Brain Mind and Markets Lab

THE UNIVERSITY OF MELBOURNE | FACULTY OF BUSINESS & ECONOMICS

FNCE30010

Algorithmic Trading

Semester 2, 2018
Project 1 - Task 1 (Induced demand-supply)
Due **11:59 pm on Monday, 27 August 2018**

Nitin Yadav and Peter Bossaerts
Brain, Mind & Markets Lab.
Department of Finance.
Faculty of Business and Economics.
The University of Melbourne.

Administrative Arrangements

Due date:	**11:59 pm on Monday, 27 August 2018**
Late assignments:	Late submission assignments may attract a penalty unless an extension has been granted. All extension requests must be made prior to the assessment due date and supported with appropriate documentation. Unless an extension has been granted, penalties to the assessment will be applied. For assignments submitted after the due date, the mark awarded will be reduced by 10% for each day the work is late. For assignments submitted later than 5 working days after the due date will not be marked and will receive no marks.
Where to submit:	Assignment submission is via the LMS Assignment Submission link for all written assignments. Please refer to the LMS Student Guide: Turnitin Assignments for detailed submission instructions, if needed. **Only one member of the group has to upload** (see Group Size below).
Group size:	For undergraduates the maximum number in each group is 3. For postgraduates this is an individual assessment.
Submission Metadata:	You should include your details as part of your submission. **Subject Code and Name** **Student Numbers of all members in group** **Full Names** **Assignment Name**
Word limit:	FBE usually as a word limit. This assignment is for computer code, however, and hence word limits do not apply.
Marks:	This assignment counts 15% towards the final mark in this subject.

The assignment must be your group's own work. Students are encouraged to discuss the assignment and to share information sources. However, the writing of each group's assignment must be conducted separately and independently.

Assignment Instructions

Trading Setting (Identical to "Instructions" for Induced Demand/Supply Manual Experiment)

You will be asked to trade in a marketplace called MarketForWidgets.

When you log in, you will see that you have been given a certain amount (maybe zero) of securities called Widgets and some (maybe zero) cash. You will then be given the opportunity to trade.

You trade by sending offers to other traders. All your offers are displayed in a common book (and visible as red and blue entries in the book). Trade takes place if an incoming buy order is for a price at least that of the best standing sell order, or *v.v.* Trade takes place at the price of the standing order.

Here is how your performance will be measured.

If you are given a positive initial allocation of Widgets, then you are a "Seller," and for each of those Widgets you can SELL, your points equal the sales price, minus a charge.

If you are not given an initial allocation of Widgets (you start with zero Widgets), then you are a "Buyer," and you will be given rewards for the **first five Widgets** you manage to BUY, minus the purchase price.

The reward (charges) for buying (Selling) do not change in a trading session.

In addition, you are allowed to "speculate" which is to buy and subsequently sell; you keep the earnings if the purchase price is below the sales price, otherwise the loss will be subtracted from your earnings. Conversely, you may be able to sell high and buy back at a lower price. The corresponding gains are added to your earnings.

Your task will be to write Python code for a bot for the trading setting described above.

Task Specification

For this task, please use the robot class template DSBot provided on the LMS. If you add new functions in the given file, please make sure you prefix your functions with an underscore ("_"). You should not remove any function from the provided template.

1. Trading and bot parameters ..... [1 mark]
    a. Price threshold. Create a global variable called `DS_REWARD_CHARGE` that stores the price threshold (in cents) for buying/selling widgets. For example, if the reward for buyer (or charge for seller) is $5, `DS_REWARD_CHARGE` should be set to 500.
    b. Trading account details. Change the variables that start with `FM_` prefix (e.g., `FM_ACCOUNT`) to the Adhocmarket details of the submitting member. Change the `MARKETPLACE_ID` variable to the ID of the given MarketForWidgets marketplace. You can find this ID from Algohost.
    c. Group details. Change the `GROUP_MEMBERS` variable to the details of the group members. This variable is a dictionary whose key is the student id and value as the name of the member (e.g., `GROUP_MEMBERS = {"123": "Name 1", "456": "Name 2"}`)
2. Role of the bot ..... [2 marks]
   The template defines an enumeration called `Role` that has two possible values, `BUYER` and `SELLER`. You should infer the role from bot's initial holdings (i.e., if the bot has positive units then it is a seller, if the bot has positive cash then it is a buyer), and accordingly update the `_role` variable in the DSBot class.
3. Type of the bot ..... [2 marks]
   The template defines an enumeration called BotType with two types. The bot can either react to orders in the order book (type `REACTIVE`) or can proactively send orders to the market (`MARKET_MAKER`). Add an extra argument called `bot_type` to the constructor of the class DSBot and store its value in a class variable called `_bot_type`. In a session, the bot can only either be of type market maker or reactive (but not both). You should provide correct implementation for both of these types.
4. Identification of profitable trade opportunities ..... [3 marks]
   The bot should identify ALL profitable trade opportunities (irrespective of weather it responds to those opportunities). Trade opportunities should be identified based on the best bid and best ask prices. The bot should print the details of the profit opportunity and if it is responding to the trade opportunity by sending an order. Additionally, a reason should be printed if the bot is unable to respond (e.g., a seller bot has no units left). To print the trade opportunity please use the provided `_print_trade_opportunity` function.

5. Order management ..... [5 marks]
    a. Market Id. The DSBot class has a variable named `_market_id` to store the market to which orders are sent. When the bot is initialised, you should extract the id of the Widget market and store it in the `_market_id` variable.
    b. Maximum number of open orders
    At any point in time the bot should have only 1 pending order in the order book. Each order should be for only 1 unit of widget. Note: Implementing this will require waiting for server to accept/reject sent orders. Please refer to the Warm-Up bot task for help.
    c. Order validity
    Before sending an order the bot should check for order validity (e.g., does it have enough units to sell, or enough cash to buy).
6. Code quality ..... [2 marks]
    Your submitted code must be clear to understand and well commented. Please read the following links for suggestions:
    – https://docs.python-guide.org/writing/style/
    – https://www.python.org/dev/peps/pep-0008/