FIN 566
Problem Set #3

Adam D. Clark-Joseph

University of Illinois Urbana-Champaign

DUE September 27, 2017

Introduction

This assignment enriches on the basic simulated market framework developed in the first two problem sets by introducing more realistic price-choosing behavior by background traders. You will be asked to examine how three of the "robot_1" market-making algorithms that you wrote for PS#2 perform in this new setting, and then to do some preliminary work that you will use on the next problem set to write an improved market-making algorithm that only participates in the market if estimates based on lagged data indicate that doing so will be profitable. I have again posted a general template for the overall simulation; re-use the scripts for the matching-engine, orderbook construction, and the three market-making algorithms from PS#2, which you already built versions of in previous problem sets. I have also included a script for the NEW background-trader behavior, "`bgt_behavior_rw_price.m`", and a "meta-script" template, which is explained in the relevant questions.

In the solutions that you turn in, please leave the "Model Parameters" unchanged, except where otherwise noted. (Note that t_max is shorter than it was on PS#1&2) For your convenience, I have included organizing data structures; please use these and populate them with the indicated data (you can also add other things if you so choose.)

For future reference:

- "`robot1_algo_mm_at_best_IC.m`" denotes the market-making algorithm from PS#2 that enters an order at the best price in the direction opposite to the sign of robot1's inventory.
- "`robot1_algo_mm_at_best.m`" denotes the market-making algorithm from PS#2 that enters an order at the best price in a randomly chosen direction.
- "`robot1_algo_mm_better_price.m`" denotes the market-making algorithm from PS#2 that enters an order in a randomly chosen direction at one dollar better than the current best price, unless that order would be marketable, in which case the order is priced at the current best.

General remark: running the meta-script iterations can take a while, particularly if you've written inefficient code. If the run-time becomes problematic for you, examine your code for gross inefficiencies. Alternatively, and/or in addition, you can leave your simulations running in the background and do something else while you wait for them to finish.

1 Market Making With Pure Random-Walk Prices

Set t_max=11322, and prob_last_order_price_resets=1

1.1 Question

Set prob_last_order_price_sets_to_price_robot_1=0.

As a warm-up, please report the quantities in the table below for the indicated market-making algorithm. Where appropriate, compute inventory value using the same mark-to-market conventions as in past problem sets.

	`mm_at_best_IC`	`mm_at_best`	`mm_better_price`
# of post-burn-in transactions			
mean bid-ask spread (post-burn-in)			
Robot1 max inventory position magnitude (dollars)			
Robot1 max inventory position magnitude (shares)			
Robot1 total trading profit			
Robot1 total trading volume			
Robot1 final inventory position (dollars)			
Robot1 final cash position			
Mean time-to-execution of Robot1's orders			
Median time-to-execution of Robot1's orders			
Std. dev. of time-to-execution of Robot1's orders			
Fraction of Robot1's orders that execute			
Fraction of orders that Robot1 places that are mispriced relative to fundamental value			

Hint: All of the above quantities can be calculated immediately using code from previous problem sets, except for "Fraction of orders that Robot1 places that are mispriced relative to fundamental value." Note that "last_order_price" at time t is what we have been calling fundamental value at time t.

1.2 Question

(a) Repeat question (1.1), but set prob_last_order_price_sets_to_price_robot_1=1.

(b) Explain why the market-maker profits from (1.2.a) are so different from those in (1.1).

Hint: This may not be entirely obvious at first. Try looking at a plot of best-bid/best-ask prices or transaction prices, and contrast the behavior between (1.1) and (1.2.a).

2 Bootstrapping

To make meaningful comparisons between the performance of different algorithms in different settings, we need to regard our simulation results as estimates of the algorithms' true performance, and we need to quantify the precision of those estimates. This section walks you through constructing simple bootstrap confidence intervals for various estimates (such as those in section 1).

The Meta-Script

The file "`FIN566_PS3_meta_script_template.m`" is a short program that runs the main_script many times and stores the results. The parameter values and script designations in the meta_script are used for each iteration of the main_script.

2.1 Question

(a) What is the purpose of the conditional ("if") statement at the beginning of the main_script template?

(b) Suppose that we are using the meta_script to run many repetitions of the main_script. Would anything change if we deleted the entire conditional statement (everything from "if" to "end") at the beginning of the main_script template?

Hint: This question is as easy as it seems—I just want to make sure that this point is clear.

Bootstrapping Procedure Sketch

When the meta_script is executed, it repeats the main_script commands "num_simulation_runs" times. For concreteness, assume that num_simulation_runs=50. The output from the meta_script will be a matrix ("meta_comparison_mat") with 50 columns, corresponding to each of the 50 simulation runs. Each row of the meta_comparison_mat corresponds to a particular statistic from the simulation results, such as the total trading profits of robot1. (The variable "est_prob_last_order_price_resets" will be discussed in later sections; it can be set equal to zero or omitted at present.)

Assume that we are interested in estimating robot1's average total trading profits. Robot1's total trading profits from each of the 50 different simulation runs are stored in one row of the meta_comparison_mat. To get a point estimate of robot1's average total trading profits, we can simply calculate the mean of that row. Now suppose that we draw 50 samples (with replacement) from the entries in that row; the mean of those 50 samples gives us another observation of robot1's average total trading profits, and this new observation comes from the same distribution as our initial point estimate. If we repeat this process of drawing 50 samples (with replacement) and computing the mean, say, 1000 times, then we will end up with 1000 observations of robot1's average total trading profits, drawn from the same distribution as our initial point estimate.

