HMK 2: PART 2.R

Harshil Patel.3001

Oct. 9th, 2018

######################################  
# HMK 2: PART 2 - STAT 3201(DONGES)  
# HARSHIL PATEL, 10/05/2018   
######################################  
#set working directory  
setwd("C:/Users/offic/OneDrive - The Ohio State University/data analytics/R programming Files/hmk2")  
#Use Monte Carlo simulation to estimate the probability, expected value, total profits of #winning bets in roulette if 60000 games were played and the bet parameters were changed.  
#The bet payout will be the different for all simulations in this entire script, but the bet #amount per game(k) is a constant.

#start by setting the seed; this ensures you'll get the same random process every time you run the code  
set.seed(35)  
simulationTrials <- 60000 #simulation trials/total iterations/games played  
betAmount<-13 #k, bet amount in dollars that player buys in to bet in single game, is a constant  
  
#the possible spin values for roulette are 00, 0, 1, 2, ..., 36,   
#For the script, value the bet value "00" will be assigned to a new bet value, which is 37.  
#hence, the line below maps bet 00 to 37 and leaves all other numbers as entered.  
betOptions<-c(0:37) #builds the simulated roulette wheel.  
  
#################################  
#Bet Scenario Simulation A  
#################################  
#Bet on Black, meaning values 2, 4, 6, 8, 10, 11, 13, 15, 17, 20, 22, 24, 26, 28, 29, 31, 33, 35 in roulette wheel,   
#Betting on black has a payout of $1:$1.  
  
betWinPayout\_1<-betAmount #in dollars, if spin is won, you win payout amount,proportional to your buy in, set to Payout of $1:$1.  
betLosePayout\_1<--1\*betAmount #in dollars, if you lose spin, you lose the buy in payment(betAmount) for the spin.  
  
nWins\_1<-0 #the number of wins (this is used to count the number of wins)  
bet\_1<-c(2, 4, 6, 8, 10, 11, 13, 15, 17, 20, 22, 24,26, 28, 29, 31, 33, 35) #choosen bet, possible values of the bet:Black   
  
winProp\_1<-vector() #storage vector for the running probalbility of winning.  
winAmount\_1<-vector() #storage vector for the total of dollars won in a game.  
winAmountTracker\_1<-vector() #storage vector for the running total of dollars won.  
averageWinnings\_1<-vector() #storage vector for the running average of dollars won.  
spinValues\_1<-vector() #storage vector for the winning numbers.  
  
for (x in 1:simulationTrials) {  
   
 spinValues\_1[x] <- sample(betOptions,1) #generate a winning number for spin x by randomly choosing a number from 0,1,2, ..., ,37("00")  
   
 if(is.element(spinValues\_1[x],bet\_1)) {   
 nWins\_1<-nWins\_1+1 #compute and store number of wins, at iteration x  
 winAmount\_1[x]<-betWinPayout\_1 #compute and store win total if game is won, at iteration x   
 } else {  
 winAmount\_1[x]<-betLosePayout\_1 #compute and store win total if game is lost, at iteration x  
 }  
 winProp\_1[x]<-nWins\_1/x #compute and store est'd win prob, at iteration x  
 winAmountTracker\_1[x]<-sum(winAmount\_1) #compute and store running win total at iteration x  
 averageWinnings\_1[x]<-sum(winAmount\_1)/x #compute and store running average winnings at iteration x  
}  
  
totalWinnings\_1<-winAmountTracker\_1[simulationTrials] #total profit at the end of 60000 simulated games of roulette with choosen bet  
  
probOfWinning\_1<-winProp\_1[simulationTrials] #expected probability of winnning for one trial(game) according to simualtion loop  
#in this simulation we know the true win prob is 18/38 => we can compute the error or absolute error of the simulation results  
trueWinProb\_1 <- 18/38  
error\_1 <- probOfWinning\_1 - trueWinProb\_1  
abs.error\_1 <- abs(error\_1)  
  
expectedValue\_1<-averageWinnings\_1[simulationTrials] #expected profits for one trial(game) according to simualtion loop  
#in this simulation we know the true expected value, E(x) is betWinPayout\*(18/38) + betLosePayout\*(20/38) => we can compute the error or absolute error of the est expected value.  
trueExpValue\_1 <- (betWinPayout\_1\*(18/38)) + (betLosePayout\_1\*(20/38))  
error.EV\_1 <- expectedValue\_1 - (trueExpValue\_1)  
abs.error.EV\_1 <- abs(error.EV\_1)  
  
#a line plot of the est'd win prob by iteration with the true win prob line shown to show convergence of probability of winning(winProb\_1)  
par(ps=15)  
plot(1:simulationTrials, winProp\_1, type='l',lwd=3, col='blue',xlab='Iteration(Games of Roulette Played)', ylab='Est. Win Prob.',main='Monte Carlo Est. of Win Prob. for Betting on Black')  
abline(h=trueWinProb\_1, col='red', lwd=2)

