Die Simulation

Ryan Mouw

February 26, 2021

1 Abstract

The goal of this paper is to explore a winning strategy to a simple die game. To understand the strategy, I will simulate the outcome of many games using random die throws. Using the CLT (Central Limit Theorem) I will create a confidence interval on the true mean winning rate of the implemented strategy.

2 The Game

Let's play a simple dice game. Our game involves two players and a single die. To play;

1. Both players start with a score of zero
2. Decide who will roll first each round, and who will roll second
3. At the start of each round, the first player rolls the die
4. The first player then decides whether they want to add the value of their die roll to their own score, or subtract it from the second player's score
5. If subtracting a die value from a score would bring the score below zero, the score is zero
6. The second player then rolls the die, and decides whether to add the die value to their score, or subtract it from their opponents score (Rule 5 still applies)
7. At the end of each round, the player's scores are noted. If one of the players has a score of 50 or greater, they win. If both players have a score of 50 or greater, then the higher of the two scores wins the game. If both players have a tie score above or equal to 50, then it is a tie game

3 Strategy

The way our game is set up, if both players choose to only add the value of their die throw to their own score, the win rate will be 50/50. This is because our game only considers the player's score after both players have thrown their die and made their decisions. This condition will lead to certain consequences when we begin considering different strategies to implement.

3.1 B is for Basic

To test how effective a certain strategy is, I decided to compare more complicated strategies against the most straightforward tactic, i.e. always adding the value of the die throw to the player's own score. As we will see, the reason I used this tactic as a sort of default was that it is the most simple while still being able to result in a win. Throughout the rest of this paper, we will assign this "default" strategy to player B, and refer to the default strategy as strategy B.

Note: A different simple tactic would be to always subtract the die throw from the opponent's score. But, it is impossible for this strategy to result in a win for the player using it, and therefore a simulation will bring about trivial results.

We will refer to the player attempting more complicated strategies as player A, and we will name the strategies as we discuss them.

3.2 V is for Value

The first strategies explored only considered the value of the die that is thrown. For example, if the value of player A's die throw is even, subtract the value from player B's score, and if the value is odd, add the value to the player A's score. Another strategy was to add the value to player A's score if the die value was greater than 3, and subtract the value from player B's score otherwise. These strategies based on the value of the die alone, we will call strategy V (value). Strategy V represents a very large family of strategies, but for the most part, these strategies seems to lose more often than win against strategy B. I quickly dismissed this approach. For better results, we leave strategy V behind.

3.3 Strategy A

The second complicated approach involved both the value of player B's current score, and the die throw. At the start of the round, Player A rolls their die first. If Player A's current score plus the value of the die throw is greater than Player B's score, then Player A adds the value of the die to their score. If Player A's current score plus the value of the die throw is less than Player B's current score, then Player A subtracts the value of the die from Player B's score.

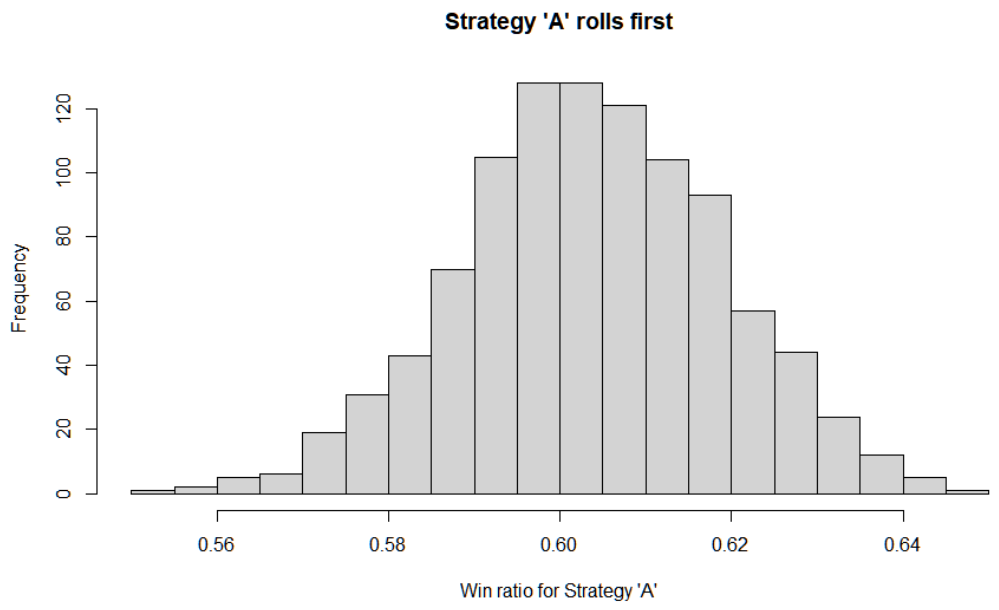
Note: In this scenario, Player A rolls first each turn, so Player A is making the decision to subtract or add without knowing what Player B has rolled this round.

