Machine Learning Engineer Nanodegree

Capstone Project

Title: Deep Reinforcement Learning for Multi-Dimensional Pairs Trading in the Brazilian stock market

Student: Tiago Monteiro Cardoso Rio de Janeiro, November 22nd, 2018

I. Definition

Project overview

In this project I propose a new automated trading strategy: a reinforcement learning agent that issues trading orders following a multi-dimensional pairs trading scheme.

Pairs Trading is an old statistical arbitrage trading strategy that uses asset pairs whose historical prices are highly correlated. That means that the prices seem to "move together". Since in the short term the correlation is usually weak, whenever the spread between the assets differs from its equilibrium level, an arbitrage opportunity appears: the most expensive asset is sold, the cheapest asset is bought and the position is undone once the spread value returns to a predetermined distance.

Recently, an expansion of the strategy - "Multi-dimensional pairs trading" or MDPT - has been proposed (1). MDPT generalizes the strategy for more than two stocks. According to (1), in order to implement it, we should first optimally model the joint distribution of returns of the group of stocks we wish to trade (in our case as in the article the simpler case of three stocks is dealt with) and measure the degrees of relative mispricing based on their joint distribution. The degrees of mispricing are expressed as conditional probabilities. So, if we are dealing with stocks X, Y and Z and if X's mispricing is 0.5, it means that X is fairly-valued relative to the other two stocks. On the other hand, a mispricing higher than 0.5 means that X is overvalued relative to Y and Z. Conversely, a mispricing lower than 0.5 indicates an undervaluation of the stock X.

Reinforcement learning is a machine learning method aiming at training an automated agent to make decisions in an environment from which it usually does not have full knowledge. The agent learns by maximizing a conveniently devised reward function. Central to every reinforcement learning implementation is the need to balance exploratory behavior (which leads to wrong choices) and exploitation of the current knowledge of successful strategies.

For a multi-dimensional pairs trading strategy to be implemented, we first need to compute and compare the conditional probabilities of the relative mispricing between the stocks, a criteria which is considerably more complicated than the condition that traditional pairs trading strategies employ to open pairs - whenever the two stocks' prices are more than a certain "distance" apart. The relevance of the strategy presented in this project lies on the fact that by employing a reinforcement learning agent we can dispense with such computations: the agent makes its decisions based only on the historical price data and on the series of returns received.

Problem Statement

In this project I tackle the problem of devising a reinforcement learning agent to execute an automated multi-dimensional pairs trading strategy. Particularly, the agent will be trained on historical records of quotations of three highly correlated stocks from the Brazilian stock market. After the training period, the agent will start to issue coupled orders: to buy the undervalued stock and sell the overvalued ones or, vice-versa, to sell the overvalued one and buy the other two (another possible action will be to maintain the current position). In the simulation we will begin with a portfolio consisting of zero shares and 100.000,00 reais (Brazilian currency). At the end of each trading day the portfolio value will be recalculated. At the end of the whole testing period, total returns will be computed and compared to two benchmarks.

Metrics

In order to evaluate our agent, we will compute, for the period under study, the portfolio's return rate obtained by the agent and compare it to the return rates obtained by the benchmarks. The return rate is given by the formula:

$RR = \frac{P_f}{P_i} - 1$, where RR is the return rate, P_i is the initial portfolio value and P_f is the portfolio value at the end of the testing period.

I postpone the discussion of the method chosen to evaluate the portfolio until section III below.

II. Analysis

Data Exploration

As described in section I above the present strategy deals with the simultaneous taking of long (bought) and short (sold) positions among a group of three stocks. I chose to work with stocks traded in the Brazilian stock market (Bovespa). I needed to choose assets that were highly correlated. In this regard every pairs trading strategy needs a preliminary step - the selection of candidates - which is the investigation of pairs that, given the respective correlation between their prices, represent good candidates for the application of the strategy. The standard procedure for that investigation is the application of cointegration tests. In the present work I skipped this preliminary step. I took advantage of the comprehensive study of the Brazilian market undertaken in (2), in which 1225 possible pairs were investigated. Of these I chose for the present study the stocks that obtained the highest Sharpe Ratio for the in-sample period. They are: ITUB4 (Itaú Unibanco Holding S.A. preference shares) and ITSA4 (Itaúsa - Investimentos Itaú S.A. preference shares). Since the present strategy relies on a multi-dimensional "pair" of three, I chose another stock from the same holding to complete the group to be studied - ITUB3 (Itaú Unibanco Holding S.A.).

Historical price data for the three stocks is public and was retrieved from <finance.yahoo.com> in csv format. Four files were downloaded, three corresponding to the stocks and one to the Ibovespa, the index of the Bovespa stock market. In the files each row of data corresponds to one trading day and there are seven columns: date, the share's open value, the highest value, the lowest value, the close value adjusted for splits, the close value adjusted for both dividends and splits and the volume of shares negotiated. The files were retrieved on 02/Oct/2018 and comprise data from 3.1.2000 to 2.10.2018.

For all the simulations in the present work the price used for buying/selling transactions was the close value adjusted for both dividends and splits.

Missing Values

Since the file for ITUB4 did not contain transaction data for the whole year 2000 and all files ended in October 2018, and since I intended the training and testing periods to coincide with calendar years, it was necessary to restrict the training and testing periods to the time window 1.1.2001 - 31.12.2017.

Outliers

As it will be apparent on the next item (Exploratory Visualization), there are periods of high volatility where the market as a whole or the assets individually may experience atypical upward or downward trends. For that matter, it was necessary that the simulations comprised diverse scenarios. In fact, the testing periods - calendar years 2011 through 2017 - include periods of market growth, decline and stagnancy.

Exploratory Visualization

In the next figure the close adjusted daily prices for the three stocks are plotted for the whole period of (2.1.2001 - 29.12.2017). On this visualization the prices are normalized relative to their initial values (on 2.1.2001), that is, they are divided by the value they had on the first day. In the simulations normalization was performed relative to the first day of the respective training period (more on that on section III below). Two characteristics here draw our attention: first, the high degree of correlation between the three stocks; second, the existence of periods of extreme behavior - chiefly during the market crash on 2008 and following its steep rebound 2016 onwards.

