ISYE 6644: ESTIMATING NUMBER OF CYCLES OF A COIN GAME

Group 128: Helen Cunningham

Abstract

There is great variation in the length of time, and amount of mental energy, that must be invested in a game. Some games are simple and are played merely to pass the time while engaged in lively conversation, or to provide a group with a mutual activity. Some games are more involved, with the possibility of extending hours - if not days.

Before committing to a game, players typically expect an estimate of playing time. A valuable piece of information would be the expected number of turns the game will take, which is the objective of this paper. This paper will focus on a two-player coin game. The game consists of each player rolling a die and taking an action to reduce, add, or maintain the number of coins in his/her hand. The game ends when a player has no coins remaining and rolls a die result requiring the player to give up coins. To reach the expected value of turns in the coin game, a program in R simulates 100,000 replications of the game and returns the mean of number of turns to be 17.6 (rounded up to 18 complete turns).

Background & Description of Problem

The simple coin game under investigation in this paper is a two-player game involving a 6-sided die and 10 coins. To setup the game, each player would take 4 coins into their "hand" and place the remaining 2 coins in a central "pot." One "cycle" of the game consists of Player A, and then Player B, rolling the die and taking an action that corresponds to the result of the die roll. If the player rolls a 1, they do nothing. The player gains all coins in the pot, or half the coins in the pot (rounded down) if they roll a 2 or 3 respectively. The player puts 1 coin in the pot if they roll a 4, 5, or 6. The game ends if two conditions are satisfied: a player rolls a 4, 5, or 6 AND the player has no coins remaining in his or her hand.

The objective of this paper is to determine the expected number of cycles the game will last for. First, we will outline the methodology of the R code used to calculate the expected value. Next, we will highlight the main findings of the program. Then, we will discuss the distribution of multiple replications of the game, as well as analyze potential fits of well-known probability distributions. Finally, we will discuss further applications and implications of the work done in this project.

Methodology

The first step was to create a function to return the result of a roll of an n-sided die. This function, "die_roll," will be used within the function "game." It takes one input, n, which returns a uniform distribution of n numbers. For our purposes, n will always be equal to 6 to reflect the result of a standard die roll.

Next, a function was created to represent one iteration of game play. This function, "game," takes on three inputs: A_coins, B_coins, and pot_coins. Based on the identified rules for the game, these values will always be 4, 4, and 2 for our purposes, reflecting the beginning balance of coins in each of Player A's hand, Player B's hand, and the pot, respectively.

The "game" function first defines a variable "nturns," which begins at zero and counts up by one each time a full cycle of the game is played. The vector "end_roll" is defined to store values of die_roll (4, 5, and 6) that represent the game ending condition when combined with 0 coins in the active player's hand.

A while loop contains a die roll by Player A, followed by interpretation of the result of the roll. If the result of the roll is in the vector "end_roll" and Player A has no coins, the loop is broken and "game" will return the value of nturns at game end. If the game ending condition is not satisfied, a series of if statements evaluates potential die rolls and adjusts the balances of coins in Player A's hand and the pot as needed. The roll of 4, 5, or 6 will result in Player A's hand increasing by 1 coin, and the pot decreasing by 1 coin. The roll of 2 will result in Player A's hand increasing by the amount of coins in the pot, and the pot decreasing to 0 coins. Finally, the roll of 3 will result in Player A's hand increasing by the floor function of half of the pot's coins, and the pot decreasing by the same amount. The roll of 1 is not reflected in an if statement as it results in no action being taken.

Following the result of Player A's die roll, and associated impact to Player A's hand and the pot, Player B's die roll is recorded, and the same game ending condition is tested to determine if the while loop will be broken as a result of the roll. Then, the same series of if statements applied to Player A's roll are applied to Player B's roll to determine the impact to Player B's hand and the pot. Finally, the variable nturns is increased by 1 before the while loop re-cycles.

Once the game ending condition occurs, the while loop is broken. The value of nturns is increased by 1 to reflect this roll as the final cycle of the game, and the value of nturns is returned.

