Statistical Arbitrage on Cryptocurrencies

Giovanni La Cagnina
École Polytechnique
Fédérale de Lausanne
Lausanne, Switzerland
Email: giovanni.lacagnina@epfl.ch

Joshua Voelkel
École Polytechnique
Fédérale de Lausanne
Lausanne, Switzerland
Email: joshua.voelkel@epfl.ch

Abstract—**This project explores a statistical arbitrage strategy using 1-minute prices on several cryptocurrency pairs. The data used is from the Binance exchange, consisting of the 13 most liquid cryptocurrencies (ETH, BNB, XRP, BTC, ADA, SOL, DOGE, TRX, BUSD, MATIC, LTC, DOT, USDC) and their prices against Tether (USDT). The report covers the data retrieval and manipulation, the theoretical aspects of the strategy, and an analysis of the results. It also discusses the successes and failures of the strategy and concludes with proposing potential improvements for the strategy used.**

I. Introduction

The cryptocurrency market has experienced strong growth in recent years, with a rising number of digital assets being traded on multiple platforms. This leads to the question whether well-established trading strategies from traditional financial markets, such as statistical arbitrage, work also well in the cryptocurrency space, hypothetizing that this early-stage environment could exhibit pricing inefficiencies that could be exploited by trading strategies such as statistical arbtitrage Fischer et al. [5]. Statistical arbitrage is a pair trading strategy which involves identifying and exploiting small price discrepancies between two assets based on a mean-reversion paradigm Avellaneda and Lee [1]. One particular implementation is the so called "two-sigma strategy", which has been demonstrated to be very effective in traditional financial markets Bertram [2], Gatev et al. [6]. In this project, we will examine the potential of the two-sigma statistical arbitrage strategy on in the cryptocurrency space using several cryptocurrency pairs with 1-minute prices from 2021. Additionally, this project explores the effects of rolling calibration hyperparameters. In section II of this report, we describe the data retrieval process, data management, and the methods considered for building usable clean datasets. Considering this we also illustrate the pipeline implemented for storing the results obtained to enhance a deep analysis of the results. Section III deals with the theoretical aspects of the trading strategy and describes its implementation in Python. In section IV, we analyze the obtained results and focus on why the strategy failed in some cases and succeeded in others. Based on our obtained results, we propose potential improvements in section V. We conclude our project in section VI.

II. Data

A. Type of data considered

We have considered the 13 most liquid cryptocurrencies in 2021 (ETH, BNB, XRP, BTC, ADA, SOL, DOGE, TRX, BUSD, MATIC, LTC, DOT, USDC) traded on the Binance exchange, one of the most used crypto exchanges available. Binance is also the source for all data we used for this project. To enhance the computational efficiency, all data has been downloaded in an automatized and parallelized way. Note that for each cryptocurrency considered we consider as reference price its price against Tether ('USDT'). This choice has been made to be able to consider each cryptocurrency price as the price in dollars, given that Tether is the most traded stable coin that pegs the dollar 1:1. For each cryptocurrency and each month, we retrieve two types of datasets, from the data collection: trade books datasets and klines datasets. An example of the information contained in each of these two datasets is reported in table IV and in table V. The trade books datasets consist of reports of the trades that took place on the exchange Binance with a frequency of 1 millisecond. In this dataset it is possible to retrieve for each trade: the price at which has been executed, the quantity exchanged, the quantity available for the given price, and two boolean variables indicating if the price considered for the trades was the best price available for the corresponding side, and the side, buy or sell, at which the trade occurred. The klines datasets contain 1-minute aggregated information about the states of the market for a given cryptocurrency. More precisely, these datasets contain the open price, the close price, the high price, the low price, the volume, the quoted volume and the number of trades executed in a given 1-minute interval.

B. Price Data Retrieval Pipeline

In Figure 1 we illustrate the process implemented to retrieve the raw data from the Binance Data Collection. As explained in the previous section the raw data consist of a data set for each cryptocurrency for each month considered. The dataset can be either a texttttrades data set or a texttklines data set, the process has been implemented to be compatible with both these two types of data sets. For each cryptocurrency and a corresponding list of dates the required data sets are retrieved using parallel computation obtaining as an output a list of data sets. These files are then saved both in an external hard drive, if available, and directly in a dedicated Google Drive folder. This last saving method has been introduced in order to allow the sharing of the data with all participants of the project. This process is then repeated for all the 13 currencies considered. All data which was used for this project (both raw and clean) can be found in the drive (Link).

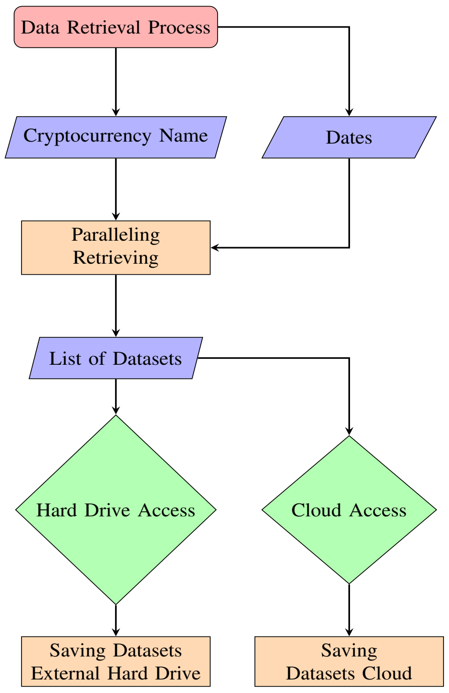


Fig. 1: Price Data Retrieval Process

C. Data Manipulation

In this section, we explain the manipulation applied after having obtained the raw data sets for each cryptocurrency and for each date, in order to obtain a unique clean data set usable for the application of the strategy. As explained in the previous section we consider two types of data sets: `klines` and `trades` datasets. The main objective of the data manipulation process is to obtain a reliable proxy of the mid-price, defined as the average between best bid and the best ask prices in a trade book, it is commonly used as a proxy for the "fair value" of a security, Easley et al. [4]. By using mid-price as the reference point, we can identify when the spread between the two crypto securities deviates from the fair value and take advantage by opening a suitable position.