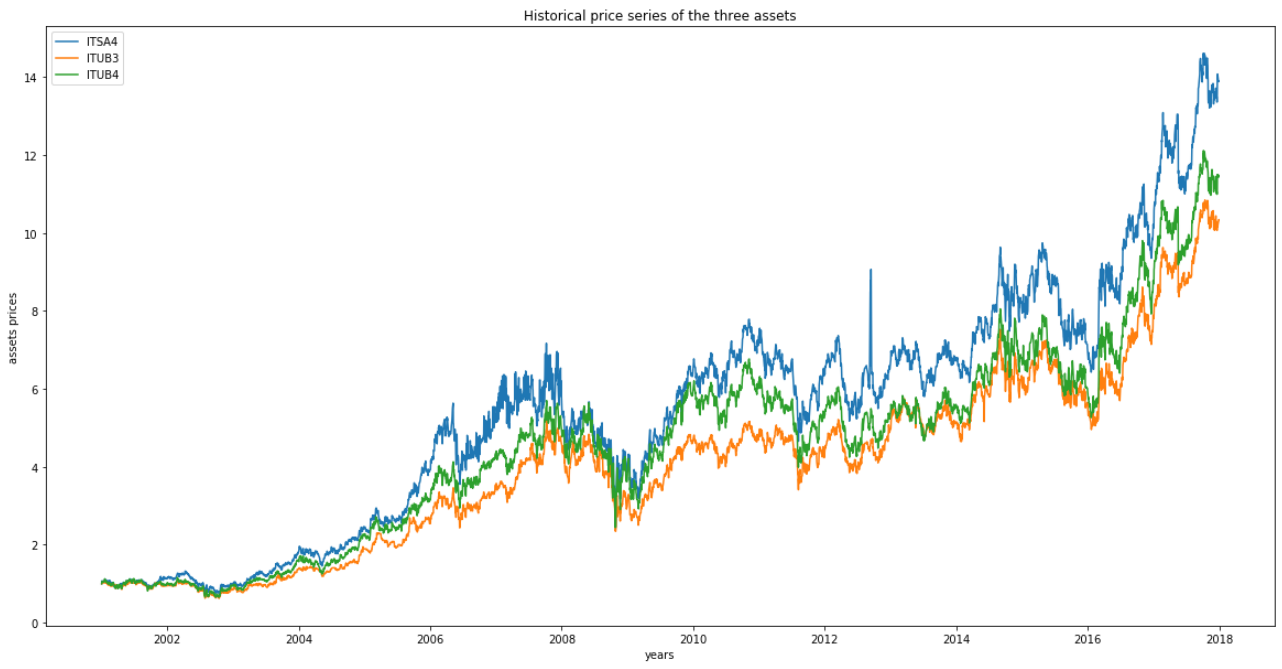

Figure 1

Algorithms and Techniques

This project consists of a reinforcement learning agent with actor and critic networks and a replay memory. At each step (trading day) the simulator will render available a new state corresponding basically to the stocks' quotations on the day. There will be 7 possible actions and we will use the daily returns to build a convenient reward function. The actor network will learn a policy function $\pi(a|s, \theta_p)$, where π is stochastic, that is, its outputs are probabilities assigned to the possible actions, and θ_P, the network parameters, are learned through gradient ascent with the goal of maximizing the long-term accumulated returns (discounted by a factor gamma). As for the critic network, it will estimate a value function $V^{\pi}(s_t)$ whose objective is the minimization of the mean-squared-error between the state's estimated value and its target value under policy π.

The algorithm to be implemented is the one suggested in (3).

The agent will control a portfolio which initially consists of 100.000 reais (the Brazilian currency) in cash and will follow the strategy outlined in (1). According to that strategy the agent will buy/sell the stock which is under/overvalued relative to the other two and will simultaneously sell/buy the other two stocks. Consequently there are in all 6 different possible operations and, in order that the strategy should remain market risk-free, the value allocated to the short position (sell) in each operation must equal the value allocated to the long position (buy). Since there is one "operation" more (i.e. neither buying nor selling), there are a total of 7 actions for our agent.

The project uses a policy gradient framework, which is especially appropriate for the continuous action space of the general case (trading whichever amount yields a higher return). However, for the moment being I decided to solve the simpler problem corresponding to a discrete action space. Thus, in each simulation step, the agent is only allowed to trade the same value (16.000 reais for the asset being bought (sold) and 8.000 reais for each of the other two assets, which are being sold (bought)), and consequently, there are only 7 actions available (as described in the previous paragraph). We will assume that transactions are free of costs. We also assume that a real market agent can make transactions using the adjusted close price.

Benchmark

At the end of the testing period, total returns will be computed and we will check whether the agent was able to outperform the three chosen benchmarks - the returns calculated with the Ibovespa index, the returns obtained by a random agent and the returns calculated by the application of Selic, Brazil's benchmark interest rate. The Ibovespa is the financial index of the Brazilian stock exchange - Bovespa - and is the standard benchmark against which Brazilian investments are evaluated. For the second benchmark, an agent will be implemented to choose randomly and with equal probability from the available actions. The third benchmark was not on the project's original proposal. However, I found that it was necessary to include it if a more realistic appraisal of the potential of the new strategy as an alternative modality of investment was needed.

III. Methodology

Data preprocessing

A module called util.py was implemented to handle the text files. An especially designed function imports the original data to a Pandas dataframe where each line corresponds to a trading day, the index is the date and the columns are the adjusted close values of each of the chosen stocks and of the Ibovespa index (which was not used in the end).

As stated on section II, item "Missing Values", the file for ITUB4 did not contain transaction data for the whole year 2000, thus it was necessary to restrict the training and testing periods to the time window 1.1.2001 - 31.12.2017. Also, since sometimes one of the stocks' files lacked trading information for a day or other, for simplification purposes, I required that the importing function should remove from the dataframe any lines that contained null or invalid data. Thus, only the days for which full data (for the three stocks and for the Ibovespa index) was available were considered in the simulations.

Implementation

The present work builds upon a previous and simpler project, which aimed at applying reinforcement learning to traditional pairs trading, the "Deep Reinforcement Learning for Pairs Trading using Actor-critic", in short RLMDP (4). I took inspiration from it in order to write the market simulator and the RL agent. In addition I employed the same neural network architecture they proposed, for which I give them due credit.

Overview of the project's modules

This project, named "Deep Reinforcement Learning for Multi-Dimensional Pairs Trading" or DRLMPT, is composed of 5 modules - runner, runner_random, util, actor_critic_agents and simulator, along with the data files.