We will call this Strategy A, and note that Player A is rolling first.

4 Central Limit Theorem

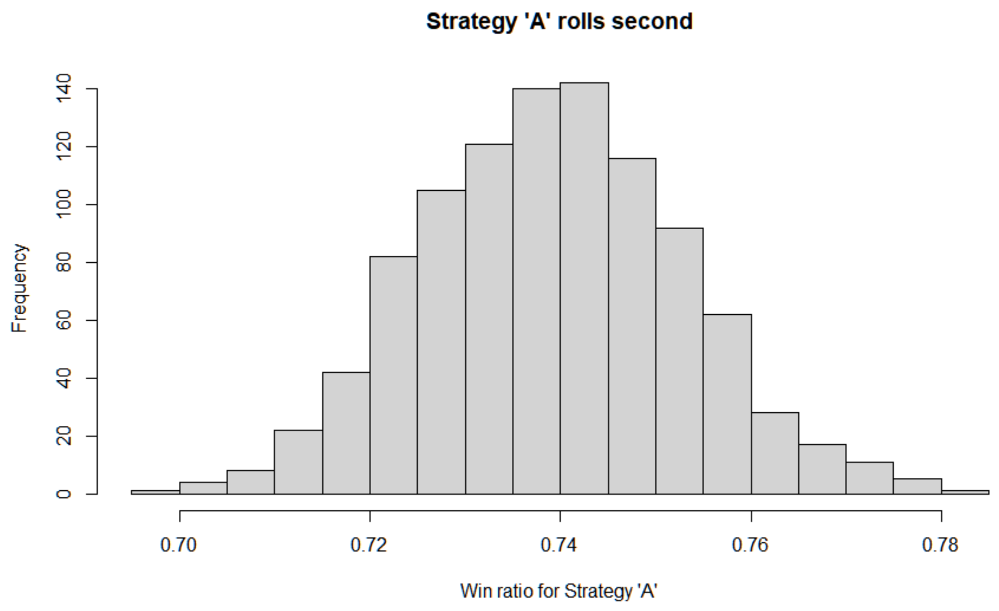
We are interested in knowing the true win rate of strategy A when played against strategy B. The "true" win rate can be thought of as the mean win rate of infinite games played using Strategy A vs Strategy B, with Player A rolling first. To approximate the true mean, we will use the CLT (Central Limit Theorem) to create a confidence interval. The CLT states that if we sample many values from an known distribution and average them, and do this multiple times, the averages will be normally distributed around the true mean of the distribution.

To convince you that the averages obtained from simulating the game many times, are normally distributed, I simulated 1000 games, 1000 times. I put the average from each 1000 simulations in the histogram below.



As we can see the histogram seems to show a normal distribution, symmetric around .6. This would suggest that the true mean of the win ratio of Strategy A vs Strategy B with A rolling first is around .6.

