Accurately Calculating the Probability of the Counterintuitive Logic That Lies in the Monty Hall Problem

Abdulaziz Al-Said, Kalvin Xiong, Weston Hawes, Anthony Lopez
California State University, Fresno
April 14, 2023

Abstract

The purpose of this report is to introduce the Monty Hall problem, but use the Monte Carlo method of simulation in order to obtain an accurate calculation of the probabilities present in the Monty Hall problem. Calculations of the more counterintuitive solutions of larger sample sizes. The method for the simulation is the scripting language Python. This report will introduce the Monty Hall problem, the advantage of using the Monte Carlo method of simulation, and an explanation of the data collected. All data present in this report was directly collected through this simulation.

1 Introduction

The goal of this simulation is to write a monte carlo python script in order to estimate the probability of winning the Montey Hall problem with three policies on switching from the original door chosen. Those policies are as follows. Policy 1 is to randomly decide to switch. Policy 2 is to never switch. Policy 3 is to always switch. This simulation will check each of these three policies with 3, 6, 9, 20, and 100 doors. There are also two variations, the traditional Monty Hall problem and the banana variant. The banana variant is where once you have selected one of the doors, the host slips on a banana peel and accidentally pushes open another door. This means there is a random chance of losing at the start of the game because the host can accidentally open the winning door, ending the game.

2 Background Material

2.1 The Monty Hall Problem

The Monty Hall problem is a classic probability puzzle that is based on a game show scenario. The problem is named after the host of the original show, "Let's Make a Deal", Monty

Hall. Suppose you're on a game show and you're given the choice of three doors. Behind one door is a prize, while behind the other two doors are goats. The doors are initially closed, and you have to pick one of the doors without knowing what's behind it. Once you've chosen a door, the host, who knows what's behind each door, will open one of the other two doors that does not have the prize behind it. At this point, there are two doors left - the one you initially chose and one other door that hasn't been opened yet. The host then offers you the opportunity to switch your initial choice to the other unopened door, or you can stick with your original choice.

2.2 Simulation Model

This simulation model is defined as a Monte Carlo and agent-based model with a discrete variable. It is Monte Carlo because of the stochastic nature of choosing a door as well as simulating the Monty Hall problem with 10, 100, 1000, and 10000 iterations. It is agent-based because of the player or contestant having different policies for switching after choosing the initial door, as well as the banana variant of the host accidentally opening a door. The discrete variable is the probability of choosing the correct door, since it can only be a finite value.

3 Approach

The Python script for the simulation begins with choosing the variation of the game, standard or banana, and with creating the array of doors to be initialized as 0 and randomly choose the index to be the winning door and set its value to be 1. After the player chooses a random door from the array, the host will open all the rest of the doors besides the winning door and 1 extra door, the door the player chose will be among these two. The player will then follow its policy to determine if it wins and the probability of it. The simulation will play out for all three policies and the second banana variation with 3, 6, 9, 20, and 100 doors over 10, 100, 1000, and 10000 iterations. All that data is saved and used to create tables using the matlab plot library

4 Results

Figure 1.1-1.3 are for the original no banana variant and for the random switch, no switch, and always switch policies respectively. Figure 2.1-3 are for the banana variant and for the random switch, no switch, and always switch policies respectively. All the data supports the earlier assumptions on winning. Always switching has the highest win rate, not switching has the highest losing rate, and randomly switching is in between. The same holds for the banana variant, the only difference is that the banana had more of an impact on losing the fewer doors there were because the host had a higher chance to accidentally open the winning door. All this data supports the logic presented earlier and succeeds at computing the probabilities at higher iterations.

