Problem Set 4

(Due Feb. 22, 1:00 PM)

Instructions

1. The following questions should each be answered within an R script. Be sure to provide many comments in the script to facilitate grading. Undocumented code will not be graded.
2. Work on git. Fork the repository found at `https://github.com/minheeseo/PS4` and add your code, committing and pushing frequently. Use meaningful commit messages – these may affect your grade.
3. You may work in teams, but each student should develop their own R script. To be clear, there should be no copy and paste. Each keystroke in the assignment should be your own.
4. If you have any questions regarding the Problem Set, contact the TAs or use their office hours.
5. For students new to programming, this may take a while. Get started.

Let's Make a Deal 2[1]

In the game show "Let's Make a Deal", the candidate gets to choose one of three closed doors, and receives the prize behind the door they choose. Behind one door is a new car; behind the other two doors are goats. After the contestant selects one of the 3 doors, the host opens one of the other two doors, and reveals a goat. Now, the candidate has the option of either sticking with the door they originally selected, or switching to the only other door that is still closed. What should the candidate do, and why? What are the probabilities of winning the car if they stay versus if they switch? This question has become known as the Monty Hall Problem.

For this problem set, you *will* solve the Monty Hall Problem via simulation. In addition to solving this problem, this assignment has a few additional requirements:

[1] `https://en.wikipedia.org/wiki/Let's_Make_a_Deal`

- You are allowed to use *no* for loops. All iterative tasks must by done with either apply-family or plyr functions.
- Your grade will be in part on your ability to show how you de-bugged your own code in the process of building the program. You are required to make commit that show the use of `debug()`, `traceback()`, and *browser()*.

Getting started

I wrote the following code as a 'first cut' at the problem. I want `myFunction` to take in a choice of a door and the location of the car. It should return a `TRUE` if they are equal and a `FALSE` if they are not. But my function doesn't seem to be working. Fix my code so it works right, and while you are at it, re-write it so it is easier to read/understand. This includes improving the naming convention, indenting, commenting, etc.

```
myFunction<-function(doorthing, doorthing2, x){
  doorthing1<-doorthing2<-sample(1:3, 1)
  if (doorthing1==doorthing2){ x<-TRUE } else { x==FALSE }
  x
}
myFunction(sample(1:3, 1), sample(1:3, 1))
# Should return a TRUE if these samples are equal and
# a false if they are not
```

Moving on

1. Define a new S4 class `door`. The class should have three slots. `chosenDoor` should be an integer (1, 2, or 3). `carDoor` should be an integer (1, 2, or 3). `switch` should be a boolean indicating whether the player adopted the strategy of staying with their first choice (`switch=FALSE`) or switching to the remaining door (`switch=TRUE`). *winner* should be a boolean indicating whether or not the door that was finally chosen was the same as the door where the car is.

    - a construction function that allows the user to create a `door` object,

- a validation function that checks whether the values stored in the slots are appropriately structured.

2. Create a method for `door` objects that is called `PlayGame` (you will first need to create the generic). This method is supposed to do the following:

    - draw a random number between 1 and 3 that presents the door behind which the car is hidden and add this to the `carDoor` slot
    - draw a random number between 1 and 3 that presents the door chosen first.
    - if `switch=FALSE`, this random draw gets added to the `chosenDoor` slot
    - if `switch=TRUE`, a door is chosen at random such that (a) it does not contain the car and (b) it is not the first door chosen by the contestant. This door is "removed" from contention. The player then chooses at random between doors not removed. This is added to the `chosenDoor` slot.
    - compare the two door slots. If they are the same, then change `winner` to TRUE. If it is not the same, change winner to `FALSE`.

Simulation

1. Run a simulation of the game 1,000 times where the players choose *not* to switch. Evaluate the percentage of the time they win the car.
2. Run a simulation of the game 1,000 times where the players choose to switch. Evaluate the percentage of the time they win the car.
3. Which strategy is best?