The Monty Hall Problem

Dublin R

May 29, 2013

1 The Monty Hall Problem

Suppose that there are three closed doors on the set of the Let's Make a Deal, a TV game show presented by Monty Hall. Behind one of these doors is a car; behind the other two are goats. The contestant does not know where the car is, but Monty Hall does.

The contestant selects a door, but not the outcome is not immediately evident. Monty opens one of the remaining "wrong" doors. If the contestant has already chosen the correct door, Monty is equally likely to open either of the two remaining doors.

After Monty has shown a goat behind the door that he opens, the contestant is always given the option to switch doors. What is the probability of winning the car if she stays with her original choice? What if she decides to switch?

1.1 The Sample Space

One way to think about this problem is to consider the sample space, which Monty alters by opening one of the doors that has a goat behind it. In doing so, he effectively removes one of the two losing doors from the sample space. We will assume that there is a winning door and that the two remaining doors, A and B, both have goats behind them. There are three options:

- The contestant first chooses the door with the car behind it. She is then shown either door A or door B, which reveals a goat. If she changes her choice of doors, she loses. If she stays with her original choice, she wins.
- The contestant first chooses door A. She is then shown door B, which has a goat behind it. If she switches to the remaining door, she wins the car. Otherwise, she loses.
- The contestant first chooses door B. She is then is shown door A, which has a goat behind it. If she switches to the remaining door, she wins the car. Otherwise, she loses.

Each of the above three options has a 1/3 probability of occurring, because the contestant is equally likely to begin by choosing any one of the three doors. In two of the above options, the contestant wins the car if she switches doors; in only one of the options does she win if she does not switch doors. When she switches, she wins the car twice (the number of favorable outcomes) out of three possible options (the sample space). Thus the probability of winning the car is 2/3 if she switches doors, which means that she should always switch doors - unless she wants to become a goatherd.

1.2 Counter Intuitive Result

This result of 2/3 may seem counterintuitive to many of us because we may believe that the probability of winning the car should be 1/2 once Monty has shown that the car is not behind door A or door B. Many people reason that since there are two doors left, one of which must conceal the car, the probability of winning must be 1/2. This would mean that switching doors would not make a difference. As we've shown above through the three different options, however, this is not the case.

1.3 R Implementation (part 1)

We have 3 doors to choose from, so we will define a numeric sequence between 1 and 3. The command `sample(,n)` takes a sample of size n from a specified set of values. Here we just want to select one door to be our ***correct door*** and another to be ***selected*** door (i.e. the door that the contestant selects).

These events are independent. For the sake of simplicity, we will perform the selection for both doors separately, but this can be implemented in one command.

```
}
Doors = 1:3
Correct = sample(Doors,1)
Choice = sample(Doors,1)
```

```
> Doors
[1] 1 2 3
> Correct
[1] 1
> Choice
[1] 3
```

1.4 R Implementation (part 2)

In this part, we will use the set theory commands `union()` and `setdiff()`. Another related command is `intersect()`, but we will not be using it here.

These commands are very simple, and can be best demonstrated with a simple example.

```
X <- 1:6
Y <- 5:9
union(X,Y)
intersect(X,Y)
setdiff(X,Y)
setdiff(Y,X)
```

The output from these commands looks like this:

```
> X
[1] 1 2 3 4 5 6
> Y
[1] 5 6 7 8 9
>
> union(X,Y)
[1] 1 2 3 4 5 6 7 8 9
> intersect(X,Y)
[1] 5 6
> setdiff(X,Y)
[1] 1 2 3 4
> setdiff(Y,X)
[1] 7 8 9
```

Going back to the Monty Hall problem. The Set Theory commands aver very useful in that they can avoid a lot of programming control statements to account for various scenarios. A wrong door must be selected to be opened. The door selected by the contestant can't be chosen. First let us select the doors that must stay closed, then find the doors that the contestant can choose from.

```
StayClosed = union(Correct, Choice)
CanOpen = setdiff(Doors, StayClosed)
```

1.5 Using the `sample` command

The `R` command `sample()` poses an interesting problem. Lets look at the help file to see what it is. Create a single element vector (let's call it **Vec**, with that single element being 4.