Figure 1.1

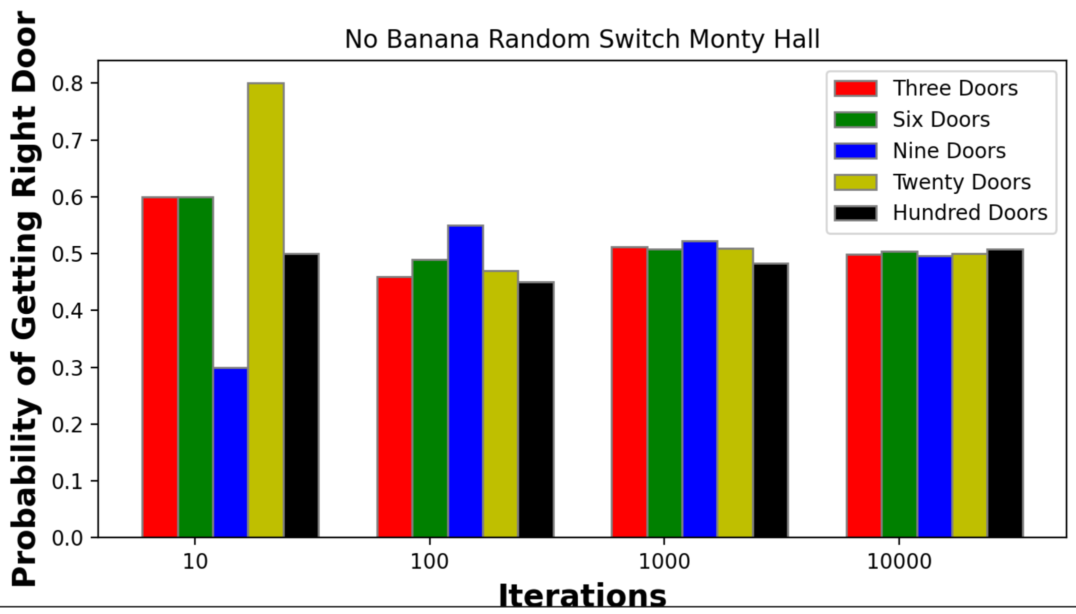


Figure 1.2

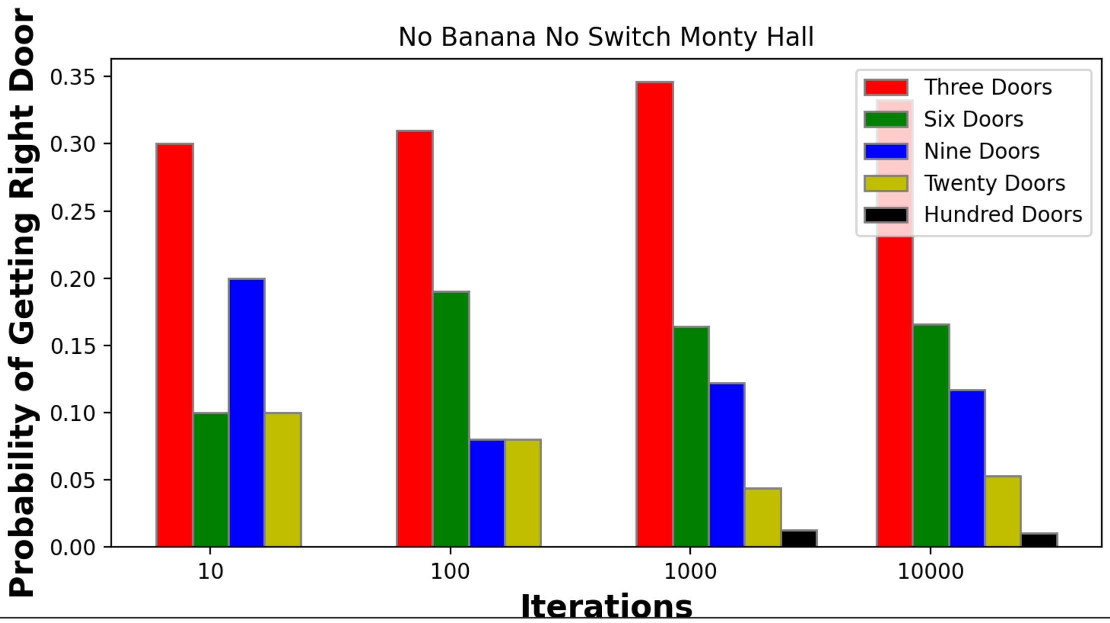


Figure 1.3

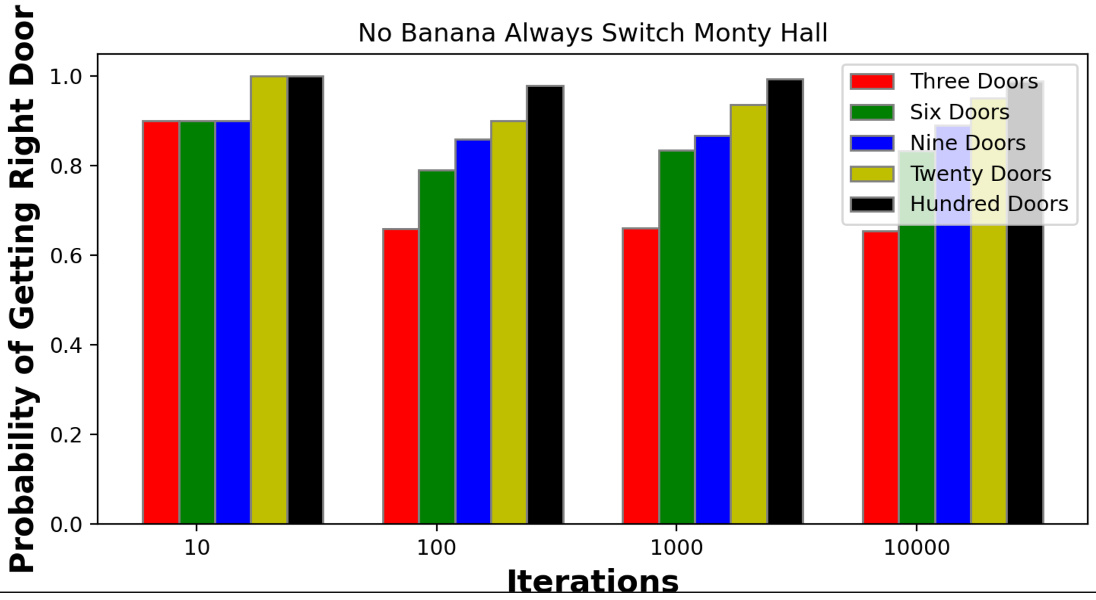


Figure 2.1

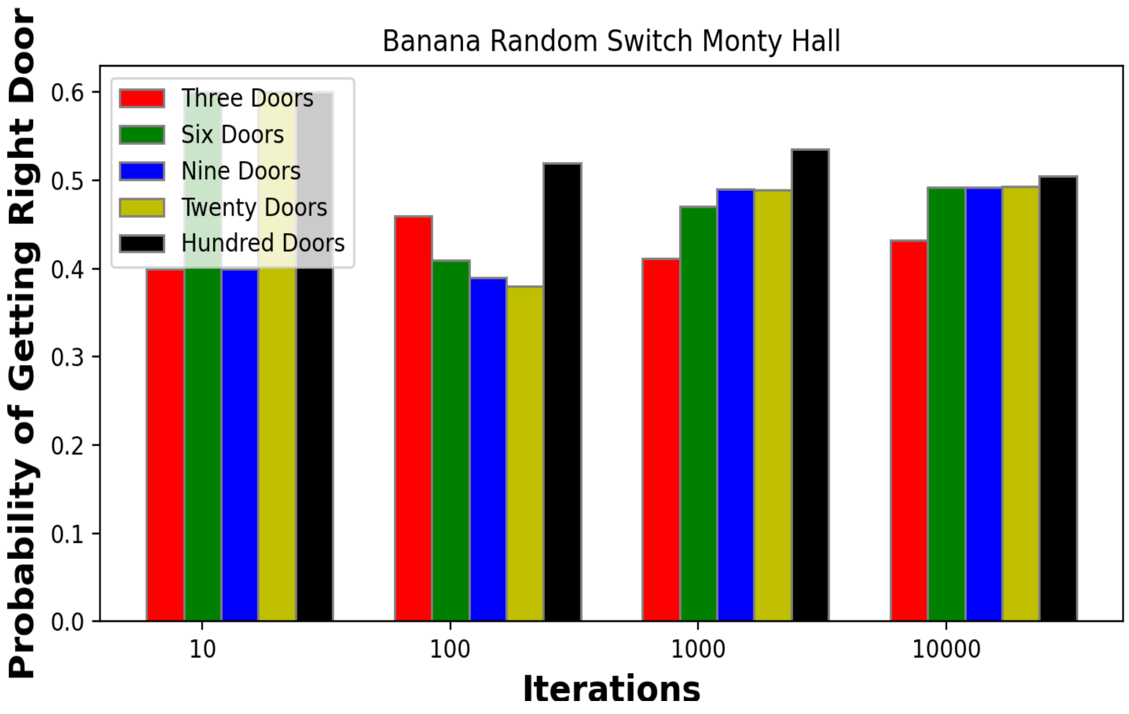


Figure 2.2

