COMSM048, Internet Economics & Financial Technology, 2020

Coursework Assessment Brief

Brief: Write a short **8-page** paper reporting on experiments with extending a simple automated market-making trading 'robot' on the Bristol Stock Exchange.

Weighting: This assessment is worth **100%** of your total unit **20 credits**.

Submission: via Blackboard submit the following:

- One PDF file containing:
  - an 8-page report in LNCS format (10% penalty for every partial page over)
  - an appendix (no length limit) detailing the full code of your new trading algorithm(s). Code can be annotated to aid explanation if you wish.
- One zip folder containing source code and data, including:
  - Python source code
  - Sample data generated by your experiments
  - Python scripts / output of your data analysis

Eight-page research paper on automated trading strategies

The *Bristol Stock Exchange* (BSE)[1] is a simple minimal simulation of the core data-structure on most technology-enabled financial markets: the *Limit Order Book* (LOB). BSE is written in Python and includes code for five simple automated trading "robots" called *Giveaway*, *ZIC, Shaver*, *Sniper* and *ZIP*. ZIC is an implementation of the Zero-Intelligence Constrained strategy introduced by Gode & Sunder; Sniper is inspired by, but not identical to, Kaplan's Sniper. Giveaway, Shaver, and ZIP are all by Dave Cliff.

Market Maker (MM):

For this assessment option, you should implement a new type of trading robot in BSE, a *market maker,* or *MM.* You should explore the behaviour of your new MM trader(s) by running sensibly structured comparative experiments, and then analyze the results from those experiments using appropriate visualization and statistical methods. Finally, you should write a report that clearly explains how your MM trader robot(s) works, the design and analysis of the experiments, and their outcome. Your report should be in the style of an academic research paper, as if you were going to submit it to a conference for peer-review.

You may want to use the existing trader code, or you might want to implement someone else's trading algorithm, such as "GD" by Gjerstad & Dickhaut (1998)[2], or "AA" by Vytelingum (2006).

All of the above robot trader strategies available in BSE have been implemented to act in a manner inspired by the job of *sales traders* in real financial markets. A sales trader receives orders to buy and/or orders to sell from her clients, and then tries to best execute the order in the market: the client supplies the limit price, and the sales trader tries to get a deal at a price better than the limit. Also, in all the literature that we looked at in the lectures, the robot trading

[1] The source-code for BSE (*BSE.py*) and its documentation (*BSEguide.pdf*) can be downloaded from GitHub. You must be using version 1.4 of BSE.py, last updated 26 Oct 2020. (On the GitHub site there is also a Wiki, which may have been updated more recently than the PDF version: check the dates.)

[2] Full details of the literature cited here are given in the "References" section of `BSEguide.pdf`.

algorithms such as (M)GD, ZIP, and AA, have been tested in their capacity as sales traders: the robot is given an order with a limit price, and then the robot tries to get a deal at a price better than the limit.

However, in most markets there are not just sales traders. Another type of trader that a financial firm might employ is a proprietary trader, or "prop trader". Prop traders buy and sell using the firm's own money: the job of the prop trader is to make money on these transactions, so basically the name of the game is to buy things when the price is low and sell them when the price has risen. Sometimes it may be necessary to sell for a lower price (e.g., if the market is crashing, it is better to sell at a small loss now than hang around and have to sell for a larger loss later). Prop traders are usually allowed to buy and sell any type of financial instrument that they can make money on, although usually they will specialise in various asset classes: one person might be a prop trader in currencies, another might prop trade in tech stocks only, and so on.

The most specialized a trader can get is to buy and sell just a single financial instrument. To distinguish this activity from prop trading, we'll refer to it as *market making*. (Actually, the full technical term is "designated market making" but we'll skip over that). Because BSE has only one tradable instrument, implementing a prop trader for BSE actually means creating a market maker for the BSE market.

DIMM01 - The Dumb Illustrative Market Maker, version 1.0

We have supplied you with a document for creating a minimal market making strategy called **D**umb, **I**llustrative, **M**arket **M**aker v01 (**DIMM01**). The logic in this prop trader is a straight clone of GVWY – this is absolutely *not* a realistic market-maker, but it provides you with a base class example of how a market making strategy can be implemented in BSE.[3]

What you are expected to do...

For a ***minimum*** you can create a market making trader that internally uses one or more of the available strategies to buy and to sell in the marketplace, holding an inventory of stock and a balance of cash, and trying to make a profit on its dealings as a market maker. The MM trader starts with an initial allocation of money and/or stock and it would need some record keeping (e.g., for each item of stock it buys, it should record the purchase price – that would then be an idea for the limit price when it comes to sell the item). You may want to add in some extra processing so that the MM trader stores a record of recent transactions, and uses that to try to predict whether the price is trending up (perhaps a good time to buy, adding to the MM trader's inventory) or down (possibly a good time to take profits, by selling off profitable items).

You may initially want to place a hard limit on the size of any one trader's inventory: this should help keep the market liquid, as it will prevent one trader amassing a huge inventory. For example, if the limit on inventory size is three, any MM trader can buy stock until it is holding three items, at which point its only allowable action is to sell. Similarly, if an MM trader sells an item and thereby reduces its inventory to zero items, its only allowable action will be to buy (that is, the MM trader cannot "short" the stock, selling items it does not currently own). Traders

[3] You may ask, *"why don't you just give us the code for DIMM01 rather than make us type it ourselves?"*. When you go through the process of implementing DIMM01, you will quickly realise the answer to this question. The answer is: *"it will greatly help you understand the BSE code and put you in a much stronger position for creating more advanced MM strategies"*. This approach is a *benefit*, not a *burden*.

holding one or two items in their inventory would be free to sell or to buy (or to simply wait), depending on what their strategy indicates is best to do, given the current market conditions.

