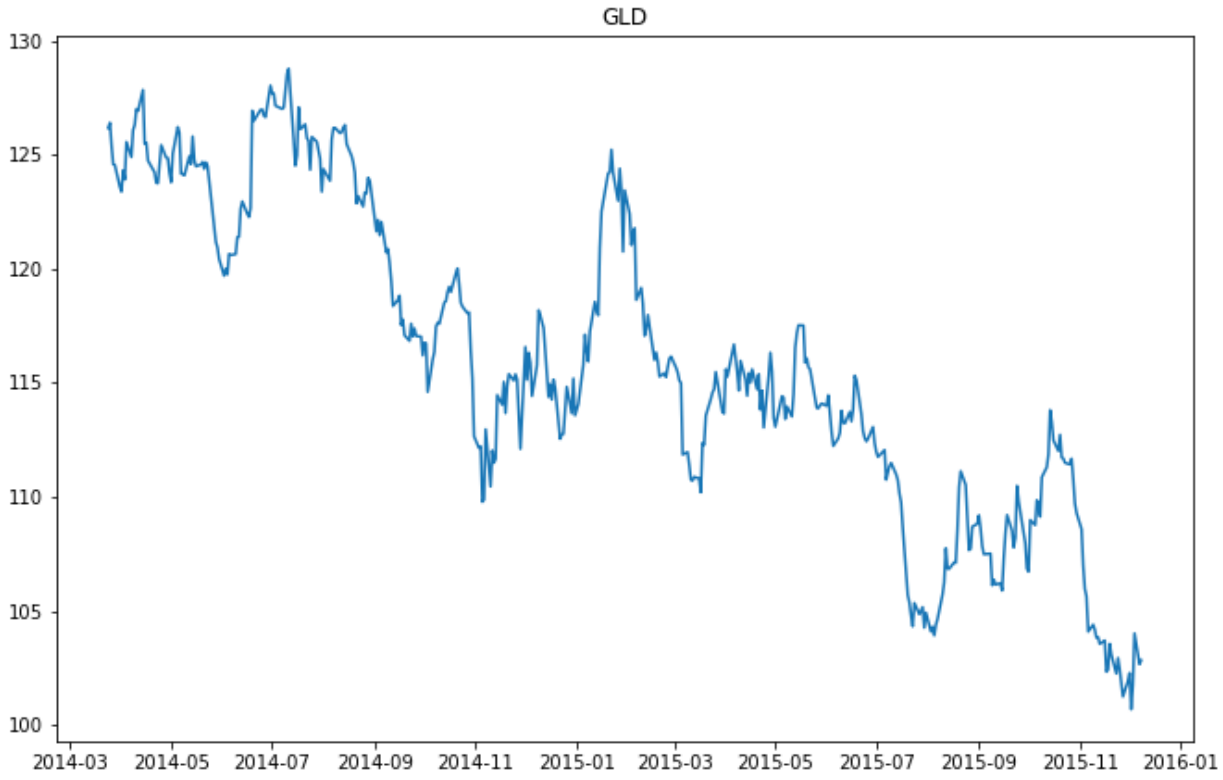
θ – long term mean level, all future trajectories of X will evolve around a mean level θ in the long run.

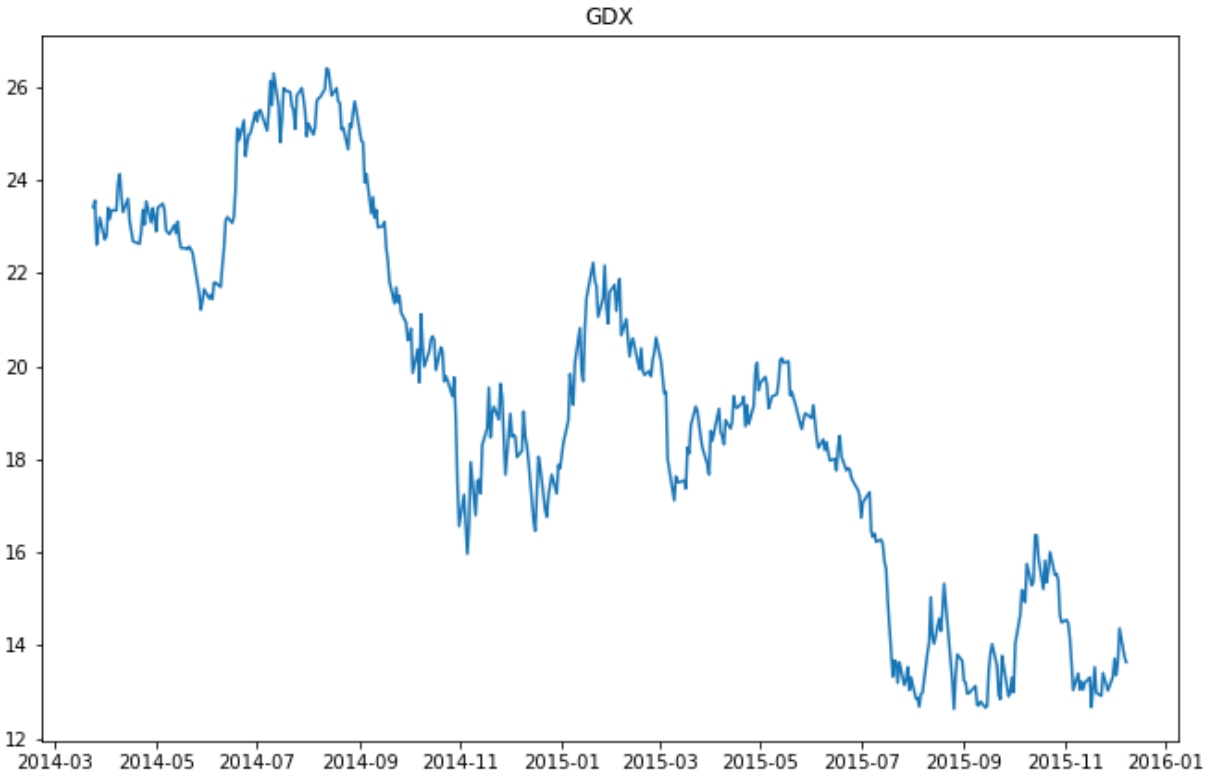
μ – the speed of reversion, characterizes the velocity at which such trajectories will regroup around θ in time.

σ – instantaneous volatility, measures instant by instant the amplitude of randomness entering the system. Higher values imply more randomness.

Note that we are using the same notation as used in the book, so the meaning behind theta and mu parameter may differ from other sources (Wikipedia, etc.).

To create an optimal portfolio using this model we can without the loss of generality presume that $\alpha = const$ while varying β, since our main goal during this step can be simplified to finding an optimal ratio between the assets.





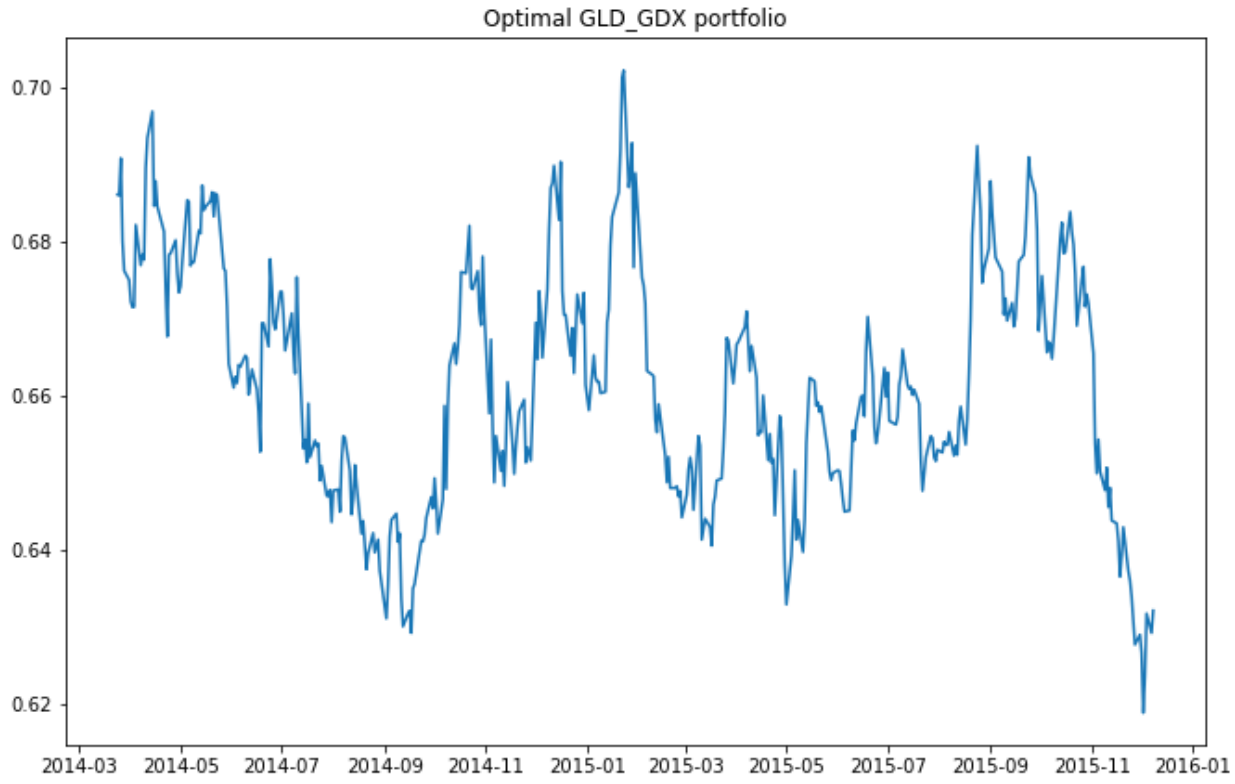
Exemplary co-moving assets.

We observe the resulting portfolio values for every strategy β realized over an n-day period. To fit the model to our data and find optimal parameters we define the average log-likelihood function. Our log-likelihood function is dependent on our β coefficient and the OU model's parameters theta, mu, and sigma.

Maximizing the log-likelihood function by applying maximum likelihood estimation(MLE) we can determine the parameters of the model and fit the observed portfolio prices to an OU process. Let's denote the maximized average log-likelihood by $\hat{\ell}(\theta^*, \mu^*, \sigma^*)$. Then for every α, we choose β^*, where:

$$\beta* = \underset{\beta}{\operatorname{argmax}}\, \ell(\theta*, \mu*, \sigma* | x_0^{\alpha\beta}, x_1^{\alpha\beta}, \cdots, x_n^{\alpha\beta})$$

To create the most fitting to an OU-model portfolio, hence optimal and more mean-reverted one.



Optimal mean-reverting portfolio constructed by maximizing the log-likelihood function.

The next step is to establish our optimal stopping problem: suppose the investor already has a position with a value process $(X_t), t > 0$ that follows the OU process. When the investor closes his position at the time τ he receives the value (X_τ) and pays a constant transaction cost $c_s \in R$. To maximize the expected discounted value we need to solve the optimal stopping problem:

$$V(x) = \sup_{\tau \in T} \mathbb{E}_x e^{-r\tau}((X_\tau - c_s)|X_0 = x)$$

where T denotes the set of all possible stopping times and $r > 0$ is our subjective constant discount rate and V(x) represents the expected liquidation value accounted with X.

The next logical step is to formalize the optimal entry problem since the future optimal value of liquidation minus current price and transaction cost constitute the cost of entering the trade:

$$J(x) = \sup_{\nu \in T} \mathbb{E}_x e^{-\hat{r}\tau}((V(X_\nu) - X_\nu - c_b)|X_0 = x)$$

Sometimes an investor would like to include a stop-loss level. If the price of a portfolio will ever reach this level, then the position will be closed immediately. So the problem is reformulated in the following way:

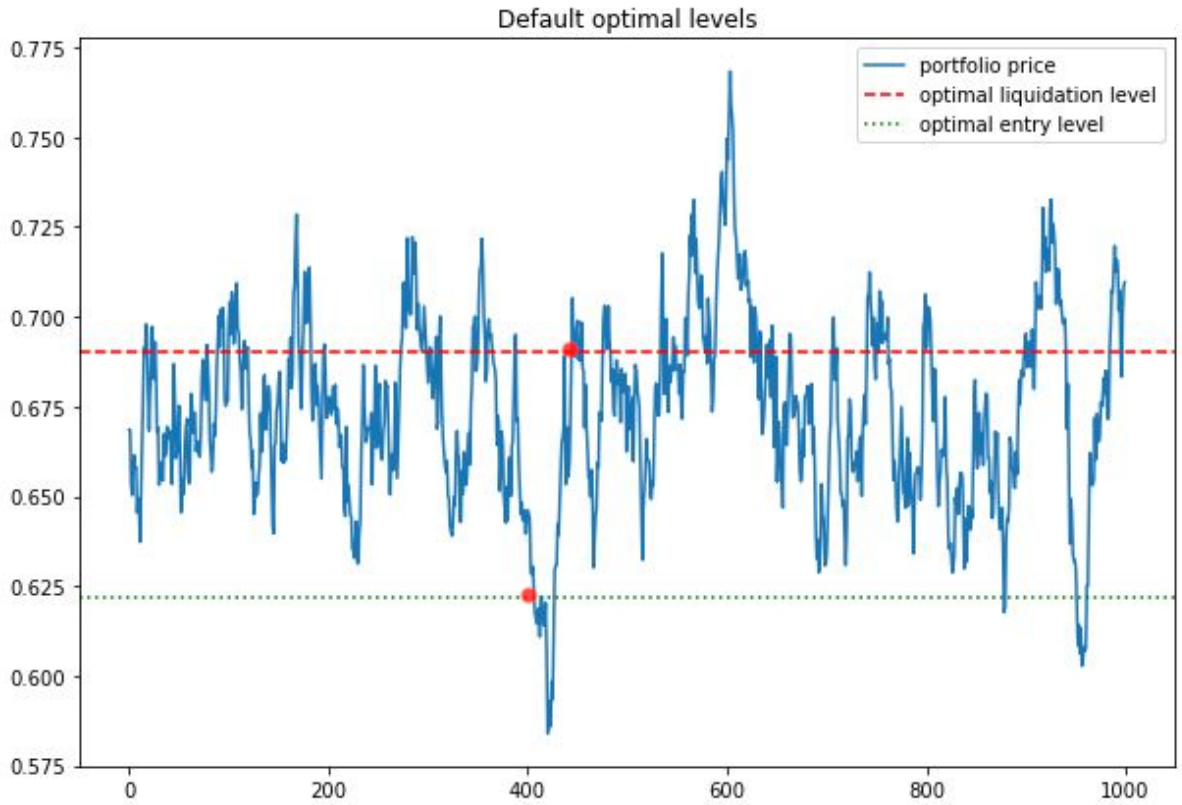
$$V(x)_L = \underset{\tau \in T}{sup} \mathbb{E}_x e^{-r\tau \wedge \tau L}((X_{\tau \wedge \tau L} - c_s)|X_0 = x)$$

$$J(x)_L = \underset{\nu \in T}{sup} \mathbb{E}_x e^{-\hat{r}\tau}((V_L(X_\nu) - X_\nu - c_b)|X_0 = x)$$

The analytical solutions for the optimal stopping problems are presented and proven in **Optimal Mean reversion Trading: Mathematical Analysis and Practical Applications by Professor Tim Leung and Xin Li**.

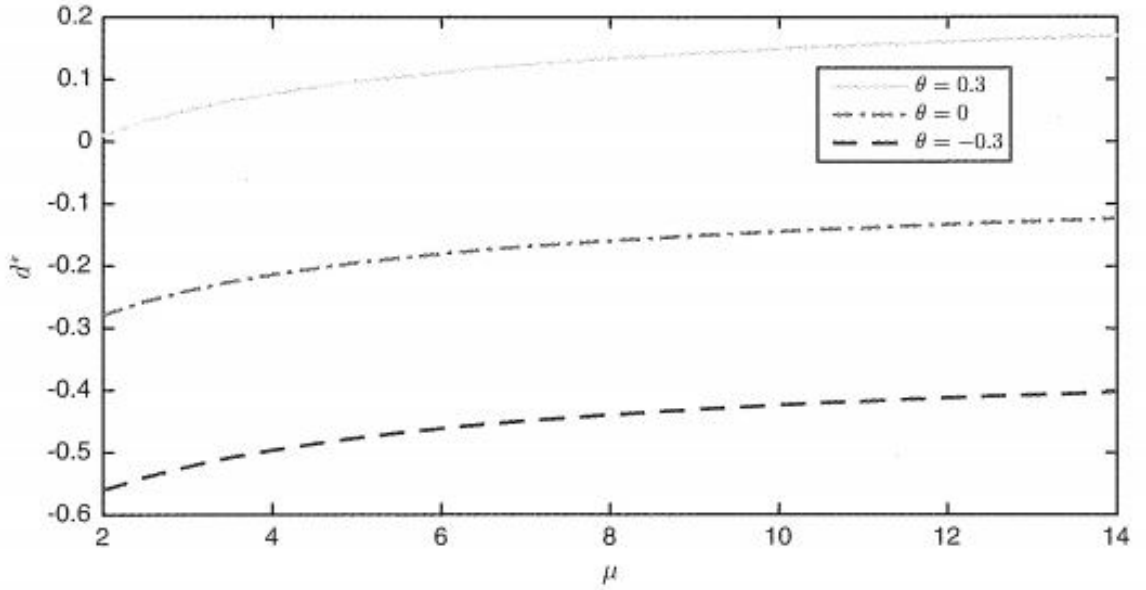
Our goal is to find the single best entry/liquidation value pair to get the biggest amount of profit from one trade.

However similar both formulations may look, the results we get are slightly different.

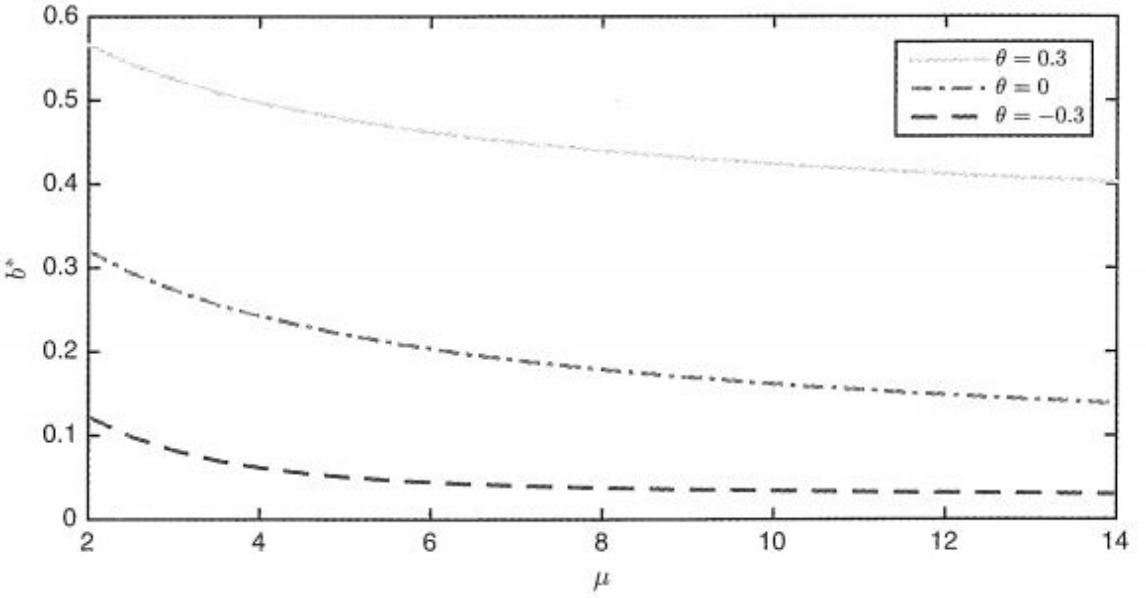


Example of Optimal Entry and Exit for a Mean-Reverting Processes

In the default formulation as a result we are getting two values – optimal entry level d^* and optimal liquidation level b^*. Since the obtained values are dependent on the OU model parameters we can observe further correlations:

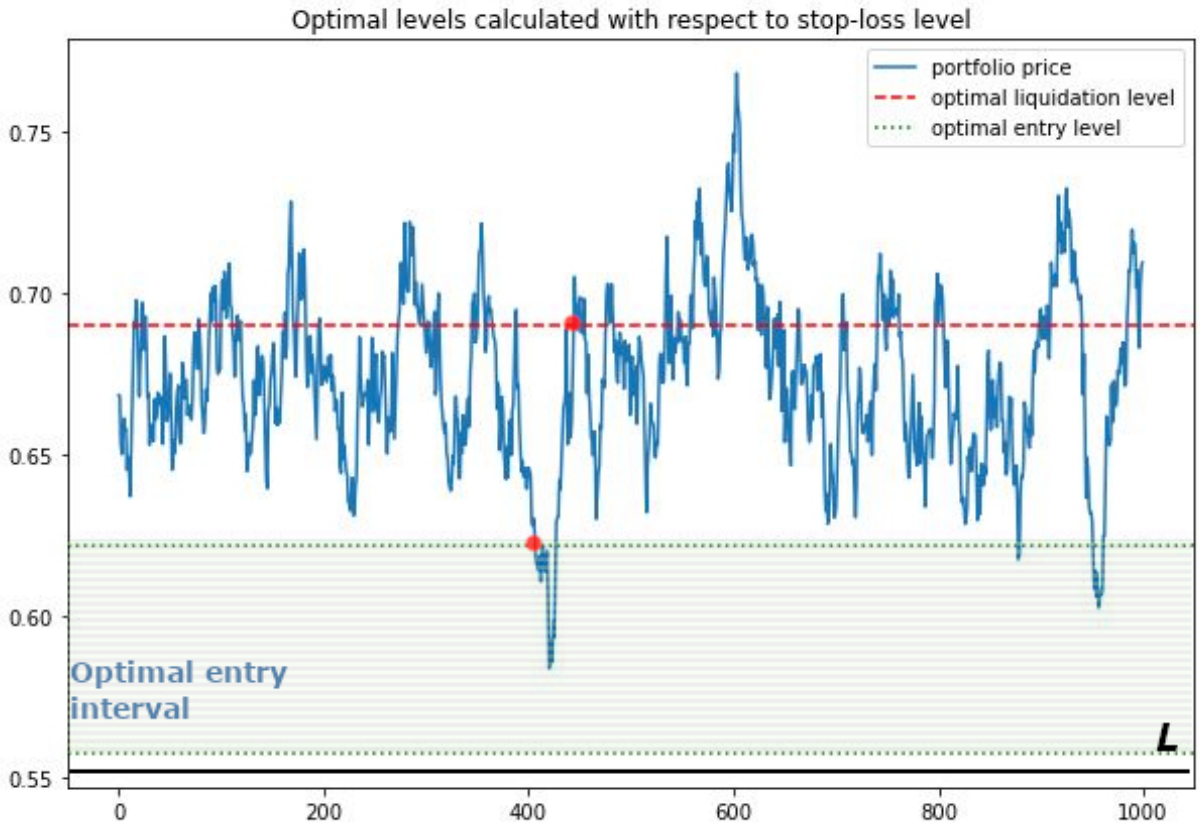


The optimal entry-level d* vs speed of mean reversion (Professor Tim Leung and Xin Li, 2015).

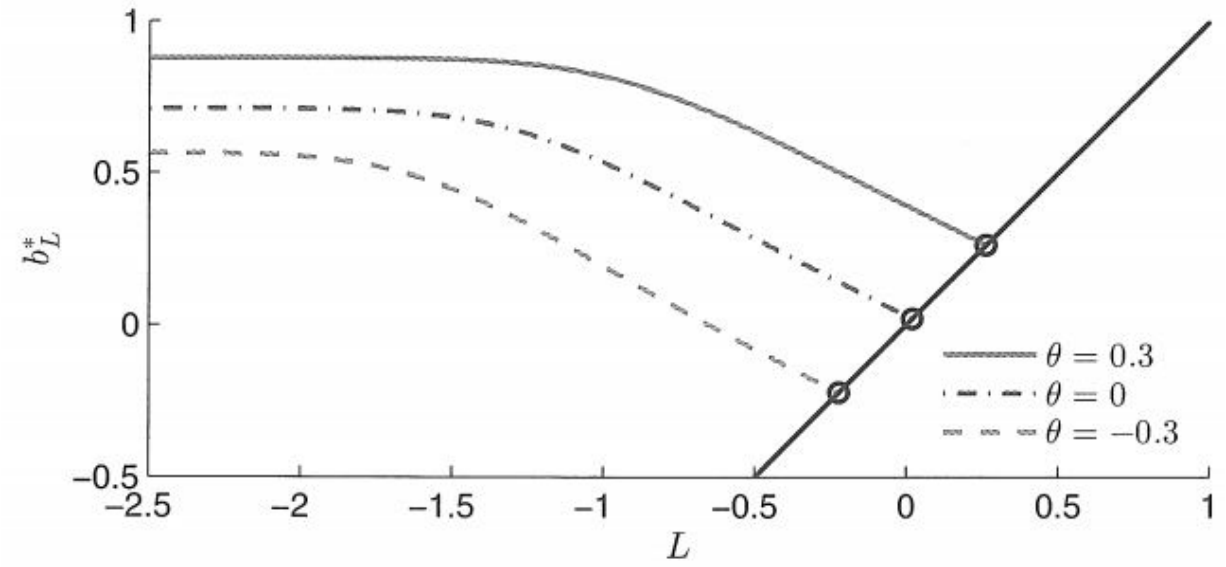


The optimal liquidation level b* vs speed of mean reversion (Professor Tim Leung and Xin Li, 2015).

- With the increase of the long-term mean both entry and liquidation value tend to be higher.
- Faster mean reversion means closer buy and sell levels (where the sell-level value is going to decrease and the entry value – increase.
- The increase in volatility sets the buy and set levels further apart. So if the volatility is high it is possible to delay both entry and exit levels to seek a wider spread.
- High transaction costs also usually mean higher sell levels since we wish to compensate for the loss on transaction costs.