The empirical distribution of our 1000 observations will approach the true distribution of our initial point estimate,[1] so if we sort the vector of our 1000 observations in ascending order, we obtain the approximate quantiles of this distribution. For example, the 10th element of this sorted vector gives the 1st-percentile value, and the 991st element gives the 99th-percentile value. Confidence intervals are easy to compute from these quantiles; for example, the 90% confidence interval for the mean is given by [50th element, 951st element].

2.2 Question

Set the following parameter values:

[1] Provided that various regularity conditions are satisfied. For the purposes of this course, unless otherwise noted, the appropriate regularity conditions are always satisfied.

- num_simulation_runs=100
- t_max=2322
- prob_last_order_price_sets_to_price_robot_1=0
- prob_last_order_price_resets=0.064

(a) Please compute 95% confidence intervals for the quantities listed in the table below for the indicated market-making algorithms.

DO NOT REPORT THESE NUMBERS IN SCIENTIFIC NOTATION! The numbers in the table below should span approximately 5 orders of magnitude. If you put all of these numbers in the same matrix in Matlab, the format in which that matrix would be displayed on the Matlab command window will omit important information by rounding and/or truncating the smaller numbers. Copy values out of the variable editor (with "numeric array format" set to "short g" or "long g") instead of out of the command window. So to reiterate, DO NOT REPORT THESE NUMBERS IN SCIENTIFIC NOTATION!

	`mm_at_best_IC`	`mm_at_best`
# of post-burn-in transactions		
mean bid-ask spread (post-burn-in)		
Robot1 max inventory position magnitude (dollars)		
Robot1 max inventory position magnitude (shares)		
Robot1 total trading profit		
Robot1 total trading volume		
Robot1 final inventory position (dollars)		
Robot1 final cash position		
Mean time-to-execution of Robot1's orders		
Median time-to-execution of Robot1's orders		
Std. dev. of time-to-execution of Robot1's orders		
Fraction of Robot1's orders that execute		
Fraction of orders that Robot1 places that are mispriced relative to fundamental value		

(b) Are the average profits of the "`robot1_algo_mm_at_best_IC.m`" algorithm significantly different from those of the "`robot1_algo_mm_at_best.m`" algorithm (which has no inventory-control mechanism)? What difference would you expect to see in the limit as $num\_simulation\_runs \to \infty$?

Explain why the inventory-control mechanism used by "`robot1_algo_mm_at_best_IC.m`" affects profits like this in the current setting, and why it did not affect profits in the uniform-price-background-trader setting.

3 Finding Necessary Conditions for MM Algorithm Profitability

The potential profitability of market-making in a given market is heavily influenced by the degree of adverse selection in that market. In the context of our simulation, the variable "prob_last_order_price_resets" parameterizes the level of adverse selection risk. Using the methods from the last two sections of this problem set, we can show that all of the market-making algorithms that we have considered lose money when prob_last_order_price_resets=1, and from the results in PS#2, it follows that all of these market-making algorithms will make money when prob_last_order_price_resets=0. In this section, you are asked to find conditions for prob_last_order_price_resets to ensure that a particular market-making algorithm will be profitable with high probability.

Economic theory vastly simplifies this problem, because we can show that expected market-maker profitability is monotone decreasing in prob_last_order_price_resets. Thus it suffices to find the maximum value of prob_last_order_price_resets for which a given market-making algorithm is expected to earn non-negative profits.

3.1 Question

Briefly explain the intuition for why expected market-maker profitability is monotone decreasing in prob_last_order_price_resets.

3.2 Question

Set the following parameter values:

- num_simulation_runs=50
- t_max=2322
- prob_last_order_price_sets_to_price_robot_1=0

Find the maximum value of prob_last_order_price_resets for which the 97.5% bootstrap lower bound for the average trading profits of the "`robot1_algo_mm_at_best_IC.m`" algorithm is greater than zero. Determine this maximum value for prob_last_order_price_resets to within 0.001. Report the 95% confidence interval for average trading profits associated to this this maximum value.

Hint 1: The results from section 2 of this problem set show that the desired value for prob_last_order_price_resets is less than 0.064.

Hint 2: This is a fairly easy problem to solve using bisection. You are welcome to use methods other than bisection if you choose, but using a grid search is a bad idea.

Hint 3: You are welcome to write a script or function to compute the answer to this question, but you are also allowed to do it manually. Using bisection, and Hint 1 above, 6 iterations should be sufficient to find the answer to the required precision.

3.3 Question

Set the following parameter values:

- num_simulation_runs=50

- t_max=2322
- prob_last_order_price_sets_to_price_robot_1=0

Find the maximum value of prob_last_order_price_resets for which the 97.5% bootstrap lower bound for the average trading profits of the "`robot1_algo_mm_at_best.m`" algorithm is greater than zero. Determine this maximum value for prob_last_order_price_resets to within 0.001. Report the 95% confidence interval for average trading profits associated to this this maximum value.

Hint 1: The results from question 2.2(b) of this problem set suggest that the desired value for prob_last_order_price_resets is smaller for the market-making algorithm without inventory control than it is for the algorithm with inventory control. The answer from question 3.2 should help you narrow your search.

Hint 2: Hints 2 and 3 from problem 3.2 still apply.

Accompanying Template Code

```
FIN566_PS3_main_script_template.m
FIN566_PS3_meta_script_template.m
bgt_behavior_rw_price.m
```

Reused Code

```
robot1_algo_mm_at_best_IC.m
robot1_algo_mm_at_best.m
robot1_algo_mm_better_price.m
FIN566_PS2_entry_only_matching_subscript.m
orderbook_depth_construction_subscript.m
```