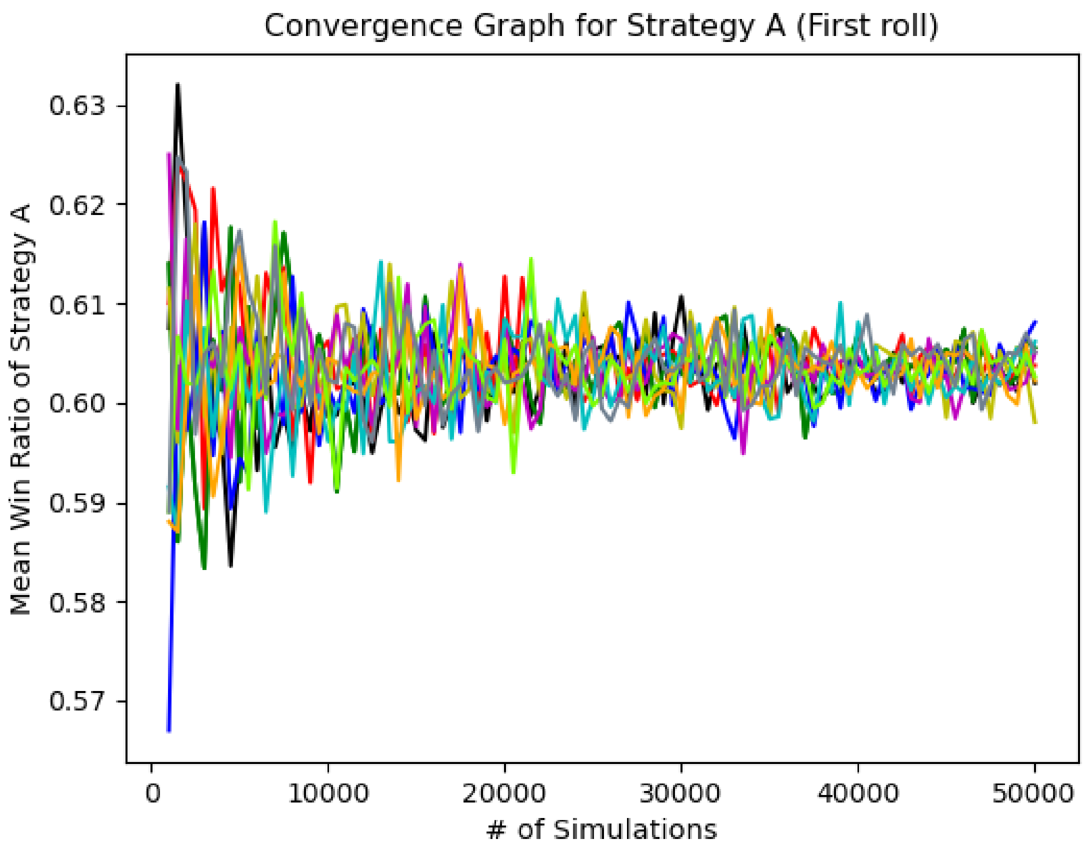
We will also explore Strategy A vs Strategy B, but with Player A rolling second, after Player B each turn. Again I simulated 1000 games, 1000 times, and took the average of each set of 1000 games. Below is the histogram.