- Given the fact that using `klines` datasets we do not have access to any information of the bid and ask books we can build several mid-price proxies:
  - Average between the close and open price for each period, in this case 1-minute.

$$p_{mid}^{klines} = 0.5 \times p_{open}^{klines} + 0.5 \times p_{close}^{klines}$$

  - Average between the low and high price for each period, in this case 1-minute.

$$p_{mid}^{klines} = 0.5 \times p_{low}^{klines} + 0.5 \times p_{high}^{klines}$$

  - Typical price, which corresponds to the average between close, high and low price

$$p_{mid}^{klines} = \frac{1}{3} \times p_{low}^{klines} + \frac{1}{3} \times p_{high}^{klines} + \frac{1}{3} \times p_{close}^{klines}$$

Unlike the `klines` dataset, in the `trades` dataset, we have more information related to the bid and ask books. This is due to the presence of the variable, `isBuyerMaker`, which allows for the identification of the side on which each trade was executed, and the variable, `isBestprice`, which enables the determination of whether the price at which the trade was conducted was the best available price for that side. Consequently, a proxy for the mid-price can be constructed by utilizing the following methodology: loading the milliseconds `trades` datasets for each asset, discarding trades that were not executed at the best available price, and subsequently dividing the dataset into two tables, one comprising of buy-side orders, and the other consisting of sell-side orders. Subsequently, to obtain datasets with a frequency of 1 minute, the data in milliseconds is resampled by averaging the price of each trade by the quantity traded. This method allows for the provision of more weight to prices that have traded a larger quantity of the asset. To generate a unique dataset that contains the mid-price, the buy and sell datasets are merged (inner merge) and the two prices are averaged for each date, in accordance with the definition of the mid-price:

$$p_{mid}^{trades} = 0.5 \times p_{buy}^{trades} + 0.5 \times p_{sell}^{trades}$$

The above-described method enables the calculation of a proxy for the mid-price that is based on the trades executed on both the buy and sell sides. It is important to note that in order to enhance the precision of this estimation and to acquire prices that more accurately reflect the state of the market during the execution of the strategy, the inclusion of non-best match prices within the dataset would be beneficial. Consequently, this methodology is reliable when applied in the context of a statistical arbitrage strategy that comprises of numerous small trades rather than large positions that have the potential to substantially influence the price and necessitate the division of the order into multiple segments to locate a corresponding offer on the opposite side of the exchange.

Given these consideration we implemented in our code both a proxy for the mid price as average between close and open prices, and also a proxy obtained with `trades` datesets. The reported results are obtained using mid-price calculated from the `trades` datasets given that we consider this proxy more realistically close to the real mid-price.

The last step in the process of data manipulation is the creation of a datasets containing the mid-price for all the coins considered. This is done by using parallelization, in order to allow the calculation of the mid price for six coins simultaneously. Once the mid-price is calculated for all pairs, for all months, we merge these datasets into one, and we save it both locally and automatically we upload it on Google Drive, to allow the sharing of the data with all the participants of the team.

D. Margin Fee

Additionally, for each crypto asset, we have retrieved the margin fee associated. The margin fee, can be defined as the expense incurred in relation to short selling of an asset. It should be noted that when an individual engages in short selling of a crypto asset, the exchange necessitates the payment of an additional fee, which can be perceived as the interest on the loan obtained for borrowing the asset. The retrieval of the margin fees has been done using web scraping. We have used Selenium to retrieve those margin fees. Note that the margin fees are not constant in time, but the Binance exchange updates them in a continuos manner. To simplify the retrieval of such data we have considered the last available margin fee for each asset.

III. Methodology

In this section, we explain in detail the technicalities of our implemented trading strategy. First, we introduce the theoretical aspects of our strategy and then we describe how we implemented it.

A. Theory

Statistical Arbitrage is one of the most popular pair-trading strategies. It is based on the idea of co-integration, that is we can write the price of the one asset (denote it as Y_t) as a linear combination of the other asset price X_t, i.e.

$$Y_t = \alpha + \beta X_t + \epsilon_t,$$

where the residuals ϵ_t are stationary. This allows us to trade the spread Z_t by going long in asset Y_t and going short β times asset X_t, that is

$$\begin{aligned} Z_t &= Y_t - \beta X_t \\ &= \alpha + \epsilon_t \end{aligned}$$

Assuming that co-integration holds, the spread will then have mean α and since ϵ_t is stationary, the spread is mean reverting. *Statistical arbitrage* exploits this mean-reversion property by trading the spread. If the spread is above its mean α, we can short the the spread (go long β times X_t and go short Y_t) and close our position once the spread returned to its mean α. If the spread is below its mean, we can go long in the spread (go long Y_t and go short β times X_t) and close our position once the spread returned to its mean α.

Clearly, we have to decide how far the spread must be away from the mean in order for us to enter a short or long position. One commonly used rule is the two-sigma (a.k.a. two standard deviations) rule proposed by Bertram [2], which states to enter a long/short position if the spread deviates more than two standard deviations from its mean and close the position once the spread returned to its mean. For the theory behind this finding, we refer to Bertram [2]. We use this two-sigma approach in our trading simulation.

B. Implementation

1) Trading signal: In the first step, we need to determine when to start trading. That is, when the asset spread is stationary (i.e. the pair is co-integrated). Therefore, we do a rolling linear regression for each possible combination of asset pairs at each minute using the past 12 hours of data (in minutes frequency). By that, we obtain an estimate of the slope $\hat{\beta}$, of the intercept ($\hat{\alpha}$), the residuals, and the standard deviation of the residuals. To test for co-integration, we test if the residuals are stationary using the Augmented Dickey-Fuller test (Dickey and Fuller [3]). If the resulting p-value is small enough, we can reject the null hypothesis that the residuals are non-stationary. Stationarity implies mean-reversion. The choice of "when the p-value is small enough" is a hyperparameter, which has to be determined in advance. We test our strategy for a total of four different p-value thresholds: $0.1, 0.5, 0.01$, and 0.005, which can be considered to be the most common ones. Once we obtain a p-value smaller than the threshold, we see this as a signal to start trading, as the time series of the spread seems to be mean-reverting.

2) Trading window: After we have obtained a trading signal, we start trading for a predefined trading period. Also this choice is a hyperparameter. Note that in order to have a realistic setting, we execute trades always in the next minute after we have obtained a signal (i.e. lagged execution of trades), as we are not able to execute trades immediately in the real world. We test our strategy on multiple trading window sizes (30 minutes, 1 hour, 2 hours, 4 hours, 6 hours, and 12 hours). At the beginning of the trading period, we define the state variable $S = 0$, which means that we do not own any assets. Then, at each time point (i.e. every minute), we calculate the spread value and as soon as the spread $Z_t > \hat{\alpha} + 2\hat{\sigma}$, we short-sell the spread by taking $\hat{\beta}$ long positions of X_t and going one-time short asset Y_t. Now, we set our state variable $S = -1$. Analogously, when we are at state $S = 0$ and the spread develops in such a way that $Z_t < \hat{\alpha} - 2\hat{\sigma}$, we expect the spread to return to its mean $\hat{\alpha}$ and thus, go long the spread by going long asset Y_t and going short $\hat{\beta} X_t$. We close our positions when one of the following three events occurs:

1) the spread returns to its mean value of $\hat{\alpha}$,
2) our positions reaches the stop-loss of -20%,
3) the predetermined trading period reaches its end.

If we close our position due to the first reason, it means that our pre-determined trading period is not over at this point. Therefore, we repeat the strategy until we reach the end of the trading period. Note that we keep the beta fixed for the trading period. Furthermore, it is important to mention that the stop-loss choice is a hyperparameter which needs to be determined in advance. We chose it to be -20% as it seemed to be a reasonable loss we could consider.

3) Cost factors: To test the strategy under realistic assumptions, we need to take into account two important cost factors, more precisely the borrowing cost when shorting assets and the transaction costs when executing trades. The borrowing cost, a.k.a. margin borrow interest is dependent on the asset and since we tested our strategy on a limited amount of assets, we retrieved the most current margin borrow interest for each of the relevant assets from Binance as described in section II-D. The margin borrows interest is a percentage value and is based on the initially borrowed money (i.e. the price of the shorted asset when opening the trade) and is calculated on an hourly basis. The interest, however, is reported as a daily rate. To incorporate this borrowing cost into our strategy, we calculated the cost when closing the trade since it is dependent on how long we keep our short position open. The other important cost factor is the transaction cost when buying or selling an asset. As described in the section II, we used also in this case the transaction fee from Binance, to have a realistic setting. The transaction fee is a percentage fee based on the asset price and is fixed for all assets with 0.1%.