#a line plot of the est'd avg win by iteration with the true expected win seen to show convergence of est'd exp value  
par(ps=15)  
plot(1:simulationTrials, averageWinnings\_1, type='l',lwd=3, col='blue',xlab='Iteration(Games of Roulette Played)', ylab='Est. Exp. Value($)',main='Monte Carlo Est. of Expected Value for Betting on Black')  
abline(h=trueExpValue\_1, col='red', lwd=2)

#a line plot of the total amount won by iteration with the break even point($ = 0)  
par(ps=15)  
plot(1:simulationTrials, winAmountTracker\_1, type='l',lwd=3, col='green',xlab='Iteration(Games of Roulette Played)', ylab='Total Amount Won($)',main='Monte Carlo Est. of Amount Won for Betting on Black')  
abline(h=0, col='red', lwd=2)

#################################  
  
#################################  
#Bet Scenario Simulation B  
#################################  
#Bet:First 12, meaning even values 1:12 in roulette wheel, # Betting on First 12 has a payout of $1:$2.  
  
betWinPayout\_2<-betAmount\*2 #in dollars, if spin is won, you win payout amount,proportional to your buy in, set to Payout of $1:$2.  
betLosePayout\_2<--1\*betAmount #in dollars, if you lose spin, you lose the buy in payment(betAmount) for the spin.  
  
nWins\_2<-0 #the number of wins (this is used to count the number of wins)  
bet\_2<-c(1:12) #choosen bet, possible values of the bet:First 12 which are even values from 1:12 in roulette wheel.  
  
winProp\_2<-vector() #storage vector for the running probalbility of winning.  
winAmount\_2<-vector() #storage vector for the total of dollars won in a game.  
winAmountTracker\_2<-vector() #storage vector for the running total of dollars won.  
averageWinnings\_2<-vector() #storage vector for the running average of dollars won.  
spinValues\_2<-vector() #storage vector for the winning numbers.  
  
for (y in 1:simulationTrials) {  
   
 spinValues\_2[y] <- sample(betOptions,1) #generate a winning number for spin x by randomly choosing a number from 0,1,2, ..., ,37("00")  
   
 if(is.element(spinValues\_2[y],bet\_2)) {   
 nWins\_2<-nWins\_2+1 #compute and store number of wins, at iteration y  
 winAmount\_2[y]<-betWinPayout\_2 #compute and store win total if game is won, at iteration y   
 } else {  
 winAmount\_2[y]<-betLosePayout\_2 #compute and store win total if game is lost, at iteration y  
 }  
 winProp\_2[y]<-nWins\_2/y #compute and store est'd win prob, at iteration y  
 winAmountTracker\_2[y]<-sum(winAmount\_2) #compute and store running win total at iteration y  
 averageWinnings\_2[y]<-sum(winAmount\_2)/y #compute and store running average winnings at iteration y  
}  
  
totalWinnings\_2<-winAmountTracker\_2[simulationTrials] #total profit at the end of 60000 simulated games of roulette with choosen bet  
  
probOfWinning\_2<-winProp\_2[simulationTrials] #expected probability of winnning for one trial(game) according to simualtion loop  
#in this simulation we know the true win prob is 12/38 => we can compute the error or absolute error of the simulation results  
trueWinProb\_2 <- 12/38  
error\_2 <- probOfWinning\_2 - trueWinProb\_2  
abs.error\_2 <- abs(error\_2)  
  
expectedValue\_2<-averageWinnings\_2[simulationTrials] #expected profits for one trial(game) according to simualtion loop  
#in this simulation we know the true expected value, E(x) is betWinPayout\*(12/38) + betLosePayout\*(26/38) => we can compute the error or absolute error of the est expected value.  
trueExpValue\_2 <- (betWinPayout\_2\*(12/38)) + (betLosePayout\_2\*(26/38))  
error.EV\_2 <- expectedValue\_2 - (trueExpValue\_2)  
abs.error.EV\_2 <- abs(error.EV\_2)  
  
#a line plot of the est'd win prob by iteration with the true win prob shown to show convergence of probability of winning(winProb\_2)  
par(ps=15)  
plot(1:simulationTrials, winProp\_2, type='l',lwd=3, col='blue',xlab='Iteration(Games of Roulette Played)', ylab='Est. Win Prob.',main='Monte Carlo Est. of Win Prob. for Betting on First 12')  
abline(h=trueWinProb\_2, col='red', lwd=2)

#a line plot of the est'd avg win by iteration with the true expected win seen to show convergence of est'd exp value  
par(ps=15)  
plot(1:simulationTrials, averageWinnings\_2, type='l',lwd=3, col='blue',xlab='Iteration(Games of Roulette Played)', ylab='Est. Exp. Value($)',main='Monte Carlo Est. of Expected Value for Betting on First 12')  
abline(h=trueExpValue\_2, col='red', lwd=2)