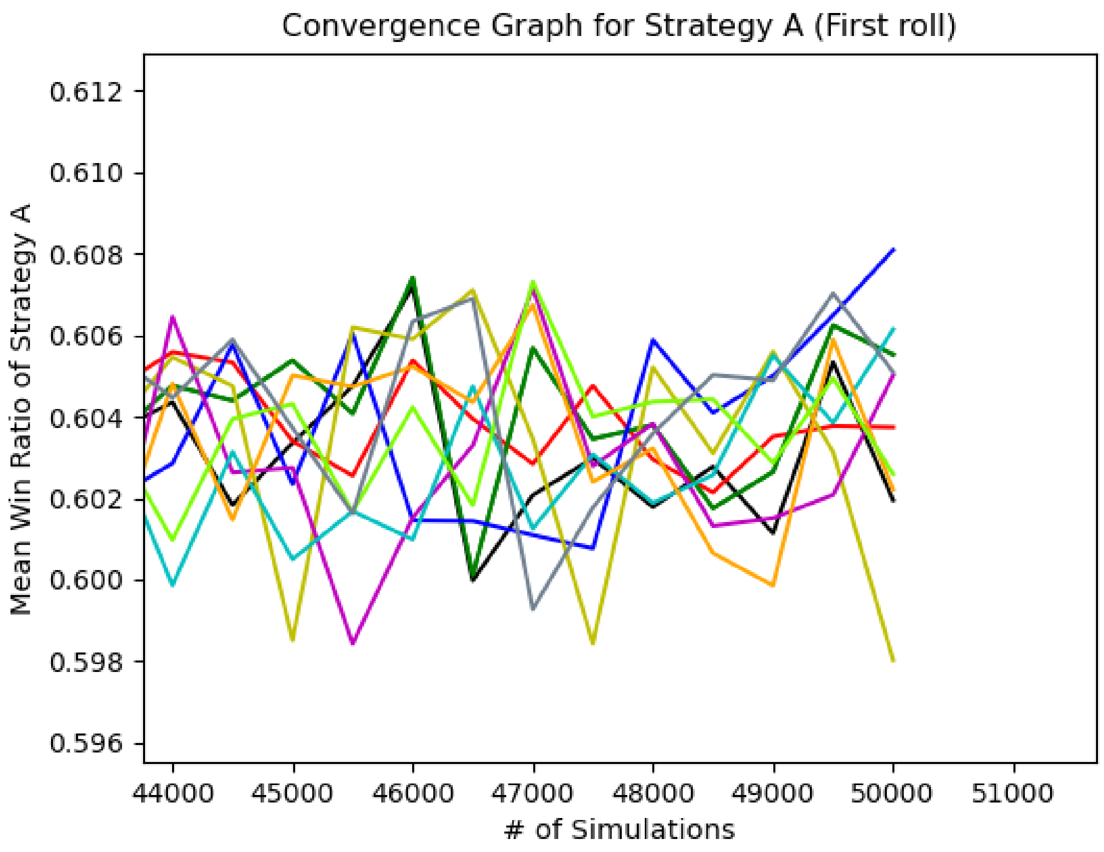
As we can see the distribution seems to be normal with a mean around .74.

5 Simulation and Convergence

Our hope is that with a higher sample size within our simulations, the mean of the simulations will converge to the true win rate of Strategy A vs Strat B. Below is a convergence graph showing ten different simulations showing Strategy A vs Strategy B, with player A rolling first.



As we can see, as the sample size increases, the mean of the win ratios is converging to some number. Below is a zoomed in version of the same graph, showing the mean win ratio of ten different simulations each involving 50,000 random games.



As we can see our mean win ratio is converging to some number. To determine this number, we will use the Confidence Interval.

6 Confidence Interval

A confidence interval is a statistical tool which will allow us to say that the true win ratio of Strategy A vs Strategy B with Player A rolling first is between two numbers with a certain degree of confidence.

What I did was to run the simulation for 10,000 different games using Strategy A vs Strategy B, with Player A rolling first. I ran this simulation 10 times. The mean of each simulation is below.

[0.5954, 0.59645, 0.5996, 0.60125, 0.6025, 0.60285, 0.6043, 0.60855, 0.6087, 0.6093]

Because these means follow the CLT, they are symmetric about the true win ratio. That implies that each mean is equally likely to be above, as it is to be below, the true win ratio. So the probability that all 10 means are on one side of the true mean is

$$\frac{2}{2^{10}} = .001953125$$

The complement of this value (.998) is the probability that the true win ratio is between the highest and lowest value value from our 10 simulations.

Putting all this together, we can say that we are 99.8% confident that the true win ratio of Strategy A vs Strategy B with player A rolling frist is between [0.5954, 0.6093].

Similarly, we can create a confidence interval for Strategy A vs Strategy B with player B rolling first. Below are the means of the 10 simulations that were run, each using 10,000 games.

[0.73425, 0.7352, 0.74215, 0.7452, 0.73775, 0.737, 0.73415, 0.73965, 0.7419, 0.74285]

This leaves us 99.8% confident that the true win ratio of Strategy A vs Strategy B with player B rolling first is between [0.73415, 0.7452].

