Recall the third prediction market we ran:

We flip coin $C$ (the real coin). Then each student $s$ in the class is assigned a number of decoys $n_s$, and we flip that many decoys $D_1, D_2, \ldots, D_{n_s}$. Then we randomly pick one of the $1 + n_s$ coins (the real coin and the $n_s$ decoys) at random, and tell them whether it is heads or tails. Students then trade contracts which are worth $100 if $C$ is heads and $0 if $C$ is tails.

All coins are fair, unless otherwise specified.

Eventually, you will be coding bots to participate in prediction markets. Your bot’s strategy should depend not only on how you expect other students’ bots to behave; equally important, it should depend on the behavior of the market maker: i.e., Gabe’s bot.

In the first part of this lab, we ask you to think about how your bot should behave for a variety of different market makers. In the second part, we ask you to consider how the behavior of the other bidders in the market, as well as your risk preferences, might affect your bot’s bidding behavior.

1. Imagine your bot is bidding against 15 other students’ bots and a bot coded by your instructors. For each of the following instructor bots, how might you code your bot to bid?

   (a) Every few seconds, the bot picks a number uniformly at random between 0 and 100, as well as “buy” or “sell” uniformly at random, and then offers to buy or sell, accordingly, that number of shares.

   (b) Every few seconds, the bot picks a number uniformly at random between 0 and 100, and then buys (or sells) at that number.

   (c) The bot never trades.

   (d) The bot is an additional bidder, just like all the rest of you, receiving the same type of information, coded by your instructors to maximize its profits.

   (e) The bot calculates the fair value of the contract given its initial information (i.e., a number of decoys and “heads” or “tails”). It then makes any trade it believes will net at least $1. In other words, if its fair value is $v$, then it is always willing to buy for $v - 1$, and sell for $v + 1$. It accepts a trade or offers one every few seconds. For this bot, $v$ never changes.
(f) The bot constructs its beliefs about the fair value of the contract just like the previous bot did. However, this bot is smart enough to know that it should update its value as trading progresses. So now $v$ is updated after every trade. More specifically, every time the bot sells to someone, $v$ goes up by $5$, and every time the bot buys from someone, $v$ goes down by $5$. As before, the bot will make any trade that it believes will net at least $1$.

The coder of this bot is convinced that their bot will make good trades, so they want their bot to make as many of these trades as possible. Therefore, this bot is not limited to trading once every few seconds. Instead, it will always offer to buy for $v - 1$ and sell for $v + 1$. If anyone trades with it, it will update $v$ to $v'$ (either by adding or subtracting $5$ to it as stated above), and then immediately offer to buy for $v' - 1$ and sell for $v' + 1$ again.

For the sake of simplicity, you may assume that this bot updates its orders faster than anyone else in the market can.

Think hard about this one. This bot is doing something seemingly reasonable. It has some belief about the fair value of the contract which it updates based on what other people do. But it has a serious flaw. When you implement your own bots, it will be on you to make sure your bots don’t make a similar mistake!

2. For each of the above markets, how would your strategy change if there were no other bidders participating in the market? What about if there was 1 other bidder in the market? What about 100?

3. In reality, people are not pure profit-maximizers. They are usually risk averse. One way to model risk aversion is to give people risk limits: Let $P$ be the number of contracts that a bidder has bought less the number of contracts that a bidder has sold. A bidder obeys a risk limit of $L$ if $|P| < L$ at all times.

Imagine your bot were required to adhere to a risk limit of $5$. How might this change your bidding behavior in the above markets?

4. Think about how you should code your bot to hedge against the possibility that you implement incorrect logic, like the logic we proposed for the very last instructor bot. Recall that even if this bot were implemented exactly as described, it would produce terrible results! The basic problem is that you think your bot will perform well, but it could perform terribly, if your logic is incorrect. One simple (and common) fix is to restrict the number of trades your bot can make each second, but that would not be ideal when your logic is correct and your bot performs well, because it will not get to make as many good trades.
How might you safeguard against serious flaws in your logic without restricting the potential for making many good trades?