In the case of the problem with the addition of the stop-loss:



The optimal exit threshold vs stop-loss level. The straight line lies where $b_L = L$. (Professor Tim Leung and Xin Li, 2015).

- All correlations from the default model still hold true.
- The optimal liquidation level is strictly decreasing with the increase of the stop-loss level.
- The optimal entry-level becomes an optimal entry interval set strictly above the stop-loss level (L). The signal for buying, in this case, is when the price of a portfolio reaches one of the interval bounds. Particular case being if the current price is between the lower bound and a stop-loss level it is still optimal to wait to avoid exiting at a loss.

"Our model can be considered as the building block for the problem with any finite number of sequential trades"

– Professor Tim Leung and Xin Li, 2015.

CONCLUSION

We encounter myriads of small optimal stopping problems during our lifetime. Asking for a raise, getting a better deal on your car, or picking the best parking spot – they exist in all the areas of life that require decision-making to maximize the good or minimize the bad. Luckily, with a help of stochastic calculus and optimal stopping theory, we can quantify our decision-making process and get the strategy that will give us the best result possible with the most probability. It is only natural that people would try to use it in a world where knowing where to stop is so crucial – quantitative finance

Following the work of Professor Tim Leung and Xin Lee, we explored how the Ornstein-Uhlenbeck process known for modelling mean-reverting interest rates, currency exchange rates, and commodity prices can be used in pairs trading and statistical arbitrage. The two-step process looks the following way: first, we fit the OU process to our pairs-trading portfolio and also choose the optimal ratio between two assets by maximizing the average log-likelihood function, achieving the best mean-reversion and the best fit at the same time. The second step is setting the optimal stopping problem and subsequently solving it. We maximize our expected discounted values of entry and liquidation to find the optimal levels at which we buy or sell our spread. It is also possible to expand the problem by adding the stop-loss level. Both solutions are found with the presumption of the single entry and exit point.

We will talk about the expansion of the optimal stopping problem – the optimal switching problem that accounts for multiple possible entry and exit points in the next blog post on XOU model.

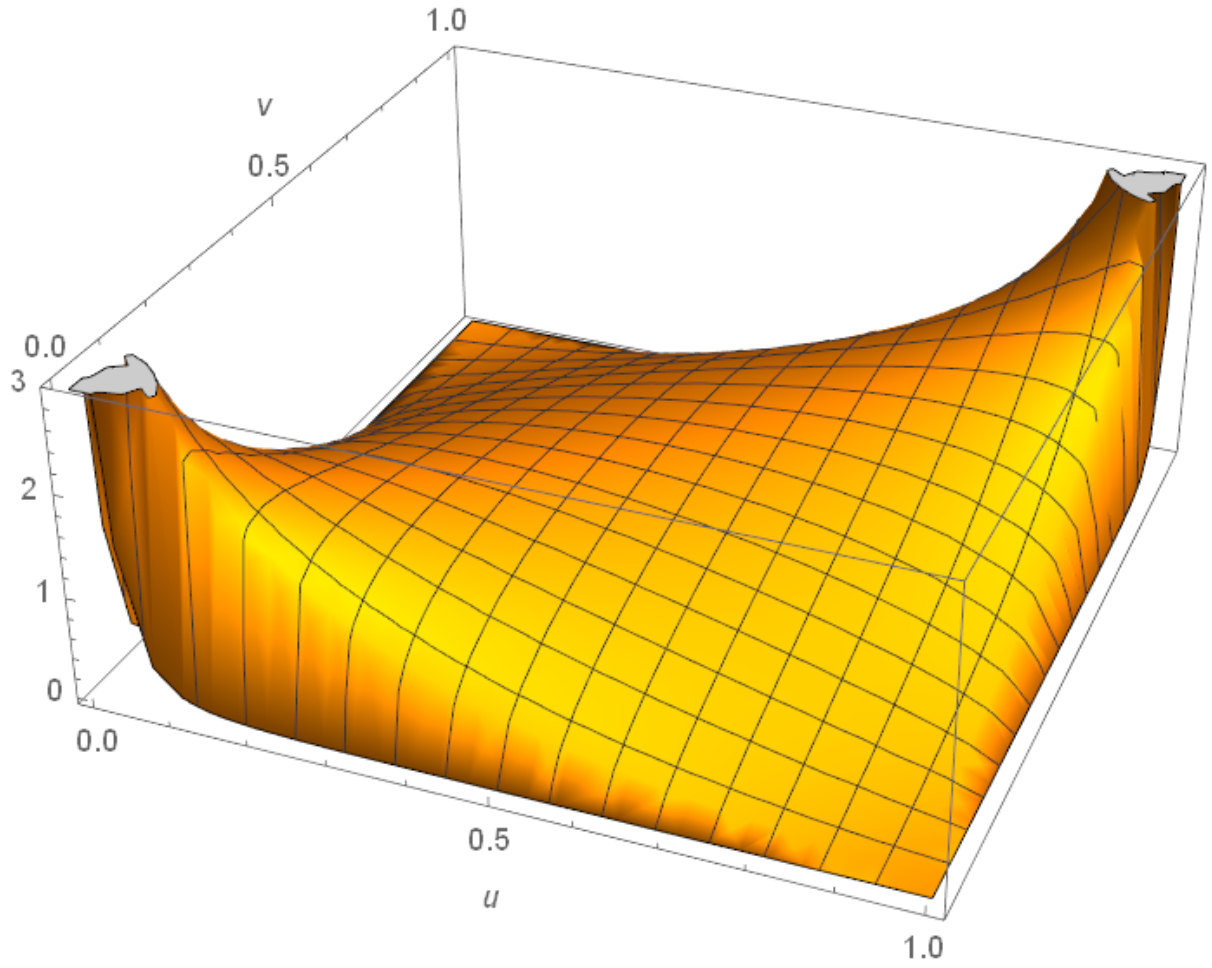
REFERENCES

1. Leung, Tim, and Xin Li. "Optimal Mean Reversion Trading: Mathematical Analysis and Practical Applications." World Scientific Publishing Company (2015).

6.0

COPULA: A DETAILED, BUT PRACTICAL INTRODUCTION

INTRODUCTION



Let's Solve a Mystery

Suppose that you encountered a promising pair of stocks that move closely together, the spread zig-zagged around 0 like some fine needle stitching that sure looks like a nice candidate for mean-reversion bets. What's more, you find out that the two stocks' prices for the past 2 years are all nicely normally distributed. Great! You can avoid some hairy analysis for now. Therefore you fit them as a joint-normal distribution for some sanity check and immediately find that it doesn't look as promising anymore:

For the past two years, there were some major market events, during which the stocks moved together upwards

or downwards, depending on if it was good news or bad news. Your bivariate Gaussian model, in contrast, says that such co-moves are very unlikely to happen since they are so close to the tails of the distribution and you better ignore it. What is more annoying is that the stocks tend to move downward together more than going upward, and the bivariate Gaussian distribution says it should be symmetric.

So what went wrong? For this mini example, there are two major pitfalls present:

Marginal random variables being normally distributed does not mean the two random variables together are jointly-normally distributed.
A bivariate normal distribution has a lot of assumptions, which may not be realistic. To list a few: linear correlation of the data from the two legs, no tail-dependence, and assumes upper-lower end symmetry (upper co-moves and co-lower moves are equally likely).
Of course, as a detail-oriented trader, you decide to improve the model to capture the non-linear dependencies, especially taking into consideration the large moves in the tail and you adjusted the model accordingly. But what if the two stocks are not normally distributed in history, to begin with? What if one stock has an upward drift and it always punches through the 3 standard deviation ceiling of your assumed marginal distribution in the testing set?

Maybe switching to a distance strategy? The simplest version of which looks at standard deviations might have already been overused since it is quite straightforward to implement. To answer the mystery one must understand the concept copula.

WHY MODEL WITH COPULA?

Well, essentially you just want to know how the two legs are "related", and then exploit the structure for profit. **Copula**, a concept that enables mathematicians to analyze the dependency structure from multiple random variables, is an ideal superhero that is specifically tailored to deal with this problem. Just to name a few advantages:

1. It separates the marginal distributions from studying their "relation", so you don't need to worry about taking into account all the possible combinations of possible types of univariate distributions, greatly simplifying the amounts of code needed.

2. You can tune the dependency structure to your own liking, especially the tail dependencies. No more linear dependence assumptions telling you some events that happen yearly should occur every 10 billion years.

3. You can work with multiple stocks in a cohort, instead of just a pair, to create the ultimate trading unit for capturing mispricing in a much higher dimension.

4. Being a relatively novel approach, the playground for copula (especially the higher dimensional stuff) is much less crowded compared to other common quant strategies like vanilla distance and cointegration. This daunting Latin word, let's be fair, scares a lot of people away.

Copula means “link” in Latin (And strictly speaking, its plural is copulae, but I will use copulas here, because I am an applied mathematician and have never used fancy stuff like lemmata or copulae for my writing). The key idea lies in modeling the relation between two random variables’ quantiles to avoid involving idiosyncratic marginal distributions.

Throughout this article, I will guide you through the important facts related to copula, with an applicative mindset, without digging too much into the bog of various mathematical analysis. This article is a definitive, conglomerate collection of important concepts regarding copula collected and derived from multiple journals and textbooks, so it has a lot of formulas and is not meant to be read line-by-line in one go, but as an article one can refer to for resources. We assume a generic understanding of applied probability and statistics. The fitting algorithms, sampling and exact trading strategies are separated for their own dedicated articles to keep the length manageable.

Those are the topics I aim to cover:

- **What is a bivariate copula and how to understand it?**
- **What is tail dependence and why it is crucial for pairs trading?**
- **What are Archimedean copulas and Elliptical copulas and how to understand a mixed copula?**
- **Features (mostly tail dependence) of common copulas related to modeling pairs trading.**
- **Interesting open problems to consider.**

However, if you are still keen on the math side of those, I highly recommend going through the seminal book for a thorough analyst approach: An Introduction to Copulas by Professor Roger Nelsen.

Also, we limit our discussion to **bivariate copulas** unless clarified otherwise to avoid swimming in notations.

AN EXAMPLE

First, let's start by doing the following calculations to try to temporarily bypass the mathematical definition, which may not help much unless we have an idea of where the copula comes from. Essentially we are just making the Q-Q (quantile-quantile) plot of two continuous random variables. Here is the process:

1. For two continuous random variables with fixed distributions, find their own marginal CDFs (cumulative density functions).
2. Sample from the two-random-variable pair multiple times.
3. Map the random samples into their quantile domain by their CDFs, you should get two uniformly distributed quantiles data in $[0, 1] \times [0, 1]$.
4. Plot the pair's quantile-quantile data.

The picture below illustrates our construction well:

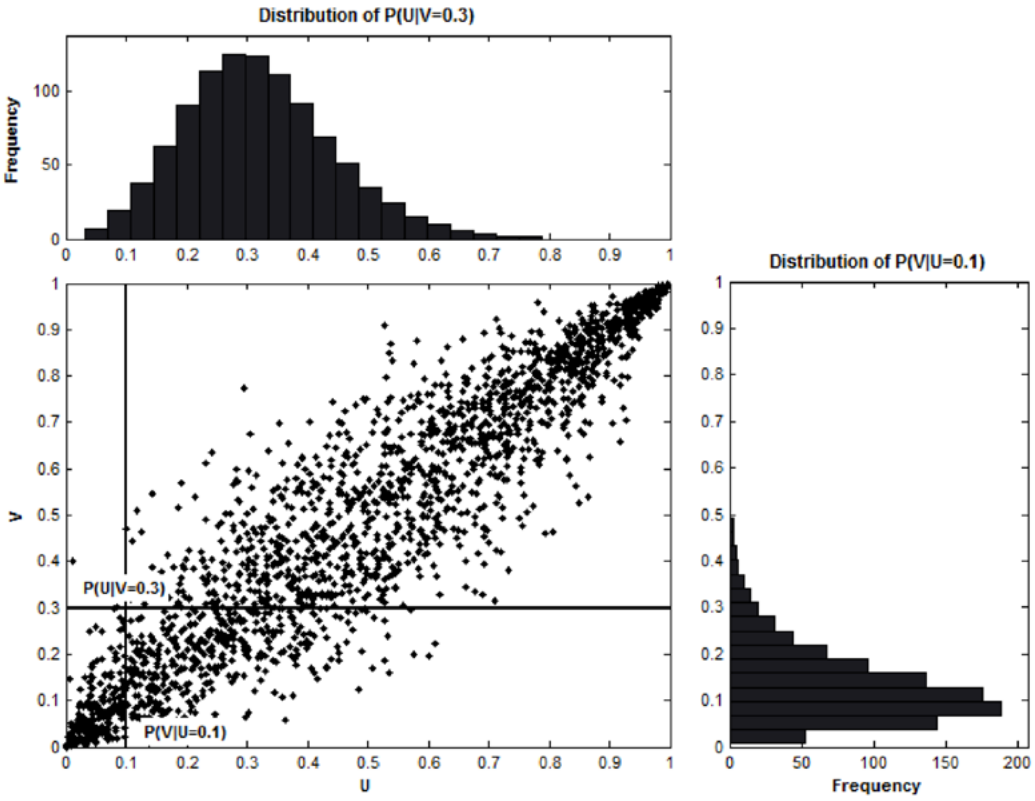


Fig. 1. An illustration of the conditional distribution function of V for a given value of U and the conditional distribution function of U for a given value of V using the N14 copula dependence structure. An example from "Trading strategies with copulas." by Stander, Yolanda, Daniël Marais, and Ilse Botha.

A few points here to clarify: First, knowing the marginal distribution does not in any way determine this Q-Q plot at all, and this Q-Q plot tells the full dependency structure between the two random variables, it tells us how they are "related". Second, we just plotted the samples from a copula according to its copula density $c(u_1, u_2)$, which is generally denoted as c. The denser the dots, the higher the copula density. It is still not the copula itself because copula is the joint cumulative density of quantiles $C(u_1, u_2)$, and is defined in the next section. $c(u_1, u_2) = \dfrac{\partial^2 C(u_1, u_2)}{\partial u_1 \partial u_2}$. Much like when you try to plot a univariate distribution, you sample from its CDF by generating uniformly from $[0, 1]$ to plot its PDF, we are doing the same thing but in 2 dimensions.

BASIC CONCEPTS

Sklar's Theorem

This definition is a natural abstraction of our sampling procedure above.

(Definition using Sklar's Theorem) For two random variables $S_1, S_2 \in [-\infty, \infty]$. S_1 and S_2 have their own fixed, continuous CDFs F_1, F_2. Consider their (cumulative) joint distribution $H(s_1, s_2) := P(S_1 \leq s_1, S_2 \leq s_2)$. Now take the uniformly distributed quantile random variable $U_1(S_1), U_2(S_2)$, for every pair (u_1, u_2) drawn from the pair's quantile we define the bivariate copula $C : [0, 1] \times [0, 1] \to [0, 1]$ as:

$$C(u_1, u_2) = P(U_1 \leq u_1, U_2 \leq u_2)$$

$$= P(S_1 \leq F_1^{-1}(u_1), S_2 \leq F_2^{-1}(u_2))$$

$$= H(F_1^{-1}(u_1), F - 1_2(u_2)),$$

where $F_1^{-1}$and F_2^{-1} are quasi-inverses of the marginal CDFs F_1 and F_2. Let's emphasize that copula is just the joint cumulative density for quantiles of a pair of random variables, do not get scared by its fancy Latin name. (I know it is a lot of symbols and there are more incoming, and if it is the first time for you to see this it might take a moment to wrap your head around all the material. It took me a few days to fully absorb this material. But trust me this theorem is the only part of the whole journey for one to know by heart for applications. For other materials in this article you only need to know them qualitatively and know where the formulas are when required.)

Sklar's theorem guarantees the existence and uniqueness of a copula for two continuous random variables, clearing the way for the wonderful world of applications. Imagine what the world would be if you have to worry about its existence and uniqueness every time you use it, ughhhh!

FORMAL DEFINITION

The following widely used formal definition focuses on the joint distribution (of quantiles) aspect of a copula.

(Formal Definition of Copula) A two-dimensional copula is a function $C : \mathbf{I}^2 \to \mathbf{I}$ where $\mathbf{I} = [0, 1]$

such that

– $(C1) C(0, x) = C(x, 0) = 0$ and $C(1, x) = C(x, 1) = x$ for all $x \in \mathbf{I}$;

– $(C2) C$ is 2-increasing, i.e., for $a, b, c, d \in \mathbf{I}$ with $a \leq b$ and $a \leq b$:

$$Vc([a, b] \times [c, d]) := C(b, d) - C(a, d) - C(b, c) + C(a, c) \geq 0.$$

Keep in mind that the definition of copula C is just the joint cumulative density on quantiles of each marginal random variable, i.e.,

$$C(u1, u2) = \mathbb{P}(U1 \leq u1, U2 \leq u2).$$

And $(C1)(C2)$ naturally comes from joint CDF: $(C1)(C2)$ is quite obvious, and $(C2)$ is just the definition of

$$\mathbb{P}(a \leq U1 \leq b, c \leq U2 \leq d).$$

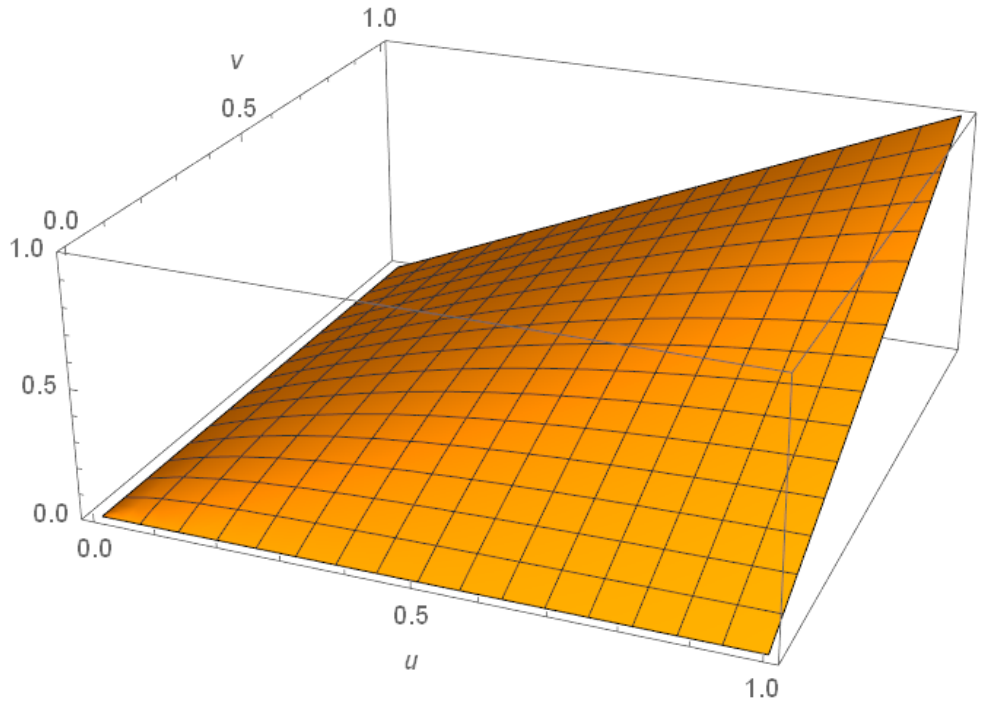


Fig. 2. Plot of C(u, v) for an N13 copula.

RELATED CONCEPTS

We define the (cumulative) **conditional probabilities**:

$$P(U_1 \leq u_1 | U_2 = u_2) := \partial C(u_1, u_2) \partial u_2,$$

$$P(U_2 \leq u_2 | U_1 = u_1) := \partial C(u_1, u_2) \partial u_1,$$

and the **copula density** $c(u_1, u_2)$:

$$c(u_1, u_2) := \partial^2 C(u_1, u_2) \partial u_1 \partial u_2,$$

which by definition is the probability density. Those two concepts are crucial for **fitting a copula to data** and its **application in pairs trading**, and we do not use the value C directly as much.

Then we define coefficients of tail dependence, which is one of the crucial reasons why you want to use copula for modeling in the first place: For a bivariate Gaussian distribution, when it is used to model two stocks' prices or returns, it does not capture co-movements when the market moves wildly altogether very well, especially when the market goes down. Loosely speaking, tail dependence quantifies the strength of large co-moves at the upper or lower tail of each leg's distribution. Picking a copula that correctly reflects one's belief in co-moves on tail events is thus the key.

(Definition of Coefficients for Lower and Upper Tail Dependence) The lower and upper tail dependence are defined respectively as:

$$\lambda l := \lim_{q \to 0^+} \mathbb{P}(U_2 \leq q \mid U_1 \leq q),$$

$$\lambda u := \lim_{q \to 1^-} \mathbb{P}(U_2 > q \mid U_1 > q).$$

And those can be derived from the copula definition:

$$\lambda l = \lim_{q \to 0^+} \frac{\mathbb{P}(U_2 \leq q, U_1 \leq q)}{\mathbb{P}(U_1 \leq q)} = \lim_{q \to 0^+} \frac{C(q, q)}{q},$$

where $\hat{C}(u_1, u_2) = u_1 + u_2 - 1 + C(1 - u_1, 1 - u_2)$ is the **reflected copula**. If $\lambda_l > 0$ then there is lower tail dependence. If $\lambda_u > 0$ then there is upper tail dependence. A reminder is that you should always calculate those limits for each copula whenever you can, and it is in general not obvious from the definition plot or density plot.

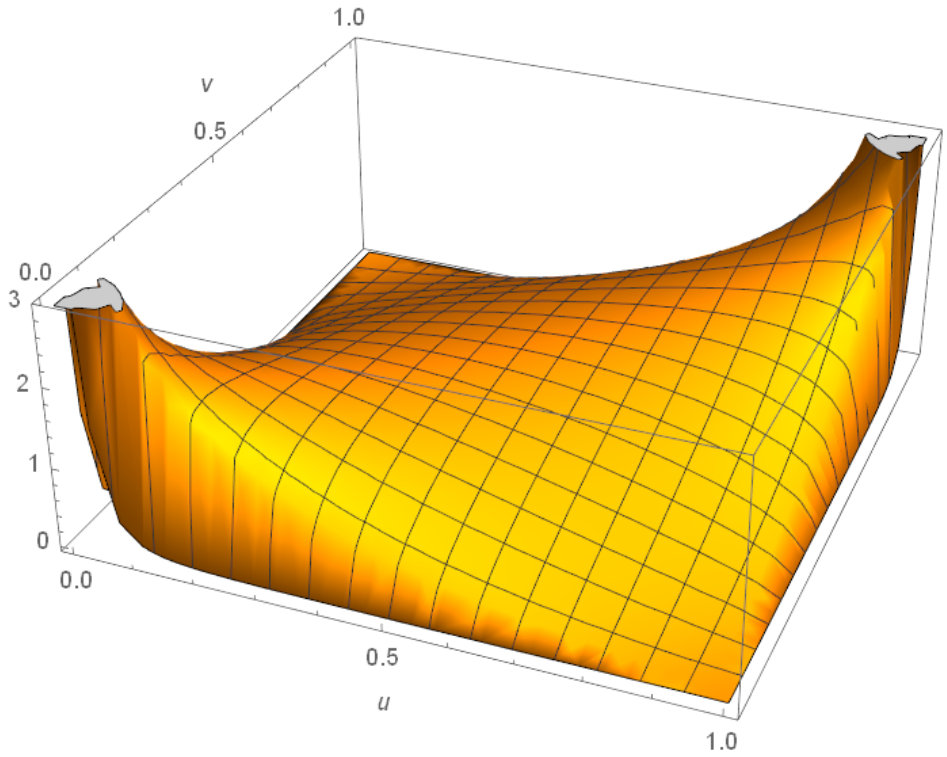


Fig. 3. Plot of c(u,v) for the infamous Gaussian copula. It is often not obvious to decide from the plot whether a copula has tail dependencies. In this case, Gaussian has none.

COPULA TYPES AND GENERATORS

The most commonly used bivariate types and those implemented in ArbitrageLab are **Gumbel**, **Frank**, **Clayton**, Joe, **N13**, **N14**, **Gaussian**, and **Student-t**. All of those except for Gaussian and Student-t copulas fall into the category of **Archimedean copulas**, and Gaussian, Student-t are regarded as **Elliptical copulas**. Any linear combinations of those copulas with positive weights adding up to 1 will lead to **mixed copulas**.

Archimedean Copulas and Their Generators

(Definition of Archimedean Copula) A bivariate copula C is called Archimedean if it can be represented as:

$$C(u_1, u_2; \theta) = \phi^{[-1]}(\phi(u_1; \theta) + \phi(u_2; \theta))$$

where $\phi : [0,1] \times \Theta \to [0, +\infty)$ is called the generator for the copula, $\phi^{[-1]}$ is its pseudo-inverse, defined as

$$\phi^{[-1]}(t;\theta) = \begin{cases} \phi^{-1}(t;\theta), & 0 \le t \le \phi(0;\theta) \\ 0, & \phi(0;\theta) \le t \le \infty \end{cases}$$

As a result, one uses a generator to define an Archimedean copula. The generators' formulae are listed below:

- **Gumbel:** $\phi(t;\theta) = (-\ln t)^{\theta}, \theta \in [1, +\infty)$
- **Frank:** $\phi(t;\theta) = -\ln\left(\dfrac{e^{-\theta t} - 1}{e^{-\theta} - 1}\right), \theta \in [-\infty, \infty) \backslash \{0\}$
- **Clayton:** $\phi(t;\theta) = \dfrac{t^{-\theta} - 1}{\theta}, \theta \in [-1, +\infty) \backslash \{0\}$
- **Joe:** $\phi(t;\theta) = -\ln(1 - (1-t)^{\theta}), \theta \in [1, +\infty)$
- **N13:** $\phi(t;\theta) = (1 - \ln t)^{\theta} - 1, \theta \in [0, +\infty)$
- **N14:** $\phi(t;\theta) = (t^{-1/\theta} - 1)^{\theta}, \theta \in [1, +\infty)$

Again, for applications, you do not need to remember all these unless you are coding the copulas themselves. The takeaway is that Archimedean copulas are parametric, and is uniquely determined by θ. Loosely speaking, θ is the parameter that measures how "closely" the two random variables are "related", and its exact range and interpretation are different across different Archimedean copulas.

