Report: Monty Hall/Monty Hall Variant

Motivation/Background Material

The Monty Hall game was a famous television game that was hosted by Monty Hall in the 1980s, it is also well-known because of its probability theory. However, During the 1990s there was a debate about whether there existed a strategy behind this game. Various scientists and mathematicians investigated this game, trying to determine a strategy/method. Shortly after, a woman by the name, of Vos Savant was the first to claim that players should always switch door choices after the host revealed one of the doors to increase the chance of winning the prize. Although it was concluded that this was the strategy for this game, various people, including myself, could not put this into perspective. For that reason, we want to create a simulation to validate this strategy, especially under different conditions. In addition, I would like to determine if adding a variant could affect the probability of winning the prize, and how much of a difference it would make to the famous game show.

Problem Statement

Visualize that you are in the Monty Hall game show, during the game show you are placed in an environment with “n” doors. The number of doors will range from three, six, nine, twenty, and hundred, hence all doors will contain a goat except one, which will contain the car (the grand price). Once all situated, you be given the option to choose a door first, but it will not be opened. Not until the host reveals one or some of the doors depending on the scenario, which will contain a goat. Keep in mind, that the game holds two policies which you either randomly switch to one of the remaining doors or stick with your original selection. Depending on your decision this will affect the probability of your chances of winning the car. However, knowing that there exists a strategy for increasing your chances of winning the car, this strategy has only been applied when playing with three doors. Will this strategy hold true when we add more doors and will it hold true when considering a variant, where the host falls and accidentally opens one of the doors?

Approach

The Monty Hall show game is based on conditional probability, therefore, to simulate this game I will be using the fundamentals of Monte Carlo when creating the Python script. This is because we will be repeating random samples (randomly placing the car) to calculate the probabilities. First, we have two main events, the initial choice of the contestant, then we have the game host revealing one of the doors that doesn’t contain the car. The contestant then is confronted with two policies, either switching to another remaining door or sticking with the original selection. Hence, I would be using two variables to record the number of wins based on whether they switched or didn’t switch (kept the original choice). Then I will use two arrays to maintain the two policies within this game show. One array will record the number of doors in the game and track the contestant’s first and second choice/decision. The second array will be used to track the number of doors and which door the host will reveal, hence it will keep track of where the car is placed assuring the host doesn’t reveal that door. All this will occur within a loop of n iterations, along with n number of doors to simulate the game. Then calculate the overall probability, which is done by the following formula (“contestant’s decision” wins)/# of iteration *100, where “contestant decision” is whether the contestant switched or stayed with the original selection.

A similar approach will be applied to Monty Hall considering the variant experiment. So the same properties will be kept from the Monty Hall experiment except for the array that tracks the doors for the host. This time it will be at random, as the host will continue to slip and fall causing him to open doors at random.

Experiment Set-up

For this experiment, I will be creating a tree probability diagram for both Monty Hall and Monty Hall with the variant. Then after creating the tree diagrams, I will use these diagrams to guide me when programming my Python scripts for each experiment. After implementing the Python script for both the Monty Hall and Monty Hall with variant, I will then run the scripts while recording the data. After, having sufficient data, I will then use this data to convert it into a bar graph, to demonstrate a clear overall representation of my experiment and the outcome.

Results and Discussion

After conducting the experiment, it was clear that switching was the better option. After analyzing

our data from the Python script, and tree probability diagram they illustrated a greater probability if you switched door choices after the host revealed one of the doors with a goat. It is true that before the host reveals one of the doors, there exists a 50/50 probability that you have chosen the car door or a door with a goat. However, this all changes when the host reveals one of the doors with a goat(Figure 1.1). Now, if you switch door choices the probability of winning the car increases much more if you were to stay with the original choice. This remains true for all n number of doors, where n ranges from 3,6,9,20, and 100 doors even when increasing the number of iterations(Figure 1.4). When considering the variant, we obtain a different outcome, compared to the original Monty Hall. When considering the variant, the probability of winning the car, whether you switch or not is the same. After analyzing our Python script, the tree probably diagrams, and data results demonstrate that the host accidentally opening the door at random does dramatically impact that game, as now you went from having a probability of 2/3 to ½ (Figure 1.3), meaning that the strategy that existed in the original Monty no longer applies to this variant, as the game bias no longer exists(Figure 1.2).

Figure 1.1 (Monty Hall Tree Diagram)

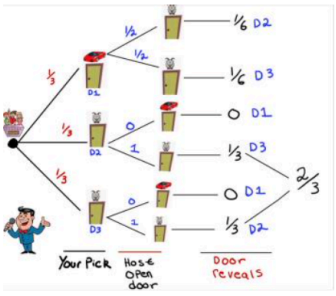

Figure 1.2(Monty Hall Variant)

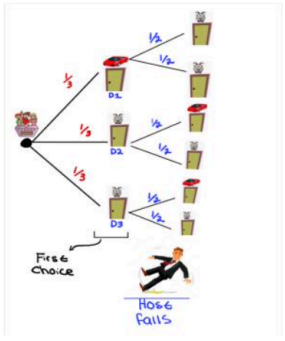

Figure 1.3(Monty Hall Variant data)

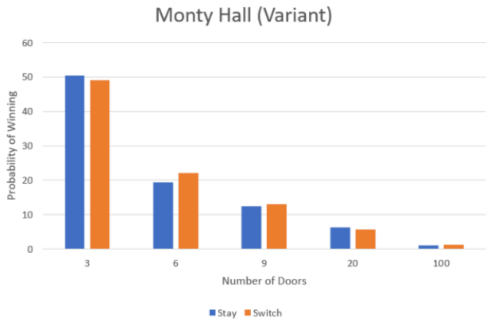

Figure 1.4(Monty Hall Variant data)