7 Something New. C is for Constant

Now that we have determined an efficient way to beat Strategy B using Strategy A, let's change things up. One thing to notice is that Strategy A's conditional statement can be expressed as;

$if\ A\ score + die\ value - B\ score \geq 0\ then\ add\ die\ value\ to\ A\ score$

Notice that the right hand side of the inequality does not have to be zero. We can increment this number with different constants (c) and see how it affects the outcome, i.e.;

$if\ A\ score + die\ value - B\ score \geq c\ then\ add\ die\ value\ to\ A\ score$

We will continue to have Player A use Strategy A, but now we will have Player B use this new strategy, Strategy C.

Below is a table with 99.8% confidence intervals for the true mean win rate of Player A, with Player A using Strategy A and Player B using Strategy C with varying constants c. I chose $c = 1, 2, 3, 4$ because these values were enough to see the general trend, and a value of $c = 6$ gave trivial results; either a 1.0 win rate for A or an infinite loop in the code. The confidence intervals were created the same way as the previous intervals, but running 10 simulations of 10,000 games and taking the mean of each. The lower bound of each interval is the lowest value mean value, and the upper bound of each interval is the highest mean value.

Note: These values are the win rates associated with **PLayer A**. This means that a value < .5 is associated with a winning strategy for Player B.

c=	1	2	3	4
A roll first	[0.3458, 0.358]	[0.3753, 0.3922]	[0.4371, 0.4514]	[0.538, 0.5496]
B roll first	[0.5978, 0.61675]	[0.57565, 0.58635]	[0.54505, 0.56875]	[0.5713, 0.5825]

8 Results Discussion

8.1 Strategy A vs Strategy B

Any confidence interval that excludes .5 is relevant to this study. If a confidence interval includes .5, we will not be able to conclude that a strategy is better or worse than the default, nonstrategic strategy. In both cases, with player A rolling first, or second, our confidence interval has led us to believe that Strategy A does win significantly more than strategy B.

There is a notable difference between the win ratios of Strategy A vs Strategy B depending on if Player A or Player B rolls first. To explain this, it may be worthwhile to consider a possible last round of play. Imagine that Player A is rolling first and rolls a 1, and that Player A's score is 48 and Player B's score is 49. According to Strategy A, Player A would add 1 to their score, resulting in Player A having a score of 49. This leaves Player B to win the game with their roll, no matter the value of the roll, because the round will end after Player B's roll.

Now imagine that Player B rolls first and rolls a 1, with a current score of 49. After adding their die roll to their score, Player B has a score of 50. Now Player A rolls, no matter what the value of the die that Player A rolls, Player A is guaranteed to either tie, win, or continue the game for another round. This is an interesting consequence of only considering the scores after each player after a round is over, as opposed to saying that any player that reaches a score of 50 or greater at any point wins the game.

8.2 Strategy A vs Strategy C

This match up has proven less interesting in some ways, and more interesting in others. For example, it seems that Strategy A vs Strategy C favors whoever is rolling second in a round. Interestingly, this had limit for Player B. Player B began to lose with $c \geq 4$, even when rolling second in a round. There was no such limit for Player A rolling second, as each value of c resulted in a winning ration for Player A.

In conclusion, who rolls first seems to have a big hand in determining winning strategies that are more complicated than the most basic (Strategy B). If one must roll first and they are using Strategy C vs Strategy A, then they are best off using a value of $c = 3$, and hope to get lucky.