People study Archimedean copulas because, in general, arbitrary copulas are quite difficult to work with analytically, whereas Archimedean copulas enable further examinations by having the nice structure above. Two of the most important features of the Archimedean copula are its **symmetry** and **scalability to multiple dimensions**, although a closed-form solution may not be available in higher dimensions.

Elliptical Copulas

For the Gaussian and Student-t copula, the concepts are much easier to follow: Suppose for a correlation matrix $R \in [-1, 1]^{d \times d}$, the multivariate Gaussian copula with parameter matrix R is defined as:

$$CR(\mathbf{u}) := \Phi_R(\Phi^{-1}(u_1), \Phi^{-1}(u_2)),$$

where Φ_R is the joint Gaussian CDF with R being its covariance matrix, Φ^{-1} is the inverse of the CDF of a standard normal. Note that using a correlation matrix instead of covariance will get identical results because copulas only care about quantiles and are invariant under linear scaling.

The Student-t copula can be defined similarly, with $\boldsymbol{\nu}$ being the degrees of freedom:

$$C_{R,\nu}(\mathbf{u}) := \Phi_{R,\nu}(\Phi_\nu^{-1}(u_1), \Phi_\nu^{-1}(u_2))$$

It should also be clear that scaling elliptical copulas to multiple dimensions is also trivial, and they are also symmetric.

Mixed Copulas

Archimedean copulas and elliptical copulas, though powerful by themselves to capture nonlinear relations for two random variables, may suffer from the degree to which they can be calibrated to data. While using a pure empirical copula may be subject to overfitting, a mixed copula that can calibrate the upper and lower tail dependencies usually describes the dependency structure well for its flexibility.

It is pretty intuitive to understand, for example, for a Clayton-Frank-Gumbel (CFG) mixed copula, one needs to specify the θ'$\boldsymbol{s}$, the dependency parameter for each copula component, and their positive weights that sum to 1. In total there are 5 parameters for CFG (3 copula parameters and 2 weights). The weights should be understood in the sense of Markov, i.e., it describes the probability of an observation coming from a component.

For example, we present the Clayton-Frank-Gumbel mixed copula as below:

$$C_{mix}(u_1, u_2; \boldsymbol{\theta}, \mathbf{w}) := w_C C_C(u_1, u_2; \theta C) + w_F C_F(u_1, u_2; \theta_F) + w_G C_G(u_1, u_2; \theta_G)$$

WHICH COPULA TO CHOOSE

For practical purposes, at least for the implemented trading strategies in ArbitrageLab, it is enough to understand tail dependence for each copula and what structures are they generally modeling because sometimes the nuances only come into play for a (math) analyst.

Here are a few key results:

1. **Upper tail dependence means the two random variables are likely to have extremely large values together. For instance, when working with returns series, upper tail dependence implies that the two stocks are likely to have large gains together.**

2. **Lower tail dependence means the two random variables are likely to have small values together. This is much stronger in general compared to upper tail dependence, because stocks are more likely to go down together than going up.**

3. **Frank and Gaussian copulas do not have tail dependencies at all. And Gaussian copula infamously contributed to the 2008 financial crisis by pricing CDOs exactly for this reason.**

4. **Frank copula has a stronger dependence in the center compared to Gaussian.**

5. **For Value-at-Risk calculations, Gaussian copula is overly optimistic and Gumbel is too pessimistic [Kole et al., 2007].**

6. **Copulas with upper tail dependence: Gumbel, Joe, N13, N14, Student-t.**

7. **Copulas with lower tail dependence: Clayton, N14 (weaker than upper tail), Student-t.**

8. **Copulas with no tail dependence: Gaussian, Frank.**

Now we can look back at the mystery at the very beginning of the article on what went wrong: The bivariate Gaussian distribution assumes 1. All marginal random variables are Gaussian; 2. The copula that relates the two Gaussian marginals is also Gaussian. A Gaussian copula does not have any tail dependence. What's worse, it requires symmetry on both upward co-moves and downward moves. Those assumptions are very rigid, and they just can't compete with dedicated copula models for flexibility.

OPEN PROBLEMS

Copula is still a relatively modern concept in probability theory and finance. There are still a lot of interesting open problems to consider, and being able to solve any of these below will guarantee an edge on someone's copula model.

1. Parametrically fit a Student-t and mixed copula.

2. Existence of statistical properties that justify two random variables to have an Archimedean copula (like one can justify a random variable is normal) [Nelsen, 2003].

3. Take advantage of copula's ability to capture nonlinear dependencies but also adjust it for time series, so that the sequence of data coming in makes a difference.

4. Analysis of copulas when it is used on time series instead of independent draws. (For example, how different it is when working with the average μ on time series, compared to the average μ on a random variable?)

5. Adjust copulas so it can model when dependency structure changes with time.

6. All copulas mentioned, "pure" or mixed, assume symmetry. It would be nice to see an analysis of asymmetric pairs modeled by copulas.

7. The common trading logics used for copulas are still relatively primitive and can be considered a fancier version of a distance strategy and hence there is room for improvements.

8. A careful analysis done on goodness-of-fit for using copulas with tail dependencies versus those without, for various markets and various times, especially when markets are having large swings.

REFERENCES

1. Liew, R.Q. and Wu, Y., 2013. Pairs trading: A copula approach. Journal of Derivatives & Hedge Funds, 19 (1), pp.12-30.
2. Stander, Y., Marais, D. and Botha, I., 2013. Trading strategies with copulas. Journal of Economic and Financial Sciences, 6 (1), pp.83-107.
3. Schmid, F., Schmidt, R., Blumentritt, T., Gaißer, S. and Ruppert, M., 2010. Copula-based measures of multivariate association. In Copula theory and its applications (pp. 209-236). Springer, Berlin, Heidelberg.
4. Huard, D., Évin, G. and Favre, A.C., 2006. Bayesian copula selection. Computational Statistics & Data Analysis, 51 (2), pp.809-822.
5. Kole, E., Koedijk, K. and Verbeek, M., 2007. Selecting copulas for risk management. Journal of Banking & Finance, 31 (8), pp.2405-2423.
6. Nelsen, R.B., 2003, September. Properties and applications of copulas: A brief survey. In Proceedings of the first brazilian conference on statistical modeling in insurance and finance (pp. 10-28). University Press USP Sao Paulo.
7. Cai, Z. and Wang, X., 2014. Selection of mixed copula model via penalized likelihood. Journal of the American Statistical Association, 109 (506), pp.788-801.
8. Liu, B.Y., Ji, Q. and Fan, Y., 2017. A new time-varying optimal copula model identifying the dependence across markets. Quantitative Finance, 17 (3), pp.437-453.
9. Demarta, S. and McNeil, A.J., 2005. The t copula and related copulas. International statistical review, 73 (1), pp.111-129.
10. Nelsen, R.B., 2007. An introduction to copulas. Springer Science & Business Media.

7.0

COPULA: SAMPLING AND FITTING TO DATA

OVERVIEW

Whether it is for pairs trading or risk management, two natural questions to ask before putting copula for use are: How to draw samples from a copula? How should one fit a copula to data? The necessity of fitting is quite obvious, otherwise, there is no way to calibrate our model for pairs trading or risk analysis using historical data.

For sampling, it is mostly for making a Q-Q plot against the historical data as a sanity check. Note that a copula natively cannot generate future price time series since it treats time series data as independent draws from two random variables, and thus has no information regarding the sequence, which is vital in time series analysis. One way to think about sampling from a copula trained by time series is that it gives the likelihood of where the next data point is going to be, regardless of the input sequence.

The former question about sampling has an analytical answer for all the bivariate copulas we care about (Gumbel, Clayton, Frank, Joe, N13, N14, Gaussian, Student-t, and any finite mixture of those components). As with pretty much all analytical approaches in applications, it requires a bit of mathematical elbow grease to wrap the head around a few concepts. But once employed, it is pretty fast and reliable.

The latter question about fitting to data is still somewhat open, and the methods are statistical. Therefore those are relatively easy to understand, but they might suffer from performance issues and may require active tuning of hyperparameters.

Throughout this post, we will discuss:

- **The method employed in sampling from a copula.**
- **Fitting a "pure" copula to data.**
- **Fitting a mixed copula to data using an EM algorithm.**

We limit our discussion to bivariate cases unless otherwise specified. You can also refer to my previous article Copula: A Detailed, But Practical Introduction to brush up on the basics.

Another note is that all the algorithms mentioned in this article will be incorporated in the copula trading module from ArbitrageLab since version 0.3.0 so that you do not need to code them by yourself, and this article is meant to be a technical discussion for interested readers who want to know how things are implemented, where the bottlenecks are in performance, and what can and cannot be achieved intrinsically based on our current understanding of copula.

SAMPLING

Suppose you have a joint cumulative density function $H(x_1, x_2) = \mathbb{P}(X_1 \leq x_1, X_2 \leq x_2)$ and you want to sample a few points from it. Mathematically it is pretty straightforward:

- **Draw a number p uniformly in $[0, 1]$ as the probability.**
- **Find the level curve $\Gamma = \{(x_1, x_2) | H(x_1, x_2) = p\}$, and draw a point uniformly from this curve via arc-length parameterization.**

Done! Can we apply this to copulas? Theoretically not really, practically a hard no. Finding the level curve is in general a bad idea from a computational point of view. There exist other algorithms that perform better than the above such as Metropolis-Hastings. However, all the copulas we care about have some nice structures, and we are gonna take advantage of them.

"Pure" Copula

Note that "pure" copula is in no way a formal name. I call Archimedean and elliptical copulas together as "pure" to distinguish them from mixed copulas (in the Markovian sense, we will address it later) of Archimedean and elliptical components.

For Archimedean copulas, the general methodology for sampling or simulation comes from [Nelsen, 2007]:

1. Generate two uniform in $[0, 1]$ i.i.d. 's $(v1, v2)$.
2. Calculate $w = K_c^{-1}(v_2), K_c(t) = t - \dfrac{\phi(t)}{\phi'(t)}$.
3. Calculate $u_1 = \phi^{-1}[v_1 \phi(w)]$ and $u_2 = \phi^{-1}[(1 - v_1)\phi(w)]$.
4. Return (u_1, u_2).

For the Frank and Clayton copula, the above method can greatly be simplified due to having closed-form solutions for step 2. Otherwise, one will have to use appropriate numerical methods to find w. Interested readers can check Procedure to Generate Uniform Random Variates from Each Copula for some of the simplified forms.

For Gaussian and Student-t copulas, one can follow the procedures below:

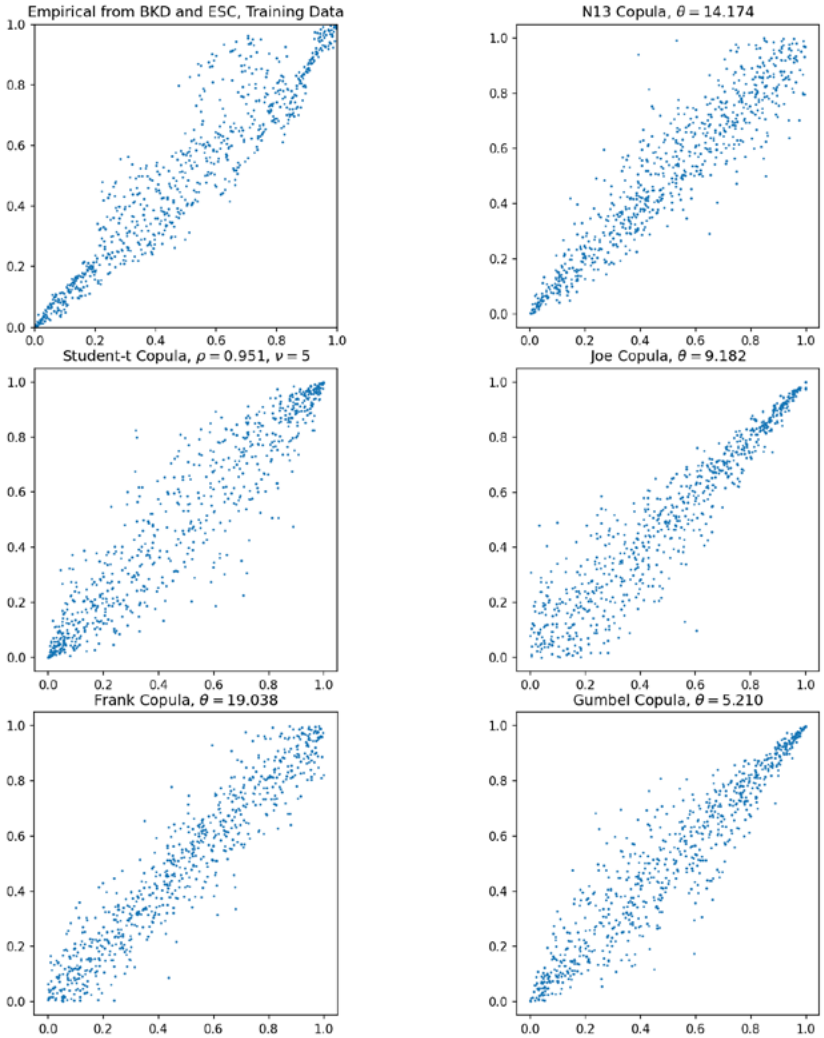
1. Generate a pair (v_1, v_2) using a bivariate Gaussian/Student-t distribution with desired correlation (and degrees of freedom).
2. Transform those into quantiles using CDF from standard Gaussian or Student-t distribution (with desired degrees of freedom). i.e., $u_1 = \Phi(v_1), u_2 = \Phi(v_2)$.
3. Return (u_1, u_2)

Mixed Copula

A mixed copula has two sets of parameters: parameters corresponding to the dependency structure for each component copula (let's call them copula parameters for reference) and positive weights that sum to 1. For example, we present the Clayton-Frank-Gumbel mixed copula as below:

$$C_{mix}(u_1, u_2; \theta, \mathbf{w}) := w_C C_C(u_1, u_2; \theta_C) + w_F C_F(u_1, u_2; \theta_F) + w_G C_G(u_1, u_2; \theta_G)$$

The weights should be interpreted in the Markovian sense, i.e., the probability of an observation coming from a component. (Do not confuse them with the BB families where the mixture happens at the generator level.) Then sampling from a mixed copula becomes easy: for each new sample, just choose a component copula with associated probability, and generate a pair from that copula.



Empirical copula and sampled N13, Student-t, Joe, Frank, and Gumbel copulas.

FITTING TO DATA

This is an active research field. Here we are just introducing some commonly used methods. Note that every fitting method should come with a way for evaluation. For the same dataset, one can use the classic sum of log-likelihood, or various information criteria like AIC, SIC, and HQIC by also taking account of the number of parameters and sample size. For evaluations across datasets, there currently are no widely accepted methods.

Now, we face two issues for fitting:

1. **How to translate marginal data into quantiles?**

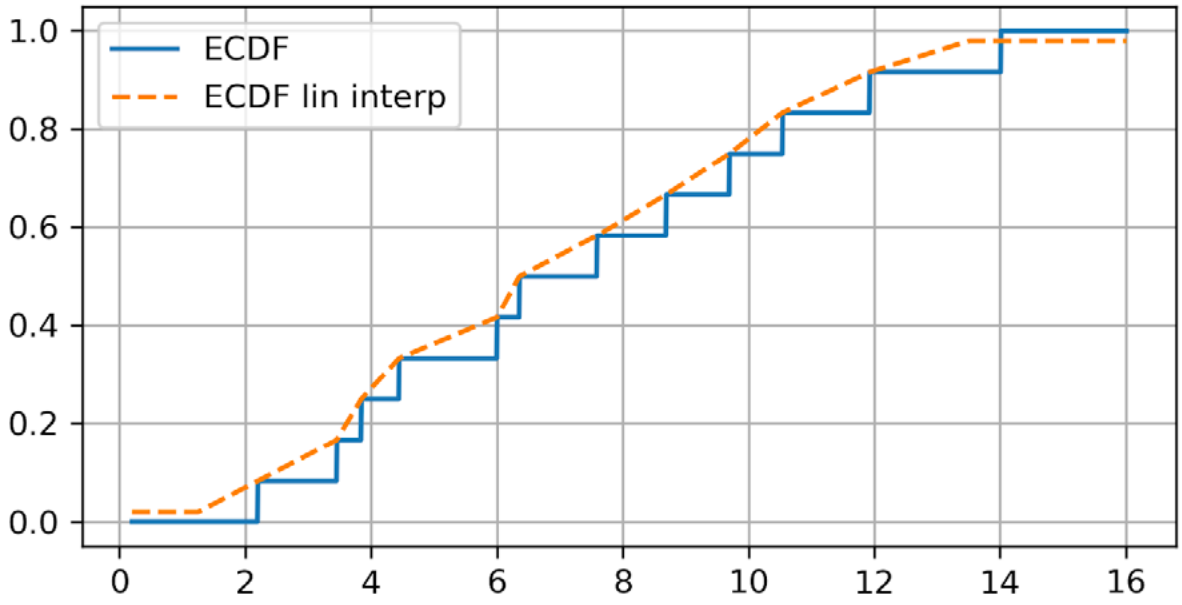
2. **How to fit a copula with quantiles?**

Maximum Likelihood and ECDF

All copulas we have mentioned so far are parametric, i.e., they can be defined uniquely by parameter(s). Hence it is possible to wrap an optimizer over the model and set the sum of log-likelihood generated from copula density $c(u_1, u_2)$ as the objective function to maximize. However this is slow, and with 4 or 5 parameters to fit even for a bivariate "pure" copula with arbitrary marginal distributions it may not be ideal. Hence we usually do not go for this approach, and the empirical CDFs (ECDFs) are generally used instead.

Note that using ECDF will require the marginal random variable to be approximately stationary, otherwise they may go out of bound of a training set infinitely often. This can happen when one fits copula to two stocks prices time series when at least one of them has a positive drift, then the marginal quantile will almost surely be one after a certain point.

Also, the ECDF function provided by a statsmodels library is a step function. And it thus does not work well where the density of data is thin. A nice adaptation is to use linear interpolations between the steps. Moreover, it makes computational sense to wrap the ECDF's value within $[\varepsilon, 1 - \varepsilon]$ to avoid edge results in 0 or 1, which may lead to infinite values in copula density and screw up the fitting process. We provide the modified ECDF in the copula module as well, and by default, all trading modules are built based on it. See the picture below for comparison:



ECDF vs. ECDF with linear interpolation. The max on the training data 14, the min is 2.2. We made the \varepsilon for upper and lower bound much larger on our ECDF for visual effect.

Pseudo-Maximum likelihood

We follow a two-step pseudo-MLE approach as below:

1. Use Empirical CDF (ECDF) to map each marginal data to its quantile.
2. Calculate Kendall's $\hat{\tau}$ for the quantile data, and use Kendall's $\hat{\tau}$ to calculate $\hat{\theta}$.

For elliptical copulas, θ estimated here is their ρ , the correlation parameter.

For Archimedean copulas, τ and θ are implicitly related via

$$\tau(\theta) = 1 + 4 \int_0^1 \frac{\phi(t;\theta)}{\phi'(t;\theta)} dt$$

Then one inversely solves $\hat{\theta}(\hat{\tau})$. For some copulas, the inversion has a closed-form solution. For others, one has to use numerical methods.

For elliptical copulas, we calculate Kendall's $\hat{\tau}$ and then find $\hat{\rho}$ via

$$\hat{\rho} = \sin\left(\frac{\hat{\tau}\pi}{2}\right)$$

for the covariance matrix $\sigma_{2\times 2}$ (though technically speaking, for bivariate copulas, only correlation ρ is needed, and thus it is uniquely determined) from the quantile data, then use $\sigma_{2\times 2}$ for a Gaussian or Student-t copula. Fitting by Spearman's ρ is the variance-covariance matrix from data for elliptic copulas is also practised by some. But Spearman's ρ is, in general, less stable than Kendall's τ (though with faster calculation speed). And using variance-covariance implicitly assumes a multivariate Gaussian model, and it is sensitive to outliers.

A Note about Student-t Copula

Theoretically speaking, for Student-t copula, determining ν(degrees of freedom) analytically from an arbitrary time series is still an open problem. Therefore we opted to use a maximum likelihood fit for ν for the family of Student-t copulas initiated by $\sigma_{2\times 2}$. This calculation is relatively slow.

EM TWO-STEP METHOD FOR MIXED COPULA

Specific Problems with Maximum Likelihood

Archimedean copulas and elliptical copulas, though powerful by themselves to capture nonlinear relations for two random variables, may suffer from the degree to which they can be calibrated to data, especially near the tails. While using a pure empirical copula may be subject to overfitting, a mixed copula that can calibrate the upper and lower tail dependence usually describes the dependency structure well for its flexibility.

But this flexibility comes at a price: fitting it to data is far from trivial. Although one can, in principle, wrap an optimization function for finding the individual weights and copula parameters using max likelihood, realistically this is a bad practice for the following reasons:

1. The outcome is highly unstable and is heavily subject to the choice of the maximization algorithm.

2. A maximization algorithm fitting 5 parameters for a Clayton-Frank-Gumbel (CFG) or 6 parameters for Clayton-Student-Gumbel (CTG) usually tends to settle to a bad result that does not yield a comparable advantage to even its component copula. For example, sometimes the fit score for CTG is worse than a direct pseudo-max likelihood fit for Student-t copula.

3. Sometimes there is no result for the fit since the algorithm does not converge.

4. Some maximization algorithms use a Jacobian or Hessian matrix, and for some copulas, the derivative

computation does not numerically stay stable.

5. Often the weight of a copula component is way too small to be reasonable. For example, when an algorithm says there is 0.1% Gumbel copula weight in a dataset of 1000 observations, then there is on average 1 observation that comes from that Gumbel copula. It is just bad modeling in every sense.

The EM Algorithm

Instead, we adopt a two-step expectation-maximization (EM) algorithm for fitting mixed copulas, adapted from [Cai, Wang 2014]. This algorithm addresses all the above disadvantages of a generic maximization optimizer. The only obvious downside is that the CTG copula may take a while to converge to a solution for certain data sets.

Suppose we are working on a three-component mixed copula of bivariate Archimedeans and ellipticals. We aim to maximize the objective function for copula parameters θ and weights $\mathbf{w}$.

$$Q(\theta, \mathbf{w}) = \sum_{t=1}^{T} \log \left[\sum_{k=1}^{3} w_k c_k(u_{1,t}, u_{2,t}; \theta_k) \right] - T \sum_{k=1}^{3} p_{\gamma,a}(w_k) + \delta \left(\sum_{k=1}^{s} w_k - 1 \right),$$

where T is the length of the training set or the size of the observations; k is the dummy variable for each copula component; $p_{\gamma,a}(\cdot)$ is the smoothly clipped absolute deviation (SCAD) penalty term with tuning parameters γ and a and it is the term that drives small copula components to 0; and last but not least δ the Lagrange multiplier term that will be used for the E-step is:

$$\delta = T p'_{\gamma,a}(w_k) - \sum_{t=1}^{T} \frac{c_k(u_{1,t}, u_{2,t}; \theta_k)}{\sum_{j=1}^{3} w_j c_j(u_{1,t}, u_{2,t}; \theta_j)}$$

E-step

Iteratively calculate the following using the old θ and $\mathbf{w}$ until it converges:

$$w_k^{new} = [w_k p'_{\gamma,a}(w_k) - \frac{1}{T} \sum_{t=1}^{T} \frac{c_k(u_{1,t}, u_{2,t}; \theta_k)}{\sum_{j=1}^{3} w_j c_j(u_{1,t}, u_{2,t}; \theta_j)}] \times [\sum_{j=1}^{3} w_j p'_{\gamma,a}(w_j)]^{-1}$$

M-step

Use the updated weights to find new θ, such that Q is maximized. One can use truncated Newton or Newton-Raphson for this step. This step, though still a wrapper around some optimization function, is much better than estimating the weights and copula parameters altogether.

We then iterate the two steps until it converges. It is tested that the EM algorithm is oracle and sparse. Loosely speaking, the former means it has good asymptotic properties, and the latter says it will trim small weights off.

Possible Issues Discussion

The EM algorithm is still not mathematically perfect, and has the following issues:

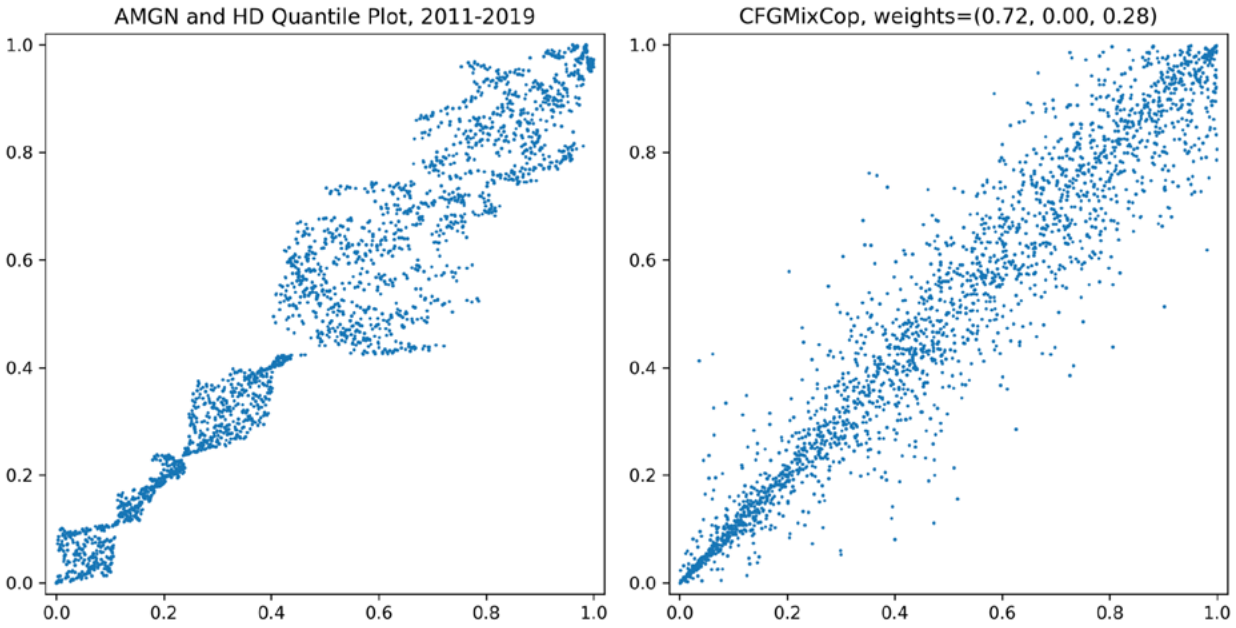
1. It is slow for some data sets. Specifically, the CTG is slow due to the bottleneck from Student-t copula.