The next step is to run the simulation of this game many times in order to estimate the expected value of cycles the game will run for. A loop runs through the 100,000 replications of the game and stores each result in a list. We can unlist the resulting vector to then plot a histogram of the results and return summary statistics. An easy check that the results look as expected is verifying that the minimum number of cycles of the game is equal to 5. Because neither player's turn impacts the number of coins in the other player's hand, the only way to achieve the game ending condition is for a player to lose all 4 coins in their hand over the course of 4 cycles of game play. Then, the player must roll a 4, 5, or 6 on their fifth turn, thus ending the game.

Main Findings

The expected value of cycles of the game is 17.6, meaning an average of 17.6 cycles of the coin game are played before the game ends. Because the number of cycles must be an integer (i.e., a cycle consisting only of Player A's turn still counts as 1 full cycle), we report the expected value as 18. The list of nturns values can be plotted to approximate the distribution of number of cycles of the game.

To determine the confidence interval for the mean number of cycles, first we calculated sample mean, sample variance, and a t-value at the .05 and .01 significance levels. At the .05 significance level, the upper bound is calculated to be 17.66 and the lower bound is calculated to be 17.53. This means we are 95% confident that the true mean number of turns lies between 17.53 and 17.66. At the .01 significance level, the upper bound is 17.69 and the lower bound is 17.50. We are 99% confident that the true mean number of turns lies between 17.5 and 17.69. Because the number of turns must be an integer, the results of our confidence interval calculations leave us very confident that a game will last for an average of 18 turns.

Analysis of the Distribution of Replications

Visual analysis of the histogram of number of turns for each of the 100,000 replications shows that the distribution appears to be Poisson, with values clustered to the left of the graph, and a long right tail of higher values, as seen in *Figure 1*. This makes sense, as the games that last far longer than the mean number of turns are likely to be outliers, intuitively.

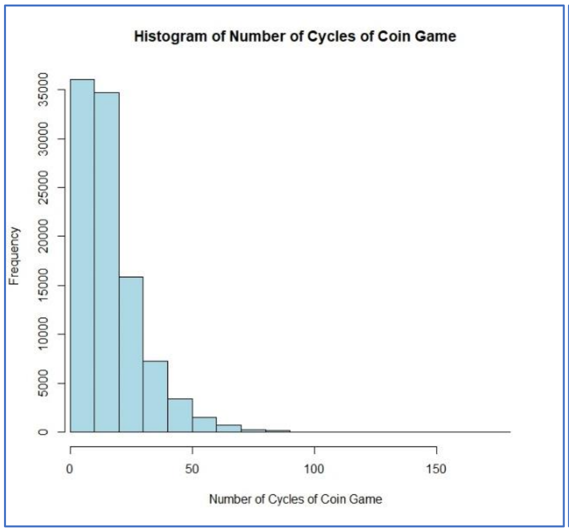

Figure 1: Histogram showing frequency of number of cycles of the coin game over 100,000 replications

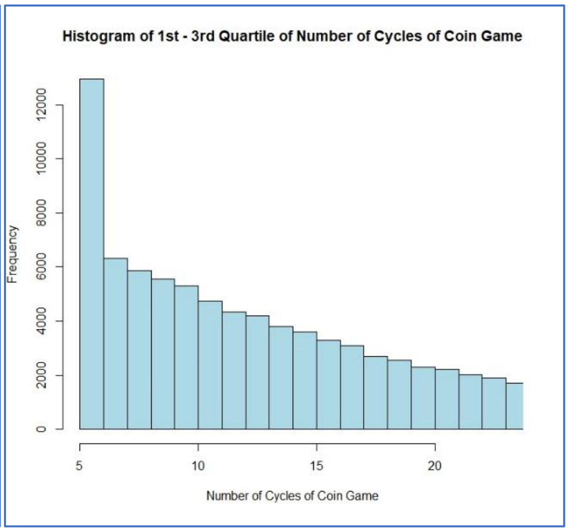

Figure 2: Histogram showing frequency of number of cycles of the coin game in the first, second, and third quartiles of the 100,000 replications' results

It is unsurprising then that, when we calculate the quantiles and median of the distribution of nturns, we see that values are clustered close together to the left of the mean and spread out to the right of the mean. There are as many values of nturns between 5 (the minimum) and 17.6 as there are between 17.6 and 180 (the maximum). The median value of nturns is 14. The mean is likely higher than the median value because of how spread out the values are toward the fourth quartile of data. When we view just the first, second, and third quartile of the replications' results, as seen in *Figure 2*, it is clear that the mode of the distribution is near the minimum of the distribution. A calculation in R shows that the mode is 6 turns. The likelihood of each turn being the last turn decreases with each successive turn.