9 Code

This Code was written in Python and includes the loops used for the histograms and game simulations that are referenced in this paper.

```
import random
import numpy as np

## Loops for histogram



CLT_array=[]
for j in range(10):
    counter_array=[]
    score_1_array=[]
    score_2_array=[]
    winner_array=[]
    for i in range(10):
        counter=0
        score_1=0
        score_2=0
        win_score=50

        while score_1 < win_score and score_2 < win_score:
            counter +=1
            die_1 = random.randint(1,6)

            if score_1 + die_1 >= score_2:
                score_1 = score_1 + die_1

            else:
                if score_2-die_1 < 0:
                    score_2=0
                else:
                    score_2 = score_2 - die_1

            die_2 = random.randint(1,6)
            score_2 += die_2

        counter_array.append(counter)
        score_1_array.append(score_1)
        score_2_array.append(score_2)

        if score_1 > score_2:
            winner_array.append(1)
        elif score_1 < score_2:
            winner_array.append(0)
        elif score_1 == score_2:
            winner_array.append(.5)

    CLT_array.append(np.mean(winner_array))

print(np.mean(counter_array))
print(np.mean(score_1_array))
print(np.mean(score_2_array))
print(np.mean(CLT_array))

CLT_array_2=[]
for j in range(10):
    counter_array=[]
    score_1_array=[]
    score_2_array=[]
    winner_array=[]
    for i in range(10):
        counter=0
        score_1=0
        score_2=0
        win_score=50

        while score_1 < win_score and score_2 < win_score:
            counter +=1

            die_2 = random.randint(1,6)
            score_2 += die_2

            die_1 = random.randint(1,6)

            if score_1 + die_1 >= score_2:
                score_1 = score_1 + die_1

            else:
                if score_2-die_1<0:
                    score_2=0
                else:
                    score_2 = score_2 - die_1


        counter_array.append(counter)
        score_1_array.append(score_1)
        score_2_array.append(score_2)

        if score_1 > score_2:
            winner_array.append(1)
        elif score_1 < score_2:
            winner_array.append(0)
        elif score_1 == score_2:
            winner_array.append(.5)


    CLT_array_2.append(np.mean(winner_array))

print(np.mean(counter_array))
print(np.mean(score_1_array))
print(np.mean(score_2_array))
print(np.mean(CLT_array_2))

print(CLT_array)
print(CLT_array_2)

# np.savetxt("CLT_array.csv", CLT_array, delimiter=",")
# np.savetxt("CLT_array_2.csv", CLT_array_2, delimiter=",")

# Strategy C for Constant subtract if B score - A score > C vs Strat A

## Player A rolls FIRST

#### PLAYER B STRAT C=1

CLT_array_3=[]
for j in range(10):
    counter_array=[]
    score_1_array=[]
    score_2_array=[]
    winner_array=[]
    for i in range(10000):
        counter=0
        score_1=0
        score_2=0
        win_score=50

        while score_1 < win_score and score_2 < win_score:
            counter +=1

            die_1 = random.randint(1,6)

            if (score_1 + die_1 ) >= score_2:
                score_1 = score_1 + die_1

            else:
                if score_2-die_1<0:
                    score_2=0
                else:
                    score_2 = score_2 - die_1

            die_2 = random.randint(1,6)
            if (score_2 + die_2 - score_1) >= 1:
                score_2 = score_2 + die_2

            else:
                if score_1-die_2<0:
                    score_1=0
                else:
                    score_1 = score_1 - die_2

        counter_array.append(counter)
        score_1_array.append(score_1)
        score_2_array.append(score_2)

        if score_1 > score_2:
            winner_array.append(1)
        elif score_1 < score_2:
            winner_array.append(0)
        elif score_1 == score_2:
            winner_array.append(.5)


    CLT_array_3.append(np.mean(winner_array))

print(np.mean(counter_array))
print(CLT_array_3)
print(min(CLT_array_3))
print(max(CLT_array_3))

### PLAYER B STRAT C=2

CLT_array_3=[]
for j in range(10):
    counter_array=[]
    score_1_array=[]
    score_2_array=[]
    winner_array=[]
    for i in range(10000):
        counter=0
        score_1=0
        score_2=0
        win_score=50

        while score_1 < win_score and score_2 < win_score:
            counter +=1

            die_1 = random.randint(1,6)

            if (score_1 + die_1 ) >= score_2:
                score_1 = score_1 + die_1