2. If the copulas mixture has similar components, for instance, Gaussian and Frank, then it cannot pick the correct component weights well. Luckily for fitting data, usually the differences are minimal.

3. The SCAD parameters \gamma and a may require some tuning, depending on the data set it fits. Some literature suggested using cross-validation to find a good pair of values. We did not implement this in the ArbitrageLab because it takes quite long for the calculation to a point it becomes unrealistic to run it.

4. Sometimes the algorithm taken from scipy.optimization package throws warnings because the optimization algorithm underneath does not strictly follow the prescribed bounds. Usually, this is not an issue that will compromise the result, and some minor tunings on SCAD parameters can solve this issue.

5. Sometimes the pattern in the data cannot be explained well by any of our copulas, even if they are initially selected by Kendall's tau or Euclidean distance, very likely to co-move together and thus have the potential for pairs trading. See below for an example of price data. Changing to returns data usually will be much better for fitting, but the data will have much more noise, and trading algorithms should be designed specifically with this into consideration.



AMGN and HD quantile plot vs CFG mixed copula sample.

REFERENCES

1. Schmid, F., Schmidt, R., Blumentritt, T., Gaißer, S. and Ruppert, M., 2010. Copula-based measures of multivariate association. In Copula theory and its applications (pp. 209-236). Springer, Berlin, Heidelberg.
2. Huard, D., Évin, G. and Favre, A.C., 2006. Bayesian copula selection. Computational Statistics & Data Analysis, 51(2), pp.809-822.
3. Nelsen, R.B., 2003, September. Properties and applications of copulas: A brief survey. In Proceedings of the first brazilian conference on statistical modeling in insurance and finance (pp. 10-28). University Press USP Sao Paulo.
4. Cai, Z. and Wang, X., 2014. Selection of mixed copula model via penalized likelihood. Journal of the American Statistical Association, 109(506), pp.788-801.
5. Nelsen, R.B., 2007. An introduction to copulas. Springer Science & Business Media.

8.0

COPULA: A UNIFIED OVERVIEW OF COMMON STRATEGIES

INTRODUCTION

Systematic approaches of pairs trading gained popularity from the mid-1980s. Gatev et al (2006) examined the profitability of a distance-based strategy on normalized prices. Cointegration is another common strategy incorporated approach as discussed in [Vidyamurthy (2004)]. Both methods are tied to the idea of a mean-reverting bet, and the trading signals are generated from the spread: when the spread widens, it is expected to narrow, and when it does happen the trader pockets the profit.

We have previously talked about several advantages from copula-based models in **Copula: A Detailed, But Practical Introduction**, and as a tool it analyzes the dependence structure among several random variables (For pairs trading it is just 2 random variables). We quickly summarize it here:

1. Non-linear relations in tail moves of stocks are naturally captured by copula models. This is often where the buy-low-and-sell-high opportunities arise.

2. Copula works with quantiles data, and thus greatly simplifies the workflow by bypassing analyzing idiosyncratic marginal distributions.

Apart from those two relatively known advantages, it has another interesting feature specifically for creating trading strategies: the signal is not generated from the spread but individually from the two legs. Therefore a trader can use different methods to assemble the signals for creating the final long/short/exit decision and thus create a huge amount of variations under the copula framework to avoid a crowded playground.

One way to understand such a feature is that copula models the relative mispricing from each stock, and the end user's work is to combine to make use of them. Being a relatively novel approach, there is no hard-and-fast rule here, and a lot of work indeed can be done to further improve those strategies especially from pairs-selection. We have looked through the most well-regarded approaches and their variations, and we found it more meaningful to look at them through a unified perspective. This is a truly vast topic and to make this article manageable, we restrict our discussions to the followings:

1. What information does copula provide for trading?

2. Common trading strategies, variations, and what are the moving parts?

3. Common pairs selection methods and what is missing.

4. Possible ideas for implementation.

KEY IDEA: CONDITIONAL PROBABILITY

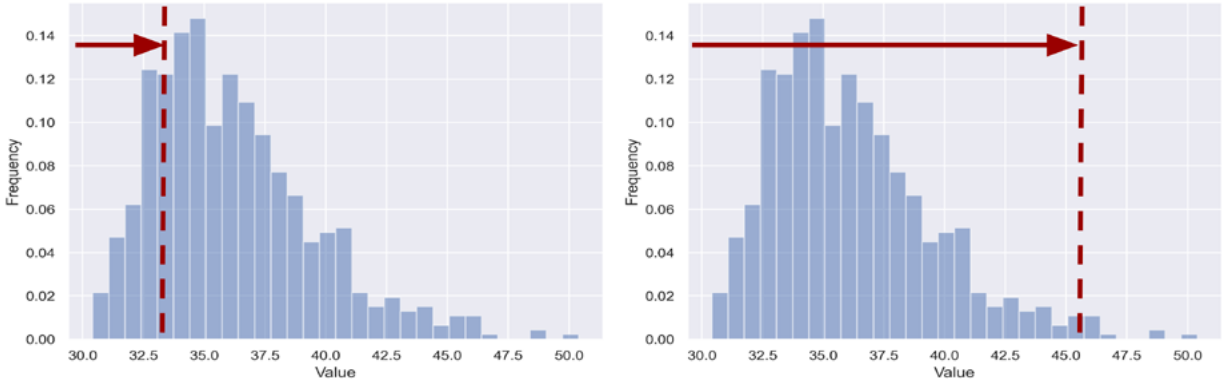
Suppose C(u1,u2) is a bivariate copula with uniform random variable input u1 and u2 as quantiles from data mapped by their marginal CDFs, the (cumulative) conditional probabilities for each leg is defined as the following:

$$P(U_1 \leq u_1 | U_2 = u_2) := \frac{\partial C(u_1, u_2)}{\partial u_2}, P(U_2 \leq u_2 | U_1 = u_1) := \frac{\partial C(u_1, u_2)}{\partial u_1}.$$

Almost all copula-based pairs trading strategies that I am aware of use conditional probabilities as the core. Their values are interpreted as below:

- When $P(U_1 \leq u_1 | U_2 = u_2) < 0.5$, then variable 1 is considered undervalued.
- When $P(U_1 \leq u_1 | U_2 = u_2) > 0.5$, then variable 1 is considered overvalued.

Similarly, this relation holds for variable 2 using $P(U_2 \leq u_2 | U_1 = u_1)$. See the picture below for a demonstration:

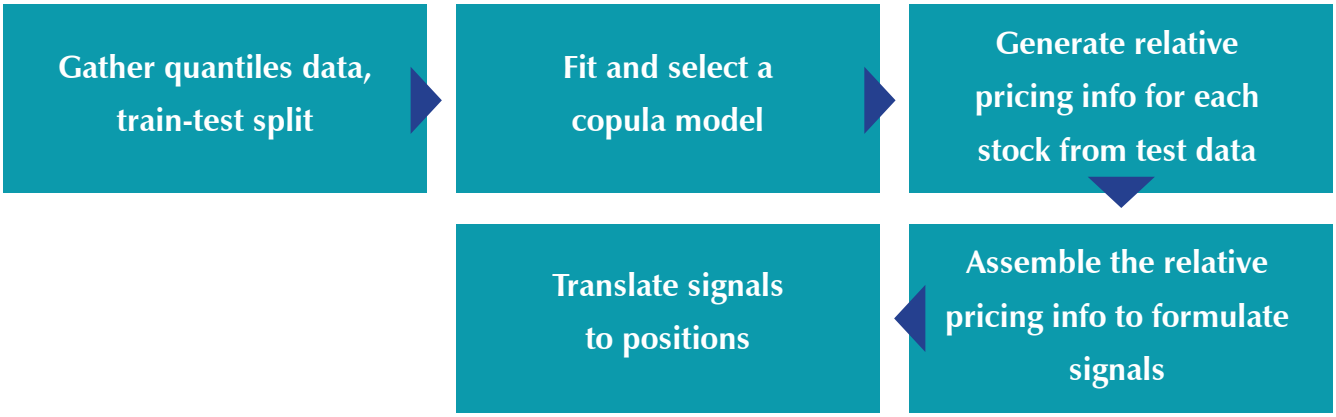


The histogram represents some independent sample draws from a fixed conditional distribution, thus the total frequency sums up to 1. When the conditional cumulative density is less than 0.5, as shown on the left, the value is considered small. Similarly on the right, it is considered a large value.

A few things to keep in mind: First, this quantity is model dependent, which means it will be almost surely different if you use another different copula. Second, to be able to use a copula for calculating returns series, one needs to fit a copula with historical data, and all the above undervalue or overvalue judgment is based

on history. Third, for a given data point (u_1, u_2), you will get conditional probabilities from each random variable individually. For example, if the input quantile comes from stock prices, then you will get information like "stock 1 is overpriced, given the current stock 2's price" and "stock 2 is underpriced, given the current stock 1's price" together. A very important note is that one does not imply another necessarily, and the copula may tell you that both stocks are overvalued/undervalued sometimes. It depends on you how you would like to assemble the information.

Therefore, all the trading strategies utilizing conditional probabilities can be unified under the following framework. In **ArbitrageLab** we provided the most commonly used ones that we will discuss for the next few sections, and a few places you can tweak the logic to formulate your strategy.



Thus, the possible moving parts are clear:

1. What data to use? Prices or returns?

2. How to generate relative mispricing information of stocks from a copula?

3. What logic to use to assemble information from the two legs?

STRATEGY 1: SIMPLE THRESHOLDS ON PRICES

This is proposed in [Liew et al. 2013] [Botha et al. 2013].

Strategy

This strategy is relatively easy to understand and implement since it works directly with the pair's prices series using the conditional probability thresholds: Suppose we define an upper threshold bup (e.g. 0.95) and a lower threshold blo (e.g. 0.05), then the logic goes as follows as described in the literature:

- **Opening rules:**
    - If $P(U_1 \leq u_1 | U_2 = u_2) \leq b_{lo}$ AND $P(U_2 \leq u_2 | U_1 = u_1) \geq b_{up}$, then stock 1 is undervalued, and stock 2 is overvalued. Hence we long the spread. (1 in position)
    - If $P(U_2 \leq u_2 | U_1 = u_1) \leq b_{lo}$ AND $P(U_2 \leq u_2 | U_1 = u_1) \leq b_{lo}$, then stock 2 is undervalued, and stock 1 is overvalued. Hence we short the spread. (-1 in position)
- **Exit rule:**
    - If BOTH/EITHER conditional probabilities cross the boundary of 0.5, then we exit the position, as we consider the position no longer valid. (0 in position).

Here the spread can come from the hedge ratio using the training data. For example, you can run a simple OLS or use something fancier like the Johansen cointegration test and pick their eigenvectors. Alternatively, you may run a dollar-neutral strategy, which is also quite popular to pair with copula-based methods.

This strategy also works with normalized prices or log prices and will produce an identically fitted copula, since copula uses quantiles data. Any strictly monotone transformation using an increasing function on the marginal data will thus yield identical results.

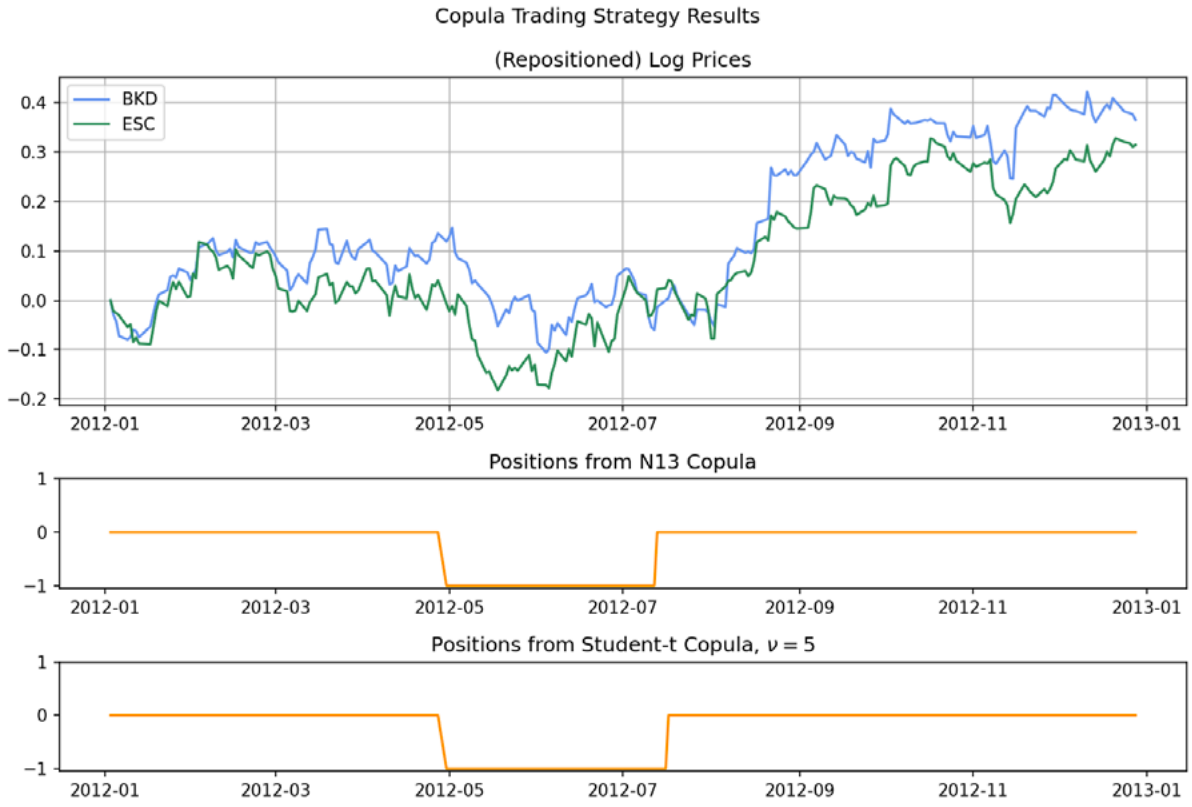
For ambiguities, the following situations are not specified:

1. When there is an open signal and an exit signal.

2. When there is an open signal and currently there is a position.

3. When there is a long and short signal together.

One can adjust the logic and specify ambiguities to see what happens. Often the change of fundamental logic for signal generation greatly influences the performance of the strategy, and wrong combinations can drive a promising strategy unprofitable. The ambiguities are less influential overall but they still should be specified for a working strategy both for interpretability and preventing unexpected actions.



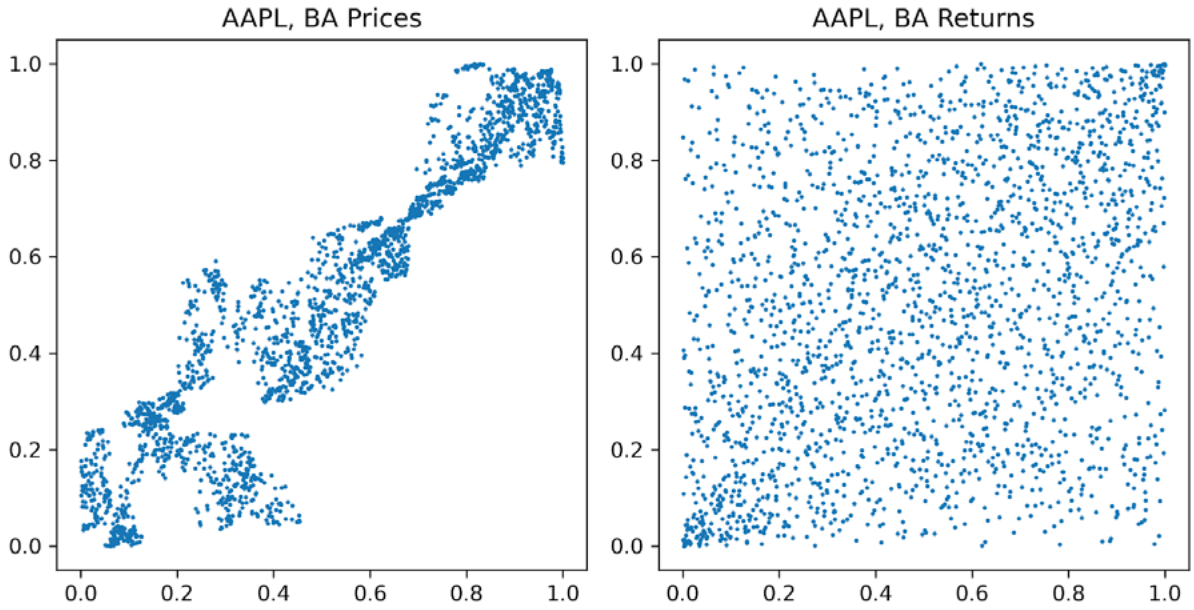
Comments

Based on our tests, we found using AND for opening, OR for exiting on average captures better trading opportunities and exits on time. This strategy is also pretty robust: using a different copula with similar fit scores usually leads to almost identical positions and P&L. However we still have a few concerns, regarding using prices series as input:

1. Prices of stocks are in general not stationary and since copula uses quantiles data, if a stock has an upward drift then the model will be broken after some time.

2. Copula assumes the input data to be independent draws from two random variables with fixed distributions, and a stock price time series definitely does not satisfy such an assumption. Instead, stock prices have term structures and they tend to aggregate in their Q-Q plot. Such a structure may not be readily described by commonly used copulas and is subject to overfitting even if someone can find a copula on the face that fits.

The first concern can somewhat be mitigated using simple methods. For example, when selecting pairs, use the Hurst exponent to filter all the non-stationary pairs; or update the training set often to keep up with the newest price. But the second concern cannot be fixed as long as we are not incorporating time-varying copula models, which is not under our current concern and is a serious topic that deserves its own discussion.



Instead, working with returns data can by large resolve the above issues: returns are in general stationary around 0, and they are much closer to the i.i.d. assumption imposed by copulas compared to prices. But in general, stocks are not traded directly on returns but prices, so how can we engineer a strategy that is based on returns?

STRATEGY 2: CUMULATIVE MISPRICING INDEX ON RETURNS

Working with returns is more common in literature for copula-based methods, and this is a building block for other more complicated multi-pairs trading strategies. See [Xie et al. 2014] [Stübinger et al. 2016] [Rad et al. 2016] [da Silva et al. 2017].

Concepts

To use returns to generate trading signals, one eventually needs to translate the information from overvalued/undervalued returns to mispricing.

Working with returns is more common in literature for copula-based methods, and this is a building block for other more complicated multi-pairs trading strategies. See [Xie et al. 2014] [Stübinger et al. 2016] [Rad et al. 2016] [da Silva et al. 2017].

Concepts

To use returns to generate trading signals, one eventually needs to translate the information from overvalued/ undervalued returns to mispricing.

Mispricing Index (MPI)

MPI is defined as the conditional probability of returns, i.e.,

$$MI_t^{X|Y} = P(R_t^X < r_t^X \mid R_t^Y = r_t^Y)$$

$$MI_t^{Y|X} = P(R_t^Y < r_t^Y \mid R_t^X = r_t^X)$$

for stocks (X, Y) with returns random variable at day t : (R_t^X, R_t^Y) and specific returns value at day t : (r_t^X, r_t^Y) The MPIs determine conditionally if the return on that day is over or under the average. Note that so far only one day's return information contributes, and naturally we want to cumulatively add a few days of MPIs up to gauge whether each stock is mispriced. This idea is not at all new, for example, when log-prices can be constructed from adding up log-returns.

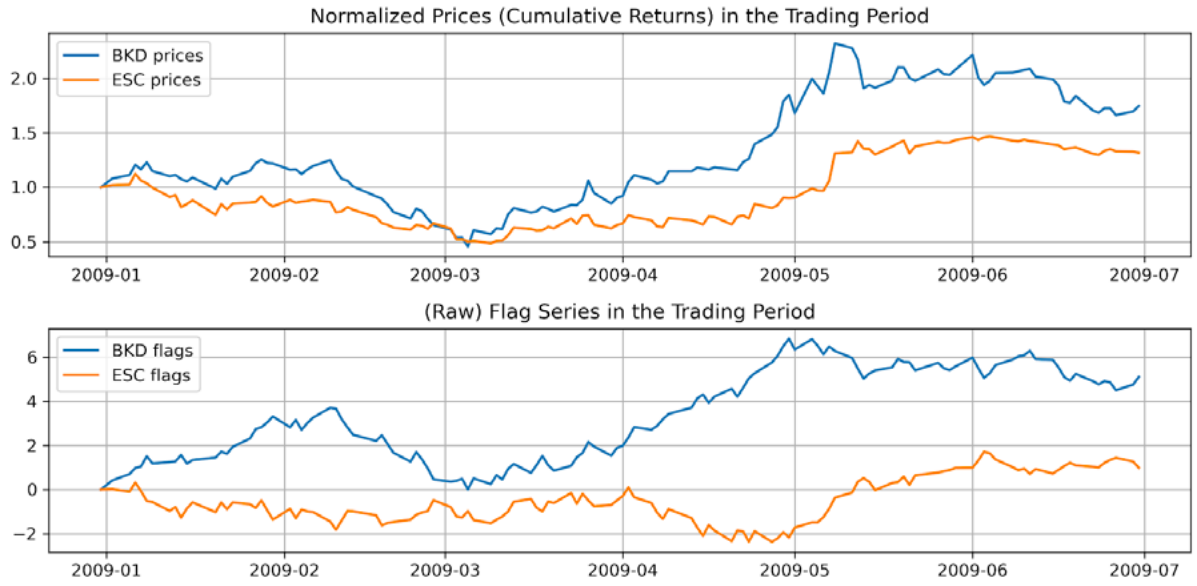
Cumulative Mispricing Index (CMPI) / Flags

Well, we are almost there. For ease of analysis, we also need to subtract the average 0.5 before adding up the daily MPIs, because they are probabilities. It is easier to think about a quantity is overvalued/undervalued if the indicator is greater/lesser than 0. Therefore we introduce the cumulative mispricing index (CMPI, in some literature and in our module it is also called Flags) series as below

$$FlagX(t) = \sum_{s=0}^{t} (MI_s^{X|Y} - 0.5)$$

$$FlagY(t) = \sum_{s=0}^{t} (MI_s^{Y|X} - 0.5)$$

If $FlagX > 0$ then stock X is overvalued; if $FlagX < 0$ then stock X is undervalued. Same for $FlagY$. We plot the raw flags series as below and they look quite similar to cumulative log-returns (just another way of saying log-prices!) from their price series, which is what they were designed to do: to reflect price information from returns.



Trading Logic

A strategy under the dollar-neutral scheme is worded as follows:

- **Opening rules:** (D=0.6 for example)
    - When $FlagX$ reaches D, short X and buy Y in equal amounts. (-1 Position)
    - When $FlagX$ reaches $-D$, short Y and buy X in equal amounts. (1 Position)
    - When $FlagY$ reaches D, short Y and buy X in equal amounts. (1 Position)
    - When $FlagY$ reaches $-D$, short X and buy Y in equal amounts. (-1 Position)
- **Exiting rules:** (S=2 for example)
    - If trades are opened based on $FlagX$, then they are closed if $FlagX$ returns to 0 or reaches the stop-loss position S or $-S$.
    - If trades are opened based on $FlagY$, then they are closed if $FlagY$ returns to 0 or reaches the stop-loss position S or -S.
- **After trades are closed, both $FlagX$ and $FlagY$ are reset to 0.**

For ambiguities we have the following:

1. When FlagX reaches D (or -D) and FlagY reaches D (or -D) together.

2. When in a long(or short) position, receives a short(or long) trigger.

3. When receiving an opening and exiting signal together.

4. When the position was open based on FlagX (or FlagY), FlagY (or FlagX) reaches S or -S.

The ambiguity cases occur much more often than the previous strategy and are a fundamental issue for this approach, since the mispricing info is generated from each stock. When assembling the information, conflicting signals should be regarded as normality, not outliers. We will discuss more issues in the comment section below.

COMMENTS

Logic

This strategy overall is way too sensitive. Using the end-of-day data, it is not uncommon to see 6 to 8 position changes in a month. This comes from how the signals are assembled. Looking at the strategy above it essentially uses an OR logic for opening: when stock X OR Y suggests an opening, then the strategy opens a position. For closing, it also uses an OR logic. Another variable is whether we reset the CMPIs to 0. In a nutshell:

1. Opening logic: AND, OR

2. Exiting logic: AND, OR

3. Whether reset: True, False

That is 8 possible combinations! Those choices are built-in for the copula module in ArbitrageLab. Just be aware of the logic that, tracking which stock opened a position only makes sense for the OR-OR open-close logic combination. For all the other three choices, it's sufficient to directly compare the CMPIs to thresholds. Not so surprisingly, to tune down the sensitivity you can change AND to OR.

Variation by Rad et al (2016)

Rad et al (2016) suggested using AND for opening, OR for exiting, and no reset. Then they did a thorough analysis by comparing this copula strategy variation with distance and cointegration methods. It is an excellent paper that sheds insight on the performance characteristics and I think all the people interested in copula methods should read it. Here are two major results:

- The performance of the copula approach is similar to the more traditional distance and cointegration methods given that the pairs do converge. Overall the copula approach has lower returns because non-convergent pairs drag the performance.

- The copula method has better performance during market downturns across all the pairs.