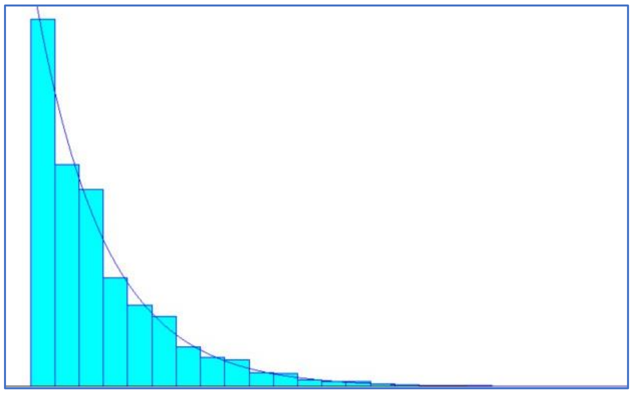

Figure 3: Result of Arena Input Analyzer fit to histogram of number of cycles of coin game

When the vector of mean number of turns over 100,000 replications is fed into Arena's input analyzer, as seen in *Figure 3,* Arena identifies the Exponential and Erlang distributions as the best fit out of its pre-programmed probability distributions and superimposes the Exponential function over the histogram of replication data. Visual analysis of the plot in *Figure 3* shows that the fit of the Exponential distribution to the data is poor, however. The histogram bars exceed the line denoting the Exponential distribution at multiple points.

Function	**Sq Error**
Erlang	0.00125
Exponential	0.00125
Beta	0.00133
Gamma	0.00572
Weibull	0.00605
Lognormal	0.0377
Normal	0.0551
Triangular	0.122
Uniform	0.15

Figure 4: Results of Arena Input Analyzer's attempt to fit all distributions to the data

A summary of the distributions tested, sorted by lowest to highest squared error value, can be seen in *Figure 4.* The worst fits to the data are the Triangular and Uniform distributions. While the exponential function 5 + Exp(12.6) is the best fit of the distributions stored by Arena, as seen in *Figure* 5, this expression yields a poor p-value < .005.

Distribution Summary	
Distribution:	Exponential
Expression:	5 + EXPO(12.6)

```
Square Error:  0.001254

Chi Square Test
Number of intervals        = 25
Degrees of freedom         = 23
Test Statistic             = 1.05e+03
Corresponding p-value   < 0.005
```

Figure 5: Result of Arena Input Analyzer fit to Exponential distribution

Because the data under observation is discrete, as all possible nturns values must be integers, it would make more sense that the distribution would be Poisson rather than Exponential or Erlang. It is also possible that different distributions, or probability mass functions, could fit the data better for defined intervals. Next steps could be to treat each quartile individually to test the fit of various distributions.

Conclusion

While the objective of this project was simple, solely to estimate the approximate number of cycles of game play for a coin game, the code generated can be used for other similar objectives. The R program is written in a general way so that it would be easy to tweak the values of n and/or initial conditions of the game. The code could then be used to approximate the number of turns that the coin game would last if the die were not 6-sided, but instead had more or fewer sides. The number of coins that Player A, Player B, and the pot begin with could be changed by altering the inputs into the game() function.

Other projects that could build off of the code created for this paper include incorporating game rule changes, increasing the number of players, and estimating the differences between these changes and the original simulation. The rules could be changed to allow for Player A and Player B to take coins from each other, or to have to split coins to the pot and to the other player. It would be interesting to see the effect of this change on the number of turns of the game. It could increase the number of turns, as players would have more opportunities to gain coins (both from the pot and from their opponent). It could also decrease the number of turns in the case that players have more opportunities to lose coins (both to the pot and to their opponent). To determine whether the new version of the game resulted in more or fewer turns, a confidence interval for the difference in two means could be calculated. If zero were *not* included in the confidence interval, a determination of which version of the game was likely to be shorter could be determined.

Works Cited

Zach. “How to Calculate the Mode in R (with Examples).” *Statology*, 9 June 2022, www.statology.org/mode-in-r/.