            else:
                if score_2-die_1<0:
                    score_2=0
                else:
                    score_2 = score_2 - die_1

            die_2 = random.randint(1,6)
            if (score_2 + die_2 - score_1) >= 2:
                score_2 = score_2 + die_2

            else:
                if score_1-die_2<0:
                    score_1=0
                else:
                    score_1 = score_1 - die_2

        counter_array.append(counter)
        score_1_array.append(score_1)
        score_2_array.append(score_2)

        if score_1 > score_2:
            winner_array.append(1)
        elif score_1 < score_2:
            winner_array.append(0)
        elif score_1 == score_2:
            winner_array.append(.5)


    CLT_array_3.append(np.mean(winner_array))

print(np.mean(counter_array))
print(CLT_array_3)
print(min(CLT_array_3))
print(max(CLT_array_3))

### PLAYER B STRAT C=3

CLT_array_3=[]
for j in range(10):
    counter_array=[]
    score_1_array=[]
    score_2_array=[]
    winner_array=[]
    for i in range(10000):
        counter=0
        score_1=0
        score_2=0
        win_score=50

        while score_1 < win_score and score_2 < win_score:
            counter +=1

            die_1 = random.randint(1,6)

            if (score_1 + die_1 ) >= score_2:
                score_1 = score_1 + die_1

            else:
                if score_2-die_1<0:
                    score_2=0
                else:
                    score_2 = score_2 - die_1

            die_2 = random.randint(1,6)
            if (score_2 + die_2 - score_1) >= 3:
                score_2 = score_2 + die_2

            else:
                if score_1-die_2<0:
                    score_1=0
                else:
                    score_1 = score_1 - die_2

        counter_array.append(counter)
        score_1_array.append(score_1)
        score_2_array.append(score_2)

        if score_1 > score_2:
            winner_array.append(1)
        elif score_1 < score_2:
            winner_array.append(0)
        elif score_1 == score_2:
            winner_array.append(.5)


    CLT_array_3.append(np.mean(winner_array))

print(np.mean(counter_array))
print(CLT_array_3)
print(min(CLT_array_3))
print(max(CLT_array_3))

### PLAYER B STRAT C=4

CLT_array_3=[]
for j in range(10):
    counter_array=[]
    score_1_array=[]
    score_2_array=[]
    winner_array=[]
    for i in range(10000):
        counter=0
        score_1=0
        score_2=0
        win_score=50

        while score_1 < win_score and score_2 < win_score:
            counter +=1

            die_1 = random.randint(1,6)

            if (score_1 + die_1 ) >= score_2:
                score_1 = score_1 + die_1

            else:
                if score_2-die_1<0:
                    score_2=0
                else:
                    score_2 = score_2 - die_1

            die_2 = random.randint(1,6)
            if (score_2 + die_2 - score_1) >= 4:
                score_2 = score_2 + die_2

            else:
                if score_1-die_2<0:
                    score_1=0
                else:
                    score_1 = score_1 - die_2

        counter_array.append(counter)
        score_1_array.append(score_1)
        score_2_array.append(score_2)

        if score_1 > score_2:
            winner_array.append(1)
        elif score_1 < score_2:
            winner_array.append(0)
        elif score_1 == score_2:
            winner_array.append(.5)


    CLT_array_3.append(np.mean(winner_array))

print(np.mean(counter_array))
print(CLT_array_3)
print(min(CLT_array_3))
print(max(CLT_array_3))

# PLayer B rolls FIRST

## Playber B Strat C=1

CLT_array_4=[]
for j in range(10):
    counter_array=[]
    score_1_array=[]
    score_2_array=[]
    winner_array=[]
    for i in range(10000):
        counter=0
        score_1=0
        score_2=0
        win_score=50

        while score_1 < win_score and score_2 < win_score:
            counter +=1

            die_2 = random.randint(1,6)
            if (score_2 + die_2 - score_1) >= 1:
                score_2 = score_2 + die_2

            else:
                if score_1-die_2<0:
                    score_1=0
                else:
                    score_1 = score_1 - die_2

            die_1 = random.randint(1,6)

            if (score_1 + die_1) >= score_2:
                score_1 = score_1 + die_1

            else:
                if score_2-die_1<0:
                    score_2=0
                else:
                    score_2 = score_2 - die_1