The first module - "runner" - is the main one. It creates instances of both the market simulator and the agent and keeps requiring the agent to make trading decisions while the final date of the requested period is not reached. The module "runner_random" instead simulates an agent who makes random action choices and, for that reason, it does not need to create an instance of the RL agent. The module util has a single function "get_prices" which is called by the simulator and is charged with reading the data files and saving the relevant information (the adjusted close prices of each stock for the required date range) to a Pandas dataframe. Since the next two modules are the central pieces of DRLMPT, I dedicate a subsection to each.

The agent

The module "actor_critic_agents" is the implementation of the agent while the module "simulator" goes through the data, supplies the agent with the daily stocks' prices, controls the opening and closing of positions and computes the portfolio value and the reward function.

The agent follows the algorithm outlined in (3), from which I extensively quote in the following. The model consists of state space S, basically consisting of the recent history of stocks' prices, a policy π: $S \rightarrow \mathcal{A}$ that governs the choice of action and a value function $\mathcal{V}$: $S \rightarrow \mathbb{R}$ under policy π. The action space $\mathcal{A}$ consists of the seven actions the agent is allowed to take (opening or closing (buying or selling) positions for each of the three stocks plus holding (i.e. neither buying nor selling anything)). The agent's goal is to maximize the long-term accumulated return $\sum_{k=0}^{\infty} \gamma^k r_k$, where the r_k compose the series of received rewards and the discount factor gamma was set to 0.9 throughout the simulations.

The agent is in principle stochastic since the output of policy function π are probabilities attributed to the seven possible actions. In this regard an important modification was introduced in the present project. Since the simulations were projected to last eleven years, of which the first ten years were the training period and the eleventh year, the testing period, the agent was implemented in a way that it followed the stochastic policy π in the beginning and became almost entirely deterministic in the eleventh year. For that matter a special function epsilon was implemented:

$$epsilon = \frac{1}{\left(\frac{day}{2400}\right)^8 + 1}$$

where "day" is the numbers of days elapsed since the beginning of the simulation. The seven probabilities calculated by the actor network are each multiplied by epsilon/7, with the exception of the largest probability, which is multiplied by a factor of (1 - epsilon + epsilon/7). The function epsilon decreases very slowly during the first 8 years but falls abruptly afterwards (each year has circa 240 trading days). Upon entering the eleventh year, the simulation has crossed the 2400-day mark and epsilon is bellow 0.5. That means that the policy becomes in practice deterministic during the eleventh year (which is the testing period). Illustrated below is the profile of epsilon function:

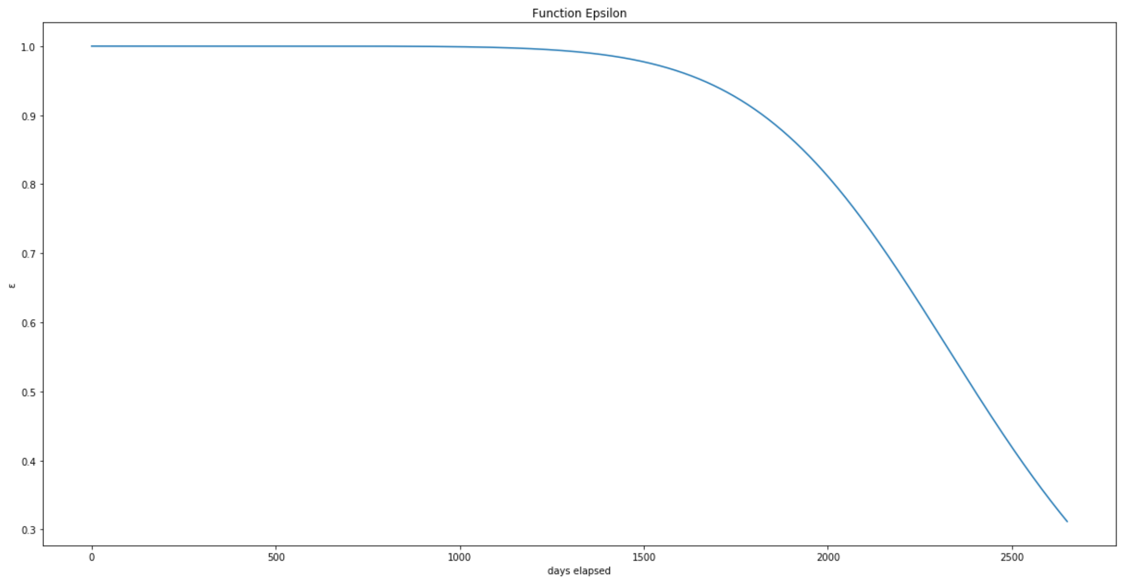

Figure 2

The actor network approximates the policy function and learns $\pi\ (a|s, \theta_p)$ through gradient ascent, whose goal is the maximization of long term reward. In the present algorithm this is achieved by updating the parameters of $\pi\ (\theta_p)$ in the direction of

$$\nabla_\theta \log \pi(a_t|s_t; \theta)(r_t + \gamma V^\pi(s_t) - V^\pi(s_{t+1}))$$

where the value function $V^\pi(s_t)$ is estimated through the minimization of the mean-squared-error between the state's estimated value and its target value under policy π. This is accomplished by a separate neural network with structure analogous to the actor's. In order to stabilize the critic network, a target network is charged with computing the value of $V^\pi(s_{t+1})$ (3).

Since the state of the environment is basically a time series of prices, two recurrent neural networks (RNN) were used to approximate the actor's policy function and the critic's value function under the policy (3). Each state s_t, which is the input to RNN, is a series of historical features comprising the latest 50 days. The daily features included in the series are simply the differences between the price of each stock and the mean value of the three stocks (the prices were normalized relative to the respective prices on the first day of the simulation). The output of policy function, as already commented, is a probability distribution for the actions.

A replay memory saves recent states, actions and rewards and a batch of groups of state, action and reward is randomly sampled from it in order to update the actor and critic networks. This technique is called "experience replay" (5) and is used to stabilize the training process (3).

The simulator

Central to the simulator is its function "step" which takes the action chosen by the agent for the current market day and returns a reward and the next state (based on next day's stock prices). The first task of the function step is the opening or closing of positions in the three stocks being traded. Here a remark is needed. In traditional pairs trading we have two stocks, say A and B, and accordingly there are only two possible scenarios for the opening and closing of positions. Either we open a position by buying A and selling B and close it by the reverse operations, i.e., selling A and buying B or we open a position by selling A and buying B and close it accordingly. In Multi-dimensional pairs trading things get a bit complicated. Here as before every operation consists of coupled transactions of equal value and opposite direction but not necessarily the positions in the three stocks will be closed simultaneously.