(Moreover, in our research, we found that this variation tends to capture the opening opportunities well, however does not exit at the right time sometimes, which brings down the performance.) We can see that the performance suffers mostly from non-convergent pairs. If the pairs are well-chosen then the copula method will enjoy a similar performance with distance and cointegration method while suffering much less from bear markets. This is no surprise since tail co-moves are baked-in for copula models. For the convergence issue, now let's look at the two CMPI series generated from the stocks pair:

First, the CMPIs are highly model-dependent. Even using copulas with similar fit scores will lead to possibly very different looking CMPIs and positions are taken based on CMPIs. Secondly, this type of strategy regardless of the logic combinations is betting on the mean-reversion of CMPIs, but they behave more like martingales. Moreover, the commonly used pairs selection methods such as Spearman's rho, or Euclidean distance do not reflect this key mean-reverting requirement for the CMPI series, also the copula's prices spread do not necessarily need to be mean-reverting for the copula strategy to be profitable.

QUICK PAIRS SELECTION

The commonly used methods are Euclidean distance ($L^2 - norm$ on normalized prices, though other P-norms are also reasonable, especially because stocks spread rarely stays at the same value for an extended time that could invalidate large P values if it ever happened.), Kendall's tau and Spearman's rho. The rank-based

methods offer some protection against extreme values natively, and they are more closely tied to how copulas are fitted.

In [Stübinger et al. 2016], other quick and seemingly promising approaches are proposed as well, such as geometric distance to the diagonal on Q-Q plot, and a χ^2 measure of independence (thus dependence).

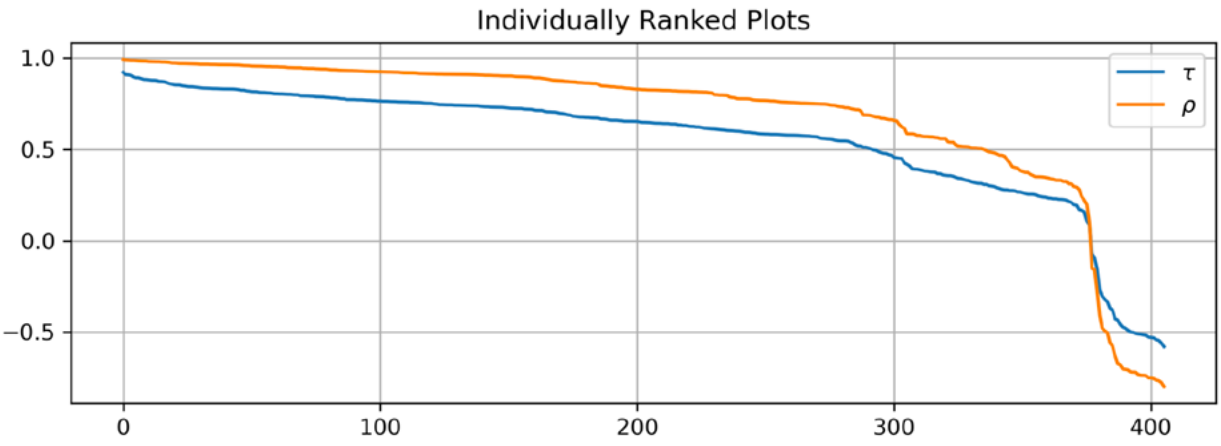
Comparisons Among the Methods

In general, Kendall's tau and Spearman's rho both use quantile data, will rank and select their pairs similarly, though Kendall's tau requires more computational time ($O(N^2)$ vs. $O(NLog(N))$) and is a tad more stable for outliers.

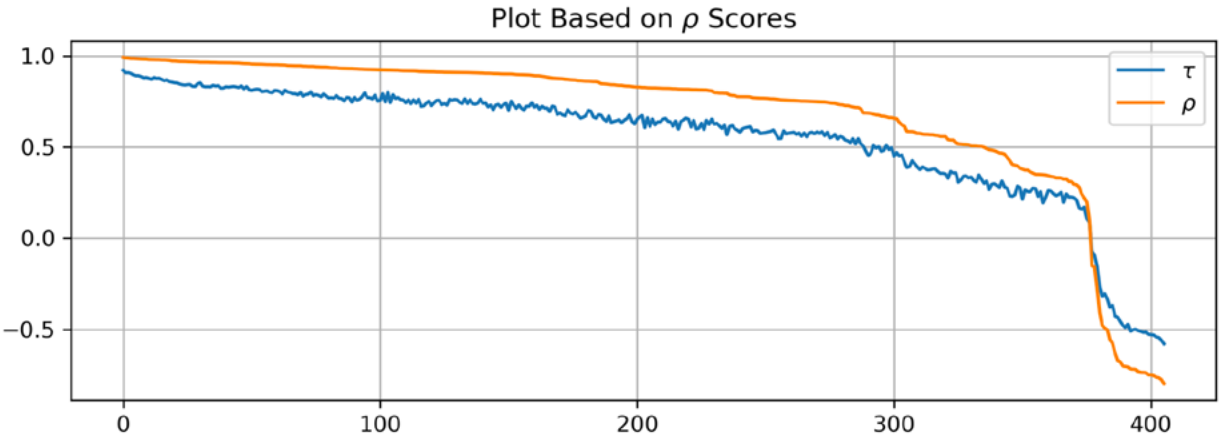
The higher the value of tau or rho, the more co-moves in their quantiles and vice versa. Also due to using percentile data, they are much more resilient to outliers, as compared to Euclidean distance. Euclidean distance will generally select very different pairs, in comparison. The smaller the Euclidean distance, the more close movements the pair of stocks have.

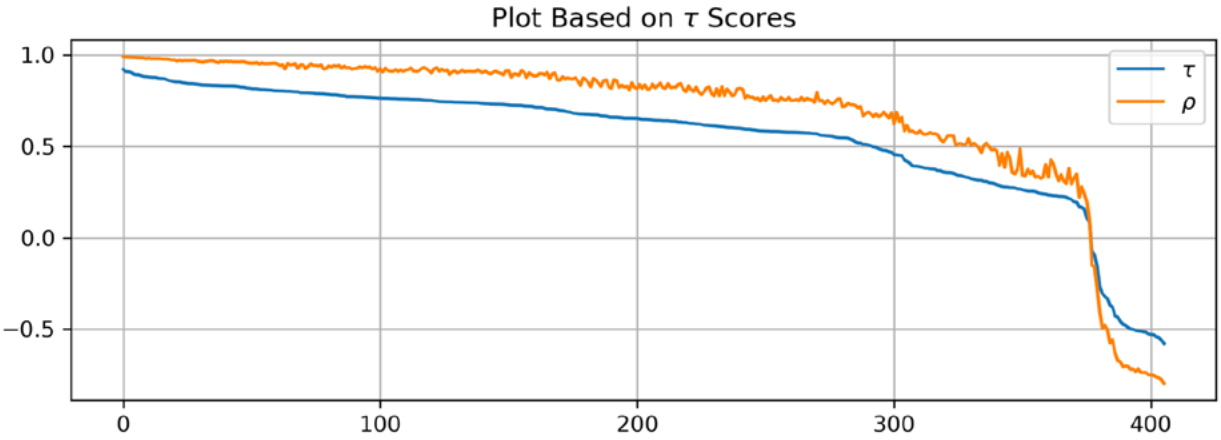
Here, I ran a mini-test for the three criteria using the 30 stocks in the Dow from 2011-2019 with adjusted closing prices for demonstration (Note: this is just for show of those methods. In reality it is a bad idea to feed 10 years of data to a static model.). The calculation is pretty fast, taking about 1 second using my office laptop, so it is readily scalable.

First, let us rank and plot Kendall's tau and Spearman's rho values individually. We can see that Kendall's tau tends to be smaller than Spearman's rho in absolute value in general.

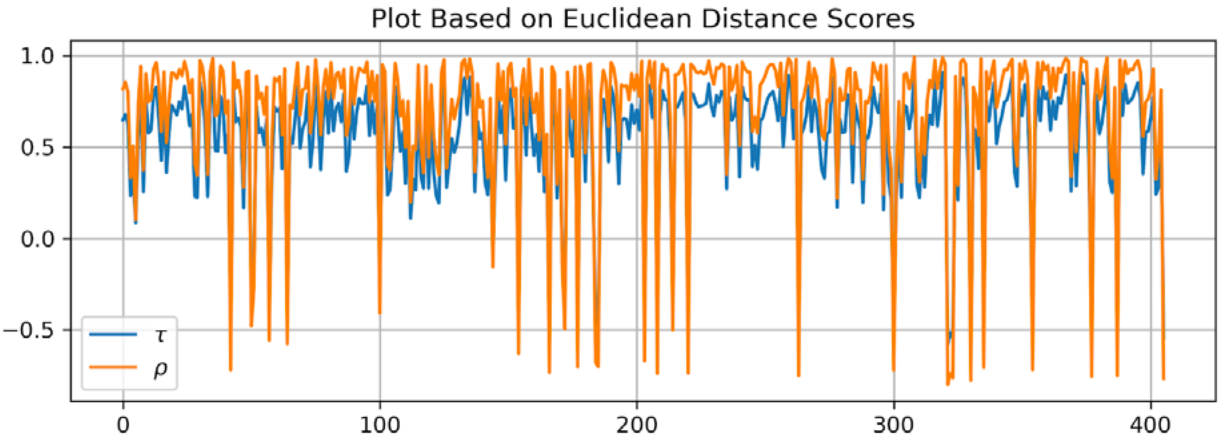


Then let's verify that they indeed select similar pairs. We do this by plotting Kendall's tau value using ranks from Spearman's rho and vice versa.

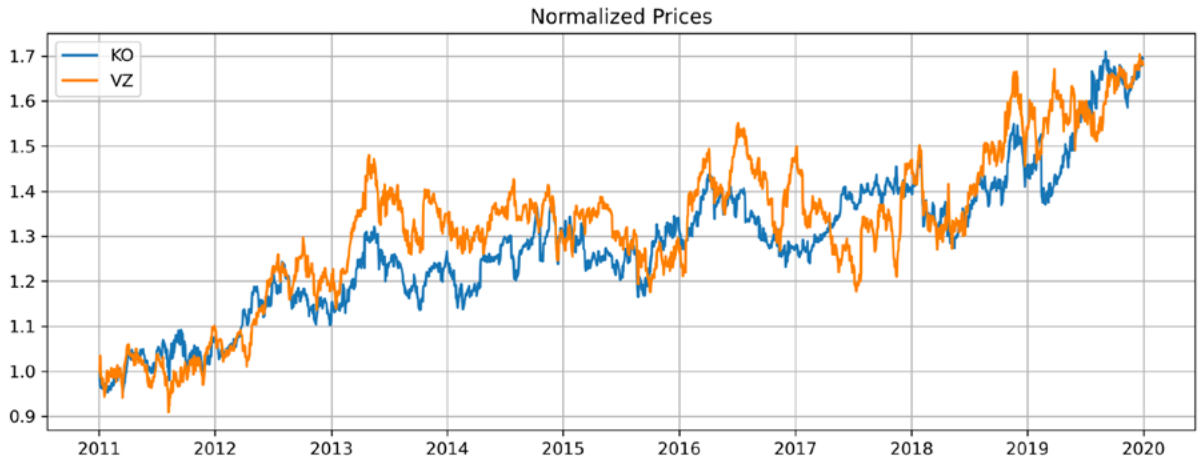


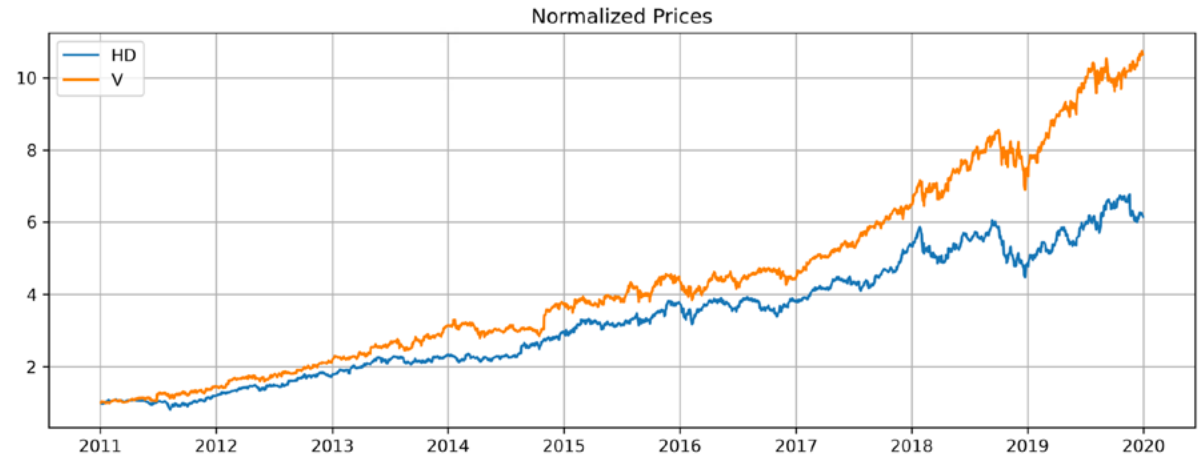


As you can see, Spearman's rho is slightly less stable than Kendall's tau, but otherwise, they generate comparable results. The Euclidean distance is very different, here is a plot of rho and tau values using ranks from the Euclidean distance.

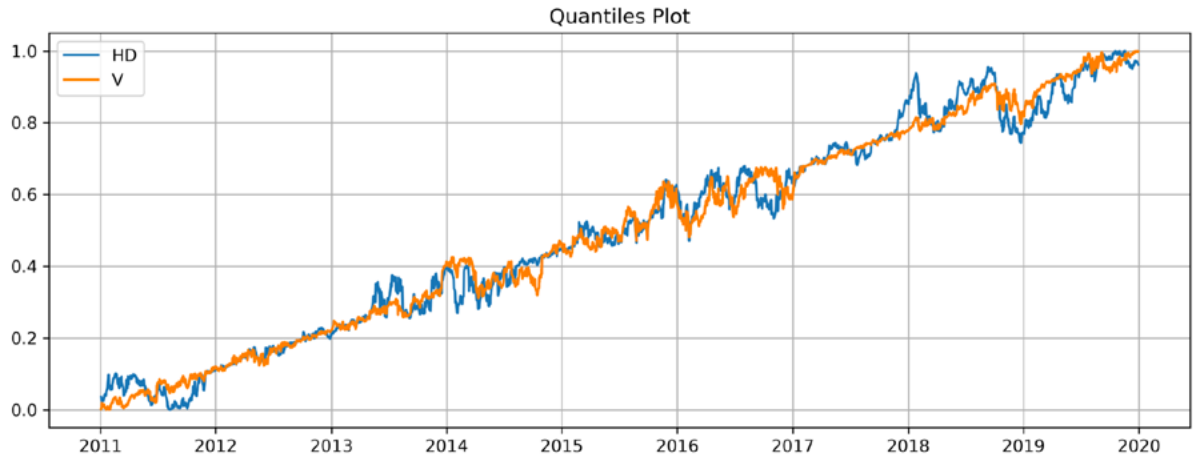


Here are some plots for prices: The KO-VZ pair is the top choice chosen by Euclidean distance, whereas the HD-V pair is chosen by Kendall's tau.





It might make more sense to look at this pair's quantile plot on prices instead:



Comments

Users should be aware that the three methods are not specifically made for copulas. When using strategy 2 the desirable property of convergent CMPIs is not captured explicitly by those methods. In [Stübinger et al. 2016], a χ^2 independence test criterion demonstrates the best performance for vine copula models, probably from accounting for extreme moves. We, therefore, encourage the reader to test this approach.

Our take overall is that copula methods should come with their own dedicated pairs selecting methods with dedicated goals to match implicit requirements posed by the copula methods. That χ^2 method in [Stübinger et al. 2016] may be closer to those goals on average, compared to the three methods defined above. We will dive deeper into this approach in an article for vine copula strategies. Indeed, copula is already a relatively novel approach and some practitioners treat it as a black box. The associated analysis for pairs selection is severely underdeveloped and needs to be taken seriously.

OTHER IDEAS

Here are a few ideas that are on my shortlist to try out. Some are from other literature that has not been applied to copula in general, some are just thoughts that are easy to implement for empirical analysis. All of the followings can be done with just a few lines of code since the CMPIs values can be directly returned in the module.

1. Use Bollinger Band for generating trading signals from each stock: when the CMPIs is greater than the band's upper bound, then generate a short signal, lesser for a long signal. Then assemble the signal via AND/OR.

2. Instead of comparing CMPIs with 0 for long/short/exit, compare the two CMPIs with each other by building a distance strategy based on it. It is natural to think about comparison of performance between the original distance strategy and a distance strategy based on CMPIs.

3. Run the CMPIs for pairs selection, and only choose those pairs that have converging CMPIs.

4. Mix the prices-based strategies and returns-based strategies, since the former is robust and the latter is sensitive.

5. Copulating with alternative data. This is easier when using higher-dimensional copula models for instance in vine copula.

CONCLUSION

Copula is no panacea for pairs trading. I hope you find it helpful for understanding these strategies after reading through this article. Essentially it is just a mathematical tool that is born to describe the dependence structure among univariate random variables. When applied to generating trading signals there are a lot of moving parts and room for improvising.

The sophistication of copula-based methods still seems somewhat lacking, given that most known methods are just variations of strategy 1 and 2, using simple conditional probabilities. There are also no dedicated copula-based strategies for trading when the tail events happen, given that copula models perform well with tail risks.

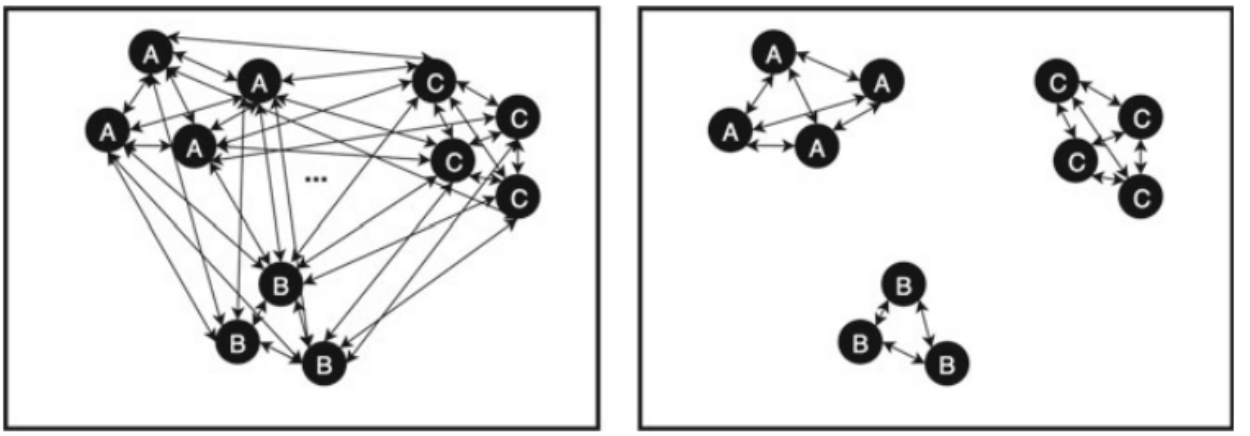
We expect further developments and popularity to emerge in this field in the foreseeable future, both in the aspect of strategy complexity (e.g., using more than just the conditional probability for signal generation) and model complexity (e.g., using vine copula for exploiting mispricing in a cohort of stocks).

REFERENCES

1. B Sabino da Silva, F., Ziegelman, F. and Caldeira, J., 2017. Mixed Copula Pairs Trading Strategy on the S&P 500. Flávio and Caldeira, João, Mixed Copula Pairs Trading Strategy on the S&P, 500.
2. Liew, R.Q. and Wu, Y., 2013. Pairs trading: A copula approach. Journal of Derivatives & Hedge Funds, 19(1), pp.12-30.
3. Patton, A.J., 2012. A review of copula models for economic time series. Journal of Multivariate Analysis, 110, pp.4-18.
4. Patton, A.J., 2009. Copula–based models for financial time series. In Handbook of financial time series (pp. 767-785). Springer, Berlin, Heidelberg.
5. Rad, H., Low, R.K.Y. and Faff, R., 2016. The profitability of pairs trading strategies: distance, cointegration and copula methods. Quantitative Finance, 16(10), pp.1541-1558.
6. Stander, Y., Marais, D. and Botha, I., 2013. Trading strategies with copulas. Journal of Economic and Financial Sciences, 6(1), pp.83-107.
7. Stübinger, J., Mangold, B. and Krauss, C., 2018. Statistical arbitrage with vine copulas. Quantitative Finance, 18(11), pp.1831-1849.
8. Vidyamurthy, G., 2004. Pairs Trading: quantitative methods and analysis (Vol. 217). John Wiley & Sons.
9. Xie, W., Liew, R.Q., Wu, Y. and Zou, X., 2016. Pairs trading with copulas. The Journal of Trading, 11(3), pp.41-52.

9.0

EMPLOYING MACHINE LEARNING FOR PAIRS SELECTION



(a) Searching for pairs without restrictions. **(b)** Searching for pairs within the same category.

Pairs Search Common Techniques (Sarmento and Horta 2020).

In this post, we will investigate and showcase a machine learning selection framework that will aid traders in finding mean-reverting opportunities. This framework is based on the book: **"A Machine Learning based Pairs Trading Investment Strategy"** by Sarmento and Horta.

A time series is known to exhibit mean reversion when, over a certain period, it reverts to a constant mean. A topic of increasing interest involves the investigation of long-run properties of stock prices, with particular attention being paid to investigate whether stock prices can be characterized as random walks or mean-reverting processes.

If a price time series is a random walk, then any shock is permanent and in the long run, the volatility of the process will continue to grow without bound. On the other hand, if a time series of stock prices follow a mean-reverting process, there is a tendency for the price level to return to a constant mean over time. Thus it is seen as a very attractive opportunity for investors because of its more reliably forecastable future returns.

PREVIOUSLY PROPOSED APPROACHES

Pairs trading methods differ in how they form pairs, the trading rules used, or both. The dominating non-parametric method was published in the landmark paper of Gatev et al: "Pairs trading: Performance of a relative-value arbitrage rule", where they propose using a distance-based approach to select the securities for constructing the pairs. The criterion used was choosing pairs that minimized the sum of squared deviations

between the two normalized prices. They argue that this approach "best approximates the description of how traders themselves choose pairs".

Overall, the nonparametric distance-based approach provides a simple and general method of selecting "good" pairs, however, this selection metric is prone to pick up pairs with a small amount of variance of the spread and therefore limits overall profitability.

Another two popular approaches are the cointegration and correlation methods. Basically, correlation reflects short-term linear dependence in returns while on the other hand cointegration models long-term dependencies in prices (Alexander 2001).

A significant issue with the correlation approach is that two securities correlated on a returns level do not necessarily share an equilibrium relationship, and there is no theoretical foundation that divergences are reversed. This potential lack of an equilibrium relationship leads to higher divergence risks and thus to significant potential losses.

High correlations may well be spurious since high correlation is not related to a cointegration relationship (Alexander 2001). Omitting cointegration testing is very much contradictory to the requirements of a rational investor.

THE PROBLEM OF MULTIPLE HYPOTHESIS TESTING

As the popularity of Pairs Trading grows, it is increasingly harder to find rewarding pairs. The simplest procedure commonly applied is to generate all possible candidate pairs by considering the combination from every security to every other security in the dataset.

Two different problems arise immediately.

First, the computational cost of testing mean-reversion for all the possible combinations increases drastically as more securities are considered.

The second emerging problem is frequent when performing multiple hypothesis tests at once and is referred to as the multiple comparisons problem. By definition, if 100 hypothesis tests are performed (with a confidence level of 5%) the results should contain a false positive rate of 5%.

This problem was tackled by (Harlacher 2016), who found that Bonferroni correction leads to a very conservative selection of pairs and impedes the discovery of even truly cointegrated combinations. The paper recommends the effective pre-partitioning of the considered asset universe to reduce the number of feasible combinations and, therefore, the number of statistical tests. This aspect might lead the investor to pursue the usual, more

restrictive approach of comparing securities only within the same sector.

This dramatically reduces the number of necessary statistical tests, consequently decreasing the likelihood of finding spurious relations. However, the simplicity of this process might also turn out to be a disadvantage. The more traders are aware of the pairs, the harder it is to find pairs not yet being traded in large volumes, leaving a smaller margin for profit.

This dilemma motivated the work of (Sarmento and Horta 2020) in the search for a methodology that lies in between these two scenes: an effective pre-partitioning of the universe of assets that does not limit the combination of pairs to relatively obvious solutions, while not enforcing excessive search combinations.

HOW CAN MACHINE LEARNING SOLVE THIS?

In this work (Sarmento and Horta 2020), propose a three-pronged process, built using unsupervised learning algorithms, on the expectation that it infers meaningful clusters of assets from which to select the pairs.

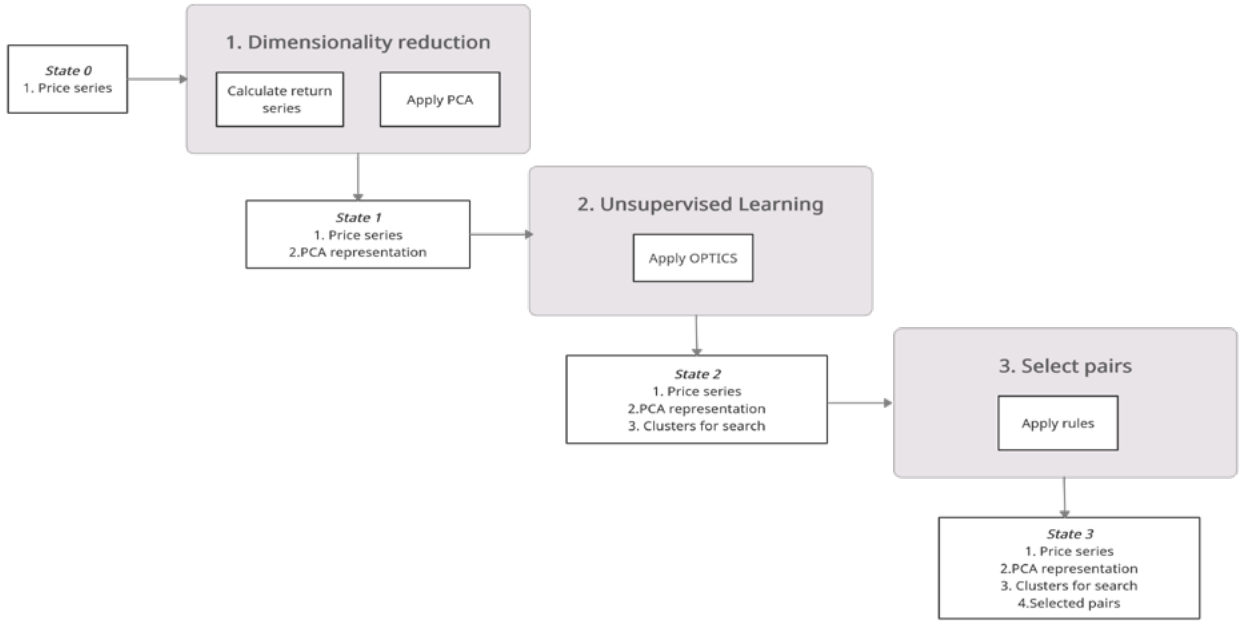


Figure 1: Proposed Framework diagram from (Sarmento and Horta 2020).

