Lab 07 - Simulation

Francisco Santamarina

November 07, 2016

Load the necessary packages and dataset.

```
##load data
library( scales )
```

Question 1

One function that sets up the game (three doors, one car, two goats).

```
createGame <- function( )
{
    a.game <- sample( x=c("goat","goat","car"), size=3, replace=F )
    return( a.game )
}

#TEST
createGame()
```

```
## [1] "car"  "goat" "goat"
```

Question 2

One function that selects a door for your first guess.

```
pickDoor <- function( a.pick=sample( x=c(1,2,3), size=1, replace=F) )
{
  #user selects 1,2, or 3
  #if( a.pick != c("1","2","3") )
  if( a.pick == "1" )
  {
    return( a.pick ) # returns 1
  }
  if( a.pick == "2" )
  {
    return( a.pick )  # returns 2
  }
  if( a.pick == "3" )
  {
    return( a.pick )  # returns 3
  } else
  {
    return( "Please choose an integer from 1 to 3")
  }
}
```

```
#TEST
pickDoor( )
```

```
## [1] 3
```

Question 3

One function that opens a door with a goat.

```
this.game <- createGame()
this.game #returns stored value for a specific game/sequence of goat-goat-prize
```

```
## [1] "goat" "goat" "car"
```

```
my.pick <- pickDoor()
my.pick #returns stored value for a specific initial door pick of 1:3
```

```
## [1] 1
```

```
openGoatDoor <- function( game, a.pick )
{
   doors <- 1:3

   if( game[ a.pick ] == "goat" )
   {
      opened.door <- doors[ doors != a.pick & game != "car" ]
   }
   if( game[ a.pick ] == "car" )
   {
      one.of.these <- doors[ doors != a.pick & game != "car" ]
      opened.door <- sample( x=one.of.these, size=1, replace=F )
   }
   return( opened.door )
}

#TEST

openGoatDoor( game=this.game, a.pick=my.pick )
```

```
## [1] 2
```

Question 4

One function that makes your final pick (depending upon if you intend to stay or switch).

```
changeDoor <- function( stay=T, opened.door, a.pick )
{
  doors <- 1:3
```

```
  # final.pick is equal to the doors value that is not the first pick and not the opened door
  if( stay == T )
  {
    final.pick <- a.pick #returns the same value as the initial pick
  }
  if( stay == F )
  {
    final.pick <- doors[ doors != a.pick & doors != opened.door]
    #will return the door number that was not the initial pick AND not the revealed door
  }
  return( final.pick )  # number between 1 and 3
}

#TEST
opened.door <- openGoatDoor( game=this.game, a.pick=my.pick )
changeDoor(stay=F, opened.door=opened.door, a.pick=my.pick)
```

```
## [1] 3
```

Question 5

One function that decides if you win the car or not.

```
determineWinner <- function( final.pick, game )
{
   if( game[ final.pick ] ==  "car")
   {
     return( "W" )
   } else
   {
     return( "L" )
   }

   #if( game[ final.pick ] == "goat")
   #{ return( "L" )  }
}

#TEST
picked.final <- changeDoor(stay=F,opened.door=opened.door, a.pick=my.pick)
determineWinner( final.pick=picked.final, game=this.game )
```

```
## [1] "W"
```

Question 6

One final function that runs the game a specified number of times (one argument of the function), allows the user to specify their strategy (stay or switch), and returns a vector of wins and losses.

```
playTheGame <- function( num.sims=10000, stay=T )
{
   results <- NULL   # create a vector to store results
   doors <- 1:3
   for( i in 1:num.sims )
   {
      game <- createGame()
      picked.door <- pickDoor()
      opened.door <- openGoatDoor( game, picked.door )
      picked.final <- changeDoor(stay=stay, opened.door, picked.door)
      win.lose <- determineWinner( picked.final, game )
      results <- c( results, win.lose )
   } # end of loop
   return( results )
} # end of function

#TEST
results.stay <- playTheGame( num.sims = 10000, stay = T  )
table(results.stay)
```

```
## results.stay
##    L    W
## 6736 3264
```

```
results.switch <- playTheGame( num.sims = 10000, stay = F)
table(results.switch)
```

```
## results.switch
##    L    W
## 3312 6688
```

Question 7

** A table or graph representing the results after you have run 10,000 trials with the strategy of stay, and 10,000 trials with the strategy of switch. Your table or graph will represent the proportion of times each strategy was successful.**

```
## STAY STRATEGY

par( mfrow = c(2,1), mar=c(3,2,2.4,2), ylim = c(0,15))
```

```
## Warning in par(mfrow = c(2, 1), mar = c(3, 2, 2.4, 2), ylim = c(0, 15)):
## "ylim" is not a graphical parameter
```

```
results.stay <- playTheGame( num.sims = 10000, stay = T  )
table(results.stay)
```

```
## results.stay
##    L    W
## 6664 3336
```

```
monty.wins.stay <- prop.table( table( results.stay ) )
rnd.monty.stay <- round( monty.wins.stay, 3 )
rnd.monty.stay <- as.numeric( rnd.monty.stay )
perc.monty.stay <- percent( rnd.monty.stay )
num.sims.stay = length(results.stay)
num.sims.stay <- comma( num.sims.stay )

plot.single <- barplot( monty.wins.stay,
        main = paste("Probability of Winning a New Buick if you Stay,\nout of",
                     num.sims.stay,
                     "simulations"),
        # Let's Make a Deal episodes in 1973 featured Buicks as the automobile prize
        axes = F,
        xlab = "",
        ylim = c(0,1),
        names.arg = c( "Losses", "Wins" ),
        font = 1,
        family = "serif",
        col = "seagreen"
        )
text(  x = plot.single,
       y = monty.wins.stay,
       labels = perc.monty.stay,
       pos=3,
       cex=1.2,
       family = "serif"
       )

## SWITCH STRATEGY
results.switch <- playTheGame( num.sims = 10000, stay = F)
table(results.switch)
```

```
## results.switch
##    L    W
## 3281 6719
```

```
monty.wins.switch <- prop.table( table( results.switch ) )
rnd.monty.switch<- round( monty.wins.switch, 3 )
rnd.monty.switch <- as.numeric( rnd.monty.switch )
perc.monty.switch <- percent( rnd.monty.switch )
num.sims.switch = length(results.switch)
num.sims.switch <- comma( num.sims.switch )

barplot( monty.wins.switch,
        main = paste( "Probability of Winning a New Buick if you Switch,\nout of",
                     num.sims.switch,
                     "simulations"),
        # Let's Make a Deal episodes in 1973 featured Buicks as the automobile prize
        axes = F,
        xlab = "",
        ylim = c(0,1),
        names.arg = c( "Losses", "Wins" ),
        font = 1,
```

```
        family = "serif",
        col = "darkorange2"
        )
text(   x = plot.single,
       y = monty.wins.switch,
       labels = perc.monty.switch,
       pos=3,
       cex=1.2,
       family = "serif"
       )
```