Let us make an example. On the first day of simulation there are still no operations opened. Let us suppose that the agent choses action "buyA". It means that the simulator shall buy 16.000 reais in A shares and sell 8.000 reais in B shares and 8.000 reais in C shares. Note that each lot of 8000 reais in shares counts as one opened position. Thus, at the end of day one the simulator will consider that there are two long positions opened for A (bought) and one short position opened for each of B and C (i.e., sold). If, on the next day, the agent chooses again "buyA", the simulator will accordingly open further positions for the three stocks. If, on the other side, the choice were "sellA", which is the exact inverse of "buyA", the simulator would close all the opened positions, since it would sell the two lots of A shares that are held in the portfolio and would buy in reverse the corresponding lots of B and C shares that had been previously sold. The specificity of multi-dimensional pairs trading is that not necessarily two symmetrical operations will follow each other. So, turning again to our example, suppose that the action chosen for day 2 is "buyB", that is, buying 16.000 reais in B shares and selling 8.000 reais in each of the other two assets. In that case, the simulator would close one of A's long positions, would close B's short position and open one long position for it, and, finally, would open an additional short position for C. Table 1 below summarizes the values computed in the beginning and at the end of days 1 and 2. It also displays the values of the variables used by the simulator to keep track of the positions ("longA", "longB" and "longC"):

Day	Action	Shares of A	Shares of B	Shares of C	A positions	B positions	C positions	longA	longB	longC
0	None	0	0	0	0	0	0	0	0	0
1	buyA	2 bought lots	1 sold lot	1 sold lot	opened 2 long	opened 1 short	opened 1 short	+2	-1	-1
2	buyB	1 bought lot	1 bought lot	2 sold lots	closed 1 long	closed 1 short and opened 1 long	opened 1 short	+1	+1	-2

Table 1

Whenever a position is opened, the simulator computes its cost and deducts it from the available portfolio cash. The cost is simply 8.000 reais for each operation opened (in either direction). Whenever a position is closed, the simulator computes its return which is then added to the portfolio cash. Turning back to stock A in our example, recalling that 2 lots were bought on day 1 and one lot was sold on day 2, and supposing that the price of A was 10,00 on day 1 and 11,00 on day 2, we would have:

Day	Action (A's portfolio)	Resultant number of shares in A's portfolio	Cost (R$)	Return (R$)	Cash (R$)
0	None	0	0	0	100.000,00
1	2 lots bought	1.600 (2x R$ 8.000,00 ÷ R$ 10,00)	16.000,00	-	84.000,00
2	1 lot sold	800	-	8.800,00 (800 shares x R$ 11,00)	92.800,00

Table 2

Next, the simulator computes the reward, which in an earlier version of the project was taken to be the net return of the operation, that is, the difference between the value of the portfolio after and before the action taken. After many simulations, I found it necessary to introduce two important modifications. The first was a change in the way the portfolio is evaluated (more on this on the item "Measurement" below). The second change was the employment of the hyperbolic tangent function and the rescaling of the net return. The reward function became:

$$reward \ = \ \tanh\left(100\ \frac{P_a - P_b}{C_i}\right)$$

where P_a, P_b and C_i are respectively the value of the portfolio after the chosen action, the value of the portfolio before the chosen action and the initial amount of cash.

Also, the simulator's function step calls another function, called "boundary", which computes possible restrictions to be imposed on the agent for the next choice action. The restrictions correspond to two parameters - "max_positions" and "min_return" - which put limits to the agent's ability to open new positions and to close existing ones. The need behind parameter "max_positions" is two-fold: on the first place, it is the way I found to "let the agent know" that the resources are finite and that it is not allowed to run into debt (negative value of the cash account); on the second place, it helps us optimize the agent's behavior vis-à-vis the time frame of the experiment (i.e., one year), for we want the agent to conclude its operations inside the time frame without leaving too many positions opened in the end. The parameter is another important feature that was added in the course of the simulations. In the earlier versions of the project (as described in the project proposal) infinite loans were allowed. Afterwards I came to the conclusion that they should be avoided. An increasingly negative cash account means increasing transaction costs. And since we are not computing transaction costs in this project, allowing the agent to incur in debts would produce artificially inflated results. The parameter was set to 6 in the simulations, which means that the agent was not allowed to keep more than 6 opened positions in any share and in any direction (long or short). As for the parameter "min_return", it forbids the closing of operations with returns inferior to the its value. In a real market application "min_return" may be set in a way so that transaction costs may be taken in account. In the present simulations it was set to zero, which means that it simply precludes the closing of operations with loss. I recall once more that in multi-dimensional pairs trading the operations are not necessarily symmetrical and the agent may at the same time open positions in some stock while closing positions in others.

Finally, the simulator's function "step" returns the reward, the next state, the number of the present day and the restrictions, if any.

Measurement

As already noted by Gatev, Goetzmann and Rouwenhorst in their classic study on pairs trading, " the calculation of the excess return on a portfolio of pairs is a nontrivial issue" (6). A first precaution would be to measure only the returns of the pairs trading strategy, not eventual gains from the increase in value of portfolio items. For this reason, I decided that 1) the portfolio should not consist of the stocks being traded, but only of cash, which is not remunerated, 2) the stocks bought and sold are only held for as long as the positions are kept open, and 3) the stocks held in portfolio are evaluated by their price of acquisition (see simulator's function "port_value"). This leads to two consequences: first, the agent's behavior is not influenced by changes in the market value of the assets: since the reward function depends on the portfolio's value, it would be affected if the stocks held in portfolio were evaluated by current market prices. The second consequence is that there are no gains or losses computed for any positions that remain open at the end of the testing period. In other words we treat those positions as if they had not been opened in the first place, which in turn means that the calculated excess returns with this method are underestimated.

This method of calculation is another important modification that was implemented in the course of the simulations. In the earlier versions of the project the portfolio stocks were evaluated by current prices (see simulator's function " port_value_for_output"). I made many trials with this method before discarding it. I suspect it lead the agent to treasure the winner stocks. It made money but not with the intended pairs strategy.