4) Final steps: After the trading period reached its end, we repeat our strategy by estimating the $\hat{\beta}$ on a rolling basis until we reach a trading signal indicated by the low p-value regarding the stationarity of residuals. We continue with this approach until we reach the end of the dataset. As a final step, we calculate the total return over all trades we have done. As mentioned before, we test our strategy with 24 hyperparameter combinations: 4 different p-value thresholds and 6 different trading window sizes. To get a quick and intuitive understanding of how well the respective hyperparameter combination performed, we create a heat map for each asset pair, as shown in figure 6. Note that we report the multiplicative returns $R_t = 1 + r_t$ as it is the most intuitive metric and you can simply multiply them to get the total return over a longer period. This means if $R_t = 2$, we double our initial investment and if $R_t = 0.4$, we lost 60% of our initial investment.

5) Results: For each pairs we save the results of the whole statistical arbitrage strategy in a dictionary, with keys corresponding to monthly date strings, for example "2021-01". This object then is saved, both on the cloud and locally using the library `joblib` that allows us to save python objects ina straight forward manner and to load them without loosing any information. Then inside of this objects it possible to access all the informations and results of the strategy. The structure of this object is illustrated in the following schema. As we can see from the schema for each month we store a list containing two object. In the first object of the list we save a data frame that is used as a summary of the month for the given pair: indeed in this data frame we store, for each combination of hyperparameter the multiplicative return of the strategy. This object is then used to build heat-map, such as 6. Another important element to save, for a result analysis is the trade book of our strategy, this is why at the second element of the above cited list we save a dictionary, where keys correspond to string that correspond to combination of the two hyperparameter, such as `{"hours":2}_0.01`. For each key of this dictionary we store a data frame contaning the trade book generated from the strategy for the given combination of hyperparamer, where it is possible to have access to single trade information together with all the information of the linear regression used to estimate the beta coefficient for that each trading window. This objects allows us to create plots like: 11.

Pair	Month	Return	T.W. (hrs:min)	p-value
BNB - TRX	2021-09	2.53278e-106	12:00	0.1
BNB - LTC	2021-06	7.18487e-100	12:00	0.1
ETH - BTC	2021-08	3.00343e-93	12:00	0.05
ETH - BNB	2021-12	4.3375e-70	12:00	0.1
XRP - LTC	2021-07	1.49876e-67	12:00	0.01
XRP - ADA	2021-06	6.06091e-67	12:00	0.05
BTC - LTC	2021-06	1.81815e-66	12:00	0.1
XRP - TRX	2021-07	5.93564e-66	12:00	0.1
ADA - LTC	2021-06	1.12755e-64	12:00	0.05
ETH - TRX	2021-06	1.08486e-61	12:00	0.005
BTC - TRX	2021-08	3.73028e-61	12:00	0.05
XRP - BTC	2021-09	1.95041e-60	12:00	0.1
BNB - DOT	2021-07	3.6863e-60	12:00	0.1
ETH - DOGE	2021-09	6.74613e-55	12:00	0.05
ETH - LTC	2021-07	7.12268e-55	12:00	0.05
TRX - LTC	2021-09	8.8517e-54	12:00	0.1
ETH - ADA	2021-07	1.08467e-51	12:00	0.1
XRP - MATIC	2021-07	1.42316e-51	12:00	0.05
ADA - TRX	2021-07	4.3359e-50	12:00	0.05
BTC - ADA	2021-09	4.76076e-50	12:00	0.1

TABLE I: Worst 20 performing pairs (monthly cumulative returns)

IV. Results

In this section, we are analyzing the results we obtained by applying our trading strategy. All obtained results are stored in the drive (Link). In total, we tested our strategy on 56 pairs using the historical price data beginning 2020-12 until the end of 2021-12. As already briefly described in section III-B, we created for each pair and each month a heatmap, with the aforementioned four different p-values on the x-axis and the six different trading periods on the y-axis as shown in figure 6. Figure 6 is a good example of how much influence the choice of hyperparameters can have on the return. While we lost nearly all of the initial investment using a p-value threshold of 10%, we could make strong positive returns when considering a p-value threshold of 1% and 0.5%. Also, the different trading periods can lead to strong variations in returns. In the example case of figure 6, we make strong returns using the p-value threshold of 0.5% for all trading periods except for the 30-minute period, where we lose roughly 60% of our initial investment.

We are reporting the 20 worst and 20 best monthly performances and the respective parameter choice for each pair over the 13 months and all hyperparameters in table I and II respectively. From table I, we can immediately see that all of the worst 20 performances have a trading window of 12 hours, i.e. the longest possible trading window. A possible reason for this fact is that the statistical behaviour of the time series changes in that time window, which would be in line with the general ever-changing statistical nature of cryptocurrencies. It must be also mentioned that 12 hours is also the rolling calibration window size we use to come up with trading signals, beta, etc. and which can be seen as a strong form of evidence that our calibration window should be at least 12 hours to capture the changing behaviour of the underlying time series. Regarding the p-value threshold, we cannot really conclude much from the table. The only thing worth mentioning is that there is only one pair (ETH - TRX), which had its overall worst perfomance with the lowest p-value threshold of 0.5%. For this reason, we plotted a histogram of all p-values for the worst monthly performances per pair in 2.

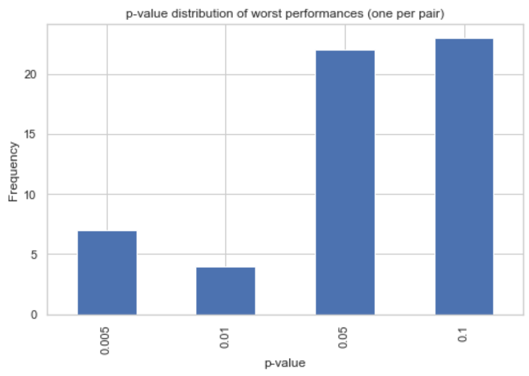


Fig. 2: p-value distribution of worst performances per pair

From the histogram, we can see that the strong majority of pairs had their worst performances with high p-value thresholds. This seems to make sense as the p-value threshold is determining how likely the it is that the spread time series is stationary (and thus mean reverting) using the past 12 hours of data (low p-value indicates higher probability of rejecting non-stationarity). Clearly, this does not indicate that the spread time series stay stationary once we start trading, however, if we are assuming that there is even a slight serial correlation in the spread price time series, we can argue that a lower p-value will indicate a higher chance of mean-reverting behaviour of the spread time series while trading. From table I, we also can see that for all of the reported 20 pairs, the worst performance resulted in a return of basicall -100%. In fact for all tested 56 pairs, the worst monthly performance was each -100%. We will analyse the possible reasons for this observation more detailed in subsection IV-B.