The premise is to extract empirical structure that the data is manifesting, rather than imposing fundamental groups each security should belong to. The proposed methodology encompasses the following steps:

1. Dimensionality reduction – find a compact representation for each security;

2. Unsupervised Learning – apply an appropriate clustering algorithm;

3. Select pairs – define a set of rules (ARODS) to select pairs for trading.

DIMENSIONALITY REDUCTION

Dimensionality reduction is the transformation of high-dimensional data into a meaningful representation of reduced dimensionality. It is important in many domains since it mitigates the curse of dimensionality and other undesired properties of high-dimensional spaces. Ideally, the reduced representation should have a dimensionality that corresponds to the intrinsic dimensionality of the data. The intrinsic dimensionality of data is the minimum number of parameters needed to account for the observed properties of the data.

In the quest for profitable pairs, we wish to find securities with the same systematic risk-exposure. Any deviations from the theoretical expected return can thus be seen as mispricing and serve as a guideline to place trades. The application of PCA on the return series is proposed to extract the common underlying risk factors for each security, as described in (Avellaneda and Lee 2010).

PCA is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of linearly uncorrelated variables, the principal components. The transformation is defined such that the first principal component accounts for as much of the variability in the data as possible. Each succeeding component, in turn, has the highest variance possible under the constraint that it must be orthogonal to the preceding components.

CLUSTERING

There exist several unsupervised learning clustering techniques capable of solving the task at hand. Two main candidates are showcased: DBSCAN and OPTICS.

DBSCAN, as the name implies, is a density-based clustering algorithm, which given a set of points in some space, it groups together points that are closely packed together.

OPTICS is an extension built upon the idea of DBSCAN, but improves on its major weakness: the problem of

detecting meaningful clusters in data of varying density. To do so, the points of the database are ordered such that spatially closest points become neighbors in the ordering. Additionally, a special distance is stored for each point that represents the density that must be accepted for a cluster so that both points belong to the same cluster.

In summary, there are a few differences:

- Memory Cost: OPTICS requires more memory as it maintains a priority queue to determine the next data point which is closest to the point currently being processed in terms of reachability distance. It also requires more computational power because the nearest neighbor queries are more complicated than radius queries in DBSCAN.
- Fewer Parameters: OPTICS does not need to maintain the epsilon parameter making it relatively insensitive to parameter settings. Thus also leading to the reduction of manual parameter tuning.
- Interpretable: OPTICS does not segregate the given data into clusters. It merely produces a distance plot and it is upon the interpretation of the programmer to cluster the points accordingly.

The framework described till now has been implemented as a part of the ArbitrageLab library that has been in development for the last few months. The steps above have been condensed down into a simple manner as follows;

```
# Importing packages
import pandas as pd
import numpy as np
from arbitragelab.ml_approach import PairsSelector


# Getting the dataframe with time series of asset returns.
data = pd.read_csv('X_FILE_PATH.csv', index_col=0, parse_dates = [0])
ps = PairsSelector(data)
```

```
# Price data is reduced to its component features using PCA
ps.dimensionality_reduction_by_components(5)


# Clustering is performed over the feature vector.
ps.cluster_using_optics({'min_samples': 3})


# Plot clustering results.
ps.plot_clustering_info(method='OPTICS', n_dimensions=3)
```

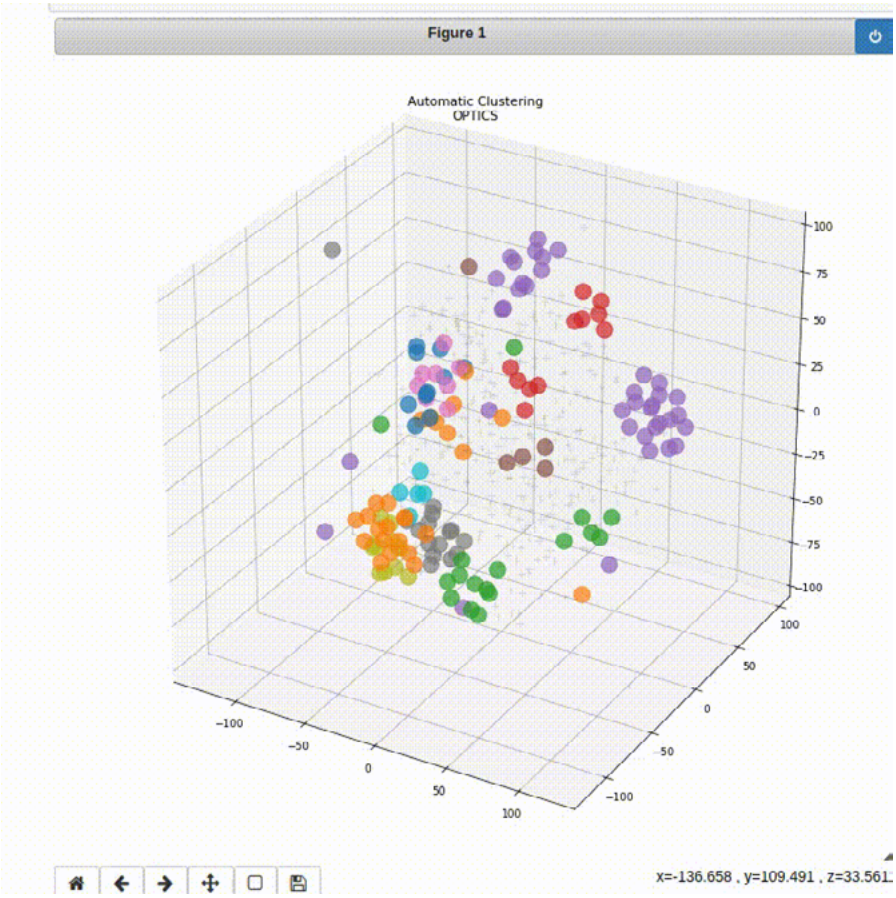


Figure 2: 3D plot of the clustering result using the OPTICS method.

ABSOLUTE RULES OF DISQUALIFICATION (ARODS)

The classical pairs selection approach encompasses two steps:

- Finding the appropriate candidate pairs and
- Selecting the most promising ones.

Having generated the clusters of assets in the previous steps, it is still necessary to define a set of conditions for selecting the pairs to trade. The most common approaches to select pairs are the distance, cointegration, and correlation approaches. The cointegration approach was selected because the literature suggests that it performs better than minimum distance and correlation approaches.

The selection process starts with the testing of pairs, generated from the clustering step, for cointegration using the Engle-Granger test. This cointegration testing method was chosen based on its simplicity. If you want to learn more about cointegration methods I recommend reading our article: **An Introduction to Cointegration for Pairs Trading.**

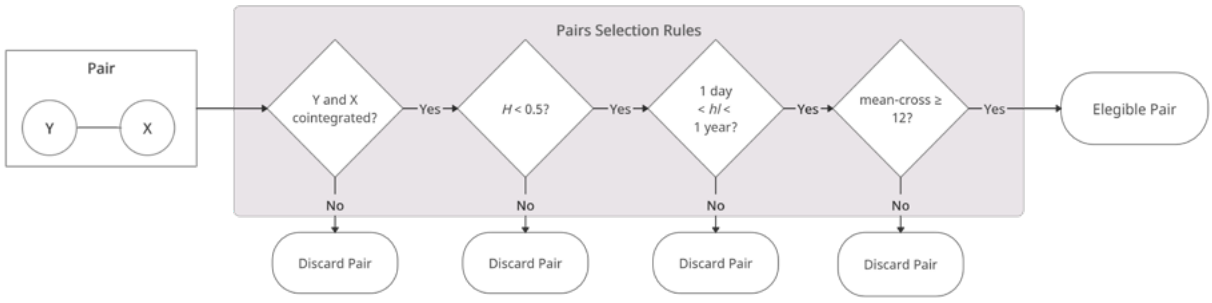


Figure 3: Pairs Selection Rules diagram from (Sarmento and Horta 2020).

Secondly, a validation step is implemented to provide more confidence in the mean-reversion character of the pairs' spread. The condition imposed is that the Hurst exponent associated with the spread of a given pair is enforced to be smaller than 0.5, assuring the process leans towards mean-reversion. The Hurst exponent is a measure that quantifies the amount of memory in a specific time series, which in this case means, how mean-reverting is the process under analysis?

In third place, the pair's spread movement is constrained using the half-life of the mean-reverting process. The

half-life is the time that the spread will take to mean-revert half of its distance after having diverged from the mean of the spread, given a historical window of data. In the framework paper, the strategy built on top of the selection framework is based on the medium-term price movements, so for this reason the spreads that either have very short (< 1 day) or very long mean-reversion (> 365 days) periods are not suitable.

Lastly, every spread is checked to make sure it crosses its mean at least twelve times a year, to provide enough liquidity and thus providing enough opportunities to exit a position.

This step can be easily done writing the following lines of code;

```
# Generated Pairs are processed through the rules mentioned above

ps.unsupervised_candidate_pair_selector()


# Generate a Panel of information of the selected pairs

final_pairs_info = ps.describe_extra()
```

	leg_1	leg_2	coint_t	pvalue	hedge_ratio	hurst_exponent	half_life	crossovers
0	AJG	ICE	-5.147984	0.000011	0.832406	0.280300	1.178147	True
1	AJG	MMC	-3.984768	0.001492	0.892408	0.333155	1.325405	True
2	ICE	MMC	-4.314326	0.000420	1.072514	0.353929	3.312397	True
3	MMC	WLTW	-4.173535	0.000730	1.758648	0.330393	2.353220	True
4	CHTR	TMUS	-3.834987	0.002569	0.166618	0.350558	2.954012	False
5	EW	FISV	-3.775258	0.003171	1.247778	0.342441	1.774284	True
6	EW	GPN	-4.614576	0.000121	2.381694	0.388706	10.865253	False
7	FISV	GPN	-3.788320	0.003029	1.890617	0.366615	1.021983	False
8	IQV	V	-3.432286	0.009906	1.353333	0.331590	1.975149	True
9	NVR	PHM	-3.553403	0.006717	0.009508	0.389903	13.683387	True
10	DE	NWSA	-3.436995	0.009761	0.014579	0.451436	5.102561	False

CONCLUSION

The success of a Pairs Trading strategy highly depends on finding the right pairs. The classical approaches to find tradable opportunities used by investors are either too restrictive, like restricting the search to inter-sector relationships, or too open, like removing any limit on the search space.

In this post, a Machine Learning based framework is applied to provide a balanced solution to this problem. Firstly with the application of unsupervised learning to define the search space. Secondly, in the grouping of relevant securities (not necessarily from the same sector) in clusters, and finally in the detection of rewarding pairs within them, that would otherwise be hard to identify, even for the experienced investor.

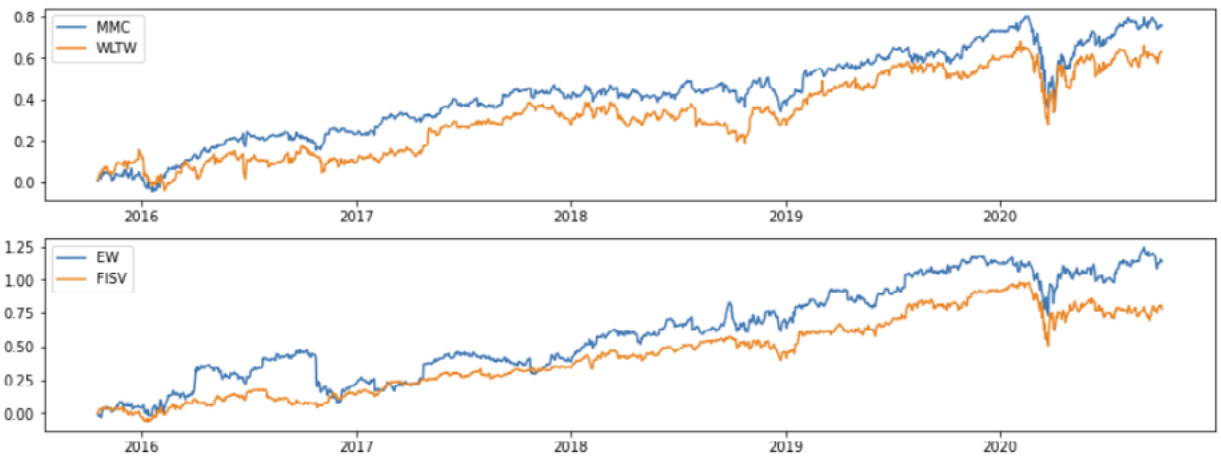


Figure 4: Example pairs found using the ArbitrageLab framework implementation.

REFERENCES

1. Ankerst, M., Breunig, M.M., Kriegel, H.P., and Sander, J., 1999. OPTICS: ordering points to identify the clustering structure. ACM Sigmod Record, 28 (2), pp.49-60.
2. Avellaneda, M. and Lee, J.H., 2010. Statistical arbitrage in the US equities market. Quantitative Finance, 10 (7), pp.761-782.
3. Alexander, C., 2001. Market models. A Guide to Financial Data Analysis, 1.
4. Gatev, E., Goetzmann, W.N. and Rouwenhorst, K.G., 2006. Pairs trading: Performance of a relative-value arbitrage rule. The Review of Financial Studies, 19(3), pp.797-827.
5. Harlacher, M., 2016. Cointegration based algorithmic pairs trading. PhD thesis, University of St. Gallen.
6. Maddala, G.S., and Kim, I.M., 1998. Unit roots, cointegration, and structural change (No. 4). Cambridge university press.
7. Ramos-Requena, J.P., Trinidad-Segovia, J.E. and Sánchez-Granero, M.Á., 2020. Some Notes on the Formation of a Pair in Pairs Trading. Mathematics, 8 (3), p.348.
8. Sarmento, S.M. and Horta, N., 2020. Enhancing a Pairs Trading strategy with the application of Machine Learning. Expert Systems with Applications, p.113490.
9. Van Der Maaten, L., Postma, E., and Van den Herik, J., 2009. Dimensionality reduction: a comparative. J Mach Learn Res, 10 (66-71), p.13.

10.0

THE CORRECT VECTORIZED BACKTEST METHODOLOGY FOR PAIRS TRADING

VECTORIZED BACKTESTING AND PAIRS TRADING

Whilst backtesting architectures is a topic on its own, this article dives into how to correctly backtest a pairs trading investment strategy using a vectorized (quick methodology) rather than the more robust event-driven architecture. This is a technique that is very common amongst analysts and is rather straightforward for long-only portfolios, however, when you start to construct long-short portfolios based on statistical arbitrage, strange little nuances start to pop up.

Please note that we are strongly advocating against research through backtesting and encourage you to turn to tools such as feature importance, to guide the process. A backtest is the very last sanity check and one needs to be very careful of generating false investment strategies.

Let's dive in!

WHAT IS AN EQUITY CURVE

Forming an equity curve is usually regarded as a sanity check on strategies to have a glance over their performance. It can be thought of as a quick backtest, usually composed of a few lines of code, focused on the strategy itself without worrying too much about VaR, transaction costs, market impact, slippage, frictions, etc., which can be handled by various full-on backtest platforms.

"Oh, it is just calculating the excess returns over some positions. What is the big deal?"

Ideally, it is simple. However, it is like one of those "trivial and left as an exercise to the reader" marks in an abstract algebra textbook. Maybe it is, for this reason, there is little discussion on it when it becomes non-trivial.

except the (j)-term, and thus we obtain

$$0 = a_{(j)} v_{j_1} \wedge \cdots \wedge v_{j_r} \wedge \cdots \wedge v_{j_n}.$$

Reshuffling $v_{j_1} \wedge \cdots \wedge v_{j_n}$ into $v_1 \wedge \cdots \wedge v_n$ simply changes the right-hand side by a sign. From what we proved at the beginning of this proof, it follows that $a_{(j)} = 0$. Hence we have proved our assertion for $1 \leqq r \leqq n$.

When $r = 0$, we deal with the empty product, and 1 is a basis for $\bigwedge^0(E) = R$ over R. We leave the case $r > n$ as a trivial exercise to the reader.

The assertion concerning the dimension is trivial, considering that there is a

Serge Lang, Algebra.

Here is one interesting problem to ponder: How to calculate returns of an arbitrage opportunity? Well, if you can make chunks of gold out of thin air, infinitely, with no cost and no risk, this kind of arbitrage has infinite returns. But, say that you are running a statistical arbitrage: pairs trade over two stocks X and Y and you find a lucrative algorithm adapted from some literature. The algorithm suggests that you should long the spread of X and Y (i.e., long X and short Y with correct hedge ratio) when the spread is $0, and exit when the spread is $1. You followed the algorithm, and to make this discussion simple, say you longed X and shorted Y by exactly 1 unit. In the end, you made 1 dollar, now the question is, what is the return? (Hint: infinity is not a correct answer.)

Through this article I aim to cover the following topics:

1. **Calculate the equity curve for a single stock.**
2. **Calculate the equity curve for two stocks, including long-only and long-short.**
3. **Interpretation of returns for a dollar-neutral strategy.**
4. **Common fallacies in the equity curve calculation.**

NON-TRIVIAL ISSUES

The major issue comes in if the assets that we calculate the equity curve upon, are not all the same. It is not difficult to calculate equity curves for a single stock, however, things change when the underlying becomes a portfolio, especially when long and short positions for each component are allowed. For pairs trading, a long-short portfolio (spread) is in a sense self-financing: You can (depending on your broker) cover the long position cost (at least partially) from the short position. And when this happens, the usual definition of returns stops working since the cost of investment becomes unclear, and it may appear one just made gold out of thin air and returns become infinity.

Another issue is about calculating the value of a portfolio. Just like the value of common stock, one wishes to see how it evolves, and visually see where to take long and short positions to capture opportunities. We will demonstrate later that this may not work for some types of strategies.

VALID METHODS FOR FORMING AN EQUITY CURVE

Here we provide a few frameworks for calculating equity curves with valid logic. A very important note here is that they work with different objects. We will discuss the common mistakes that arise from misuse.

Returns: Single Stock

This is rather straightforward, but note that stock prices cannot go negative. In this case, we are essentially capturing the daily returns, and thus forming an equity curve on returns.

1. Get the stock price S.
2. Get stock's daily returns series.

$$r(t) = \frac{S(t)}{S(t-1)} - 1$$

3. Get the positions $P(t)$ from a strategy, for the position held on that day (Be very careful not to introduce look-ahead bias in this step!).
4. Calculate the daily returns $r_s(t)$ from our strategy. It is the pointwise multiplication

$$r_s(t) = r(t)P(t), \text{ for each } t$$

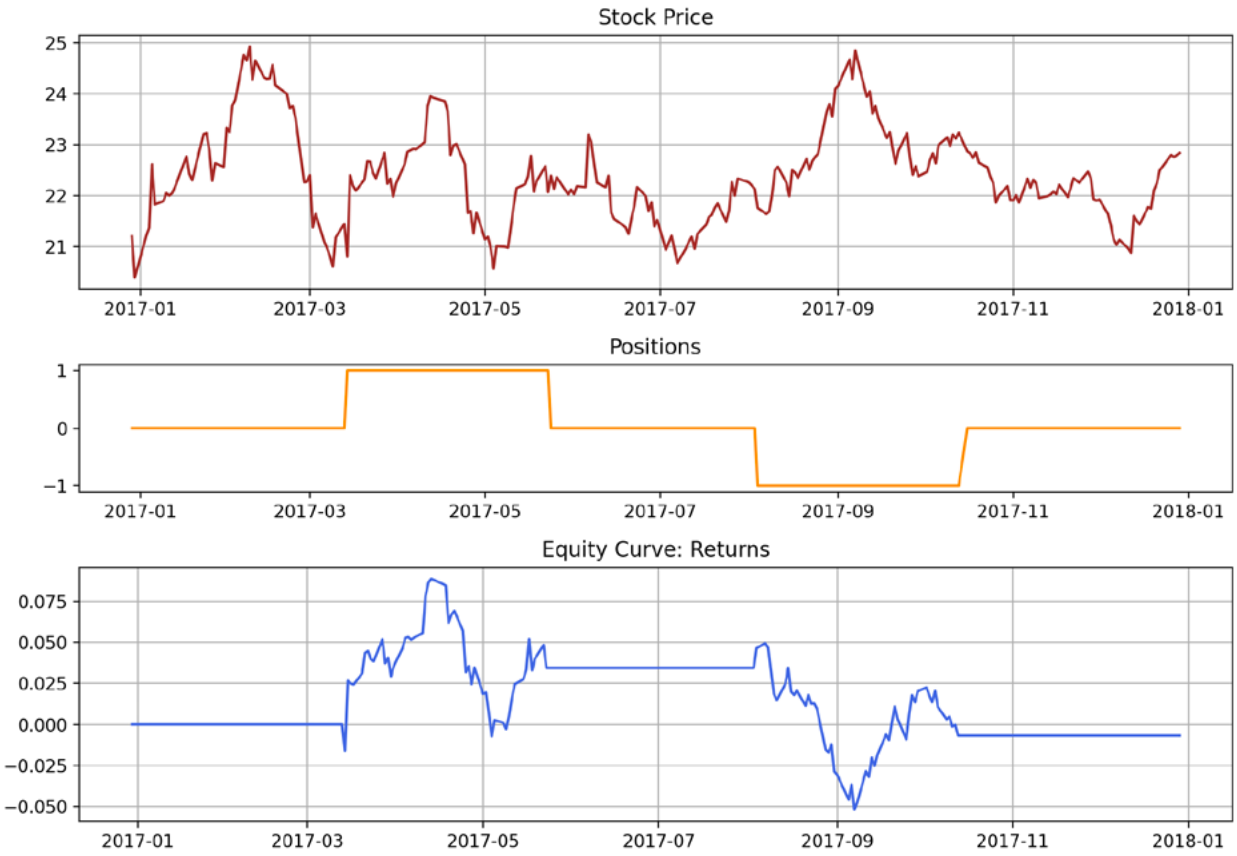
5. Then we use daily returns $r_s(t)$ to cumulatively reconstruct our portfolio's equity curve on returns:

$$\mathcal{E}(t) = (\prod_{\tau=0}^{t}[r_s(\tau) + 1]) - 1$$

Note: *We assume the positions series P(t) is a time series of (bounded) real numbers, indicating how many units of the portfolio to hold at time t, suggested by some strategy. For some strategies, it only takes values from \{0, 1, -1\}, representing no position, 100% long and 100% short respectively.*

```
# 1. Portfolio time series

prices = testing_data['GDX']


# 2. Portfolio's returns

rts = prices.pct_change()
```

```
# 3. Suggested positions by some strategy

# I am just making the positions up for ease of demonstration.

position_data = [0]*50 + [1]*50 + [0]*50 + [-1]*50 + [0]*53

positions = pd.Series(data=position_data, index=prices.index)


# 4. Returns from our strategy

rts_strat = rts * positions


# 5. (Normalized) P&L

equity_normalized = (1 + rts_strat).cumprod() - 1
```



Forming a Portfolio from a Pair of Stocks

To keep things simple for now, suppose we have 2 stocks in the portfolio, with non-negative value time series S_A, S_B. The portfolio's value time series is defined as

$$\Pi(t) = w_A(t)S_A(t) + w_A(t)S_B(t),$$

where w_A and w_B are units held for S_A and S_B, and they can be any real number: They do not necessarily sum to 1 and they do not need to be positive.

For a long-only portfolio, we assume the units held are non-negative. Moreover, it is common to normalize by letting the units be a partition of $[0, 1]$ if they are constants, and they effectively become weights, i.e.,

$$w_A + w_B = 1; \quad w_A, w_B \geq 0.$$

For a long-short portfolio, hedge ratio $h(t)$ is generally used for the number of units held. Here we use it as:

$$w_A = 1, \quad w_B(t) = -h(t).$$

Generally, h should be a function of time, but occasionally it is constant during the trading period (for instance, one calculates it from OLS on training data for benchmarking). The long-short portfolio formed from the hedge ratio thus has a value time series as:

$$\Pi(t) = S_A(t) - h(t)S_B(t),$$

Returns: Long-Only Portfolios

Suppose we have positive units held $w_A(t)$, $w_B(t)$ on two strictly positive value time series (stocks, funds, etc.). The method applied to a single stock still works.

