Assignment 6: Dice Simulation

Authors: David Chiang, Moses Jung

Due Date: 5/22/2020

Another Strategy

In this strategy, both players will only add the face value of the dice roll to their scores if they roll higher than a certain number or subtract from the opponent's score otherwise. In this particular game, we will allow player 1 to only add the face value to their score if the roll is higher than 3. If the roll is not higher than 3, then the roll will instead be subtract from the other player's score. We will allow player 2 to only add the face value to their score if the roll is higher than 5. Since the only other possible roll is 6, we will subtract from the opponent's score if they roll anything other than a 6. If at any point of the game a player's score reaches below 0, it is replaced with 0. The first player to attain a score of at least 20 wins the game.

To simulate this strategy, we will run 10,000 simulations of the game 10,000 times. We will utilize most of the same logic and structure of the code from the previous strategy, but adjusting a few lines of code to accomodate this strategy. Here is the code:

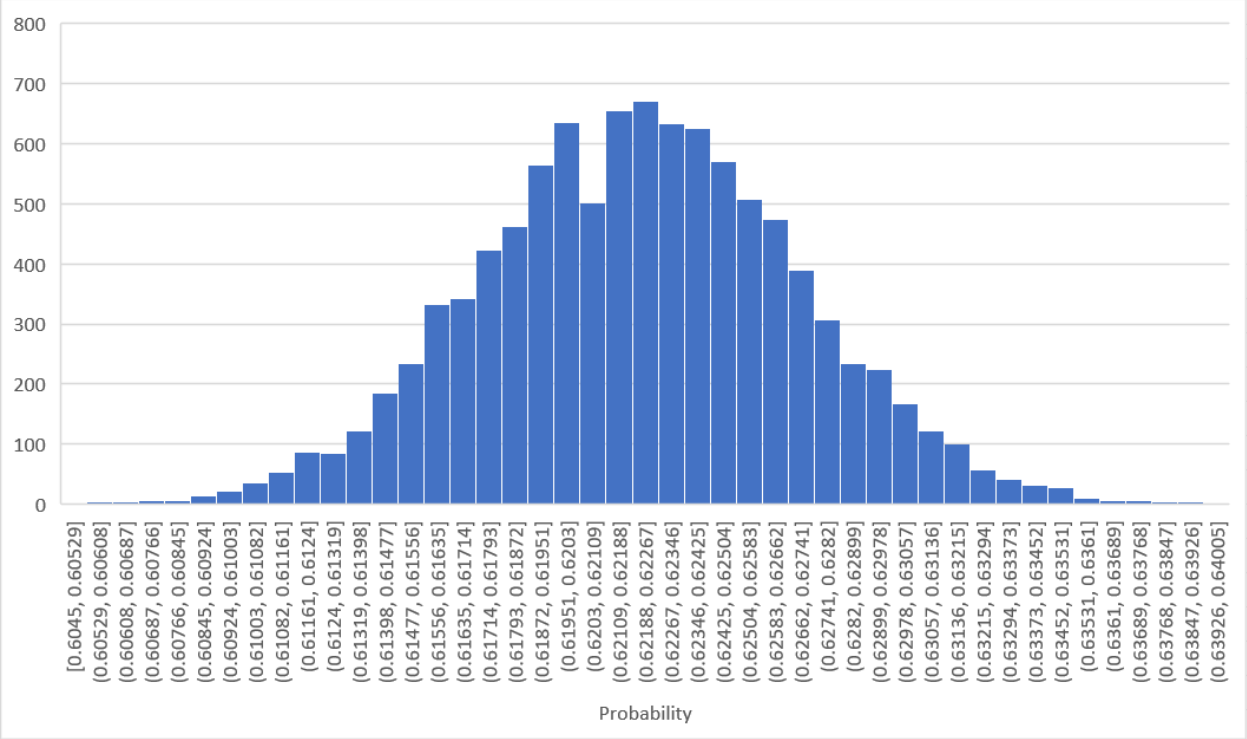
```
//repeated loops
      for (int j = 0; j < 10000; j++) {
        wins = new int[nPlayers];

        for (int i = 0; i < nGames; i++) { //simulation loop

         int pIndex = 0; //keeps track of player
         while ((pScores[0] < 20) || (pScores[1] < 20)) { //main game loop
            //System.out.println(Arrays.toString(pScores));
            Random r = new Random();
            int roll = r.nextInt(6) + 1;

            //updates player turn
            if (pIndex == 0) { //player 1 turn (if roll is bigger than 3 then
                               add to score, else sub from enemy player)
               pIndex++;
               if (roll > 3) {
                  pScores[0] += roll;
               } else {
                  if (pScores[1] - roll < 0) {
                     pScores[1] = 0;
                  } else {
                     pScores[1] -= roll;
                  }
               }
            }
            else { //player 2 turn (if roll is bigger than 5 then add to score,
                     else sub from enemy player)
```

```
            pIndex = 0;
            //pScores[1] += roll;
            if (roll > 5) {
               pScores[1] += roll;
            } else {
               if (pScores[0] - roll < 0) {
                  pScores[0] = 0;
               } else {
                  pScores[0] -= roll;
               }
            }
         }
      }
      //System.out.println(pIndex);
      //tracks wins
         if (pIndex == 1) {
            wins[1]++;
         } else {
         wins[0]++;
      }
      pScores = new int[2]; //resets scores
   }
   //prints outout to text doc
   //System.out.println(Arrays.toString(wins));
   //System.out.println(wins[0]);
   double p1 = new Double(wins[0]) / nGames;
   out.println(p1);
   System.out.println(p1);
```

After this code is executed, the results are printed and look like this:

```
(10,000 lines of output)
0.6262
0.6309
0.6115
0.6189
.
.
.
0.629
```