By taking a look at table II, we can obeserve that we have an incredibly outstanding return of more than $2e+107$ for one pair. We further explore how this result occured in subsection IV-A. Also for the other listed pairs, we see that that all of them had very strong performances in their strongest month. By taking a look at the trading window, we cannot really see a clear pattern. Therefore, we report the histogram of trading window sizes for all best monthly performance per pair in figure 3.

Pair	Month	Return	T.W. (hrs:min)	p-value
ADA - DOGE	2021-05	2.04209e+07	04:00	0.1
ADA - SOL	2021-05	405.638	12:00	0.005
ADA - MATIC	2021-04	148.837	01:00	0.005
MATIC - DOT	2021-01	66.6564	01:00	0.01
DOGE - MATIC	2021-02	46.1147	02:00	0.05
SOL - DOT	2021-01	35.3516	01:00	0.1
DOGE - LTC	2021-05	34.8946	02:00	0.005
BNB - DOGE	2021-05	29.8484	06:00	0.05
SOL - MATIC	2021-04	19.4446	04:00	0.05
XRP - DOGE	2021-05	16.2515	01:00	0.1
LTC - DOT	2021-01	13.6122	02:00	0.005
SOL - TRX	2021-02	13.232	06:00	0.05
DOGE - TRX	2021-04	13.2251	04:00	0.1
ADA - TRX	2021-02	12.6084	12:00	0.01
ADA - LTC	2021-02	10.1533	00:30	0.005
ETH - DOGE	2021-05	9.18611	12:00	0.01
ADA - DOT	2021-06	8.29277	06:00	0.005
SOL - LTC	2021-05	8.07913	06:00	0.1
DOGE - DOT	2021-01	7.47317	06:00	0.05
TRX - MATIC	2021-04	6.15108	12:00	0.05

TABLE II: Top 20 performing pairs (monthly cumulative returns)

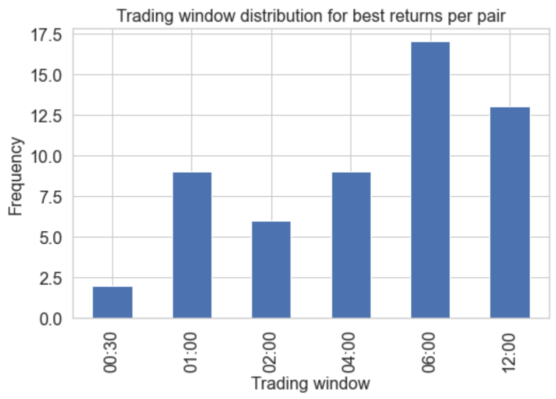


Fig. 3: trading window distribution of best performances per pair

Here, we can surprisingly see that the best performing trading windows are 6 and 12 hours while the worst performing trading window size is 30 minutes. In subsections IV-A and IV-B, we further elaborate this observation. For the p-value threshold, we can can observe that there are many best performances with the smalles p-value threshold of 0.5%. To get an even more clear understanding of the the distribution, we plotted the distribution of different p-values for the best performances per pair in figure 4. Here, we see that the best performing p-value threshold is indeed 0.5%, i.e. the smallest. This is in line with the oservations of the worst performing pairs in the previous paragraph. However, interestingly, the worst performance is 1% and then 10%.

In the following subsections, we are going to analyze both the performance of the best and worst performing pair in more detail to get an understanding, why our strategy works well in some cases and why it works terrible in other cases.

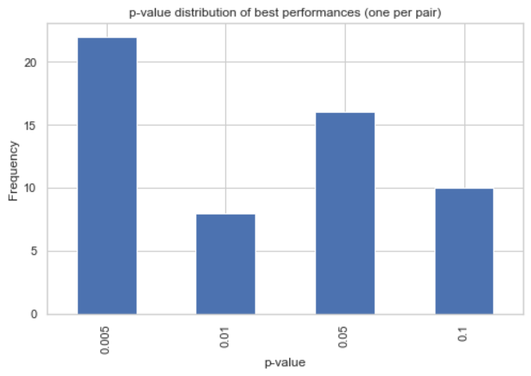


Fig. 4: p-value distribution of best performances per pair

A. Best result

From table II, we can see that the best performing pair, ADA-DOGE, did an outstanding return of more than 2e+07. We want to analyze how this could happen. In figure 8 and figure 9, we can see the two trading periods, which contributed the most to the aforementioned total monthly return. Starting with figure 8, we can see well, how the prices move very similar. However, the spread increases slowly and thus, we enter a short position on the spread between 08:30 and 09:00 with the goal that the spread goes to zero again. We keep this position open until around 10:40, where the price of DOGE makes a huge jump within less of one minute. This is a signal for us to close our position, since both prices have crossed (i.e. spread goes to zero). As described in section III, we always execute our strategy the minute after we obtained a signal in order to have a environment as close as possible to the real world, where we cannot execute a trade immediately. In this case, this is very beneficial for us, because we close our position at the perfect time which captures almost all of the price jump. After we closed our position, we immediately get a signal that we should open an inverted position, i.e. go long the spread. We do this in the next minute. And indeed, the prices cross again and we close our position. This is an example example of how a statistical arbitrage strategy should look like in an ideal scenario. From table II, we can see that the best performing pair, ADA-DOGE, did an outstanding return of more than 2e+07. We want to analyze how this could happen. In figure 8 and figure 9, we can see the two trading periods, which contributed the most to the aforementioned total monthly return. Starting with figure 8, we can see well, how the prices move very similar. However, the spread increases slowly and thus, we enter a short position on the spread between 08:30 and 09:00 with the goal that the spread goes to zero again. We keep this position open until around 10:40, where the price of DOGE makes a huge jump within less of one minute. This is a signal for us to close our position, since both prices have crossed (i.e. spread goes to zero). As described in section III, we always execute our strategy the minute after we obtained a signal in order to have a environment as close as possible to the real world, where we cannot execute a trade immediately. In this case, this is very beneficial for us, because we close our position at the perfect time which captures almost all of the price jump. After we closed our position, we immediately get a signal that we should open an inverted position, i.e. go long the spread. We do this in the next minute. And indeed, the prices cross again and we close our position. This is an ideal scenario where we benefit a lot from the lagged execution. But even if we would not consider the lagged trade execution, these trade illustrate very well how a statistical arbitrage strategy should look like in an ideal scenario.

Now, taking a look at figure 9, we see another great example of how the statistical arbitrage strategy is supposed to work. We open and close long and short positions of the spread every couple of minutes and make an outstanding return from that. Also note that the R^2 is extremely close to 1 ($R^2 = 0.988$), so we capture almost perfectly the variance of ADA using DOGE as regression variable which is clearly a desirable property as it indicates how well the one time series can explain the variance of the other time series. The figure 10 is a good example where R^2 is close to zero. In this case, the price movements of DOGE are extremely small compared to the price movements of ADA and thus, the DOGE time series is not able to explain the variance of ADA well. It would be therefore a good idea to control for R^2 for the underlying linear regression before start trading. We will come back to this point in section V.

Not all trades we did with the pair ADA-DOGE resulted in positive returns. Taking a look at 11, we see a very common situation, where we open a short position in the spread and the spread does not converge to zero in the pre-defined trading window period. In this particular case, the spread even diverged and we closed it at the end of the trading period with a loss of roughly -7.5%.