#a line plot of the total amount won by iteration with the break even point($ = 0)   
par(ps=15)  
plot(1:simulationTrials, winAmountTracker\_2, type='l',lwd=3, col='green',xlab='Iteration(Games of Roulette Played)', ylab='Total Amount Won($)',main='Monte Carlo Est. of Amount Won Betting on First 12')  
abline(h=0, col='red', lwd=2)

#################################  
  
#################################  
#Bet Scenario Simulation C  
#################################  
#Bet on Green, meaning even values between 0 and 00 in roulette wheel, but 00 maps to value 37 on our simulated roulette wheel(betOptions)  
#Bet on Green has payout of $1:$17  
  
betWinPayout\_3<-betAmount\*17 #in dollars, if spin is won, you win payout amount,proportional to your buy in, set to Payout of $1:$17  
betLosePayout\_3<--1\*betAmount #in dollars, if you lose spin, you lose the buy in payment(betAmount) for the spin  
  
  
nWins\_3<-0 #the number of wins (this is used to count the number of wins)  
bet\_3<-c(0,37) #choosen bet, possible values of the bet:Green which are values 0 and 37(00) in simulated roulette wheel.  
  
  
winProp\_3<-vector() #storage vector for the running probalbility of winning.  
winAmount\_3<-vector() #storage vector for the total of dollars won in a game.  
winAmountTracker\_3<-vector() #storage vector for the running total of dollars won.  
averageWinnings\_3<-vector() #storage vector for the running average of dollars won.  
spinValues\_3<-vector() #storage vector for the winning numbers.  
  
for (z in 1:simulationTrials) {  
   
 spinValues\_3[z] <- sample(betOptions,1) #generate a winning number for spin x by randomly choosing a number from 0,1,2, ..., ,37("00")  
   
 if(is.element(spinValues\_3[z],bet\_3)) {   
 nWins\_3<-nWins\_3+1 #compute and store number of wins, at iteration z  
 winAmount\_3[z]<-betWinPayout\_3 #compute and store win total if game is won, at iteration z   
 } else {  
 winAmount\_3[z]<-betLosePayout\_3 #compute and store win total if game is lost, at iteration z  
 }  
 winProp\_3[z]<-nWins\_3/z #compute and store est'd win prob, at iteration z  
 winAmountTracker\_3[z]<-sum(winAmount\_3) #compute and store running win total at iteration z  
 averageWinnings\_3[z]<-sum(winAmount\_3)/z #compute and store running average winnings at iteration z  
}  
  
totalWinnings\_3<-winAmountTracker\_3[simulationTrials] #total profit at the end of 60000 simulated games of roulette with choosen bet  
  
probOfWinning\_3<-winProp\_3[simulationTrials] #expected probability of winnning for one trial(game) according to simualtion loop  
#in this simulation we know the true win prob is 2/38 => we can compute the error or absolute error of the simulation results  
trueWinProb\_3 <- 2/38  
error\_3 <- probOfWinning\_3 - trueWinProb\_3  
abs.error\_3 <- abs(error\_3)  
  
expectedValue\_3<-averageWinnings\_3[simulationTrials] #expected profits for one trial(game) according to simualtion loop  
#in this simulation we know the true expected value, E(x) is betWinPayout\*(18/38) + betLosePayout\*(20/38) => we can compute the error or absolute error of the est expected value.  
trueExpValue\_3 <- (betWinPayout\_3\*(2/38)) + (betLosePayout\_3\*(36/38))  
error.EV\_3 <- expectedValue\_3 - (trueExpValue\_3)  
abs.error.EV\_3 <- abs(error.EV\_3)  
  
#a line plot of the est'd win prob by iteration with the true win prob shown to show convergence of probability of winning(winProb\_3)  
par(ps=15)  
plot(1:simulationTrials, winProp\_3, type='l',lwd=3, col='blue',xlab='Iteration(Games of Roulette Played)', ylab='Est. Win Prob.',main='Monte Carlo Est. of Win Prob. for Betting on Green')  
abline(h=trueWinProb\_3, col='red', lwd=2)

#a line plot of the est'd avg win by iteration with the true expected win seen to show convergence of est'd exp value  
par(ps=15)  
plot(1:simulationTrials, averageWinnings\_3, type='l',lwd=3, col='blue',xlab='Iteration(Games of Roulette Played)', ylab='Est. Exp. Value($)',main='Monte Carlo Est. of Expected Value for Betting on Green')  
abline(h=trueExpValue\_3, col='red', lwd=2)

#a line plot of the total amount won by iteration with the break even point($ = 0)  
par(ps=15)  
plot(1:simulationTrials, winAmountTracker\_3, type='l',lwd=3, col='green',xlab='Iteration(Games of Roulette Played)', ylab='Total Amount Won($)',main='Monte Carlo Est. of Amount Won for Betting on Green')  
abline(h=0, col='red', lwd=2)

#################################