Macquarie University

STAT 683: Introduction to Probability

Assignment 1 solution, First semester 2018

Assignment By	Sukhdeep Singh
Student ID	44442467

Game 1: Betting on red

A. Expected value: This is a Bernoulli trial therefore expected value can be given by the following expression

$$E(X) = \frac{18}{37} * (1) + \frac{19}{37} * (-1) = -\frac{1}{37}$$

Where 18/37 is the probability of winning, 19/37 is probability of losing, 1 and -1 are the amount gained in a scenario of a win/loss respectively.

Simulation results for 100000 trials: -0.02642

$$(\,percentage\ error\ = \frac{Exact\ value - Estimate\ value}{Exact\ value} * 100)$$

Percentage error=2.14%

B. Proportion of wins: Number wins for this game will be theoretically equal to

$$P(W) = \frac{18}{37} \approx 0.48648$$

Simulation results for 100000 trials: 0.48651

Percentage error = -0.0061%

C. Expected playing time = 1 bet

D. Maximum money player can lose = $1

E. Maximum money player can earn = $1

Game 2: Betting on a number

A. Expected value: This is a Bernoulli trial so the value of the winnings can be given by the following expression

$$E(X) = \frac{1}{37} * (35) + \frac{36}{37} * (-1) = -\frac{1}{37} \approx -0.0270$$

Where 1/37 is the probability of winning, $35 amount won, 36/37 is the probability of losing,-1 is the money lost.

Simulation results for 100000 trials = -0.0265

$$(\,percentage\ error\ = \frac{Exact\ value - Estimate\ value}{Exact\ value} * 100)$$

Percentage error = 1.85%

B. Proportion of wins: Theoretically the proportion of winning can be given by

$$P(W) = \frac{1}{37} \approx 0.0270$$

Simulation results for 100000 trials = 0.02584

$$(\,percentage\ error\ = \frac{Exact\ value - Estimate\ value}{Exact\ value} * 100)$$

Percentage error=4.296%

C. Expected playing time: 1 bet

D. Maximum amount player can lose= $1

E. Maximum amount player can earn= $35

Game 3: Martingale system of betting

A. Expected winnings by simulation = -1.82907
B. Proportion of wins by simulation = 0.9138
C. Expected playing time by simulation = 16
D. Maximum amount player can lose = $127

Consecutive losses since start = -1 -2 -4 -8 -16 -32 -64 = -127

E. Maximum amount player can win= $10

Game 4: Labouchere System

A. Expected winnings by simulation = -3.32318
B. Proportion of wins by simulation = 0.95636
C. Expected playing time by simulation = 4
D. Maximum amount of money player can lose = $4940
Consecutive losses since start = -5 -6 -7 -8 … -99

E. Maximum amount of money player can earn =$10

Game ID	Exp winnings (min,max)	Prop wins (min,max)	Exp play time(min,max)
Game 1	-0.02992,-0.023	0.48412,0.48795	1,1
Game 2	-0.02656,-0.012	0.02708,0.2748	1,1
Game 3	-2.08799,-1.8787	0.90861,0.9112	15,24
Game 4	-3.69058,-3.40586	0.95569,0.95768	2,20

Game ID	Winnings (mean, std dev)	Prop wins (mean, std dev)	Play time (mean, std dev)
Game 1	-0.02604, 0.9996659	0.48667, 0.499824	1, 0
Game 2	-0.019, 5.861227	0.02758, 0.164722	1, 0
Game 3	-1.98986, 38.03104	0.92482, 0.274563	17.46590,4.190035
Game 4	-3.897, 70.4128	0.95674, 0.204374	8.57188, 7.751844

Labouchere System's winning is most variable with std dev 70.41.28

Labouchere System's play time is most variable with std dev 7.751844

Rcode: Github Link

https://github.com/sedhasukhdeep/Roulette-   (Copy and paste the url in address bar of browser)