It is fine if you want to start by using a previously-published algorithm like ZIP or MGD or AA, and then alter or extend it – that's how a lot of progress in science and engineering is made. But if you want to start from scratch, that is absolutely fine too. We're not *requiring* you to write a new (or revised) algorithm, but you can (probably) pick up some extra points by at least trying to do so: your algorithm certainly doesn't have to be world-beating, but you *are* expected to explain the design choices you made and to show that you know how to experimentally evaluate a new trading algorithm in the relatively simple context of BSE.[4]

Your report must include enough detail to enable a replication of your work (i.e., you must explain your experimental methodology clearly). Where possible, you should also support your robot design, experiment design, evaluation, results, and conclusions, by making reference to the academic literature and the wider context of trading in the financial markets.

Your robot code must be written in Python 3 and it has to be a subclass of the `Trader` class defined in BSE.py (in the same way that `Trader_ZIP`, `Trader_DIMM01`, and all the other traders discussed in this document are). Along with your paper, we would like you to submit your python source-code, any python files you use for your analysis, and a (small) sample of data files generated from your experiments.

It is up to you to decide how many MM traders you want to develop. Creating more will not necessarily result in more marks. It is better to have one MM that is well evaluated than two (or three) MMs that are poorly evaluated. Since your report is limited by space, the more traders you develop the less space you will have to evaluate each. This is an important tradeoff to consider when writing your report.

Code submission: to check that the results in your paper are independently replicable, we will be running tests with your MM trader submissions using the open-source release of BSE.py (with modifications described in the DIMM01 creation guide). If we can't replicate your results within reasonable error margins, the grade you get will be **severely** reduced. Therefore, please make sure that your MM trader does not rely on alterations to other parts of the BSE code (e.g., by "hacking" the limit order book to give your MM an unfair advantage). If your MM traders all extend the `Trader` class, and you are only editing the experiment configurations in `main()` and the `populate_market()` method to initialize your new MM classes, then you will be fine. But please check this before submitting.

Paper format: Your paper should be formatted according to the Springer *Lecture Notes in Computer Science* (LNCS) conference-proceedings format, details of which are available here: http://www.springer.com/computer/lncs?SGWID=0-164-6-793341-0. Your paper should be no longer than eight pages in LNCS format, including all figures, references, and any segments of code you include to explain your algorithm. The full code for your algorithm should then be included as an appendix, for completeness. The additional pages for your code are not counted in the 8-page limit. That is, you can write 8 pages of text and graphs and tables explaining your work, and then the content on Page 9 and onwards should be your trading-agent code.

[4] Hint: Remember that the performance of *any* trading strategy depends on the strategies of other traders in the market and also on the market dynamics (is it rising, falling, oscillating, fixed, etc.). A robust strategy evaluation will take account of these (and other) considerations.

Marking Scheme		
Experiment Writeup	25%	How well is the paper structured? How clearly does it explain what you have done?
Presentation of Results	25%	How thoroughly is the experiment carried out? How clearly are the results presented? Clarity of figures, etc.
Statistical Analysis and Conclusions	25%	Is an appropriate statistical analysis chosen and conducted correctly? Are correct conclusions clearly drawn from the analysis and explained?
Challenge and Originality	25%	How challenging was the task you set yourself? How complex were the agents and the experimental setup? How extensive was the analysis?

Marking criteria	
+80%	Outstanding report. Extensive exploration and analysis demonstrating deep understanding and reading outside of the lectures.
70 - 80%	Excellent report detailing motivation, MM trader design, experiment method, appropriate statistical analysis and results. Clear demonstration of MM trading algorithm exploration and design choices, analysis, and performance optimisation. Some novelty in design choices. Clear demonstration of excellent understanding of concepts presented in the unit.
60 - 70%	Report of correct length detailing motivation, MM trader design, experiment method, appropriate statistical analysis and results. The report is lacking one area: MM trader is simple, a lack of adequate experiments, analysis lacking.
50 - 60%	Report of correct length detailing motivation, MM trader design, experiment method, appropriate statistical analysis and results. The report is lacking some of the following ways: MM trader is very simple, a lack of adequate experiments, analysis lacking, results interpreted incorrectly, poorly written report.
<50% (fail)	Report is not at an appropriate standard. Poor demonstration of understanding. Objectives of the assignment have not been demonstrated.

For fun: a tournament of submissions

For fun only (i.e., this is ***not*** part of the formal assessment and does *not contribute* in any way towards your grade) we plan to run a tournament of the market making robots that are submitted for assessment. The full details of this tournament are yet to be decided, but it may run a bit like the football World Cup: a random draw to assign each trader to a group; a round-robin contest within the group to determine the top performers in each group; and then a knockout contest leading to quarter finals, semi-finals, and then a grand final contest (plus a playoff for third place). For a comparison of two market makers, experiments would involve varying the market participants and altering the nature of the supply and demand schedules.

Note that when you submit your MM robot algorithm for this assessment, that submission is taken as your definite consent that the algorithm you submit can be used in this way, and that full details of your algorithm (including source code listing) can be given in any publications (e.g. print, and/or online) that describe the outcome of the tournament.

(If you happen to come up with a market making algorithm that outperforms all the others, we would like to be able to write that up as a research paper, with you as one of the co-authors of course.)

We will make the outcome of the tournament available only after marks are released, so do not expect to hear about this until February 2021, onwards. Remember, this tournament has no bearing on your coursework mark. It is for fun only - and potentially interesting research.