Simulation of the Casino Game of Craps

Tolu Omotunde

Last compiled on 29 December, 2020

Abstract

The casino game of craps is played as follows: 1.The player rolls a pair of standard 6-sided dice and takes their sum a. If the sum is 7 or 11, then the player wins and the game is over b. If the sum is 2, 3, or 12, then the player loses (this is called 'crapping out') and the game is over c. If the sum is anything else, then we record the sum (lets call it 'X') and continue to the next step 2. The player then rerolls the dice and takes their sum a. If the sum is X, the player wins and the game is over b. If the sum is 7, the player loses and the game is over c. If the sum is anything else, repeat step 2. Now suppose that you notice something odd - one of the two dice isn't balanced that well, and always comes up in the range 2-5 (with equal probability) but never 1 or 6.

1. For each number between 2 and 12, what is the probability of rolling the dice so that they sum to that number?

```r
get_dice_probs <- function(N, outcome) {
    #' Computes probabilty for the roll sum being number 2 to 12
    #'
    #' This is a generic function which computes probabilty of 2 dices roll sum
    #'
    #' @param N - Integer, The amount of rollsums to simulate
    #'
    #' @param outcome Vector of experiment outcomes
    #'
    #' @return prob - Vector of probabilities for each sum 2:12

    ## Allocates memory space for vector of dice roll sums outcomes
    rolls <- NULL
    probs <- c()

    for (i in 1:N) {
        ## Sum of rolls on each dice
        rollSum <- sample(1:6,1) + sample(2:5,1)

        ## Append roll sums to the list of rolls
        rolls[i] <- rollSum
    }
    for (i in 2:12) {
        ## Calculate probabilty for the roll sum being number 2 to 12
        probs[i-1] <-  sum(rolls==i)/N
    }
    probs
}


## Outcome vector
outcome <- 2:12

## Number of rolls to simulate
N <- 10000

## Use 100 bootstrap experiments
no_bootstrap <- 100

## Calculate the dice roll sum probabilities using the two helper functions:
## gen_bootstrap_matrix and gen_bootstrap_summary created earlier
bootstrap_matrix <- gen_bootstrap_matrix(
    num=no_bootstrap,
    N=N,
    rename_columns=T,
    method="prob",
    outcome=outcome
)
dice_data <- gen_bootstrap_summary(
    matrix=bootstrap_matrix,
    vec_list=outcome,
    method="prob")
head(dice_data$output.df,11)
```

```
##    Sum Probability Standard_Error
## 1    2    0.000000   0.0000000000
## 2    3    0.041543   0.0002010591
## 3    4    0.083583   0.0002603729
## 4    5    0.125000   0.0003014912
## 5    6    0.167080   0.0004199880
## 6    7    0.166646   0.0003628156
## 7    8    0.166728   0.0003024549
## 8    9    0.124508   0.0003046630
## 9   10    0.083092   0.0002510329
## 10  11    0.041820   0.0002120844
## 11  12    0.000000   0.0000000000
```

2. a. What’s the probability of winning on the very first roll?

b. What’s the probability of losing (“crapping out”) on the very first roll?

```r
get_craps_probs <- function(N,outcome) {
    #' Computes probabilty for the craps game different outcomes
    #'
    #' @param N - Integer, The amount of rollsums to take
    #'
    #' @param outcome Vector of experiment outcomes
    #'
    #' @return prob - Vector of probabilities for each game outcome

    ## Allocates memory space for vector of game outcomes
    game <- NULL
    game_probability <- list()

    for (i in 1:N) {
        ## Sum of rolls on each dice
        rollSum <- sample(1:6,1) + sample(2:5,1)

        ## First condition is the player craps out on the first roll
        ## Second condition is if the player wins on the first roll
        ## Third condition is if the game continues into the point phase
        if (rollSum %in% c(2,3,12)) {
              result <- outcome[1]
        } else if (rollSum %in% c(7,11)) {
              result <- outcome[2]
        } else {
              ## Sum of rolls on each dice
              nextRollSum <- sample(1:6,1) + sample(2:5,1)

              ## Keep rolling dice till player gets a 7 or the previous point
              while (!(nextRollSum %in% c(7,rollSum))) {
                  nextRollSum <- sample(1:6,1) + sample(2:5,1)
              }
              ## If a 7 the player loses
              ## Otherwise the player wins
              if (nextRollSum == 7) {
                  result <- outcome[3]
              } else {
                  result <- outcome[4]
              }
        }
        ## Append the result of the game to outcome list
        game[i] <- result
    }
    for (i in outcome) {
        ## Calculate probabilty for the roll sum being number 1 to 12
        game_probability[i] <-  sum(game==i)/N
    }
    game_probability
}

## Outcome vector
outcome <- c(
    "Craps on First",
    "Wins on First",
    "Craps after First",
    "Wins after First"
)

## Use 20 bootstrap experiments
no_bootstrap <- 20

## Number of rolls to simulate
N <- 100000

## Calculate the game probabilities using the two helper functions:
## gen_bootstrap_matrix and gen_bootstrap_summary created earlier
bootstrap_matrix2 <- gen_bootstrap_matrix(
    num=no_bootstrap,
    N=N,
    rename_columns=F,
    method="game",
    outcome=outcome
)
game_data <- gen_bootstrap_summary(
    bootstrap_matrix2,
    vec_list=outcome,
    method="game"
)
game_data$output.df[c(1:2),]
```

```
##          Outcome Probability Standard_Error
## 1 Craps on First   0.0414025    0.000102782
## 2  Wins on First   0.2086205    0.000270984
```

3. Suppose that on the first roll, you do not win or lose, but rather, you get the sum

X, which has roll probability p. Given that you have already made it to this point, what's your chance of winning going forward?

```
game_data$output.df[4,]
```

```
##              Outcome Probability Standard_Error
## 4 Wins after First   0.3292925   0.0003516262
```

4. If you bet a dollar on a game of craps with these two dice, what is the expected return? (If you win, you will get your original dollar back plus a second dollar; if you lose, then you do not get your dollar back)

a. What is the expected return on that dollar?

```
output.prob <- unname(game_data$output.prob, force = TRUE)
prob_losing <- output.prob[1] + output.prob[3]
prob_winning <- output.prob[2] + output.prob[4]
expected_return <- -1*prob_losing + 1*prob_winning
expected_return
```

```
## [1] 0.075826
```

Thus, you expect to gain around 7.58 cents on the dollar at this game.