        counter_array.append(counter)
        score_1_array.append(score_1)
        score_2_array.append(score_2)

        if score_1 > score_2:
            winner_array.append(1)
        elif score_1 < score_2:
            winner_array.append(0)
        elif score_1 == score_2:
            winner_array.append(.5)


    CLT_array_4.append(np.mean(winner_array))

print(np.mean(counter_array))
print(CLT_array_4)
print(min(CLT_array_4))
print(max(CLT_array_4))

## Playber B Strat C=2

CLT_array_4=[]
for j in range(10):
    counter_array=[]
    score_1_array=[]
    score_2_array=[]
    winner_array=[]
    for i in range(10000):
        counter=0
        score_1=0
        score_2=0
        win_score=50

        while score_1 < win_score and score_2 < win_score:
            counter +=1

            die_2 = random.randint(1,6)
            if (score_2 + die_2 - score_1) >= 2:
                score_2 = score_2 + die_2

            else:
                if score_1-die_2<0:
                    score_1=0
                else:
                    score_1 = score_1 - die_2

            die_1 = random.randint(1,6)

            if (score_1 + die_1) >= score_2:
                score_1 = score_1 + die_1

            else:
                if score_2-die_1<0:
                    score_2=0
                else:
                    score_2 = score_2 - die_1

        counter_array.append(counter)
        score_1_array.append(score_1)
        score_2_array.append(score_2)

        if score_1 > score_2:
            winner_array.append(1)
        elif score_1 < score_2:
            winner_array.append(0)
        elif score_1 == score_2:
            winner_array.append(.5)


    CLT_array_4.append(np.mean(winner_array))

print(np.mean(counter_array))
print(CLT_array_4)
print(min(CLT_array_4))
print(max(CLT_array_4))

## Playber B Strat C=3

CLT_array_4=[]
for j in range(10):
    counter_array=[]
    score_1_array=[]
    score_2_array=[]
    winner_array=[]
    for i in range(10000):
        counter=0
        score_1=0
        score_2=0
        win_score=50

        while score_1 < win_score and score_2 < win_score:
            counter +=1

            die_2 = random.randint(1,6)
            if (score_2 + die_2 - score_1) >= 3:
                score_2 = score_2 + die_2

            else:
                if score_1-die_2<0:
                    score_1=0
                else:
                    score_1 = score_1 - die_2

            die_1 = random.randint(1,6)

            if (score_1 + die_1) >= score_2:
                score_1 = score_1 + die_1

            else:
                if score_2-die_1<0:
                    score_2=0
                else:
                    score_2 = score_2 - die_1

        counter_array.append(counter)
        score_1_array.append(score_1)
        score_2_array.append(score_2)

        if score_1 > score_2:
            winner_array.append(1)
        elif score_1 < score_2:
            winner_array.append(0)
        elif score_1 == score_2:
            winner_array.append(.5)


    CLT_array_4.append(np.mean(winner_array))

print(np.mean(counter_array))
print(CLT_array_4)
print(min(CLT_array_4))
print(max(CLT_array_4))

## Playber B Strat C=4

CLT_array_4=[]
for j in range(10):
    counter_array=[]
    score_1_array=[]
    score_2_array=[]
    winner_array=[]
    for i in range(10000):
        counter=0
        score_1=0
        score_2=0
        win_score=50

        while score_1 < win_score and score_2 < win_score:
            counter +=1

            die_2 = random.randint(1,6)
            if (score_2 + die_2 - score_1) >= 4:
                score_2 = score_2 + die_2

            else:
                if score_1-die_2<0:
                    score_1=0
                else:
                    score_1 = score_1 - die_2

            die_1 = random.randint(1,6)

            if (score_1 + die_1) >= score_2:
                score_1 = score_1 + die_1

            else:
                if score_2-die_1<0:
                    score_2=0
                else:
                    score_2 = score_2 - die_1

        counter_array.append(counter)
        score_1_array.append(score_1)
        score_2_array.append(score_2)

        if score_1 > score_2:
            winner_array.append(1)
        elif score_1 < score_2:
            winner_array.append(0)
        elif score_1 == score_2:
            winner_array.append(.5)


    CLT_array_4.append(np.mean(winner_array))

print(np.mean(counter_array))
print(CLT_array_4)
print(min(CLT_array_4))
print(max(CLT_array_4))
```