Now sample a single value from that data set. You may expect to get 4 each time, as it is the only element in the set but `R` doesnt work that way in this instance.

```
Vec=c(4)
sample(Vec,1)
```

Instead `R` interprets the argument 4 as the upper bound of a sequence of four integers (i.e. the sample space is 1 to 4)

```
> sample(Vec,1)
[1] 3
> sample(Vec,1)
[1] 4
> sample(Vec,1)
[1] 1
```

Problems will arise when the number of elements in the **CanOpen** set of numbers is one. When there are more than one, things will work fine. To overcome thisproblem, we need some control statements.

- If there are two elements in the data set, we just select one of them at randomm using the `sample()` command.
- If there is only one element in the data set we avoid using the `sample()` command, and instead use the assignment operator.

We wish to select a door for opening, to reveal the goat.

```
if(length(CanOpen)>1)
{
Open = sample(CanOpen,1)
}else {Open=CanOpen}
Open # This door reveals a goat
```

1.6 Stick or Switch?

There are two unopened doors. One is the correct door, but which one? Should the contestant **stick** or **switch** ?

```
NotOpen = setdiff(Doors,Open)

Stick = Choice
# Stick with the original choice

Switch = setdiff(NotOpen,Choice)
# Switch to the other door
```

Was sticking the right decision? Or was switching?

```
# Was sticking the right decision? Or was switching?
# The following logicial statements will tell us.
Stick==Correct
Switch==Correct
```

An answer of 1 means that decision would have been correct. Equivalently, a response of 0 indicates a bad decision.

1.7 Writing a Function

We are going to create a function `MHfunc()` to help us simulate a solution for the Monty Hall Problem. The function is constructed using R code we have seen already. The output of the function is returned as a data frame, with 7 columns:

- Correct : The number of the correct door.
- Choice : The door that the contestant chose originally, and the door selected if the contestant decided to "stick".
- Open : The door Monty Hall opens to reveal a goat
- StickDoor : The door that the contestant chose originally, and the door selected if the contestant decided to "stick".
- SwitchDoor : The door selected if the contestant chose to "switch".
- Stick: Was Sticking a good decision (1 for yes, 0 for no)
- Switch : Was Switching a good decision (1 for yes, 0 for no)

The output looks like this:

```
> MHfunc()
   Correct     Choice Open(goat)  StickDoor SwitchDoor      Stick     Switch
         2          2          1          2          3          1          0
> MHfunc()
   Correct     Choice Open(goat)  StickDoor SwitchDoor      Stick     Switch
         1          2          3          2          1          0          1
> MHfunc()
   Correct     Choice Open(goat)  StickDoor SwitchDoor      Stick     Switch
         1          2          3          2          1          0          1
```

```
MHfunc <- function(numdoors=3){
Doors = 1:numdoors
Correct = sample(Doors,1)
Choice = sample(Doors,1)
StayClosed = union(Correct, Choice)
CanOpen = setdiff(Doors, StayClosed)

#Open the Door with a Goat
if(length(CanOpen)>1)
{
Open = sample(CanOpen,1) #Problem here
}else {Open=CanOpen}

NotOpen = setdiff(Doors,Open)

#Stick or Switch
Stick = Choice
Switch = setdiff(NotOpen,Choice)
Stick.Decision= as.numeric(Stick==Correct)
Switch.Decision = as.numeric(Switch==Correct)

#Preparing the Output
MHoutput=c(Correct,Choice,Open,Stick,Switch,
 Stick.Decision,Switch.Decision)

names(MHoutput)= c("Correct","Choice","Open(goat)","StickDoor",
 "SwitchDoor","Stick","Switch")
return(MHoutput)
}
```

1.8 A Simulation Study of the Monty Hall Problem

Let us perform a simulation study of the Monty Hall problem. We are putting the code we have written already, placed in a for loop, although we alter the ending to make for more readable output.

```
#Initialise Counters
StickCount = 0
SwitchCount = 0

#How many Trials?
M=1000

# a For loop
for(i in 1:M)
{
output=MHfunc()

StickCount = StickCount + output[6]
SwitchCount = SwitchCount + output[7]
}
StickCount
SwitchCount
```

The output of this program would look like this. The proportion is approximately 2 to 1 in favour of the switching option.

```
> StickCount
[1] 323
> SwitchCount
[1] 677
```