A second necessary measurement precaution is to avoid both capital underinvestment and capital overinvestment. Since the portfolio consists basically of cash, it is important that at any time a reasonable number of operations are opened so that all or most of the capital is invested, not idle. However, another situation is equally possible, that is, when the agent runs into debt (negative cash account) in order to open new positions (what I called "overinvestment"). In order to avoid both bad scenarios, two parameters of the model had to be carefully adjusted: the volume of each operation, which is the amount (in Brazilian reais) of shares bought/sold in each position being opened; and the maximum number of operations that the agent is allowed to keep open at the same time. In the simulations I set the first parameter to 8.000,00 (R$) (remember that the initial portfolio is 100.000,00 (R$) in cash); while the second parameter was set to 6. Those settings guaranteed that the average idle cash (averaged for all the days of the testing period and for all runs of simulation) was close to but not bellow zero. Finally, in order to compute the final percentage return, the initial invested capital was considered to be the initial cash (R$ 100.000,00) minus the average idle cash.

Each testing period lasts one entire calendar year and is preceded by a 10-year training period. Since the collected data ranges from mid-2000 to October 2018, a total of 7 testing periods could be investigated: from 2011 to 2017. For each one of them I made 20 runs of the DRLMPT agent. The final return for each year is the average of the returns of the 20 runs. Analogously there were 20 runs for each year for the random agent. The results of all runs are registered in text files, the output of the simulator module.

IV. Results

Model Evaluation and Validation

My main concern throughout this project was to build a reinforcement learning system able to be profitable in a real market application. It was not an easy task. I had to try many different functional forms for the reward function as well as different values for some key parameters like the amount of cash traded in each operation. Also, in the preceding subsections I outlined how difficult it is to measure the excess returns of a pairs trading strategy and explained how measurement requirements were the reason behind crucial modifications made to the earlier versions of this project.

In order to calibrate the model's parameters I worked with only two training periods - 2001 to 2010 and 2006 to 2015 - which corresponded to testing periods 2011 and 2016 respectively. My first intention was to restrict this project to a demonstration of the method in three years, i.e., three testing periods. However, early in the process it became clear to me that, in order to avoid potential data snooping concerns I would better apply the model to every single testing period afforded by the available data. As already explained on section III, subsection "measurement" there were a total of 7 testing periods - years 2011 through 2017. Another reason for the extension of the evaluation to more testing periods was the need to deal with perturbations of the input space, like periods of unusual highs or lows.

As a matter of fact, as seen on section II, subsection "Exploratory Visualization", the stocks used in this project display diverse behaviors throughout the interval of testing periods used: thus, there were periods of market stagnation or decline, but the interval also included the momentous crash year of 2008 and two post-Brazilian crisis years of steep rebound (2016 and 2017).

The simulations showed that the model generalized well to all testing years (cf. subsection "Results and Justification" below). A particular characteristic of the model, on which I will dwell longer on section V, is its robustness: the model yielded returns in the same, stable level throughout periods in which the Brazilian market experienced extreme volatility.

Results and Justification

The output files of all simulations were collected in the attached folder "results". In addition the results of each simulation along with the calculation of its returns are included in the appendix at the end of the present report.

The DRLMPT model's results were compared to three benchmarks - the Ibovespa index, the Selic (Brazilian basic interest rate) and the returns obtained by a random agent - in the seven testing years. The model beat the random agent in all years, the Selic in four years and the Ibovespa in five years. The model's return rates and those of the benchmarks are listed on table 3 below. I highlighted in red only the benchmarks/periods which were able to beat the model.

YEAR	RETURN (%)			
	DRLMPT	IBOVESPA	SELIC	RANDOM
2011	11.70	-5.26	11.62	-0.88
2012	15.56	-5.25	8.48	-1.45
2013	7.43	-20.28	8.21	0.46
2014	13.53	-1.54	10.91	0.37
2015	8.54	-13.86	13.29	0.53
2016	18.20	60.05	14.03	0.05
2017	6.09	31.30	9.96	1.43

Table 3

The results are encouraging particularly if we bear in mind that the expressive performance of the Ibovespa index in 2016 and 2017 represents the rebound of a more than 5-year low. For this reason, maybe a more insightful analysis should take into account not the separate annual returns, but the cumulative ones. In the next figure, the cumulative returns of the DRLMPT and the benchmarks for the seven years are displayed.

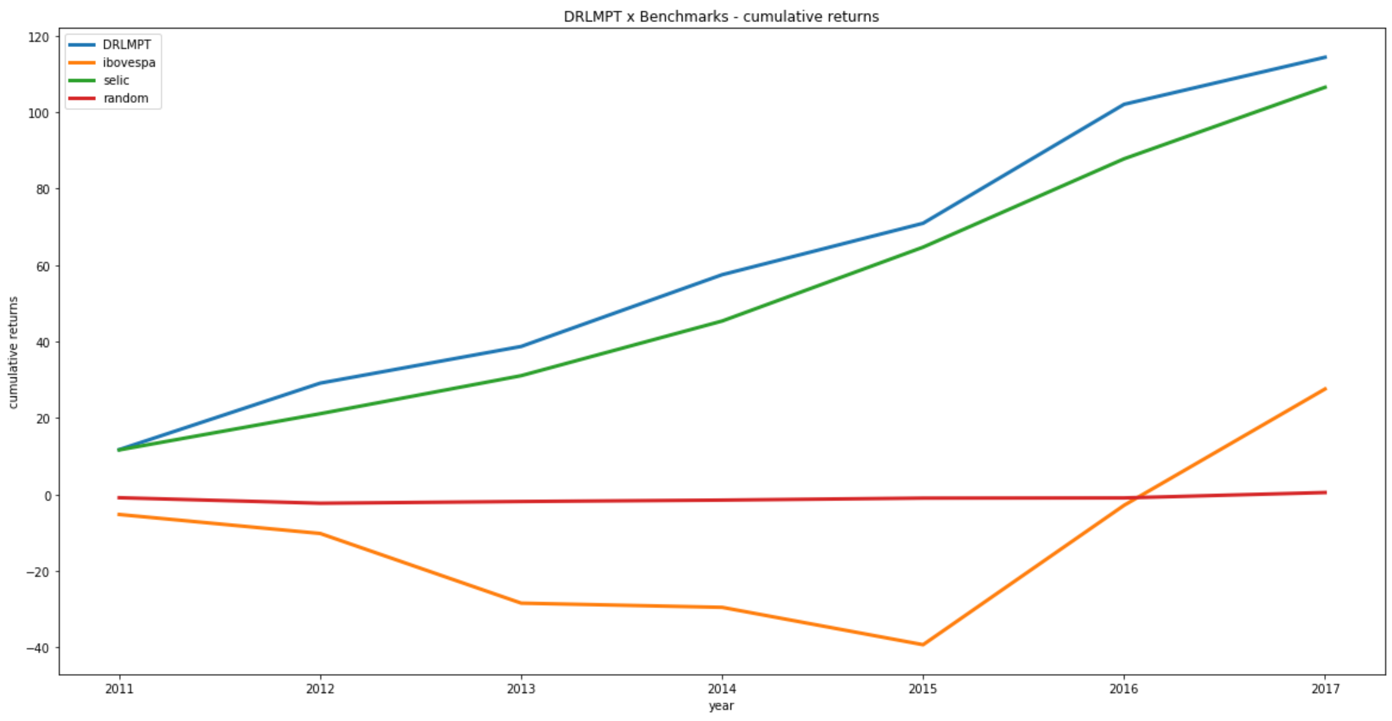

Figure 3

As can be seen, DRLMPT beats all benchmarks in all cumulative periods up to the whole 7-year period.