B. Worst result

We have seen in table I that the worst monthly return of any pair is -100%, i.e. we lose all of our initial investment. In this subsection, we will analyze the worst monthly performance of all pairs, namely the one of BNB-TRX, in the month of 2021-09 using a trading window of 12 hours and a p-value threshold of 0.1. From figure 12, it seems like our strategy is working fairly well. We make good returns for many trades. However, we close our open position at the end of the trading window which results in a loss and the stop loss is triggered 3 times (at times 04:12, 08:18 and 09:59). In these cases, we had opened our position just a few minutes before, however the spread evolved in the opposite direction than expected and thus we closed our position. This behavior is problematic, because it means that we will open the same position again immediately after we closed the previous one because the spread diverged even more, which is a signal for us to buy. This clearly does not make any sense and the idea of a stop loss is getting side-stepped. In section V, we analyze how this could be avoided. This effect is also strongly visible in figure 13.

Another phenomenon we observed is the bad performance of trades where the underlying R^2 is close to zero, as already discussed in section IV-A. This contributed strongly to the overall bad performance of the pair BNB-TRX in the month 2021-09. We will come back to this R^2 problem in section V, where we propose potential improvements to our strategy.

C. Rolling calibration

The calibration component is a critical aspect of statistical arbitrage strategies, which involves a large variety of hyperparameter decisions such as the amplitude of the trading window, the p-value threshold for the mean reversion test utilized as a signal for opening a trading window, the statistical test used for evaluating mean reversion of residuals, and the amplitude of the calibration window for calculating the intercept and beta coefficients. In this project, a decision was made to fix the calibration window to 12 hours and focus on two hyperparameters: the p-value threshold for the Augmented Dickey Fuller test and the trading window amplitude. The former determines, at every minute, whether a trading window should be opened or not, while the latter represents the duration for which our strategy trades, given a positive signal. The objective was to investigate the potential success of a monthly rolling calibration by implementing the strategy for all pairs, for each month, and considering 24 combinations of the p-value threshold and trading window amplitude through a grid search approach. This allowed for the examination of the performance of each combination and ultimately the feasibility of monthly rolling calibration of these two hyperparameters. The p-value threshold values considered were 0.1, 0.05, 0.01, and 0.005, while the trading window sizes considered were 30 minutes, 1 hour, 2 hours, 4 hours, 6 hours, and 12 hours. The monthly rolling calibration is a key aspect in implementing a real statistical arbitrage strategy. It involves having two machines, one that executes trades on the exchange market using the algorithm, and another that simulates the strategy on a range of hyperparameters. The results from the simulated strategy are then used to determine the best combination of hyperparameters to use for the following month. Note that this calibration scheme can also be performed at different frequencies, such as weekly. Therefore this rolling calibration could be successful if, month by month, the combination that gives the best performance on the previous month, leads to good performance for the following month. In order to analyze the results obtained we have used heat maps, one example is table **??**. The analysis of results has been done in a qualitative way, that is for each couple we have analyzed the series of heat-maps and the result obtained is clear: a monthly rolling calibration does not lead to consistent positive performance. To visualize this fact we can consider the Table III. In this table we report the performance of the previous month, for the same combination of hyperparameters for all the entries in Table II. What we can observe is that for all the entries of Table III the performance are below 1. This means that if we would have used a monthly rolling calibration strategy to select the hyperparameters to use for each month, we could have never obtained the returns reported in Table II. To get an even better understanding if this rolling calibration works, we show the first 6 month return heatmaps of the pair ADA-MATIC in figures 5-7. We can see that the highest returns in month 2021-02 and 2021-03 are obtained with the same hyperparameter combination. So, the rolling calibration would have worked in these months, however, for the other months, the optimal hyperparameters are changing, so a rolling calibration approach would lead to a poor performance. Clearly, these 6 heatmaps are only examples and we showed them for illustrative purposes, as we have observed the same behaviour for all other pairs. This observation could be explained by the type of assets considered, cryptocurrecies, which have high volatility and that within a month can be stressed heavily by several factors. Another aspect that could be the cause of this negative result is the number of hyperparameters considered, for example a possible extension of this project could involve a rolling calibration of more than 2 hyperparameters.

Pair	Month	Return	T.W. (hrs:min)	p-value
ADA_DOGE	2021-04	0.373935	04:00	0.1
ADA_SOL	2021-04	0.106204	12:00	0.005
ADA_MATIC	2021-03	0.815887	01:00	0.005
MATIC_DOT	2020-12	7.54585e-05	01:00	0.01
DOGE_MATIC	2021-01	0.336501	02:00	0.05
SOL_DOT	2020-12	0.000257917	01:00	0.1
DOGE_LTC	2021-04	0.000624725	02:00	0.005
BNB_DOGE	2021-04	0.540348	06:00	0.05
SOL_MATIC	2021-03	0.000855229	04:00	0.05
XRP_DOGE	2021-04	0.134121	01:00	0.1
LTC_DOT	2020-12	0.250872	02:00	0.005
SOL_TRX	2021-01	0.0177032	06:00	0.05
DOGE_TRX	2021-03	0.31981	04:00	0.1
ADA_TRX	2021-01	8.99006e-25	12:00	0.01
ADA_LTC	2021-01	0.344561	00:30	0.005
ETH_DOGE	2021-04	0.953357	12:00	0.01
ADA_DOT	2021-05	0.0330939	06:00	0.005
SOL_LTC	2021-04	0.146112	06:00	0.1
DOGE_DOT	2020-12	0.0600023	06:00	0.05
TRX_MATIC	2021-03	0.767946	12:00	0.05

TABLE III: Previous month top 20 performing pairs (monthly cumulative returns)

V. Improvements

We have seen in section IV that we often make a loss because we arrive at the end of the predefined trading window, even though we often had many successful trades before in the same trading period. One potential way to avoid this outcome is to stop trading after we closed one trade inside the trading window. Clearly, this does not mean that we realize a loss anymore since it is possible that we open a trade and never close it before the end of the trading period. However, with this less greedy approach, we stop after one successful trade and then wait for the next upcoming trading signal, which increases the amount of times we re-calibrate our model and eventually should result in better results.

Another advantage of this strategy is the reduced dependence on the selection of the hyperparameter known as the "trading window size." This serves as a secondary mechanism, enabling the closure of the trading window in the event of a successful trade completion or the activation of a stop loss, thus avoiding the repeated initiation of positions subsequent to the activation of the stop loss.

On top of that, this proposed "one-trade-strategy" would also be extremely useful with regards to the problem we observed with "side-stepping the stop-loss". That is, we close our position because our loss reaches the predefined stop-loss threshold. However, in many cases, the spread is still more than two standard-deviations away from its intercept in the next minute, which is why we enter a the same position we closed before. This does not only side-step the stop loss intention but it also increases the loss as we have to pay the trading fee once we close and re-open the position respectively. Using the "one-trade-strategy", we would close the trade and then re-calibrate our whole model, i.e. re-estimate the α and β and wait until we receive a new trading signal. Thus, if we close our position because of a stop-loss, we do not continue trading directly after and therefore are not exposed to "side-stepping" the stop-loss mechanism.

