The Monty Hall Problem: A Python Simulation Perspective

Rahul Kumar,
Research Student,
Centre for Fundamental Research and Creative Education, Bangalore, India
email: rahulkumar@cfrce.in

Abstract:
The Monty Hall problem is a mathematical problem involving probability. It is often referred to as a paradox, for it is rather counterintuitive. This problem has confused almost all who first attempt to understand it. In order to better understand it, a Python simulation has been made and run to find the correct answer to the Monty Hall problem.

Monty Hall Problem:
The Monty Hall problem itself is rather simple: You are a contestant in a game show, and have to pick one out of three possible doors. Behind one door is a car (the desirable prize) and behind the other two doors are goats (the undesirable prize). After you choose one door, the host of the game show (who knows beforehand which prize is behind which door) opens one of the other doors and reveals a goat (he will always reveal a goat, and never a car). He then asks you if you wish to switch doors. What gives you better odds, switching or not switching?

A common misconception is that the odds stay 50/50 whether or not you switch. As the reasoning goes, there are now two doors and you have chosen one of them. Therefore, you have a 50% chance of winning no matter what you do. This misconception goes on to say that the door that Monty Hall (the host) revealed has no value, as it does not contain the car, nor does it have anything to do with your original guess. However, this path of reasoning is wrong, as this paper will prove.

Python programs:
Two programs were made in Python for the purpose of this paper. The first one simulates the game show: It asks the user for their choice, reveals the goat behind one of the other two doors, and asks if the user wishes to switch. The user is then told whether or not he has won. The user can play as many times as he wishes, and at the end will be given the percentage of winning if he switched or did not switch.

However, this first program is subject to small sample size, unless the user has the time to play over a hundred times. In small samples of a few dozen times or less, the answer may not match up with the actual possibilities, due to luck and normal variance. For this reason, I created a second program, where the user merely inputs how many times he wishes the program to run. For each time through the loop, the car is set in a random door, and the
choice of the participant is also randomized. After that, a goat is revealed. If the winning decision was to not switch, the variable stay would be increased. If the winning decision was instead to switch, the variable switch would be increased. By dividing the variables stay and switch over the variable count (times the program was run), the odds of winning if you switch and didn’t switch can be found accurately. As the number of times the program ran increased, the probability of winning if you switch grew closer and closer to 66.6% (2/3), while the probability of winning if you didn’t switch grew closer and closer to 33.3% (1/3).

Both programs are available in the appendix.

Results:

<table>
<thead>
<tr>
<th>Percentage of Winning per Strategy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of times program ran</td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td>Switch</td>
</tr>
<tr>
<td>Percentage</td>
</tr>
</tbody>
</table>

Analysis:
As you can see, the simulation proves that the correct choice is always to switch, as counterintuitive as that might seem. The key to understanding this is to realize that although there are only two choices, one of those two choices is more likely to occur. For example, on any given moment, there are only two possibilities that could occur regarding lightning striking you: either lightning hits you, or it doesn’t. However, it would be extremely inaccurate to conclude that you have a 50% chance of being hit by lightning each second. In fact, the likelihood of getting hit by lightning is 1 in 700,000 each year. Assuming that the likelihood of getting hit is the same in every second, and that there are 31536000 seconds in a year, the likelihood of getting hit in a particular second is the infinitesimal value of 1/220752000000000.
While the Monty Hall problem is perhaps not so extreme, the same principal applies. While there are only two choices, it is more likely that the correct choice is the door you didn’t choose. Since the host is required to open a door containing a goat, if you originally pick a door containing a goat, the host is forced to open the other goat-containing door, leaving the door with the car the door you could switch to. On the other hand, if you originally picked a car, the host is free to pick either of the two remaining doors to show you, and switching will result in you receiving the goat. Therefore, if you originally pick a goat, you should switch, and if you originally pick the car, you should not switch. All of this seems obvious, but is the key to solving the problem.

Since there are three doors, and only one of them contains the car, you have a 33% chance of picking the car, in which case you would lose if you switch. However, you have a 66% chance of picking the goat, in which case you win if you switch. Therefore, the better choice is always to switch.