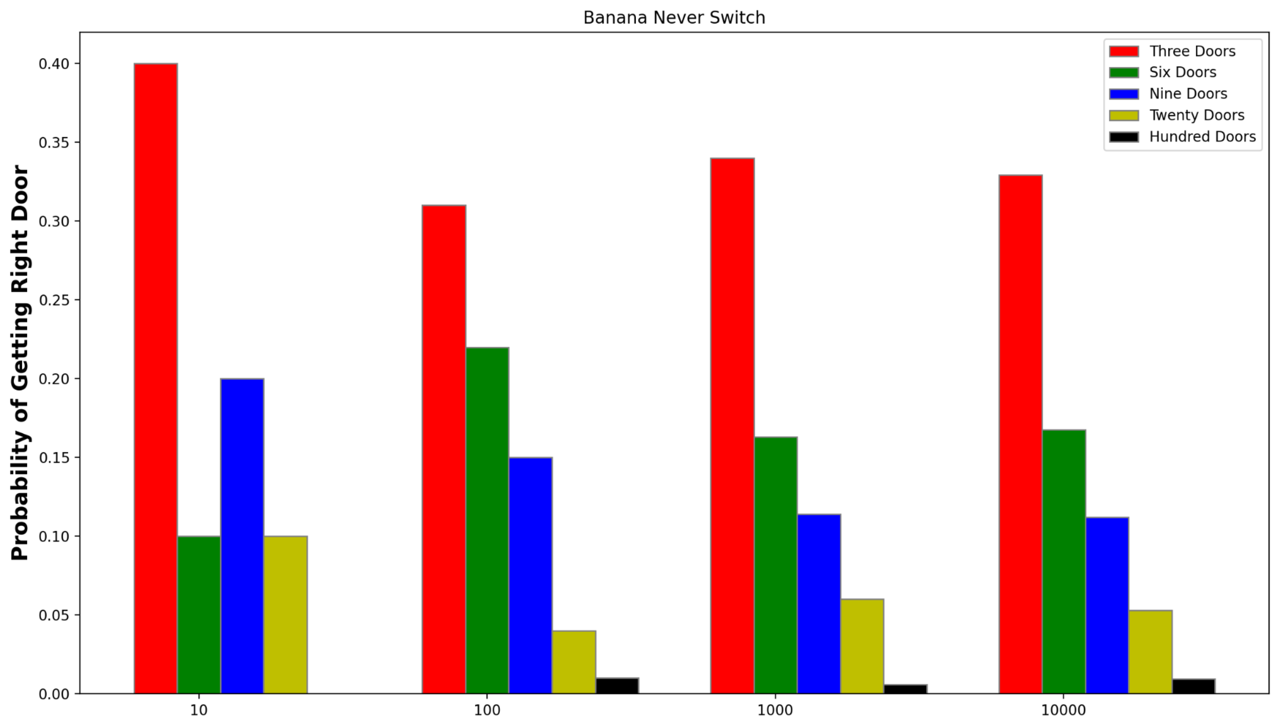
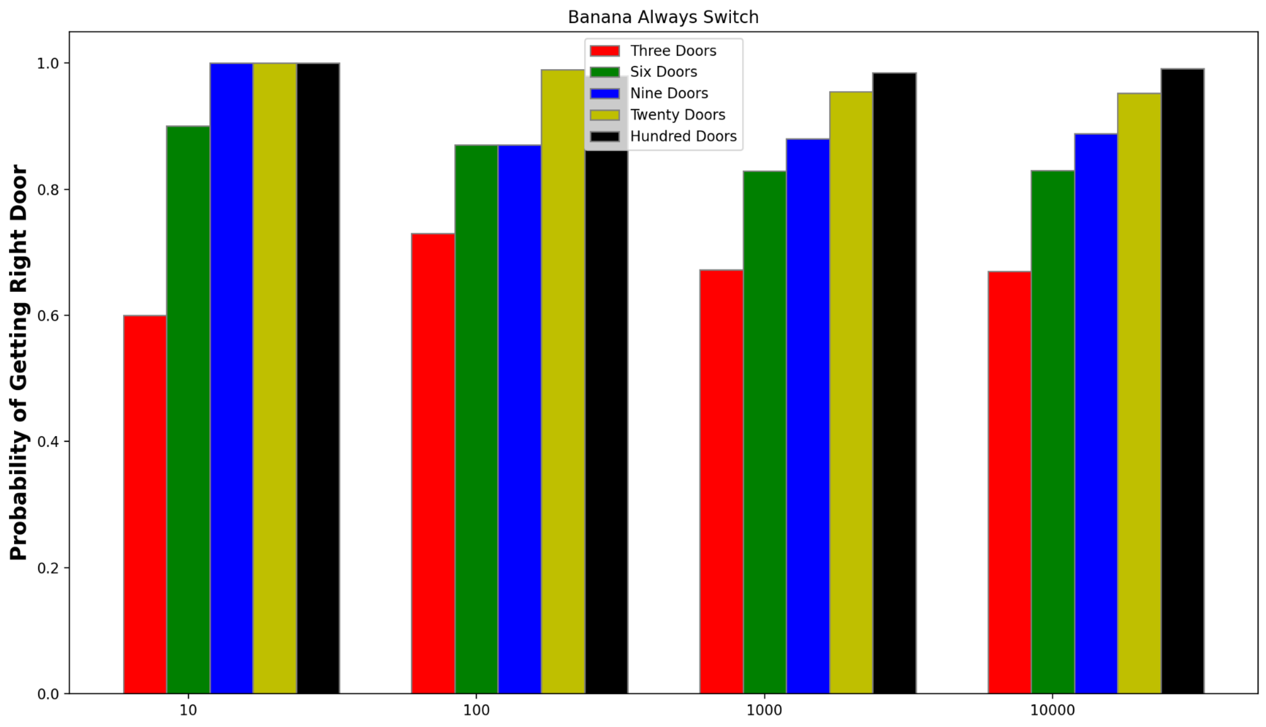


Figure 2.3



5 Conclusion

In conclusion running the Monty Hall problem through a Monte Carlo simulation allows for accurate calculations of these probabilities that would normally seem unconventional. The data obtained followed the logic applied to the problem and further supports the policy to always switch. As well as how far following a simple train of logic can get you.

6 References

1. Weisstein, E. W. (n.d.). Monty Hall Problem. Mathworld.wolfram.com. https://mathworld.wolfram.com/MontyHallProblem.html

7 Reflections

After presenting we edited our slides in order to be more streamlined and included a short but detailed description of the implementation.