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
```

```r
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
```

```r
#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)

#################################
```