Applied Statistical Programming - Spring 2022

Problem Set 3

Due Wednesday, March 2, 10:00 AM (Before Class)

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/johnsontr/AppliedStatisticalProgramming2022 and add your code for Problem Set 3, committing and pushing frequently. Use meaningful commit messages because these will affect your grade.
3. You may work in teams, but each student should develop their own Rmarkdown file. To be clear, there should be no copy and paste. Each keystroke in the assignment should be your own.
4. For students new to programming, this may take a while. Get started.

Let's Make a Deal[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 is known as the Monty Hall Problem.

Your tasks

For this problem set, you will not solve the Monty Hall Problem, but you will have to code a slightly simplified version of the "Let's Make a Deal" game. More specifically, you will set up a new class, which contains information regarding the door a player chooses, and a method that simulates a modified version of the game. You will have to do this using the S3 class system. Here are the specific instructions:

1. Define a new class: `door`. Objects of this class simply take on one numeric value: 1, 2, or 3 – indicating which door a candidate chooses.
2. Create a method for `door` objects that is called `PlayGame`. This method is supposed to do the following:
    - take the numeric value that is stored in the `door` object,
    - draw a random number between 1 and 3 that presents the door behind which the car is hidden,
    - compare the two numbers, and print a message congratulating a winning candidate that chose the correct door, or expressing sympathies for a losing candidate that chose the wrong door.
3. Write:
    - a construction function that allows the user to create a `door` object,
    - and a validation function that checks whether the value stored in `door` is actually an integer

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

1.

```
# Create a new object called ContestantDoor ContestantDoor has value of 1, 2,
# or 3, indicating the door chosen The options are listed here. Player can see
# these option and choose which ContestantDoor to 'open', by deleting the
# comment symbol of the chosen ContestantDoor and making sure the others are
# commented-out. For now, I will choose ContestantDoor = 1 (leaving it
# un-commented)
ContestantDoor <- 1
# ContestantDoor <- 2 ContestantDoor <- 3

# create class 'door' and assign it to the object ContestantDoor
class(ContestantDoor) <- "door"

# Test object ContestantDoor for correct value and class ContestantDoor
# str(ContestantDoor)
```

2.

```
#Create a new method called PlayGame
PlayGame <- function(door) {
  UseMethod("PlayGame")
} #end of PlayGame function to create method


#Fashion the method PlayGame for when it is used on objects of door class
PlayGame.door <- function(door) {
  #Store the value of Door chosen by the contestant, and store it as Chosen
  Chosen <- door

  #Draw a random number, 1-3, to indicate where the prize actually is (behind door 1, 2, or 3).
  #Call this random number Prize
  #Use sample function to draw from the numbers 1,2,3, with a sample size of 1.
  Prize <- sample(c(1, 2, 3), 1)

  #Create an if statement
  #If the values for Chosen and Prize are the same
  if (Chosen == Prize) {
    #Print a "congratulations" message.
    print("Congratulations! You've won yourself a new car!!")
  } #end of "if" statement

  #If the values for Chosen and Prize are not the same
  else if (Chosen != Prize) {
    #print a "sympathies" message.
    print("We're sorry. You did not win this time. Please try again!")
  }#end of "else if" statement

} #end of PlayGame function


#Test PlayGame method with object ContestantDoor. If the random prize matches
#the value of the ContestantDoor object, then the function will print congrats message.
#If it does not match, the function will print sorry message.
#For example: If ContestantDoor in question 1 was chosen to have a value of 1,
```

```r
#then PlayGame will print the congrats message if the random prize matches 1.
#If the random prize is 2 or 3, then the sorry message will print
#PlayGame(ContestantDoor)


#Now create a function that allows the user to create their own door object, called Door
Door <- function(x) {
  #Make sure that the function's input is a numeric. If it is not, throw an error
  if (!(is.numeric(x))) {
    stop("Error: x is not a numeric")
    }

  #Make sure that the function's input is of value 1, 2, or 3. If it is not, throw an error.
  #To do this create a vector of values 1, 2, and 3, called CorrectValue
  CorrectValue <- c(1, 2, 3)
    #Now, compare the input of the function to the CorrectValue object
  #If the input is not contained in CorrectValue (meaning, it is not the number 1, 2, or 3),
  if (!(x %in% CorrectValue)) {
    #throw an error message
    stop("Error: x does not have a value of 1, 2, or 3")
  }

  #Assign class of "door" to the input
  class(x) <- "door"

  return(x)
} #end of Door function


#Test the Door function with different values
#x <- "a"
#Door(x) should throw an error for not being a numeric
#Door(x)
#Door(12) should throw an error for being out of range
#Door(12)
#Door(1) should work
#Door(1)
#Door(2) should work
#Door(2)
#Create object d, using Door function, with value of 3
#d <- Door(3)
#should be value 3 and of class "door"
#d
#Door(2.5) should throw an error for not being a whole number
#Door(2.5)


#Test the door function with the PlayGame method
#Test PlayGame method on the Door object created
#PlayGame(Door(1))
#Test PlayGame method on object d created above with the Door function
#PlayGame(d)
```

```r
#Create a validator function, which will verify that the Door object is an integer
#In this case, integer just means a whole number (instead of integer "type")
validate_door_integer <- function(x){
  #verify that the input is a number
  #if the type input is not a numeric
  if(!(is.numeric(x))) {
    #throw an error message
    stop("Object is not a numeric")
  }

  #verify that the input is a whole number
  #use the remainder function
  #if the input has a remainder that does not equal 0
  if(x %% 1 != 0) {
    #throw an error message
    stop("Object is not a whole number")
  }

  #if it is in the correct form, a numeric and a whole number
  else {
    #print an affirmation message
    print("Yes, this is in integer form")
  }
} #end of validator function

#Test validator
#Make an object, MyDoor equal to 1.5
#MyDoor <- 1.5
#validotor should throw a code for MyDoor not being a whole number
#validate_door_integer(MyDoor)
#Test with ContestantDoor created in question 1
#validator should print an affirmation message
#validate_door_integer(ContestantDoor)



#Create a validator function, which will verify that the object is of class "door"
validate_door <- function(x){
  #if the class is not "door" throw error message
  if(class(x) != "door") {
    stop("Object is not of class door")
  }

  #if the class is "door" print an affirmation message
  else { print("Yes, this is a door")
  }
} #end of validator function

#Test validator with Door
#validate_door(Door(1))
```