DRLMPT has displayed exceptional results as data make clear. Particularly it was able to maintain a steady rate of returns throughout a period of high market instability as is dramatically illustrated by the Ibovespa line (the orange one) on the graph. However, a broader selection of testing periods is needed before a finer evaluation of the method can be reached.

V. Conclusion

Reflection and Free-form Visualization

In this project I developed an automated trader system which employs a reinforcement learning agent and is based on a multi-dimensional pairs trading strategy. The main goal was to build a system that could be profitable in a real market situation. As was commented on section IV, subsection "Results and Justification", the DRLMPT agent was able to beat all three chosen benchmarks in the analysis of cumulative returns. However, in order to demonstrate profitability in real market conditions, we would still need to compute transaction costs, something that was out of the scope of the present work.

I would like to draw attention to a significant characteristic that DRLMPT displayed in the simulations. As mentioned before, the Brazilian market displayed considerable instability in the years 2011 - 2017, with the Ibovespa index scoring anywhere between -20% and +60%. Notwithstanding the instability, the agent maintained a stable level of returns throughout the period as illustrated in the graph below:

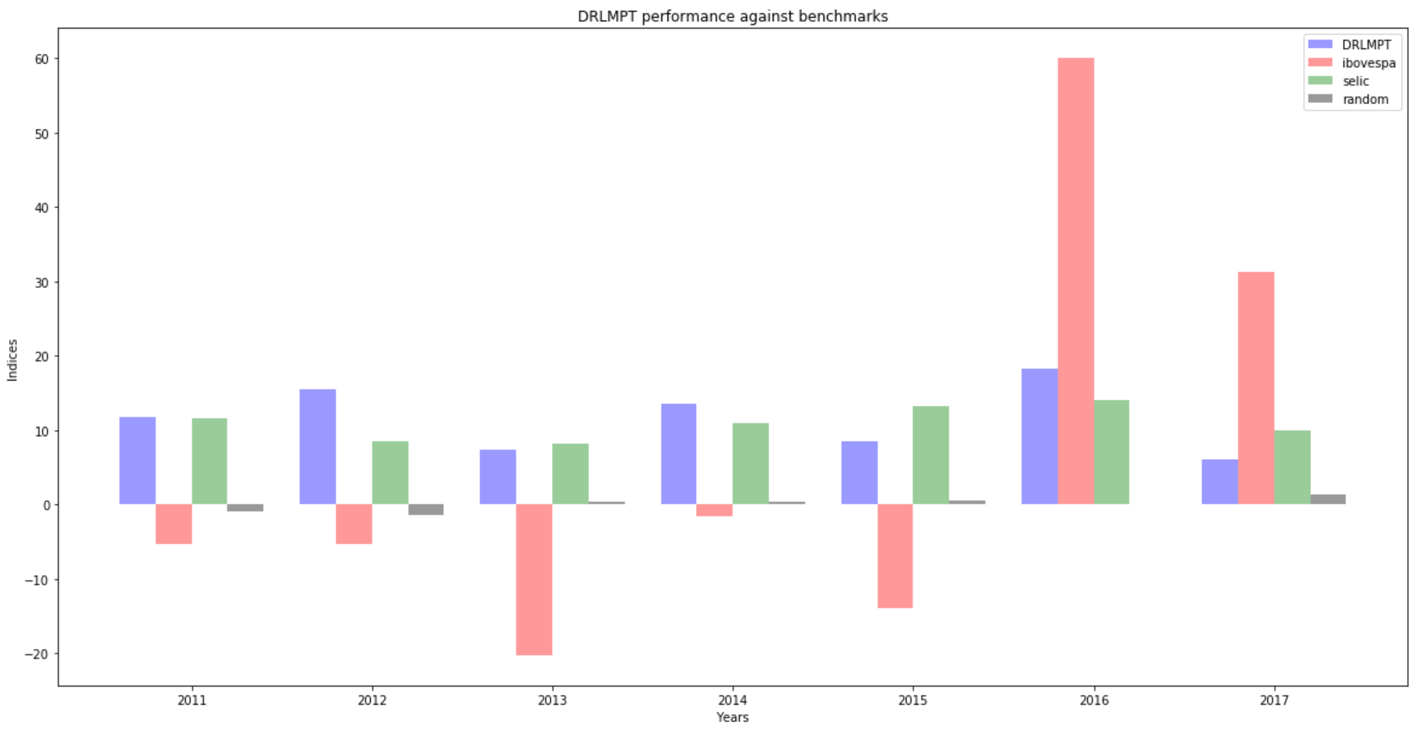

Figure 4

The returns made by the DRLMPT agent are thus shown to be uncorrelated with market movements. This is an essential property of the classic pairs trading strategy that is here demonstrably present in its multi-dimensional counterpart as well: both strategies are able to make money either in bull or bear markets.

Improvement

There are two main directions in which the present project can be improved. The first one would be the implementation of a continuous action space - one in which the transaction value were not fixed but a real number chosen by the agent from a pre-specified interval. The second improvement would be the inclusion of transaction costs. The two improvements would approximate the agent to a real market agent and would enable us, as a consequence, to make a more realistic estimate of returns.

References

(1) Lau C A, Xie W, Wu Y. *Multi-Dimensional Pairs Trading Using Copulas*. Retrieved from < http://www.efmaefm.org/0EFMAMEETINGS/EFMA%20ANNUAL%20MEETINGS/2016-Switzerland/papers/EFMA2016_0390_fullpaper.pdf> in 02/10/2018.

(2) Caldeira J, Moura, G V, *Selection of a Portfolio of Pairs Based on Cointegration: A Statistical Arbitrage Strategy*. Available at SSRN: https://ssrn.com/abstract=2196391 or http://dx.doi.org/10.2139/ssrn.2196391.