This may become clearer if the game is expanded. Assume there are 500 doors, and only one of them has the car. You make a random guess, and the host then opens all but one of the other doors, revealing goats in all of them. If you had the chance to switch, you would definitely do so, with a 99.5% chance of winning, even though it is still a choice between two doors. While the three-door version isn’t that obvious, it follows the same principle. Since the host knows what is behind all of the doors, and has to open all but one of the doors you do not pick (There are two doors left and he opens one), the prize will be behind the other door, unless you are lucky enough to have guessed correctly with your first pick, which is unlikely.

Finally, to make the answer completely clear, I will use another example, with a slightly altered version of the Monty Hall problem, which I will call the Double Monty problem. In this problem, the rules are identical to the original Monty Hall problem, with two exceptions. Firstly, the host allows you to pick two doors, instead of one. Secondly, instead of revealing the goat from a door you didn’t choose, he reveals a goat from a door you did choose. If he offers to allow you to switch your other choice, would you take the offer? The answer to this is clearly no – You had a 2/3 chance of choosing the car, and so you win unless you were unlucky enough to miss with both your attempts, which is unlikely. This problem is the exact opposite of the Monty Hall problem -> the two doors you choose in this problem are equivalent to the two doors you didn’t choose in the Monty Hall problem, and the door you don’t choose in this problem is equivalent to the door you choose in the original problem. Just like you don’t switch here because it’s unlikely you didn’t choose the car, it is better to switch in the original problem, because it’s unlikely you did choose the car.

Conclusion:
I have proven the fact that switching is always preferable by both computer simulations, and logic. I have addressed common misconceptions about the Monty Hall problem, and given example problems that help clear them up.
Appendix:

Program 1 (Simulation of Game show):

```python
import random
from random import randrange

count = 0
stay = 0
switch = 0
choice = 0
exit = 0
switch_win_percent = 0
switch_lose_percent = 0
stay_win_percent = 0
stay_lose_percent = 0
switch_percent = 0
stay_percent = 0
win_if_switch = 0
win_if_stay = 0
total = 0
change = 0
lists = [1, 2, 3]
for i in range(len(lists)):
    lists[i] = int(lists[i])
while True:
    lists = [1, 2, 3]
    choice = input("Pick a number between 1 and 3. If you wish to exit, type done.")
    if choice == "done":
        break
    key = randrange(1, 4)
    choices = int(choice)
    lists.remove(choices)
    print("There is a goat in Door", random.choice(lists))
    change = input("Would you like to switch? Answer yes or no.")
    if change == ("no"):
        if choices == key:
            print ("You win")
            stay_win_percent = stay_win_percent + 1
        else:
            print ("You lose")
            stay_lose_percent = stay_lose_percent + 1
            stay_percent = stay_percent + 1
    if change == ("yes"):
        if choices != key:
            print ("You win")
            switch_win_percent = switch_win_percent + 1
            switch_percent = switch_percent + 1
        else:
            print ("You lose")
            switch_lose_percent = switch_lose_percent + 1
            switch_percent = switch_percent + 1
```

total = total + 1

win_if_switch = (switch_win_percent / switch_percent)
win_if_stay = (stay_win_percent / stay_percent)

win_if_switch = round(win_if_switch, 2)
win_if_stay = round(win_if_stay, 2)
print("If you switch, you win", (win_if_switch * 100), "percent of the time")
print("If you stay, you win", (win_if_stay * 100), "percent of the time")

Program 2(Runs Monty Hall problem as many times as the user wants)

import random
from random import randrange
count = 0
stay = 0
switch = 0
choice = 0
stay_percent = 0
switch_percent = 0
remove = 0
lists = ['car','goat','goat']
repeat = input("How many times would you like to run this program?")
repeat1 = int(repeat)
while count <= repeat1:
    lists = ['car','goat1','goat2']
    key = ('car')
    choice = random.choice(lists)
    if choice == 'goat1':
        lists.remove('goat2')
    if choice == ('goat2'):
        lists.remove('goat1')
    if choice == ('car'):
        remove = randrange(1,3)
        if remove == 1:
            lists.remove('goat1')
        else:
            lists.remove('goat2')
    if choice == key:
        stay = stay + 1
    if choice != key:
        switch = switch + 1
    count = count + 1
switch_percent = (switch/count) * 100
stay_percent = (stay/count) * 100
switch_percent = round(switch_percent, 2)
stay_percent = round(stay_percent, 2)
print("If you switch, then you win", switch_percent, "percent of the time.")
print("If you don't switch, then you win", stay_percent, "percent of the time.")
References: