The Monty Hall Problem: A Study

Michael Mitzenmacher

June 25, 2005

Abstract

The Monty Hall problem is based on apparent paradox that is commonly misunderstood, even by mathematicians. In this paper we define the Monty Hall problem and use a computer simulation to shed light on it. We then provide a mathematical explanation that fits the experimental results.

1 Introduction

The *Monty Hall problem* is based on the following scenario familiar to those who have watched the show *Let's Make a Deal*. The host (Monty) points to three doors and tells you that behind two of the doors await hungry goats, while behind the third door sits a brand new automobile. If you choose the automobile, you get to keep it, and otherwise you are stuck with a goat. Monty asks you to choose a door. After you make your choice, Monty opens a different door and shows you a goat.[1] He now gives you the option of changing doors. What should you do?

A common response, even among professional mathematicians, is to say that there is no difference between switching and not switching. After all, we knew that there was a goat behind one of the two doors we did not choose anyway. Monty's showing us a goat does not give us any new information. Since there are two doors left, the probability of the car being behind each remaining door is now fifty percent. In fact, this argument is wrong, as we shall demonstrate. The correct strategy is to switch, and the odds of winning the car double when the switch is made.

In the rest of this paper, we define the problem carefully, and show the results of a computer simulation. We then explain our results, demonstrating the fallacy in the above argument.

[1] A possible misconception is that Monty opens a random door for you. This is not the case; Monty, knowing what lies behind all doors, chooses a door with a goat behind it in order to build suspense for both the studio audience and the viewers at home.

2 Materials and Methods

We began to test two possible strategies (always switch or always stay) using playing cards. Three cards (an ace, king, and queen) were shuffled randomly by a dealer who knew the position of each card, while a player tried to find the ace. Preliminary results demonstrated that the strategy of always switching fared better, but playing it out by hand proved too slow to get meaningful results.

To get more data, we wrote a program in C to play the game repeatedly. The program uses pseudo-random numbers generated by the functions *rand* and *srand* [2], available in standard Unix-based systems. Although the numbers generated are not truly random, they are sufficiently random for the purposes of this experiment. The program is designed so that it is easy to change the number of times the game is played and the number of cards used.

3 Results and Data

The program was set to generate samples of one million games. In a half dozen runs, the percentage of times the switching strategy proved successful ranged from 66.586% to 66.687%. The strategy of always staying with the original choice succeeded between 33.313% and 33.414%.

We also changed the program to simulate the game with thirteen cards. Here the switching strategy still provides an edge, although less dramatically so. The switching strategy succeeded approximately 8.3 to 8.4% of the time, while staying succeeded approximately 7.7% of the time.

4 Discussion

We now provide both the intuition and a mathematical justification for our results. The fallacious argument described in the introduction suggests that by showing us a goat, Monty has not given us any new information, since we knew there was a goat behind one of the doors anyway. This reasoning is faulty. Simply by listing the possibilities one finds in the three-door case one finds that the switching strategy works whenever the staying strategy fails. Staying clearly succeeds one time out of three, so switching works two times out of three, matching the experimental evidence.

One way to see the problem with this logic is to expand the problem to one hundred doors. After we choose a door, suppose Monty shows us ninety-eight goats, leaving just our door and one other. Surely one would want to switch in this case. Even though we knew that ninety-eight doors hid goats, by showing us the goats, it seems that Monty has given us something.

The key to the problem is a *restricted choice*. Suppose you choose the first door. If it hides a goat, Monty now has no choice of which door to open for you. His choice was thus

restricted. If instead your door holds the car, Monty does have a choice of doors to open for you.

So suppose that Monty opens the second door and shows you a goat. You have, in fact, gained some information, namely that the second door did not originally hide the car. But why does this make the third door more likely to hold the car? Because if a goat waited behind the third door, Monty might have shown you it instead of opening the second door. This is a difficult point to understand, yet it is the key to the problem. Essentially, it pays to assume Monty did not have a choice of which door to open, because if he did, he might have chosen differently. This phenomenon of restricted choice occurs in many game situations. For example, it is a well-known phenomenon in the game of bridge. [1]

A more mathematical derivation is provided in the appendix.

5 Conclusion

We have explained the Monty Hall problem and given evidence based on a computer program for the correct answer to the puzzle. Besides providing a mathematical treatment, we suggest that the intuitive concept of restricted choice is the key to understanding the Monty Hall problem and similar situations.

6 Acknowledgements

I would like to thank the other teachers for offering valuable corrections to the early drafts of the paper.

References

[1] The Official Bridge Encyclopedia, 1982.

[2] UNIX manual pages.

Appendix

We give a mathematical justification of our results. More formally, suppose one begins with n doors. The probability of choosing correctly to begin with is thus $1/n$ and the probability of choosing incorrectly is $(n-1)/n$. When you are wrong initially, if you switch after Monty shows a goat you win $1/(n-2)$ of the time. Thus the total probability of success by switching is $(n-1)/(n \times (n-2))$. A simple check shows that switching is preferable and that this formula matches our results extremely well.