Simulation Code

```
play_game1 <- function() {
  win_amount = sample(c(1, -1),
                      1,
                      replace = TRUE,
                      prob = c(18 / 37, 19 / 37))

  #In game 1 the number of bets are constant for each game i.e 1

  number_of_bet <- 1

  return (c(win_amount, number_of_bet))


}

#Calls the game1 function and checks for proportion of wins

proportion_win_game1 <- function() {
  wins <- 0
  gamecounter <- 1

  #looping and checking the first element of the game for 1

  while (gamecounter <= 100000) {
    if (play_game1()[1] == 1) {
      wins = wins + 1
    }

    gamecounter = gamecounter + 1

  }

  return (wins / 100000)
}

#This function calculates the expected winnings in game 1
expected_game1 <- function() {
```

```
  total_win <- 0
  gamecounter <- 1

  while (gamecounter <= 100000) {
    total_win = total_win + play_game1()[1]
    gamecounter = gamecounter + 1
  }

  return (total_win / 100000)
}

#This function plays the game 2 once and returns the winnings and numbers of
bets
play_game2 <- function() {
  win_amount = sample(c(35, -1),
                      1,
                      replace = TRUE,
                      prob = c(1 / 37 , 36 / 37))

  #In game 1 the number of bets are constant for each game i.e 1

  number_of_bet <- 1

  return (c(win_amount, number_of_bet))

}

#Checking for proportion of wins for game 2
proportion_win_game2 <- function() {
  wins <- 0
  gamecounter <- 1

  #looping and checking the first element of the game for win amount == 35

  while (gamecounter <= 100000) {
    if (play_game2()[1] == 35) {
      wins = wins + 1
    }

    gamecounter = gamecounter + 1

  }

  return (wins / 100000)

}

#This function checks the expected winnings in game 2
expected_game2 <- function() {
  total_win <- 0
  gamecounter <- 1

  while (gamecounter <= 100000) {
    total_win = total_win + play_game2()[1]
    gamecounter = gamecounter + 1
  }
```

```
  return (total_win / 100000)

}

#This game plays the martindale system game once and returns the winning
amount and number of bets played
play_game3 <- function() {
  win_amount <- 0
  number_bets <- 0
  bet_amount <- 1

  while (win_amount < 10 && bet_amount < 100) {
    current_winnings = play_game1()[1]
    if (current_winnings == -1) {
      win_amount = win_amount - bet_amount
      bet_amount = bet_amount * 2
    }
    else{
      win_amount = win_amount + bet_amount
      bet_amount = 1
    }
    number_bets = number_bets + 1
  }

  return(c(win_amount, number_bets))
}

#This function returns the expected amount of winnings for game 3 when it's
played for 100000 times
expected_game3 <- function() {
  counter <- 1
  win_amount <- 0

  while (counter <= 100000) {
    win_amount = win_amount + play_game3()[1]
    counter = counter + 1
  }
  return (win_amount / 100000)
}

#This function calculates the proportion of wins for game 3
proportion_win_game3 <- function() {
  counter <- 1
  win_amount <- 0
  number_of_wins <- 0

  while (counter <= 100000) {
    win_amount = play_game3()[1]
    if (win_amount == 10) {
      number_of_wins = number_of_wins + 1
    }
    counter = counter + 1
  }

  return (number_of_wins / 100000)
}
```

```r
#This function plays the lebouchere system of game and returns the win amount
and number of bets
play_game4 <- function() {
  list_game = c(1, 2, 3, 4)
  win_amount <- 0
  bet_amount = list_game[1] + list_game[length(list_game)]
  number_bets <- 0

  while (length(list_game) >= 1 && bet_amount < 100) {
    current_winnings <-
      sample(
        c(-bet_amount, bet_amount),
        1,
        replace = T,
        prob = c(19 / 37, 18 / 37)
      )

    if (current_winnings > 0) {