Another potential improvement to our strategy would be to control for R^2, that is we could pre-define a R^2 threshold which needs to be exceeded in order to generate a trading signal. We have seen in many cases that the most successful trades were done when the underlying linear regression showed a very high R^2 (i.e. close to 1). And we have also seen that in many cases, a very low R^2 (i.e. close to 0) leads to strong negative returns. Thus, if we started trading only if we observe a very high R^2 (such as > 0.98), then we could enhance our performance by focusing only on pairs which are able to explain each other sufficiently good in terms of variance.

VI. Conclusion

In this project, we analyzed the performance of a 2σ-statistical arbitrage strategy on pairs of the most liquid cryptocurrencies using 1-minute data. We have seen that the strategy performs well for some pairs, months and specific hyperparameter choices such as trading window size and p-value threshold, however, the optimal hyperparameter choices do not stay constant over time. In fact, we showed that it is not possible to predict the best-perfoming hyperparameter choice for the next month using the best-performing hyperparameter choice from previous month. This can be seen as a result of the fast changing statistical properties of the underlying spread/asset time series. However, we could observe that negative relationship between the returns and the p-value threshold, that is, higher p-value thresholds lead on average to worse performance compared to lower p-value thresholds. We also proposed potential improvements to our strategy, which go beyond the scope of this project and require further testing. Generally, it must be stated that based on the obtained results, we question the overall fit of this pairs trading strategy due to the fast-changing statistical properties of cryptocurrencies.

References

[1] Marco Avellaneda and Jeong-Hyun Lee. Statistical arbitrage in the us equities market. *Quantitative Finance*, 10(7):761–782, 2010.

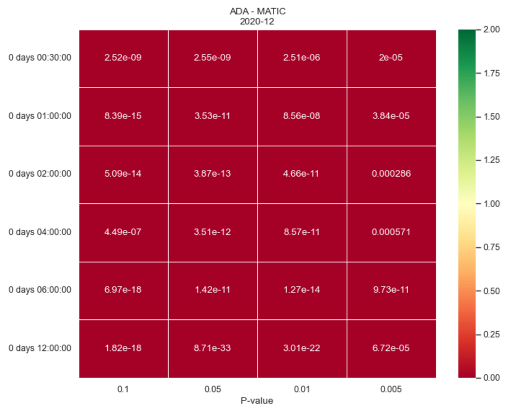
[2] William K Bertram. Analytic solutions for optimal statistical arbitrage trading. *Physica A: Statistical mechanics and its applications*, 389(11):2234–2243, 2010.

[3] David A Dickey and Wayne A Fuller. Distribution of the estimators for autoregressive time series with a unit root. *Journal of the American statistical association*, 74 (366a):427–431, 1979.

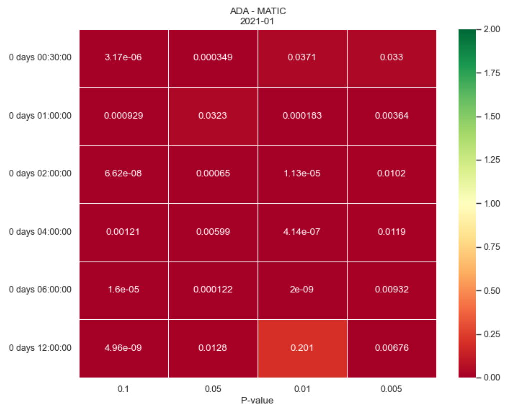
[4] David Easley, Marcos M Lopez De Prado, and Maureen O'Hara. The microstructure of the "flash crash": flow toxicity, liquidity crashes, and the probability of informed trading. *The Journal of Portfolio Management*, 37(2):118–128, 2011.

[5] Thomas Günter Fischer, Christopher Krauss, and Alexander Deinert. Statistical arbitrage in cryptocurrency markets. *Journal of Risk and Financial Management*, 12(1):31, 2019.

[6] Evan Gatev, William N Goetzmann, and K Geert Rouwenhorst. Pairs trading: Performance of a relative-value arbitrage rule. *The Review of Financial Studies*, 19(3):797–827, 2006.

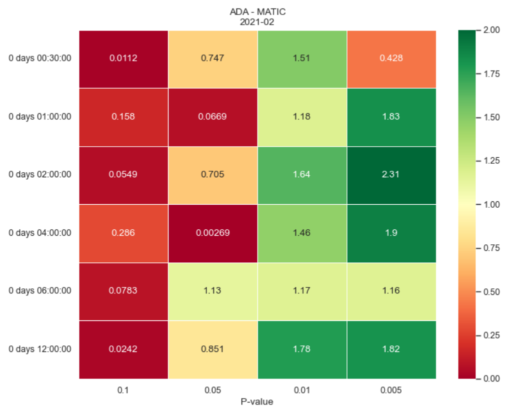


(a) ADA - MATIC returns for the month 2020-12

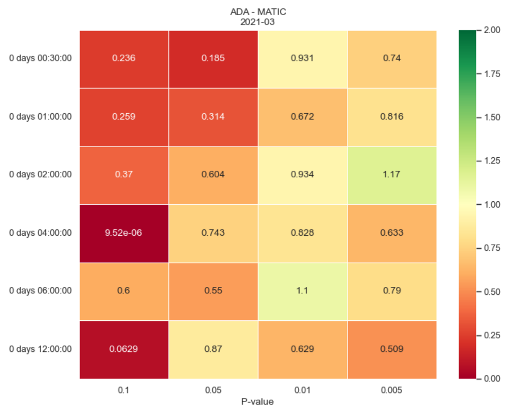


(b) ADA - MATIC returns for the month 2021-01

Fig. 5

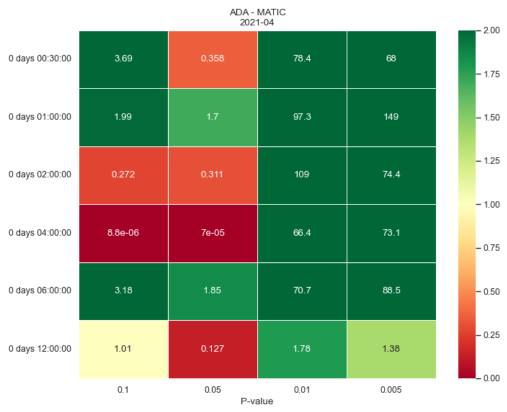


(a) ADA - MATIC returns for the month 2021-02

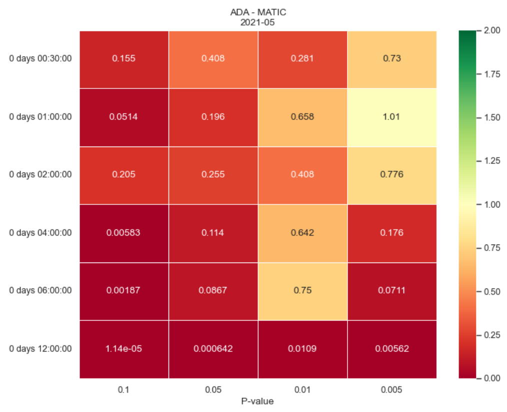


(b) ADA - MATIC returns for the month 2021-03

Fig. 6



(a) ADA - MATIC returns for the month 2021-04



(b) ADA - MATIC returns for the month 2021-05

Fig. 7

	Id	price	qty	quoteQty	isBuyerMaker	isBestMatch
2021-08-01 00:00:00.229000+00:00	227770439	1.3193	116.92	154.253	True	True
2021-08-01 00:00:00.229000+00:00	227770440	1.3193	677.25	893.496	True	True
2021-08-01 00:00:00.229000+00:00	227770441	1.3191	1600.09	2110.68	True	True
2021-08-01 00:00:00.229000+00:00	227770442	1.3191	100	131.91	True	True
2021-08-01 00:00:00.229000+00:00	227770443	1.319	358.91	473.402	True	True