1.Construct the portfolio (price series) Π.
$\Pi(t) = w_A(t)S_A(t) + w_B(t)S_B(t)$

2. Get the portfolio's daily returns series.
$$r(t) = \frac{\Pi(t)}{\Pi(t-1)} - 1$$

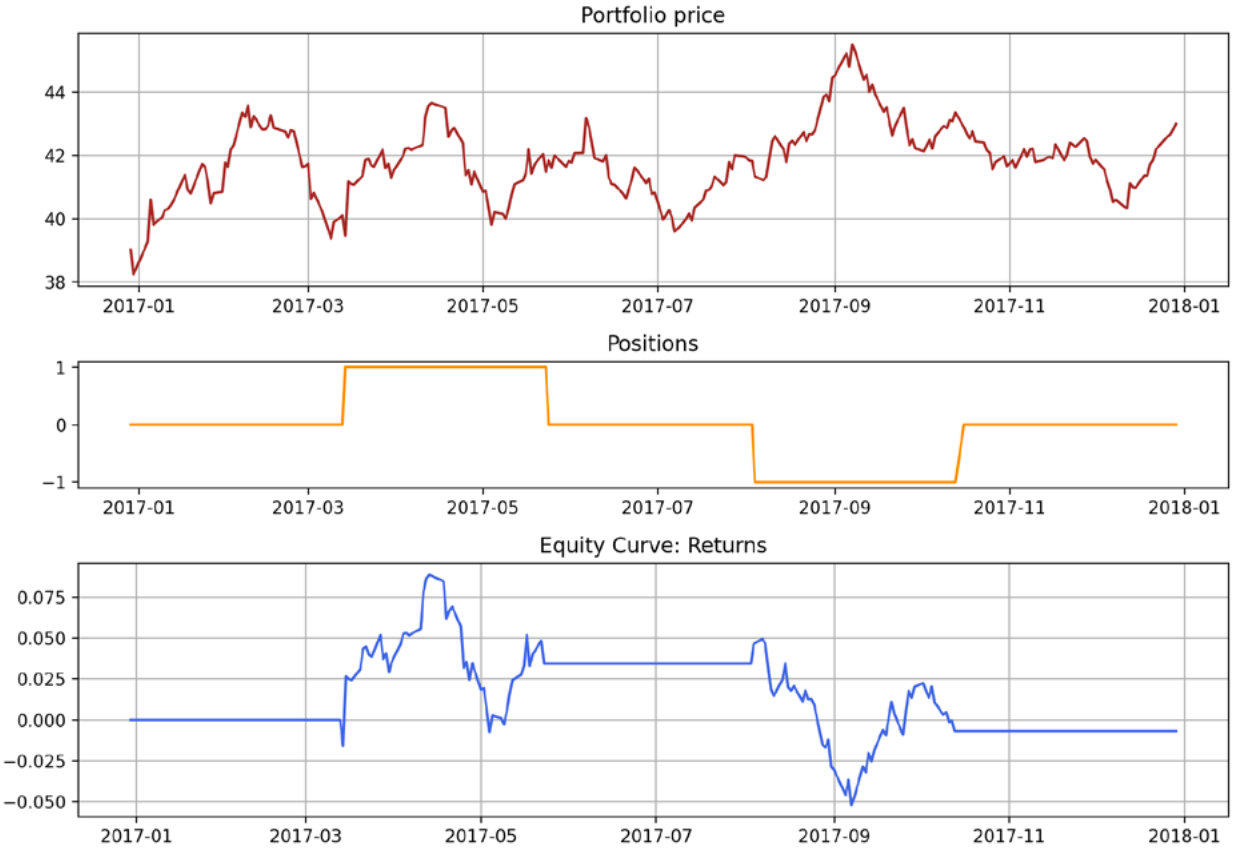
3.Get the positions $P(t)$ from a strategy, held on that day.

4. Calculate the daily returns $r_s(t)$ from our strategy. It is the pointwise multiplication
$r_s(t) = r(t)P(t), \text{ for each } t$

5. Then we use daily returns $r_s(t)$ to reconstruct our portfolio's equity curve on returns:

$\mathcal{E}(t) = (\prod_{\tau=0}^{t}[r_s(\tau) + 1]) - 1$

```
# 1. Portfolio time series
# Weights are arbitrarily chosen as a partition of [0, 1] for demonstration
w1 = 0.8  # Positive weight
w2 = 0.2  # Positive weight
portfolio = w1 * testing_data['GDX'] + w2 * testing_data['GDX']


# 2. Portfolio's returns
rts = portfolio.pct_change()


# 3. Suggested positions by some strategy
# I am just making the positions up for ease of demonstration.
position_data = [0]*50 + [1]*50 + [0]*50 + [-1]*50 + [0]*53
positions = pd.Series(data=position_data, index=portfolio.index)


# 4. Returns from our strategy
rts_strat = rts * positions


# 5. (Normalized) P&L
equity_normalized = (1 + rts_strat).cumprod() - 1
```



Daily Profit and Loss (P&L): Long-Short Portfolios and Others

Daily profit and loss (P&L) is the most general approach and it should work for all portfolios. Specifically for a long-short pair's trading portfolio, where the portfolio value time series can be positive, negative, or switching around 0, this method will provide the correct information. We provide the calculation procedure for a long-short stocks pair portfolio:

1. Construct the portfolio (price series) Π with some hedge ratio.
   $\Pi(t) = S_A(t) - h(t)S_B(t)$

2. Get the portfolio's daily revenue (price difference, daily P&L for one unit) series.
   $R(t) = \Pi(t) - \Pi(t-1)$

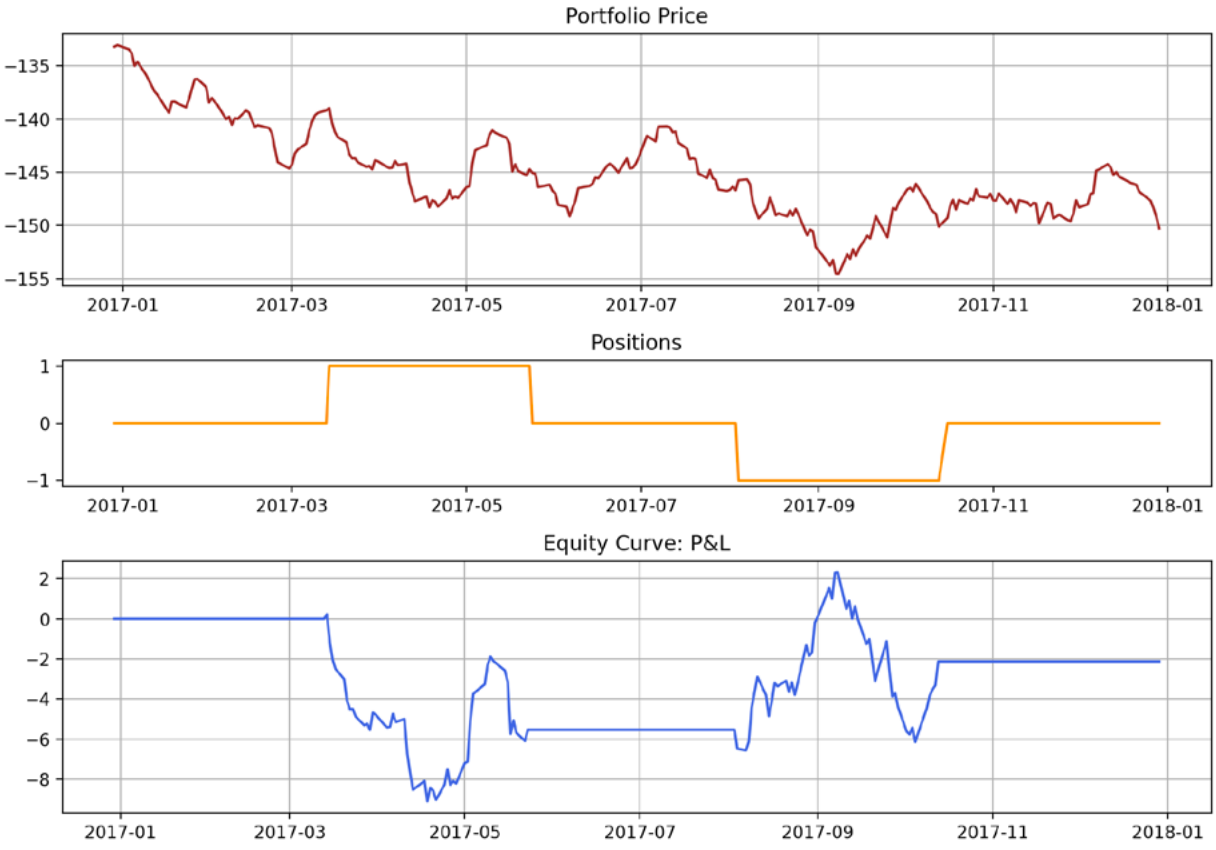
3. Get the positions $P(t)$ from a strategy.

4. Calculate the daily returns $Rs(t)$ from our strategy. It is the pointwise multiplication
   $R_s(t) = R(t)P(t), \text{ for each } t$

5. Then we use daily P&L $R_s(t)$ to reconstruct our portfolio's equity curve on P&L:
   $\mathcal{E}(t) = \sum_{\tau=0}^{t} R_s(\tau)$

This can be interpreted as the (cumulative) P&L for holding a single unit of the portfolio, with given positions suggested by the strategy.

```
h_ols = 1.4  # Hedge ratio suggested by OLS on training data

# 1. Portfolio time series

portfolio = testing_data['GDX'] - h_ols * testing_data['GDX']


# 2. Portfolio's daily P&L for holding 1 unit

pnl = portfolio.diff()


# 3. Suggested positions by some strategy

# I am just making the positions up for ease of demonstration.

position_data = [0]*50 + [1]*50 + [0]*50 + [-1]*50 + [0]*53

positions = pd.Series(data=position_data, index=portfolio.index)


# 4. Daily P&L from our strategy

pnl_strat = pnl * positions


# 5. Equity curve

equity_pnl = (pnl_strat).cumsum()
```

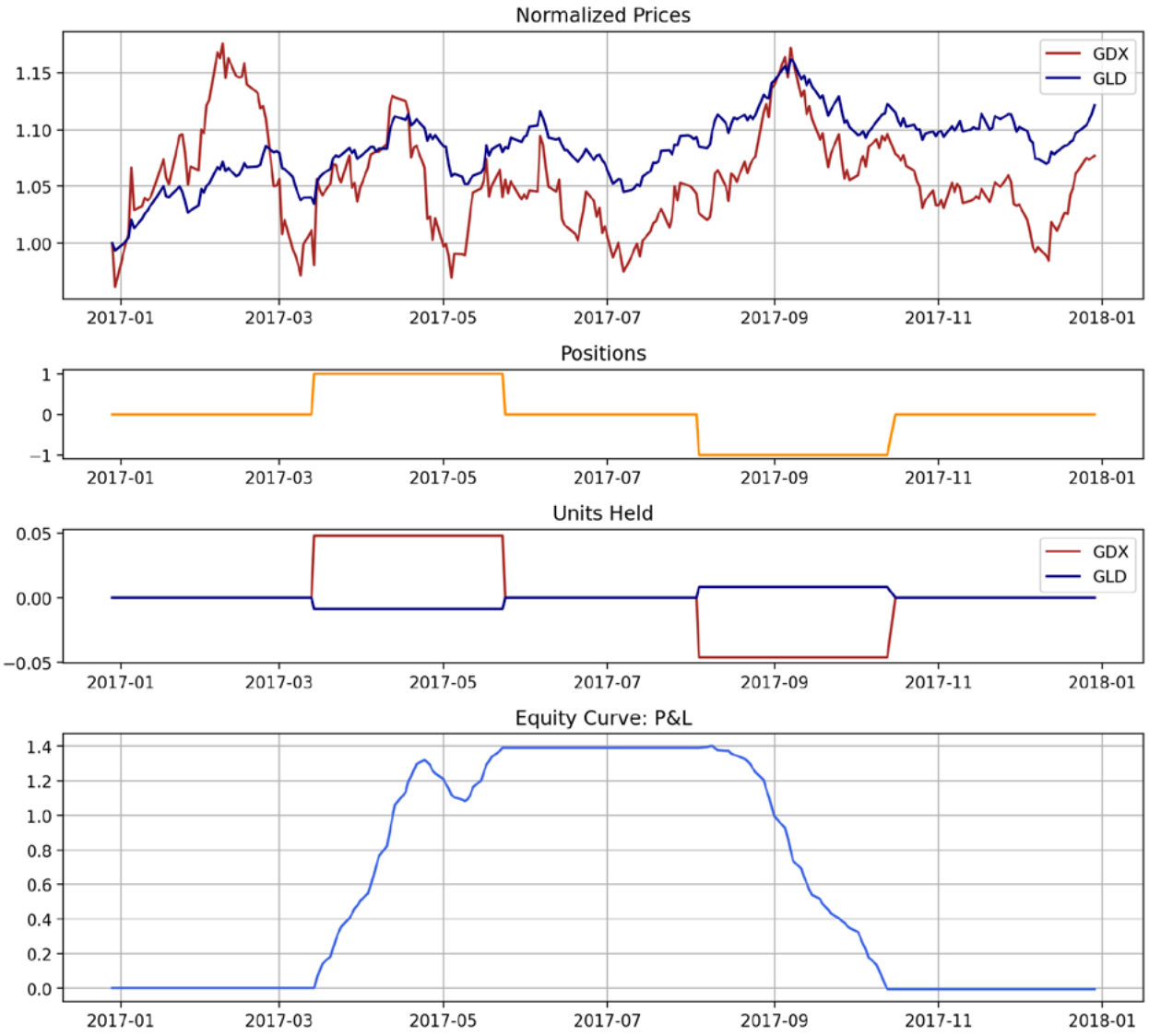


Returns-ish: Dollar-Neutral

As mentioned in [Gatev et al., 2006] for its proposed distance-based strategy, the trader takes 1 dollar short in the higher-priced stock and 1 dollar long for the lower-priced stock. In this case, there is no such thing as a standalone portfolio value time series evolving by itself.

We can still calculate the daily P&L, and then form an equity curve. Note that the P&L is computed over long-short positions for 1 dollar, it can somewhat be interpreted as "returns". However, we still need to keep in mind that this is fundamentally different from the returns for the case when someone just bought 1 dollar worth of some stocks because this 1 dollar is not the actual principal, it is just an arbitrary bound.

```
# Use the MPI module in the copula approach library for calculating dollar neutral units
CS = csmpi.CopulaStrategyMPI()

# I am just making the positions up for ease of demonstration.
position_data = [0]*50 + [1]*50 + [0]*50 + [-1]*50 + [0]*53
positions = pd.Series(data=position_data, index=portfolio.index)

# The method used below splits 1 dollar into long and short positions equally, hence adjust
# multiplier = 2
units = CS.positions_to_units_dollar_neutral(prices_df=testing_data, positions=positions,
                                             multiplier=2)

# Get the portfolio's daily P&L, which is the number of units held multiplying with prices
portfolio_pnl = units['GDX'] * testing_data['GDX'] + units['GDX'] * testing_data['GDX']

# Cumulatively sum the daily P&L for the equity curve
equity = portfolio_pnl.cumsum()
```



COMMON MISTAKES

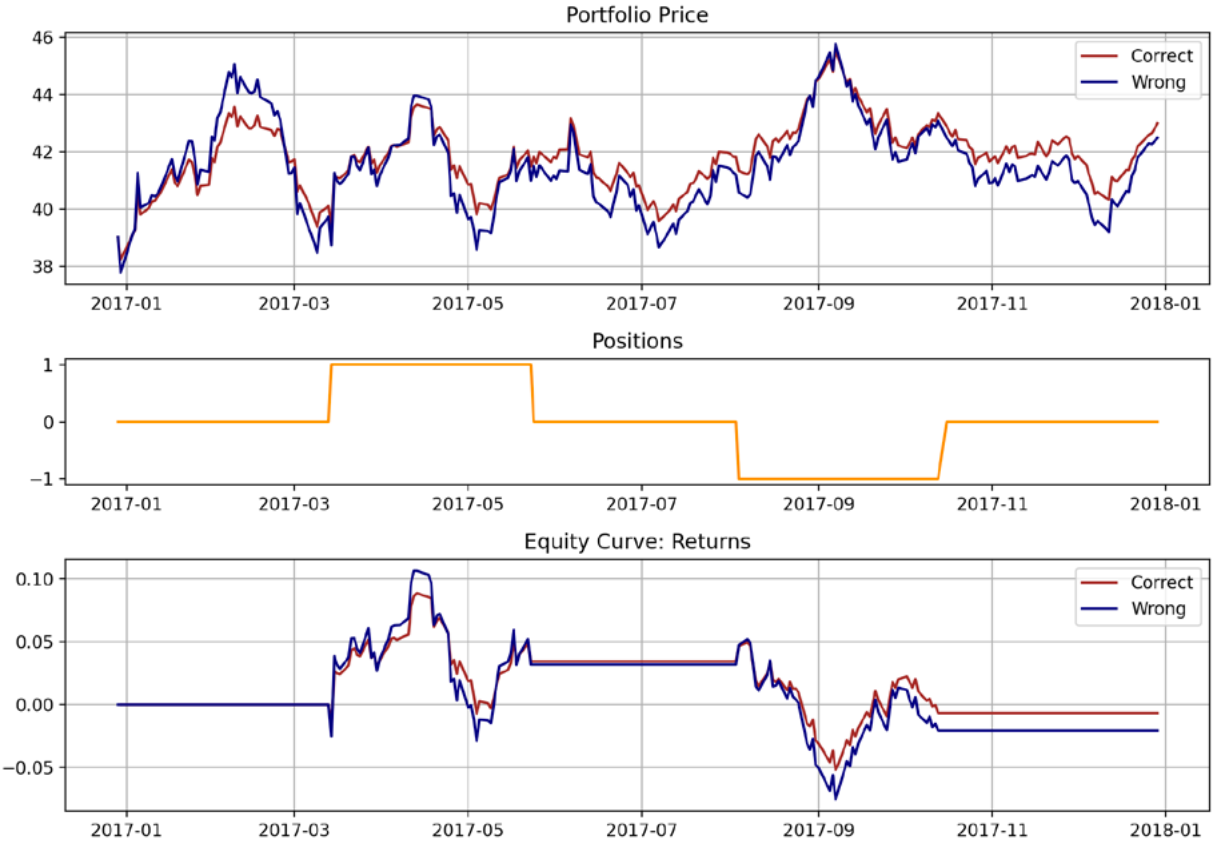
Linearly Combining Returns

When calculating the portfolio prices by itself, the returns $r(t)$ is

$$r(t) = \frac{\Pi(t)}{\Pi(t-1)} - 1$$

$$r(t) = \frac{w_A(t)S_A(t) + w_B(t)S_B(t)}{w_A(t-1)S_A(t-1) + w_B(t-1)S_B(t-1)} - 1 \neq w_A(t)r_A(t) + w_B(t)r_B(t)$$

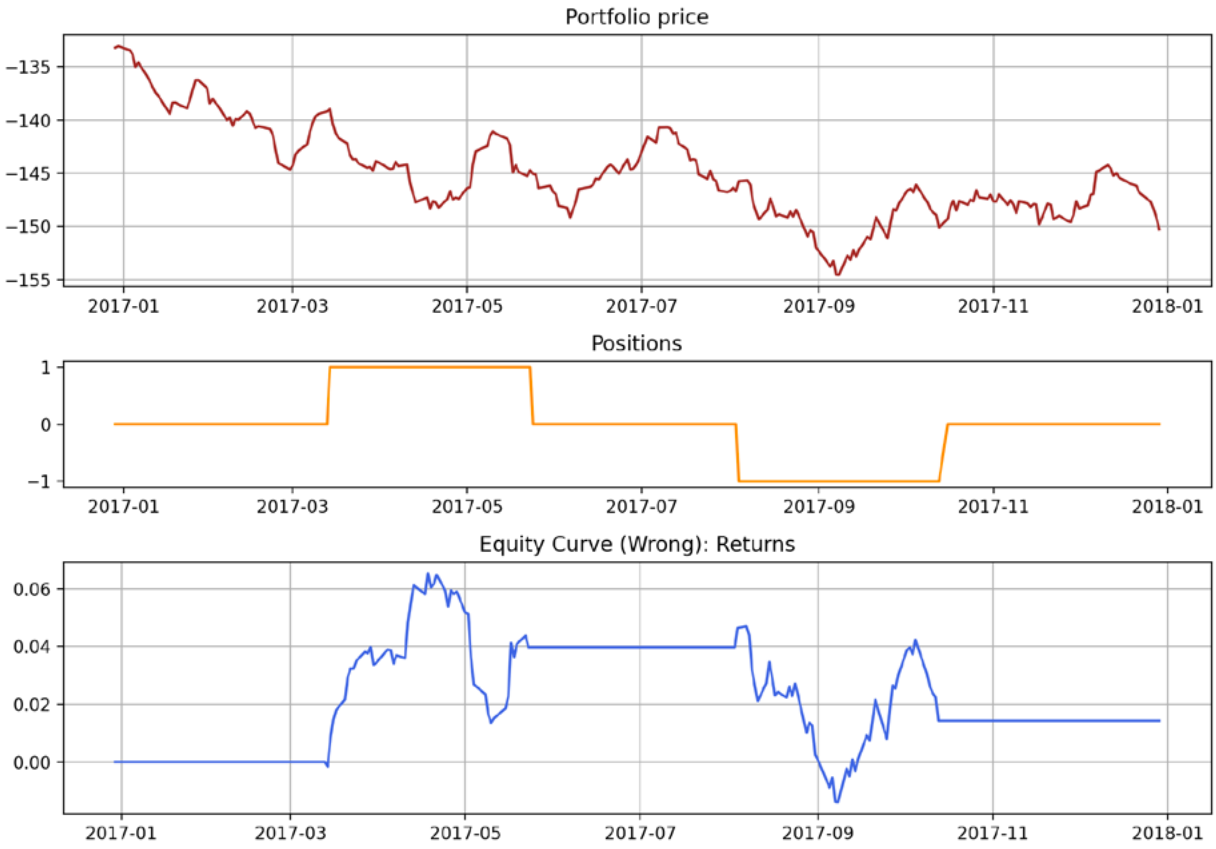
In plain English, it means you cannot linearly combine the returns from each component to calculate returns for the portfolio. If the weights are chosen well, the wrong equity curve will be around the correct equity curve so it is not completely the end of the world. The correct way to calculate $r(t)$ is the left side of the above equation: by forming the portfolio's value series from each component, then calculate the excess returns. The weights need to be positive as well.



Using the Long-Only Method on Other Portfolios

The method for calculating long-only portfolios should be applied to long-only portfolios. If used on a long-short portfolio, or where a component's value time series does not stay positive, it will break down.

See below for a wrong implementation on a negative time series using the method that was intended for long-only portfolios. We end up getting the opposite of what we are looking for.



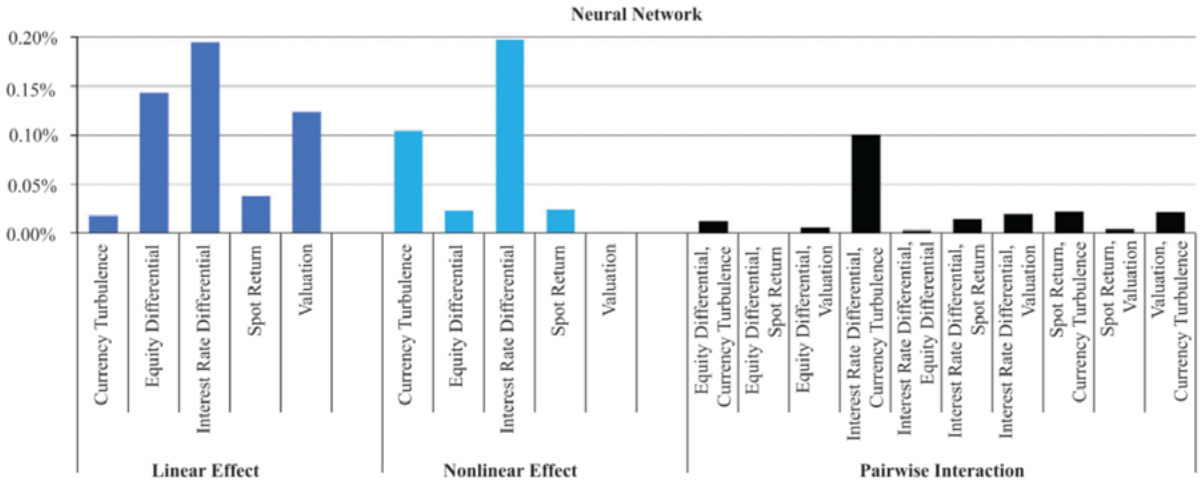
The logic breaks down when calculating returns of the portfolio, due to \Pi(t) not being strictly positive, and this invalidates the calculation of the return.

REFERENCES

1. **Gatev, E., Goetzmann, W.N. and Rouwenhorst, K.G., 2006. Pairs Trading: Performance of a Relative-Value Arbitrage Rule. The Review of Financial Studies, 19 (3).**

11.0

MODEL INTERPRETABILITY: THE MODEL FINGERPRINT ALGORITHM



Model Fingerprint (Li, Y., Turkington, D. and Yazdani, A., 2020)

"The complexity of machine learning models presents a substantial barrier to their adoption for many investors. The algorithms that generate machine learning predictions are sometimes regarded as a black box and demand interpretation. Yimou Li, David Turkington, and Alireza Yazdani present a framework for demystifying the behaviour of machine learning models. They decompose model predictions into linear, nonlinear, and interaction components and study a model's predictive efficacy using the same components. Together, this forms a fingerprint to summarize key characteristics, similarities, and differences among different models." [2]

One of the key principles in financial machine learning is:

"Backtesting is not a research tool. Feature importance is."

(Dr. Marcos Lopez de Prado)

The research guided by feature importance helps us understand the effects detected by the model. This approach has several advantages:

1. The research team can improve the model by adding extra features which can describe patterns in the data. In this case, adding features is not a random generation of various technical and fundamental indicators, but rather a deliberate process of adding informative factors.

2. During drawdown periods, the research team would want to help explain why a model failed and some degree of interpretability. Is it due to abnormal transaction costs, a bug in the code, or is the market regime not suitable for this type of strategy? With a better understanding of which features add value, a better answer to drawdowns can be provided. In this way models are not as 'black box' as previously described.

3. Being able to interpret the results of a machine learning model leads to better communication between quantitative portfolio manager and investors. Clients feel much more comfortable when the research team can tell a story.

There are several algorithms used to generate feature importances for various types of models.

1. **Mean Decrease Impurity (MDI)**. This score can be obtained from tree-based classifiers and corresponds to sklearn's feature_importances attribute. MDI uses in-sample (IS) performance to estimate feature importance.

