CS 470 Programming Assignment: Probability Estimation

This assignment involves using Monte-Carlo sampling to estimate probabilities of certain events. There is no provided code for this assignment, you are welcome to code in whatever language you like. As always, any code you write should be your own.

There are three different problems you will need to solve in order to complete this assignment. Make sure you answer each question for each part in order to get full credit.

Turn in: You should turn in a single `.pdf` write-up that includes your answers to all of the questions for each section as well as an appendix with your code.

1 St. Petersburg Paradox [35 points]

You have the chance to play the following game of chance. A single coin will be tossed until it comes up tails, and the number of tosses it takes will be recorded. If it takes k tosses to see the first tails, then you win 2^k dollars. For example, if the first toss is tails, then you win $2^1 = 2$ dollars, while if it takes 10 tosses to first see a tails, then you win $2^{10} = 1024$ dollars.

Your task is to simulate this game to determine the average amount of money you would expect to win while playing it.

You should repeat this (at least) 3 times, simulating the game 100 times, 10,000 times, and 1,000,000 times. For each case (and any others you attempt), report the following:

- The average amount of money won per game
- The most money ever won in a game

How much money would you pay for the chance to play this game?

2 Monty Hall Problem [35 points]

From Wikipedia:

> The Monty Hall problem is a brain teaser, in the form of a probability puzzle, loosely based on the American television game show Let's Make a Deal and named after its original host, Monty Hall. The problem was originally posed (and solved) in a letter by Steve Selvin

to the American Statistician in 1975. It became famous as a question from a reader's letter quoted in Marilyn vos Savant's "Ask Marilyn" column in Parade magazine in 1990:

> Suppose you're on a game show, and you're given the choice of three doors: Behind one door is a car; behind the others, goats. You pick a door, say No. 1, and the host, who knows what's behind the doors, opens another door, say No. 3, which has a goat. He then says to you, "Do you want to pick door No. 2?" Is it to your advantage to switch your choice?

Your task is to answer this question using simulation. You should write a program which simulates the entire scenario and tries each of the two strategies (keep choice or switch choice). Your code should

- Randomly place the car behind one of the three doors
- Randomly pick a door to "guess"
- Randomly open one of the other doors that has a goat
- Then see if switching or staying would win the car
- Record this and compute the average number of cars won for each strategy

Play at least 1000 games and report the percentage of times each strategy wins the car. How many games do you need to simulate to start to see definitively which strategy is better? (Plotting this out might help) What is the actual (unestimated/exact) percentage that each strategy will win? Please give an explanation/derivation for this answer.

3 RISK Battles [30 points]

In the board game of RISK, battles over territories proceed by rolling dice. In a battle over a territory there is an attacking player (trying to take the territory) and a defending player (currently in the territory). Each player has some number of armies that will be involved in the battle. The battle proceeds in stages, with each stage requiring each player to roll a set of dice. The attacking player can roll up to three dice, but must always have at least 1 more army than the number of dice rolled. The defending player can roll up to two dice, and must have at least as many armies as dice rolled.

When all of the dice are rolled, the dice for each player are sorted and then matched up from highest to lowest. For each pair of dice (one from each player) the player with the lower number loses an army. In the case of a tie, the attacker loses an army.

A few examples to hopefully make this concrete:

- Attacker rolls three dice {4,6,2}, Defender rolls two dice {3,5}. The 6 and 5 are matched up, and the 4 and 3 are matched up, and the defender loses two armies
- Attacker rolls three dice {2,3,1}, Defender rolls one die {4}. The 3 and 4 match up and the attacker loses one army.

You will use Monte Carlo simulation to estimate the probability of different outcomes in battles in RISK. In each of the following problems you should determine the appropriate number of samples to use and include a brief justification of that number.

- If the attacker rolls n_a dice and the defender n_d, what are the probabilities of the different outcomes (number of armies lost by each player)? Repeat for the different possible combinations of n_a (1,2,3) and n_d (1,2). Is it ever advantageous for a player to roll less than the most dice they are allowed by the rules?

- Assume that the defender starts out with 5 armies in the territory. Your task is to estimate, for all numbers of attacker armies between 2 and 20, the probability that the attacker will win the territory. The attacker wins the territory if all of the defender armies are lost. As the battle progresses, the dice rolling will continue, with each player using the maximum number of dice allowed given their number of armies, until either the defender has lost all of their armies or the attacker only has 1 army remaining. Again, the attacker wins the territory if the defender armies are entirely lost. Plot out your results showing this attacker-win-probability (y-axis) for each different number of attacker armies (x-axis). Make sure your plot is labeled appropriately. What is the minimum number of armies the attacker needs to guarantee a 50% chance of winning the territory? How about to guarantee an 80% chance of winning?

- Assume that both the attacker and defender have 10 armies at the beginning of the battle, and assume that the attacker continues the battle until the territory is won or the attacker only has a single army remaining. Your task is to estimate the probability for each possible number of remaining armies of each player at the end of this battle. The possibilities are the attacker could have any number two through ten, and the defender zero, or the attacker could have only one army remaining and the defender could have any number, one through ten. Display these probabilities nicely either in a table or a plot.

4 Notes

If you decide to use Python to implement this lab, the `random` (for random number generation) and `matplotlib.pyplot` (for making plots) modules will be helpful. Internet searches should turn up lots of resources for helping you use these modules.