      win_amount = win_amount + current_winnings
      list_game = list_game[-c(1, length(list_game))]

      if (length(list_game) == 1) {
        bet_amount <- list_game[1]

      }

      else {
        bet_amount <- list_game[1] + list_game[length(list_game)]
      }

    }

    else{
      win_amount = win_amount + current_winnings
      list_game = c(list_game, abs(current_winnings))
      bet_amount = list_game[1] + list_game[length(list_game)]
    }

    number_bets = number_bets + 1

  }

  return (c(win_amount, number_bets))

}

#This function returns the expected winnings of game 4
expected_game4 <- function() {
  counter <- 1
  win_amount <- 0

  while (counter <= 100000) {
    win_amount = win_amount + play_game4()[1]
    counter = counter + 1
  }
  return (win_amount / 100000)
```

```
}

#This function returns the proportion of wins
proportion_win_game4 <- function() {
  counter <- 1
  win_amount <- 0
  number_of_wins <- 0

  while (counter <= 100000) {
    win_amount = play_game4()[1]
    if (win_amount == 10) {
      number_of_wins = number_of_wins + 1
    }
    counter = counter + 1
  }

  return (number_of_wins / 100000)
}


#This function is calculating the required min-max values of expected
winnings, playtime and winning proportions of game 1,2,3 and 4
table_1_stuff<-function(){
game1_expected_wins = replicate(5, expected_game1()[1])


game2_expected_wins = replicate(5, expected_game2()[1])


game3_expected_wins = replicate(5, expected_game3()[1])


game4_expected_wins = replicate(5, expected_game4()[1])



game1_propwins = replicate(5, proportion_win_game1())


game2_propwins = replicate(5, proportion_win_game2())


game3_propwins = replicate(5, proportion_win_game3())


game4_propwins = replicate(5, proportion_win_game4())


game1_playtime <- 1
game2_playtime <- 1
game3_playtime = replicate(5, play_game3()[2])


game4_playtime = replicate(5, play_game4()[2])
```

```
return (list(min(game1_expected_wins),max(game1_expected_wins)
             ,min(game2_expected_wins),max(game2_expected_wins)
             ,min(game3_expected_wins)
             ,max(game3_expected_wins),min(game4_expected_wins)
             ,max(game4_expected_wins),min(game1_propwins)
             ,max(game1_propwins),min(game2_propwins)
             ,max(game2_propwins),min(game3_propwins),max(game3_propwins)
             ,min(game4_propwins)
             ,max(game4_propwins),min(game1_playtime)
             ,max(game1_playtime),min(game2_playtime)
             ,max(game2_playtime),min(game3_playtime)
             ,max(game3_playtime),min(game4_playtime)
             ,max(game4_playtime)))

}

game1_wins <- function() {
  g1 <- replicate(100000, play_game1()[1])

  return(list(mean(g1), sd(g1)))
}

game2_wins <- function() {
  g2 <- replicate(100000, play_game2()[1])

  return(list(mean(g2), sd(g2)))
}

game3_wins <- function() {
  g3 <- replicate(100000, play_game3()[1])

  return(list(mean(g3), sd(g3)))
}

game4_wins <- function() {
  g4 <- replicate(100000, play_game4()[1])

  return(list(mean(g4), sd(g4)))
}

prop1_wins <- function() {
  g1 <- replicate(100000, proportion_win_game1())

  return(list(mean(g1), sd(g1)))
}

prop2_wins <- function() {
  g2 <- replicate(100000, proportion_win_game2())

  return(list(mean(g2), sd(g2)))
}

prop3_wins <- function() {
  g3 <- replicate(100000, proportion_win_game3())

  return(list(mean(g3), sd(g3)))
}
```

```
prop4_wins <- function() {
  g4 <- replicate(100000, proportion_win_game4())

  return(list(mean(g4), sd(g4)))
}

playtime_Game3 <- function() {
  x = replicate(100000, play_game3()[2])
  return(list(mean(x), sd(x)))
}

playtime_Game4 <- function() {
  x = replicate(100000, play_game4()[2])
  return(list(mean(x), mean(y)))

}
```