2. **Mean Decrease Accuracy (MDA).** This method can be applied to any classifier, not only tree-based. MDA uses out-of-sample (OOS) performance in order to estimate feature importance.

3. **Single Feature Importance (SFI)**. MDA and MDI each suffer from substitution effects: if two features are highly correlated, one of them will be considered as important while the other one will be redundant. SFI is an OOS feature importance estimator which doesn't suffer from substitution effect because it estimates each feature importance separately.

Recently, we found a great article in a Journal of Financial Data Science:**Beyond the Black Box: An Intuitive Approach to Investment Prediction with Machine Learning. By Yimou Li, David Turkington and Alireza** Yazdani. Which provides a technique for feature importance by breaking any model down to its **linear**, **non-linear** and **pairwise interaction** effects for each feature. In this way, each model has a type of "fingerprint".

THE MODEL FINGERPRINTS ALGORITHM

Let's discuss at a high level the idea of the Fingerprints algorithm. For each feature X_i in a dataset we loop through possible feature values x_k and generate an average model prediction whilst keeping the other features fixed. As a result, for each feature k we have a set of x values and corresponding average prediction when feature values are fixed. This dependency is known as the partial dependence function $f_k(x)$.

The partial dependence function provides an intuitive sense for the marginal impact of the variable of interest, which we may think of as a partial prediction. The partial dependence function will have small deviations if a given variable has little influence on the model's predictions. Alternatively, if the variable is highly influential, we will observe large fluctuations in prediction based on changing the input values. [2]

Estimating linear/non-linear effect

In order to estimate linear and non-linear effect, for each feature we first fit a regression line for the partial dependence function. The regression line for feature k is denoted as l_k[x_{k,j}]. The authors of the paper define the linear effect as the mean absolute deviations of the regression line predictions around their partial dependence function.

$$\begin{aligned} &Linear\ \ prediction\ \ effect\ (x_k) \\ &= \frac{1}{N}\sum_{i=1}^{N} abs\left(\hat{l}_k[x_{k,i}] - \frac{1}{N}\sum_{j=1}^{N}\hat{f}_k[x_{k,j}]\right) \end{aligned}$$

The non-linear effect is defined as the mean absolute deviation of the total marginal (single variable) effect around its corresponding linear effect. [2]

$$Nonlinear\ prediction\ effect\ (x_k) = \frac{1}{N}\sum_{i=1}^{N} abs(\hat{f}_k[x_{k,i}] - \hat{l}_k[(x_{k,i})])$$

Estimating pairwise interaction effect

In order to estimate pairwise effect, we need to fix both feature values and generate partial dependence function value for a pair of feature values x_i, x_j. Interaction effect is defined as demeaned joint partial prediction of the two variables minus the demeaned partial predictions of each variable independently.

$$\begin{aligned} &Pairwise\ interaction\ effect\ (x_k, x_l) \\ &= \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} abs[\hat{f}_{k,l}(x_{k,i}, x_{l,j}) - \hat{f}_k(x_{k,i}) - \hat{f}_l(x_{l,j})] \end{aligned}$$

WHY THE MODEL FINGERPRINTS ALGORITHM?

The Model Fingerprint gives us the information not only about feature importance, but also how feature values influence model predictions. Does volatility have a low linear effect, but a high non-linear impact? Can it explain poor algorithm performance during range periods? How informative is 50-day volatility in a pair with the RSI indicator? Maybe a volume change itself is not that informative however in pair with price change it has high interaction effect. The algorithm not only highlights important features but also explains the nature of feature importance. This information can be used to better enable research to create future research roadmaps, feature generation plans, risk-models, etc.

The Model Fingerprints algorithm was implemented in the latest **mlfinlab** release. In this post we would like to show how MDI, MDA, SFI feature importance and Model Fingerprints algorithms can give us an intuition about a **Trend-Following** machine learning model.

TREND-FOLLOWING EXAMPLE

Import modules

```
import numpy as np
import pandas as pd
import timeit
from sklearn.ensemble import RandomForestClassifier, BaggingClassifier
from sklearn.metrics import roc_curve, accuracy_score, precision_score, recall_score,
                            f1_score, log_loss
from sklearn.tree import DecisionTreeClassifier
import matplotlib.pyplot as plt
import mlfinlab as ml
from mlfinlab.feature_importance import (mean_decrease_impurity, mean_decrease_accuracy,
                                         single_feature_importance, plot_feature_importance)
from mlfinlab.feature_importance import ClassificationModelFingerprint
```

Read the data, generate SMA crossover trend predictions

```
# compute moving averages

fast_window = 20

slow_window = 50


data['fast_mavg'] = data['close'].rolling(window=fast_window, min_periods=fast_window,

                                          center=False).mean()

data['slow_mavg'] = data['close'].rolling(window=slow_window, min_periods=slow_window,

                                          center=False).mean()

data.head()


# Compute sides

data['side'] = np.nan


long_signals = data['fast_mavg'] >= data['slow_mavg']

short_signals = data['fast_mavg'] < data['slow_mavg']

data.loc[long_signals, 'side'] = 1

data.loc[short_signals, 'side'] = -1


# Remove Look ahead bias by lagging the signal

data['side'] = data['side'].shift(1)
```

Filter the events using CUSUM filter and generate Triple-Barrier events

```
daily_vol = ml.util.get_daily_vol(close=data['close'], lookback=50)

cusum_events = ml.filters.cusum_filter(data['close'], threshold=daily_vol.mean() * 0.5)


t_events = cusum_events

vertical_barriers = ml.labeling.add_vertical_barrier(t_events=t_events, close=data['close'],

                                                     num_days=1)


pt_sl = [1, 2]

min_ret = 0.005

triple_barrier_events = ml.labeling.get_events(close=data['close'],

                                               t_events=t_events,

                                               pt_sl=pt_sl,

                                               target=daily_vol,

                                               min_ret=min_ret,

                                               num_threads=3,

                                               vertical_barrier_times=vertical_barriers,

                                               side_prediction=data['side'])

labels = ml.labeling.get_bins(triple_barrier_events, data['close'])
```

Feature generation

```
X = pd.DataFrame(index=data.index)

# Volatility

data['log_ret'] = np.log(data['close']).diff()

X['volatility_50'] = data['log_ret'].rolling(window=50, min_periods=50, center=False).std()

X['volatility_31'] = data['log_ret'].rolling(window=31, min_periods=31, center=False).std()

X['volatility_15'] = data['log_ret'].rolling(window=15, min_periods=15, center=False).std()

# Autocorrelation

window_autocorr = 50

X['autocorr_1'] = data['log_ret'].rolling(window=window_autocorr,

                                          min_periods=window_autocorr,

                                          center=False).apply(lambda x: x.autocorr(lag=1),

                                                              raw=False)

X['autocorr_2'] = data['log_ret'].rolling(window=window_autocorr,

                                          min_periods=window_autocorr,

                                          center=False).apply(lambda x: x.autocorr(lag=2),

                                                              raw=False)

X['autocorr_3'] = data['log_ret'].rolling(window=window_autocorr,

                                          min_periods=window_autocorr,

                                          center=False).apply(lambda x: x.autocorr(lag=3),

                                                              raw=False)

X['autocorr_4'] = data['log_ret'].rolling(window=window_autocorr,

                                          min_periods=window_autocorr,

                                          center=False).apply(lambda x: x.autocorr(lag=4),

                                                              raw=False)

X['autocorr_5'] = data['log_ret'].rolling(window=window_autocorr,

                                          min_periods=window_autocorr,

                                          center=False).apply(lambda x: x.autocorr(lag=5),

                                                              raw=False)
```

```
# Log-return momentum
X['log_t1'] = data['log_ret'].shift(1)
X['log_t2'] = data['log_ret'].shift(2)
X['log_t3'] = data['log_ret'].shift(3)
X['log_t4'] = data['log_ret'].shift(4)
X['log_t5'] = data['log_ret'].shift(5)


X.dropna(inplace=True)
labels = labels.loc[X.index.min():X.index.max(), ]
triple_barrier_events = triple_barrier_events.loc[X.index.min():X.index.max(), ]
X = X.loc[labels.index]
X_train, _ = X.iloc[:], X.iloc[:] # take all values for this example
y_train = labels.loc[X_train.index, 'bin']
```

Fit and cross-validate the model

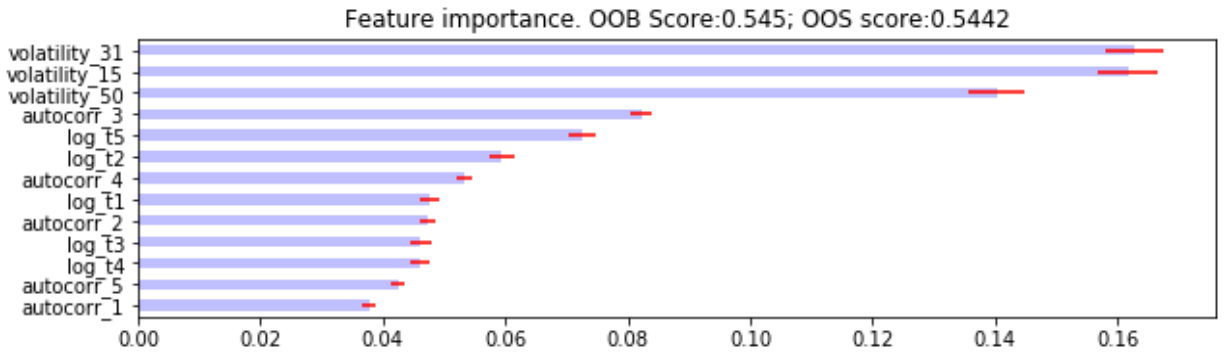
```
base_estimator = DecisionTreeClassifier(class_weight = 'balanced', random_state=42,
                                        max_depth=3, criterion='entropy',
                                        min_samples_leaf=4, min_samples_split=3,
                                         max_features='auto')
clf = BaggingClassifier(n_estimators=452, n_jobs=-1, random_state=42, oob_score=True,
                        base_estimator=base_estimator)
clf.fit(X_train, y_train)
```

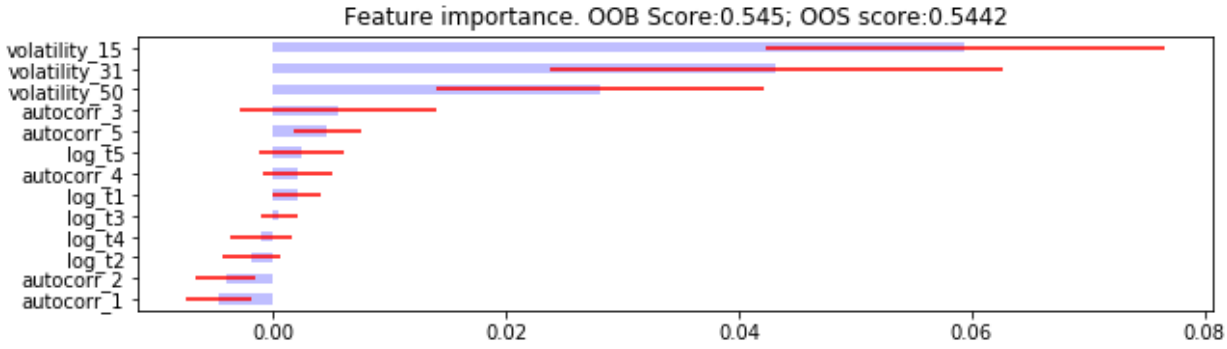
```
cv_gen =  ml.cross_validation.PurgedKFold(n_splits=5,
samples_info_sets=triple_barrier_events.loc[X_train.index].t1, pct_embargo = 0.02)


cv_score_acc = ml.cross_validation.ml_cross_val_score(clf, X_train, y_train, cv_gen,
sample_weight_train=None, scoring=accuracy_score, require_proba = False)


cv_score_f1 = ml.cross_validation.ml_cross_val_score(clf, X_train, y_train, cv_gen,
sample_weight_train=None, scoring=f1_score, require_proba = False)
```

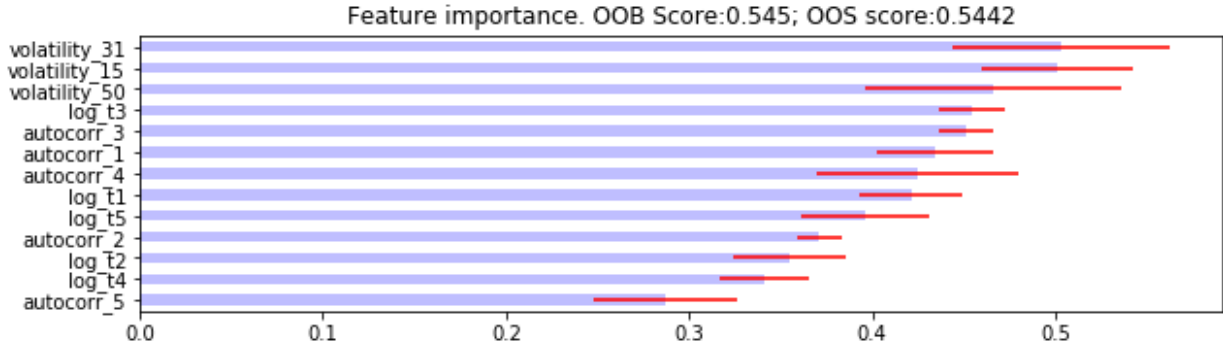
Generate MDI, MDA, SFI feature importance plots

```
mdi_feature_imp = mean_decrease_impurity(clf, X_train.columns)
mda_feature_imp = mean_decrease_accuracy(clf, X_train, y_train, cv_gen, scoring=log_loss)
sfi_feature_imp = single_feature_importance(clf, X_train, y_train, cv_gen, scoring=log_loss)


plot_feature_importance(mdi_feature_imp, oob_score=clf.oob_score_,
oos_score=cv_score_acc.mean())
plot_feature_importance(mda_feature_imp, oob_score=clf.oob_score_,
oos_score=cv_score_acc.mean())
plot_feature_importance(sfi_feature_imp, oob_score=clf.oob_score_,
oos_score=cv_score_acc.mean())
```

Interpreting Results

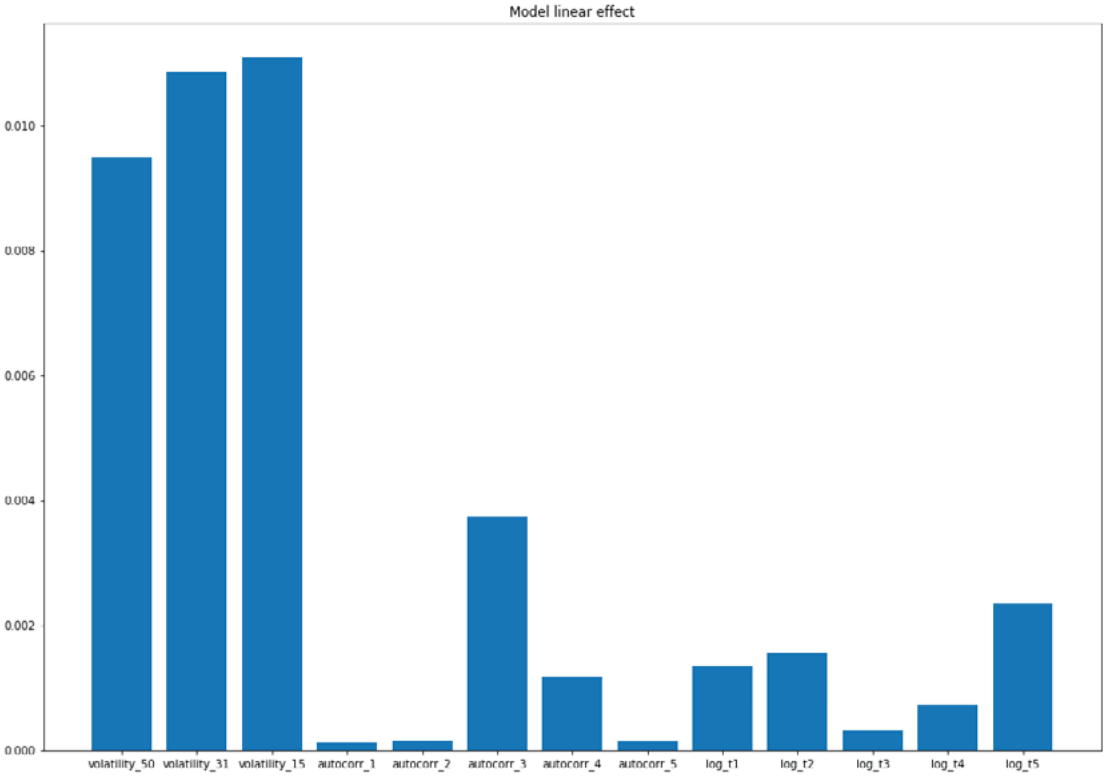


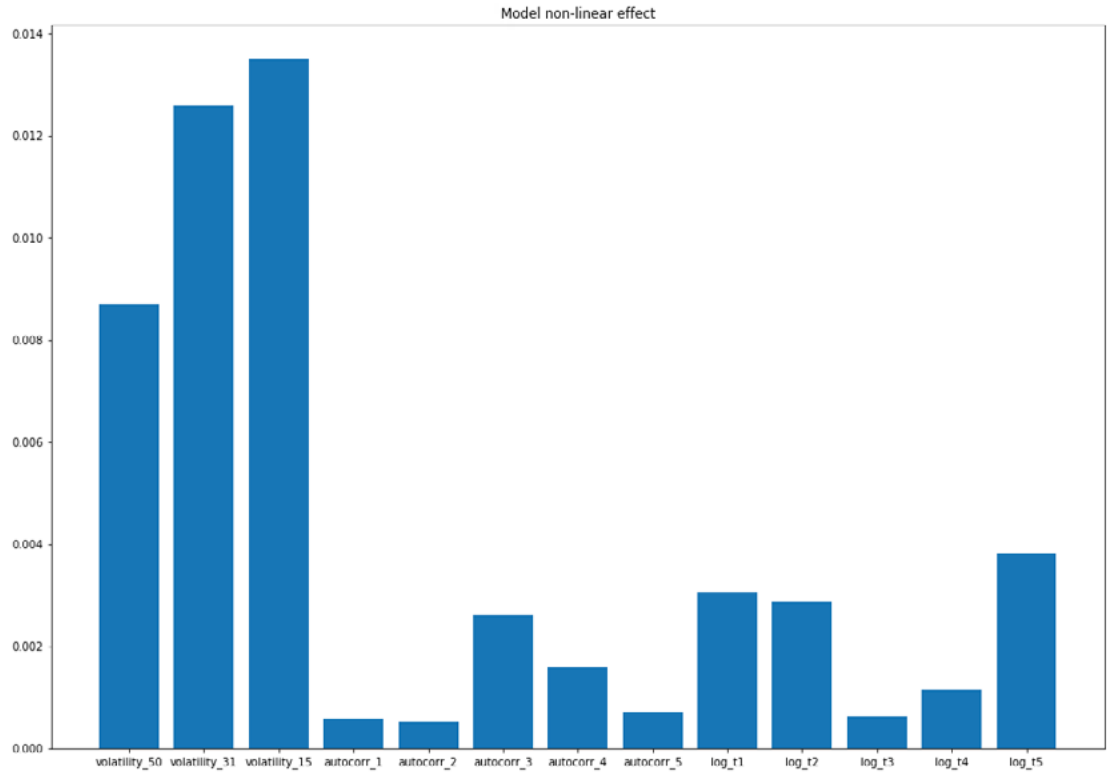


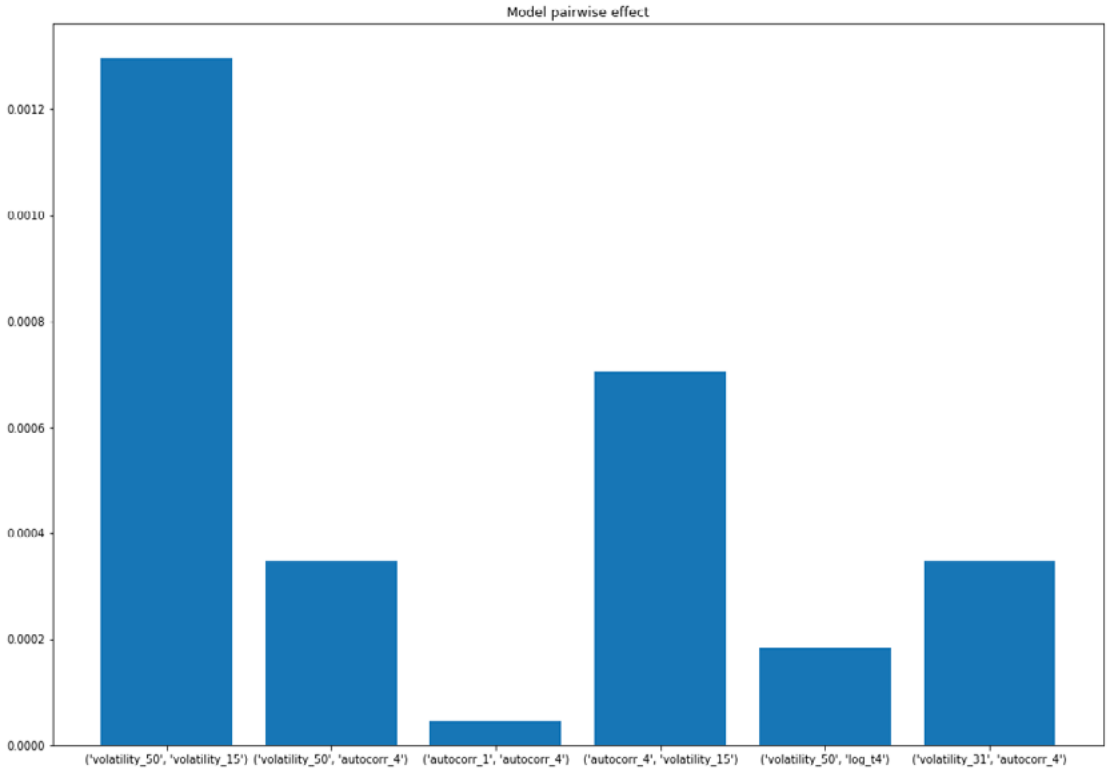


What we can see from all the feature importance plots that the volatility features provide the model with the most value. We would like to use the Fingerprints algorithm to generate more insights about model predictions.

```
clf_fingerpint = ClassificationModelFingerprint()
clf.fit(X_train, y_train)

feature_combinations = [('volatility_50', 'volatility_15'), ('volatility_50', 'autocorr_4'),
                        ('autocorr_1', 'autocorr_4'), ('autocorr_4', 'volatility_15'),
                         ('volatility_50', 'log_t4'), ('volatility_31', ('autocorr_4'))]

clf_fingerpint.fit(clf, X_train, num_values=50, pairwise_combinations=feature_combinations)



# Plot linear effect
plt.figure(figsize=(17, 12))
plt.title('Model linear effect')
plt.bar(*zip(*clf_fingerpint.linear_effect['raw'].items()))
plt.savefig('linear.png')

# Plot non-linear effect
plt.figure(figsize=(17, 12))
plt.title('Model non-linear effect')
plt.bar(*zip(*clf_fingerpint.non_linear_effect['raw'].items()))
plt.savefig('non_linear.png')

# Plot pairwise effect
plt.figure(figsize=(17, 12))
plt.title('Model pairwise effect')
plt.bar(*zip(*clf_fingerpint.pair_wise_effect['raw'].items()))
plt.savefig('pairwise.png')
```


Model linear effect
0.010
0.008
0.006
0.004
0.002
0.000
volatility_50
volatility_31
volatility_15
autocorr_1
autocorr_2
autocorr_3
autocorr_4
autocorr_5
log_t1
log_t2
log_t3
log_t4
log_t5


Model non-linear effect
0.014
0.012
0.010
0.008
0.006
0.004
0.002
0.000
volatility_50
volatility_31
volatility_15
autocorr_1
autocorr_2
autocorr_3
autocorr_4
autocorr_5
log_t1
log_t2
log_t3
log_t4
log_t5



What we can see from Fingerprints algorithm is a strong linear, non-linear and pairwise effect for volatility features. It also coincides with MDI, MDA and SFI results. What can a researcher conclude from this? Volatility is an important factor used to determine the probability of success of SMA crossover trend system.

We may also conclude that other features are not that informative for our model. In this case, the next research step would be to concentrate more on volatility type of features and extract more information from them. Try estimating the entropy of volatility values, generate volatility with bigger look-back periods, take the difference between volatility estimates and so on.

CONCLUSION

In this post we have shown how the **mlfinlab** package is used to perform feature importance analysis using MDI, MDA, SFI and the Model Fingerprints algorithm. The approach described above is just one of many ways to interpret a machine learning model. One can also use Shapley values, and the LIME algorithm to get the intuition behind the model's predictions.

REFERENCES

1. Marcos Lopez de Prado. Advances in Financial Machine Learning. Wiley, 2018.

2. Journal of Financial Data Science:Beyond the Black Box: **An Intuitive Approach to Investment Prediction with Machine Learning. By Yimou Li, David Turkington and Alireza Yazdani**

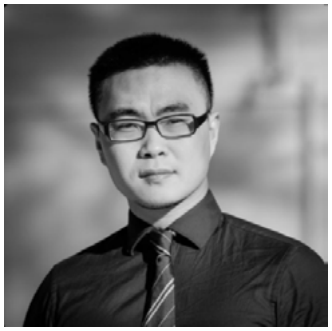
We want to thank everybody at Hudson & Thames for making this book possible!

JACQUES JOUBERT
CEO & Co-founder

OLEXANDR PROSKURIN
Co-founder & Researcher

ILLYA BARZIY
Quantitatve Research Team Lead at Hudson & Thames

VALERIIA PERVUSHYNA
Quantitative Researcher at Hudson & Thames



HANSEN PEI
Applied Math PhD Candidate at University of Delaware, Researcher at Hudson & Thames

YEFENG WANG
Ph. D. Candidate in Health Informatics at University of Minnesota-Twin Cities, Researcher at Hudson & Thames


HUDSON
AND THAMES