Playing and analyzing the Monty Hall problem

Table of Contents

```
%Clear the workspace before running anything
clear
close all
```

Description of the Problem

The Monty Hall problem is a well known situation where you are a participant in a game show, where there are three doors closed. Behind one door, there's a Sports Car, and behind the other two, there are goats. The sequence of events are as follows:

1. You as a contestant can choose a door - and that's currently the one you guess to have the Sports Car behind it.
2. Then, the host of the game show opens a door, specifically one which you have not chosen - and shows you that there's a goat behind it. The option you have now as the contestant is that you can either choose to keep your current door choice, or switch to the other door.
3. The question now is - would you choose to keep the existing door, or switch? In either case, what is the rationale to do so?

More specifically, lets say there are three doors and only the host knows that one of them has the sports car behind it. As a contestant, you then choose a door - not knowing which of the three doors hides the car. Let's say that you chose door 2. Now, the host knows that door 3 does not have a car, and so goes ahead and opens it.

The question for you then is, should you stick on to your door, or, switch? In this instance, the right choice is to switch - but you didn't know which door held the sports car. Would you switch then?

Let's play this game, shall we?

First, we will setup the game:

```
nDoors = 3;
game = montyGame(nDoors);
```

Then, we choose a door (integer, between 1 and 3):

```
selectedDoor = 3;
```

Now, we ask the host to open a door:

```
game = game.openDoor(selectedDoor);
fprintf("Opened door is: %d\n", game.openedDoor)
```

```
Opened door is: 1
```

Now - will you keep your original door, or, switch?

```
% Indicate the new door to open - keep same as selectedDoor if you're not
% switching. Make sure this isn't the opened door - you're guaranteed a
% goat, then!
switchedDoor = 2;
```

```
% Then, reveal the doors.
game.revealDoor(switchedDoor)
```

```
Yay - you won the car!
```

Some intuition for Monte Carlo Simulations

So the larger question then is - how can we make this choice? Is there a strategy that works better than the other more generally? Is this a matter of random choice, and so either of the two choices is the same then?

One way to find out what is a good strategy is to simulate this for a large number of repeats, and then find out which of the two choices is a good one. In the Mathematics world, such a method is called a Monte-Carlo simulation.

Here is how these experiments can be run:

To generalize, if there are nDoors number of doors:

```
randi(nDoors, 1)
```

will return a uniformly sampled random number between 1 and nDoors. We can use this random number generator to generate a door number that the game show organizers choose to hide the Sports Car behind, another of these samples to simulate what a contestant will choose - as we are assuming there is no prior knowledge about any of the doors, and another one, to choose the door that the host opens. Then, we can finally also use another of these random numbers to choose the door the contestant switches to - in case they so choose to. All this while, we make sure that the actual door which hides the Sports Car isn't the one that is opened. If the random number that is generated to open the door matches the actual door, we simply re-generate the random number.

Setting up the Experiment

By doing this over and over again, and recording how many times the contestant won the Sports Car and how many times they did not win, we can generate an estimate for the probability of winning based on the strategy used - either switch or not. This is what is done below:

```
% Number of times experiment repeats.
```

```
numRepeats = 10000;

% This code handles generalized number of doors. In the gameshow it is 3.
nDoors = 3; % put a slider here TODO
```

Run the experiment

We've written a function called montyHall which does the computations for generating one run of this experiment. Its inputs include the number of doors as the first input (a positive integer), and a flag to decide if the strategy is to switch to another door, or, to stay put with the original selection (logical scalar). In the first set of lines, we simulate the switching strategy, and count the number of wins (or proportion/percentage).

```
% After switching doors.
numWins = 0;
for idx = 1:numRepeats
    numWins = numWins + montyHall(nDoors, true);
end
fprintf("Percentage of wins: %f\n", numWins*100/numRepeats)
```

```
Percentage of wins: 66.310000
```

Now, we redo the experiment without switching the door.

```
% No switching doors.
numWins = 0;
for idx = 1:numRepeats
    numWins = numWins + montyHall(nDoors, false);
end
fprintf("Percentage of wins: %f\n", numWins*100/numRepeats);
```

```
Percentage of wins: 33.550000
```

Notice that the chances of winning in the case where the switching strategy is used is always higher than that of the 'stay put' strategy. This is a non-intuitive side effect of the change in probability of which door could have the Sports Car hidden behind it after the host opens one of the doors that does not have the car behind it. In fact, the ratio of the number of wins to the number of repeats of the experiment is an *estimate* of the probability of that event occurring.

Helper functions:

```
function out = montyHall(nDoors, switchDoor)

    % Decide which door actually has the Sports Car.
    actualDoor = randi(nDoors, 1);

    % User selects a random door.
    selectedDoor = randi(nDoors, 1);

    % Choose which door to open.
    openedDoor = randi(nDoors, 1);
    while (openedDoor == actualDoor) || (openedDoor == selectedDoor)
        openedDoor = randi(nDoors, 1);
    end

    % Make decision to remain fixed or switch.
    if switchDoor
        % Any other door that's switched too is fine.
        oldSelectedDoor = selectedDoor;
        switchDone = false;
        while ~switchDone
            selectedDoor = randi(nDoors, 1);
            if (selectedDoor ~= oldSelectedDoor) && (selectedDoor ~= openedDoor)
                switchDone = true;
            end
        end
    else
        % do nothing.
    end

    % Check if the actual door is still the selected door.
    if actualDoor == selectedDoor
        out = 1;
    else
       out = 0;
    end
end
```