(3) Shen Y, Zhao Y. *Deep Reinforcement Learning for Pairs Trading Using Actor-critic*. Retrieved from < > in 04/10/2018.

(4) https://github.com/shenyichen105/Deep-Reinforcement-Learning-in-Stock-Trading

(5) Lin L-J. *Self-improving reactive agents based on reinforcement learning, planning and teaching*. Machine learning, 8(3-4):293–321, 1992.

(6) Gatev E, Goetzmann W N, Rouwenhorst K G, Pairs trading: Performance of a relative-value arbitrage rule. Review of Financial Studies 19(3):797-827 (2006).

Appendix

I here collect the results of DRLMPT and of the random agent for every simulation run. There were 20 runs of the program for each agent and for each testing period. On the following tables there are 3 columns of data for each agent: "Cash", which records the average value of the cash account during the testing year, "DRLMPT" (or "Random"), which is the value of the portfolio at the end of the testing period, and "Rate", which is the return rate for the run and is calculated with the formula:

$$Rate = 100\frac{DRLMPT - 100.000,00}{100.000,00 - CASH}$$

(the above formula is valid for the DRLMPT agent. For the random agent we substitute the values of column "Random" for "DRLMPT"). The final rate is just the mean value of the rates obtained in all runs.

Testing period: 2011 (Training period: 2001 - 2010)					
Cash	DRLMPT	Rate	Cash	Random	Rate
-2384.54	106103.92	5.96	-265674.51	95371.01	-1.27
7274.22	109754.99	10.52	-57208.55	92486.67	-4.78
13918.21	117687.37	20.55	-46777.65	100474.09	0.32
5485.93	100015.14	0.02	-462345.75	95511.71	-0.80
5485.93	100015.14	0.02	-61312.64	102858.53	1.77
10821.46	111383.70	12.77	-206129.01	107987.55	2.61
8882.98	113236.45	14.53	-101136.84	101955.69	0.97
10164.88	116431.80	18.29	-136593.97	113710.86	5.80
13918.21	117687.37	20.55	-144212.49	101654.37	0.68
13712.29	118526.37	21.47	-82564.16	94199.61	-3.18
9647.21	114094.40	15.60	-68781.59	101087.80	0.64
1594.03	103396.96	3.45	-178152.79	99413.48	-0.21
11799.17	114816.19	16.80	-325486.29	96464.15	-0.83
7658.65	116092.78	17.43	-246161.21	102477.84	0.72
7274.22	109754.99	10.52	-44338.69	92714.64	-5.05
5485.93	100015.14	0.02	-47451.25	94734.26	-3.57
13918.21	117687.37	20.55	-84400.65	88340.03	-6.32
-1345.85	103491.36	3.44	-166527.88	109647.18	3.62
5485.93	100015.14	0.02	-15454.31	90241.03	-8.45
13825.37	118512.63	21.48	-58639.32	99528.45	-0.30
	Average	**11.70**		**Average**	**-0.88**

Testing period: 2012 (Training period: 2002 - 2011)

Cash	DRLMPT	Rate	Cash	Random	Rate
8122.28	118444.59	20.08	-41471.40	102360.81	1.67
12323.79	115298.25	17.45	-28443.13	96008.46	-3.11
-3560.25	107116.51	6.87	-225007.24	78301.77	-6.68
15598.81	114640.02	17.35	-91594.58	97974.14	-1.06
4398.96	112353.42	12.92	-49171.59	100924.39	0.62
14498.01	113020.29	15.23	-320944.00	86312.84	-3.25
9342.33	112058.67	13.30	-226496.88	89406.39	-3.24
14572.36	119279.56	22.57	-104079.96	113577.41	6.65
10660.12	110323.41	11.56	-309231.13	75724.75	-5.93
8122.28	118444.59	20.08	-82578.36	94717.92	-2.89
-3346.23	107263.78	7.03	-68505.50	98188.16	-1.08
7930.55	118863.74	20.49	-1306.68	109114.72	9.00
8983.85	118785.15	20.64	-150164.80	96907.83	-1.24
10773.91	114661.02	16.43	-144575.27	91178.33	-3.61
12956.83	119414.40	22.30	-243233.99	110460.02	3.05
12388.90	110977.16	12.53	-25066.54	94747.71	-4.20
3220.69	114429.26	14.91	-75702.76	88983.84	-6.27
10189.74	110215.96	11.38	-288382.43	86512.80	-3.47
11295.39	115199.53	17.13	-131327.57	96360.95	-1.57
2135.14	110676.67	10.91	-276866.45	91144.33	-2.35
	Average	**15.56**		**Average**	**-1.45**

Testing period: 2013 (Training period: 2003 - 2012)

Cash	DRLMPT	Rate	Cash	Random	Rate
5233.29	103732.76	3.94	-73561.03	103064.65	1.77
1999.11	104551.82	4.64	-47815.22	99787.06	-0.14
5493.11	107653.56	8.10	-31424.80	101936.97	1.47
3803.39	107713.41	8.02	-816.14	97932.11	-2.05
5621.04	109250.08	9.80	-261398.96	98711.87	-0.36
5621.04	109250.08	9.80	-76593.19	97955.49	-1.16
9147.09	107209.68	7.94	-74877.51	101382.43	0.79
5648.39	104640.28	4.92	-249513.79	103072.70	0.88
11123.80	109203.04	10.35	-230954.98	106936.14	2.10
9860.68	109090.52	10.08	-160231.67	104742.24	1.82
3696.28	107731.78	8.03	-260293.10	103306.57	0.92
11524.44	108989.85	10.16	-289809.72	105845.83	1.50
5474.95	108012.25	8.48	-16060.26	99552.41	-0.39
2130.77	104788.07	4.89	-80859.95	86582.75	-7.42
5233.29	103732.76	3.94	-40925.46	102997.17	2.13
2195.40	105835.44	5.97	-37396.03	106480.01	4.72
4722.85	108234.03	8.64	-72773.61	103431.04	1.99
3035.58	106150.79	6.34	-77879.50	102283.52	1.28
2130.77	104788.07	4.89	-17825.79	100329.73	0.28
5107.02	109174.74	9.67	-216133.99	97367.70	-0.83
	Average	**7.43**		**Average**	**0.46**

Testing period: 2014 (Training period: 2004 - 2013)