TABLE IV: Tradebook data example

	opentime	open	high	low	close	vol	closetime	quotevol
2021-08-01 00:55:59.999000+00:00	1.62778e+12	1.3491	1.3493	1.3477	1.3477	87475.7	1.62778e+12	117971
2021-08-01 00:56:59.999000+00:00	1.62778e+12	1.3476	1.3485	1.3475	1.3484	174738	1.62778e+12	235540
2021-08-01 00:57:59.999000+00:00	1.62778e+12	1.3484	1.349	1.348	1.3486	60050.8	1.62778e+12	80981.3
2021-08-01 00:58:59.999000+00:00	1.62778e+12	1.3486	1.3486	1.3474	1.3479	73713.4	1.62778e+12	99358.7
2021-08-01 00:59:59.999000+00:00	1.62778e+12	1.3478	1.3484	1.3461	1.3482	197711	1.62778e+12	266390

TABLE V: Klines data example

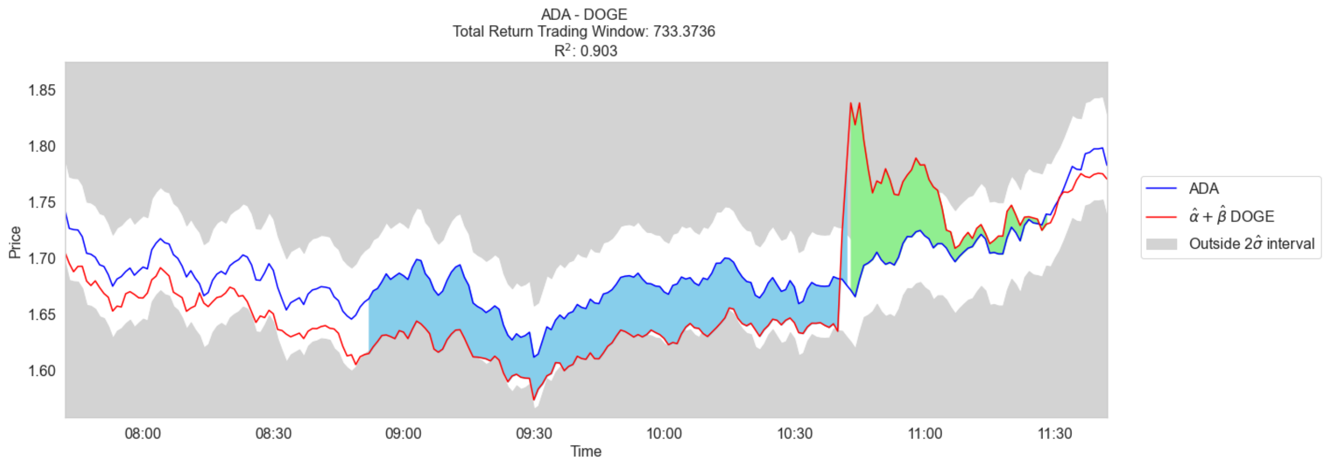


Fig. 8: ADA - DOGE example trade (Tr. window: 4 hrs; p-val threshold: 0.1)

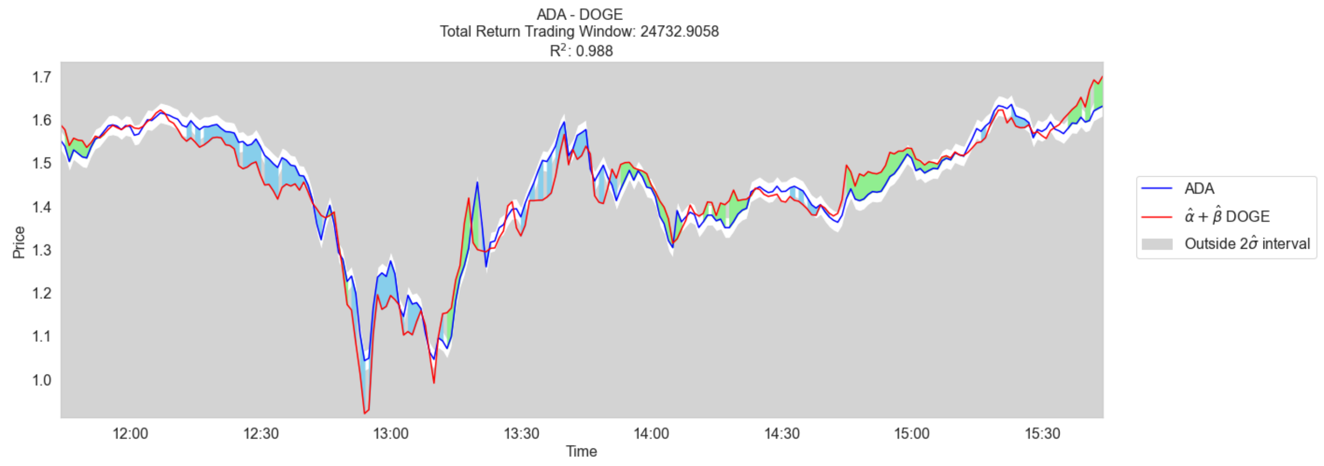


Fig. 9: ADA - DOGE example trade (Tr. window: 4 hrs; p-val threshold: 0.1)

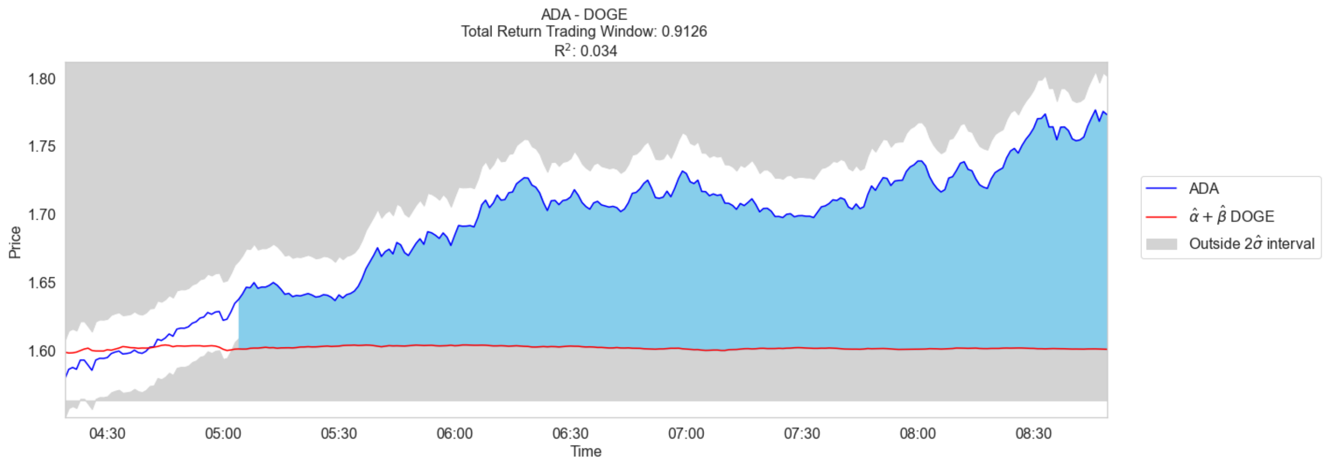


Fig. 10: ADA - DOGE example trade (Tr. window: 4 hrs; p-val threshold: 0.1)

