Learning Outcomes

1. Become familiar with writing a small program in R Studio
2. Begin to be exposed to R syntax and use of functions
3. Understand how to break a problem down into small steps

Today we are going to write a program to simulate the Monty Hall Game.

**Initial Conditions**

The first thing we want to do is to set up our three doors. The following code creates a vector named door that contains the numbers 1, 2, and 3.

door <- c(1,2,3)

Next, we need to randomly select one of the doors to have the car behind it with the following code.

cardoor <- sample(door,1)

The variable cardoor now contains a randomly selected value from those in the vector door.

We now need to randomly select the contestants choice of door using similar code.

choice <- sample(door,1)

Now we have a vector of doors, a variable indicating which door has the car, and a variable indicating which door was chosen by the contestant.

Run this block of code and make sure these three objects appear in the Global Environment

**The Reveal**

If the contestant has chosen the door with the car, we can reveal either of the goat doors. However, if the contestant chose a door with a goat, we have only one option for a door to reveal (since we will never reveal the car.

We can handle this with an if else loop.

First, lets create a new vector that holds the values corresponding to goats using the following code:

goatdoors <- setdiff(door, cardoor)

Type ?setdiff in the command panel to learn more about what this function does.

Next, we will identify the options we have for the reveal using similar code.

reveal\_options <- setdiff(goatdoors, choice)

Now we can implement our if else loop.

First, we have the situation where there are two goats to choose from, and we select one randomly and assign it to the variable reveal.

if (choice == cardoor) {

reveal <- sample(reveal\_options,1)

}

Alternatively, there will be only a single element in reveal\_options which we assign to reveal.

if (choice == cardoor) {

reveal <- sample(reveal\_options,1)

} else {

reveal <- reveal\_options

}

Finally, we can create a new vector which identifies the two remaining unrevealed doors

remaining\_doors <-setdiff(door, reveal)

Run this block of code and make sure all the objects you expect appear in the Environment.

**Switch or Stay**

Lets create a new variable recording the final choice of door if the contestant switches.

newchoice <- setdiff(remaining\_doors, choice)

Recall that the variable choice records their original door.

We can now print the results of the game using two more if else loops.

(note: if you copy past this, make sure your quotation marks are showing up as straight and not curly in the R window. Otherwise you may get an error.)

if (choice == cardoor) {

print(“Stay: You got a car”)

} else {

print(“Stay: You got a goat”)

}

if (newchoice == cardoor) {

print(“Switch: You got a car”)

} else {

print(“Switch: You got a goat”)

}

Run this block of code and make sure all it works.

**Multiple Games**

So far, we have written code for a single play of the game. But if you want to see how the game turns out over multiple plays, you need to do a little more work. The code below runs 100 games and prints the results. Try to understand what each piece of code is doing.

n\_stay <- 0

n\_switch <- 0

for ( i in 1:100) {

door <- c(1,2,3)

cardoor <- sample(door,1)

choice <- sample(door,1)

goatdoors <- setdiff(door, cardoor)

reveal\_options <- setdiff(goatdoors, choice)

if (choice == cardoor) {

reveal <- sample(reveal\_options,1)

} else {

reveal <- reveal\_options

}

remaining\_doors <-setdiff(door, reveal)

newchoice <- setdiff(remaining\_doors, choice)

if (choice == cardoor) {

n\_stay <- n\_stay + 1

}

if (newchoice == cardoor) {

n\_switch <- n\_switch + 1

}

}

print(n\_stay/100)

print(n\_switch/100)