Cash	DRLMPT	Rate	Cash	Random	Rate
11356.67	113586.22	15.33	-121975.20	98837.90	-0.52
6838.26	115480.83	16.62	-11079.50	98831.96	-1.05
11038.55	115592.40	17.53	-170277.06	111927.81	4.41
1933.69	103756.35	3.83	-264774.79	75770.70	-6.64
3628.66	109261.13	9.61	-71487.04	101510.81	0.88
7285.18	108499.99	9.17	-54388.75	92168.90	-5.07
10954.09	110516.83	11.81	-91380.97	93141.70	-3.58
2713.51	109349.05	9.61	-225753.83	117397.83	5.34
11294.19	117795.73	20.06	-39376.98	103225.82	2.31
7285.18	108499.99	9.17	-101945.32	101555.86	0.77
7891.56	109340.21	10.14	-209696.65	79418.17	-6.65
11356.67	113586.22	15.33	-300818.22	86654.61	-3.33
11913.90	110350.96	11.75	-390618.68	105718.83	1.17
11356.67	113586.22	15.33	-99661.77	104831.13	2.42
4332.64	107655.76	8.00	-161906.97	107693.62	2.94
6838.26	115480.83	16.62	-197777.58	135574.65	11.95
6417.44	117516.76	18.72	-281066.38	100368.75	0.10
6838.26	115480.83	16.62	-83702.24	99191.07	-0.44
10344.10	116870.21	18.82	-55404.00	97327.86	-1.72
6838.26	115480.83	16.62	-105813.15	108591.10	4.17
	Average	**13.53**		**Average**	**0.37**

Testing period: 2015 (Training period: 2005 - 2014)

Cash	DRLMPT	Rate	Cash	Random	Rate
5918.67	104050.23	4.31	-128764.69	98570.23	-0.62
10179.40	111468.24	12.77	-111625.66	93137.37	-3.24
355.11	105628.72	5.65	-281743.77	85391.74	-3.83
7358.03	111226.05	12.12	-55155.30	99856.39	-0.09
-772.01	106747.45	6.70	-77859.39	94594.68	-3.04
1781.99	103952.41	4.02	-132568.75	103853.02	1.66
10173.91	113228.48	14.73	-52656.90	101942.96	1.27
5918.67	104050.23	4.31	-26674.57	95975.98	-3.18
2128.75	107825.98	8.00	-268801.89	102888.88	0.78
1584.20	107181.29	7.30	-429419.88	101515.36	0.29
10289.83	111369.68	12.67	-194372.36	97485.84	-0.85
355.11	105628.72	5.65	-310574.84	109582.15	2.33
10179.40	111468.24	12.77	-237856.72	118838.10	5.58
12266.91	111459.57	13.06	-258994.76	95425.68	-1.27
9612.81	106226.93	6.89	-27815.60	101685.11	1.32
9051.31	108534.76	9.38	-68759.78	111486.16	6.81
6791.27	105014.41	5.38	-78615.28	99566.14	-0.24
-806.26	103170.50	3.15	-106645.64	108416.79	4.07
12332.11	114500.45	16.54	-90485.24	101324.06	0.70
6791.27	105014.41	5.38	-186451.86	106073.75	2.12
	Average	**8.54**		**Average**	**0.53**

Testing period: 2016 (Training period: 2006 - 2015)

Cash	DRLMPT	Rate	Cash	Random	Rate
-4392.10	103807.07	3.65	-251838.18	110171.48	2.89
-3938.23	103676.41	3.54	-472348.14	82652.32	-3.03
14890.95	122065.16	25.93	-208974.13	89841.19	-3.29
19784.48	122692.69	28.29	-184315.57	107341.03	2.58
12518.67	122973.47	26.26	-17555.77	104939.34	4.20
9019.55	114265.31	15.68	-164934.69	93490.11	-2.46
10636.57	116355.81	18.30	-77274.60	95547.01	-2.51
-1845.79	105207.18	5.11	-121971.29	114655.63	6.60
10056.48	108693.35	9.67	-131308.07	89634.78	-4.48
12162.54	116393.32	18.66	-135191.94	113587.06	5.78
9150.49	114405.91	15.86	-228872.26	63008.86	-11.25
10714.08	120752.92	23.24	-229593.58	122532.87	6.84
8809.46	114271.52	15.65	-68328.75	99992.36	0.00
13565.28	115098.29	17.47	-94720.51	101759.66	0.90
12188.40	118167.30	20.69	-239742.10	122761.86	6.70
18555.29	123217.24	28.51	-78877.32	92531.12	-4.18
13706.35	123739.30	27.51	-201223.72	85574.16	-4.79
15056.38	119270.56	22.69	-45583.47	107778.74	5.34
19030.65	121066.72	26.02	-198437.92	91941.95	-2.70
11354.45	110088.75	11.38	-152024.77	94519.12	-2.17
	Average	**18.20**		**Average**	**0.05**

Testing period: 2017 (Training period: 2007 - 2016)

Cash	DRLMPT	Rate	Cash	Random	Rate
10409.37	115104.11	16.86	-169583.75	118599.73	6.90
2284.48	108507.36	8.71	-248212.20	113894.90	3.99
12277.79	109171.20	10.45	-50587.64	103450.23	2.29
6963.38	100952.74	1.02	-361808.68	103548.48	0.77
-9490.49	100217.05	0.20	-154199.10	101138.46	0.45
6963.38	100952.74	1.02	-87732.76	91544.58	-4.50
1801.97	100835.60	0.85	-31824.53	107704.53	5.84
-9490.49	100217.05	0.20	-245048.04	90799.11	-2.67
8136.33	104022.19	4.38	-115824.30	105800.74	2.69
1801.97	100835.60	0.85	-160085.19	96145.77	-1.48
561.27	102422.33	2.44	-74419.60	103917.20	2.25
11360.96	112452.34	14.05	-135304.58	101886.77	0.80
5748.30	107980.46	8.47	-67732.76	97415.42	-1.54
5547.31	108381.04	8.87	-177121.94	109250.47	3.34
-9490.49	100217.05	0.20	-295662.36	102828.38	0.71
5163.02	113423.34	14.15	-321562.61	104572.90	1.08
9677.52	106824.99	7.56	-76421.22	101728.74	0.98
5547.31	108381.04	8.87	-237927.85	105542.43	1.64
5609.35	111141.58	11.80	-195361.07	107705.59	2.61
1801.97	100835.60	0.85	-68416.16	104089.61	2.43
	Average	**6.09**		**Average**	**1.43**