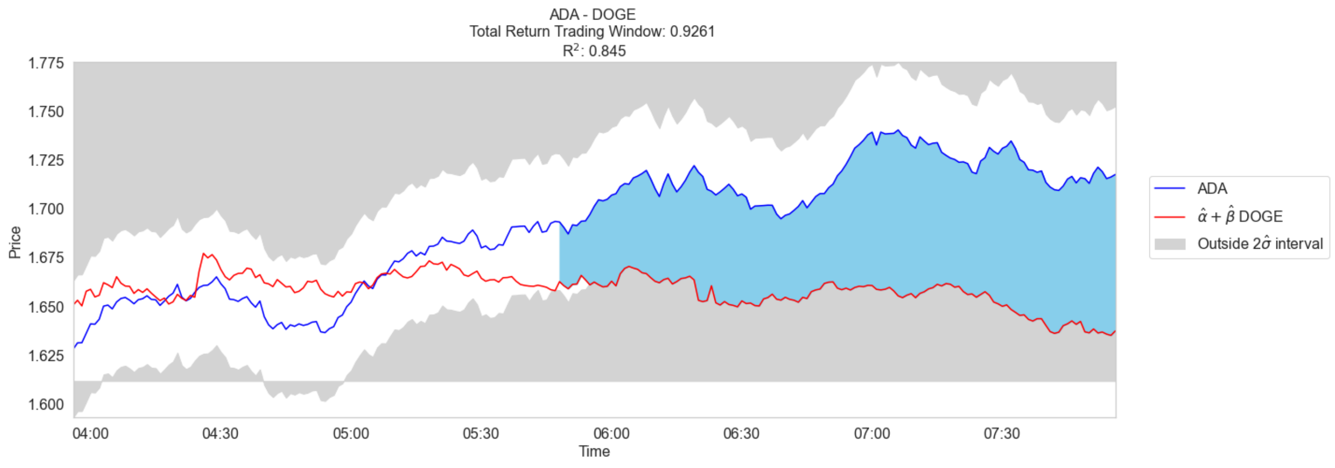


Fig. 11: ADA - DOGE example trade (Tr. window: 4 hrs; p-val threshold: 0.1)

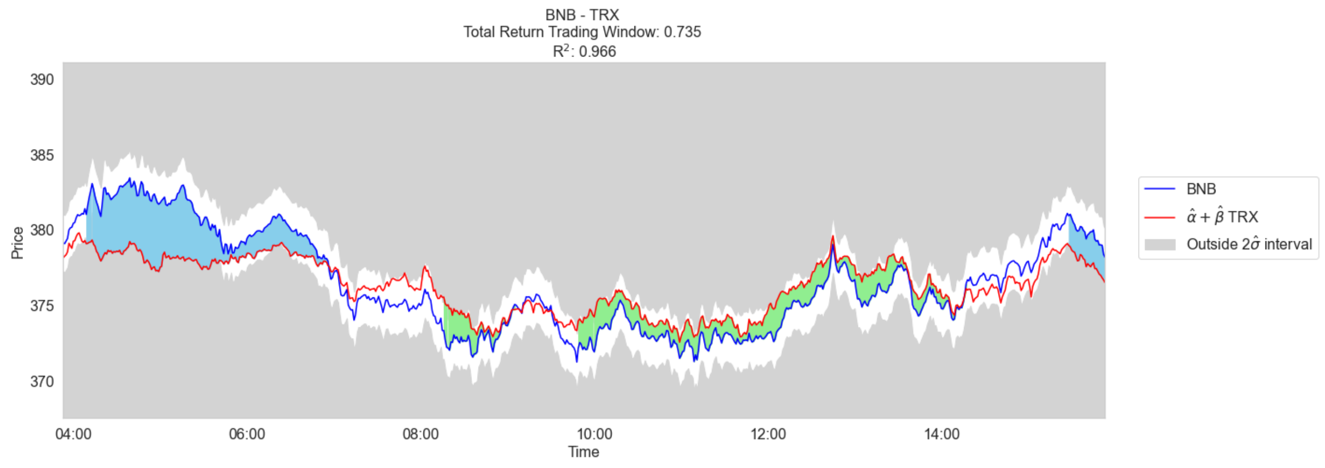


Fig. 12: BNB - TRX example trade (Tr. window: 12 hrs; p-val threshold: 0.1)

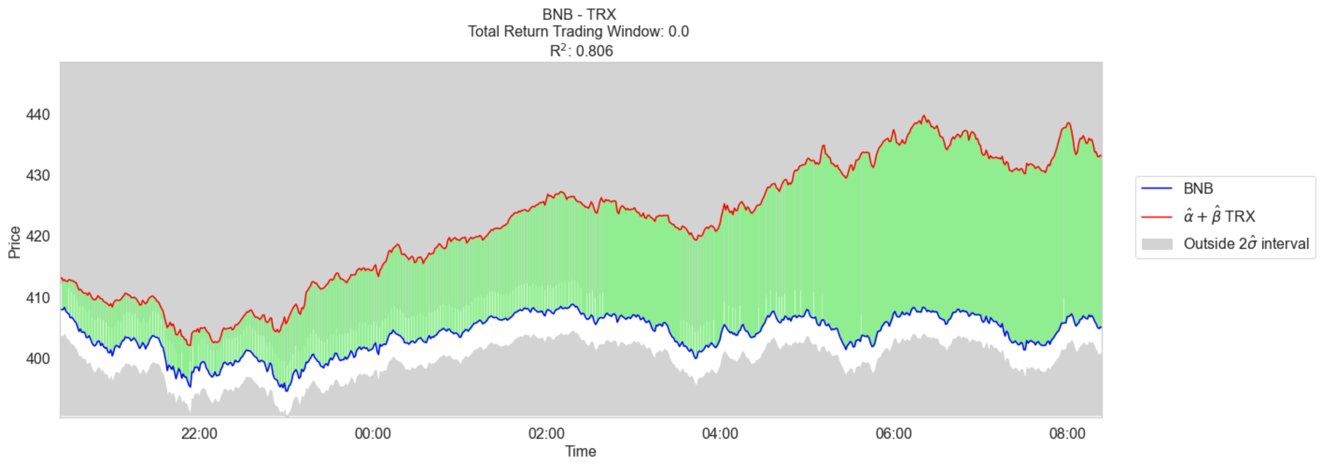


Fig. 13: BNB - TRX example trade (Tr. window: 12 hrs; p-val threshold: 0.1)