We will focus on the probability that player 1 wins. Here is a histogram of the probability that player 1 wins:



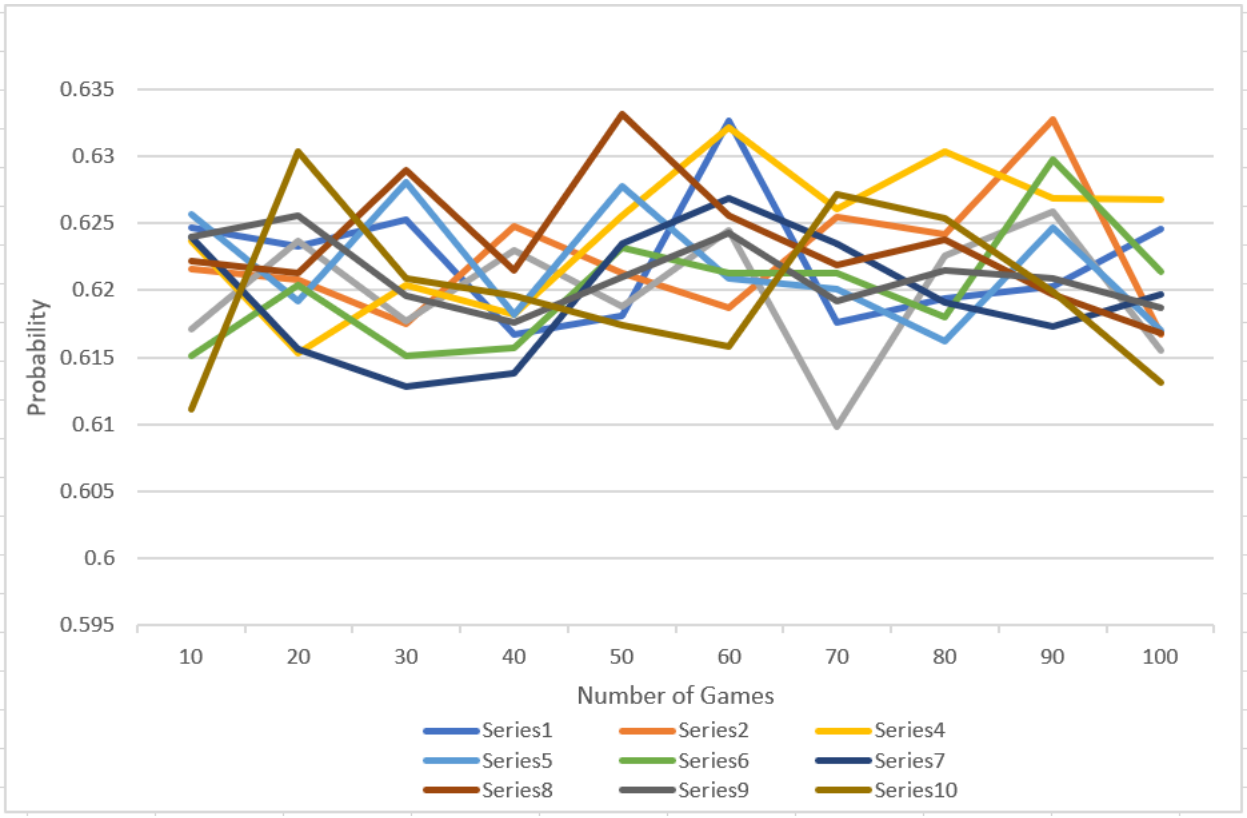
From the histogram, we see that the distribution is normal as it follows a bell shaped curve with no such skews. We see that the mean distribution lies at the center of the graph, being the interval [0.62188, 0.62267]. This means that the mean probability of player 1 winning is between 0.62188 and 0.62267. Player 1 has an advantageous strategy by only adding the face value if the roll is greater than 3 and subtract from the opponenent's score otherwise over player 2 but with a roll of 5. Player 2 should avoid choosing a higher number such as 5 as the probability of winning would be 1 - that of player 1, so 1 - [0.62188, 0.62267] would be [0.37733, 0.37812], a much lower chance.

We can utilize the *Central Limit Theorem* here to analyze the results. This theorem means that the distribution of simple random samples will be normal with a large enough sample size. That is, the larger the sample size, the better the approximation. This is why we ran 10,000 simulation of the game 10,000 times so that we are able to analyze more representative pieces of data. We can use the fact that normally distributed implies that symmetry around the mean to estimate the probability that all of our estimates are on one side of the truly probability as:

$$2/10^{10} \approx 0.001953125$$

This number is the probability that all of the estimates are greater, or all of the estimates are less than the true probability. So we can say with 99.8% confidence that the true probability lies between the confidence interval of the mean of the simulations as 0.62188 plus or minus 0.001953125, which yields [0.6199269, 0.6238331].

We can also make a convergence graph to show that our runs are long enough. We can graph the results of 100 runs of the 10,000 games each. Here is the graph that shows the convergence of the 10 runs. Once again, we will focus on player 1's probability of winning:



Looking at the graph, the 10 runs converge to a probability of around 0.62. We can utilize the *Law of Large Numbers* to illustrate this convergence. This law states that if I repeat an experiment independently a large number of times and average the result, then the result should be close to the expected value. This law is similar to the *Central Limit Theorem* in that we examine the sample size as it approaches infinity, but the *Law of Large Numbers* states that the sample mean will equal to the population mean. This converged probability value of 0.62 confirms the confidence interval that we calculated as the value lies within our confidence interval of [0.6199269, 0.6238331]. Thus, we are 99.8% confident that the true probability will lie between the confidence interval.