A Simple Coin Game: First Step Analysis and Simulation

Ross Roberts

December 2021

Abstract

This work analyzes a simple coin game involving two players, a pot, several coins, and a die. Each player starts the game with some number of coins. The remaining coins are left in the pot. The players take turns rolling the die and redistributing the coins according to a set of rules. The game ends when one of the players cannot perform a required action (i.e., putting a coin in the pot when they have none). First Step Analysis is used to determine the expected length of the game in cycles where a cycle is defined as both players completing a turn. A brief overview of First Step Analysis is provided. Finally, the distribution of cycle counts is estimated using parametric probability density estimation on the output of a game simulated with Python. With an initialization of each player having 4 coins and 2 in the pot (10 total), the expected number of cycles is around 17.54. The distribution of cycle counts is shown to approximately follow an exponential distribution with estimated parameters $\hat{\mu} = 5$ and $\hat{\beta} = 12.54$.

Background

Rules

The rules of the game will be defined as follows: if a player rolls a 1, nothing happens, if a player rolls a 2, the player takes all the coins in the pot, if a player rolls a 3, the player takes half of the coins in the pot (rounded down), and if a player rolls a 4, 5, or 6, the player puts 1 coin in the pot. If the pot is empty and a player rolls a 2 or 3, nothing happens (the game continues). If a player rolls a 4, 5, or 6 and the player doesn't have any coins, the game ends. A cycle is defined as a set of two turns, one by player A, then one by player B. If the game ends on player A's turn, that turn is counted as a full cycle. For the remainder of this analysis, assume that each game is played with a total of 10 coins.

Notation

The game can be viewed as a sequence of states that transition from one to the next. A state is a set of variables that contain enough information to describe the system (game) at any point in time. In this case, each state can be characterized by the number of coins in each player's possession at the beginning of a cycle. From this, the number of coins in the pot can be deduced. Consider a game with 10 total coins: knowing that player A has 3 coins and player B has 2 coins at the beginning of a cycle is enough information to deduce that the pot has 5 coins. This state will be notated as **A03B02**. Now consider, starting with state **A03B02**, that player A rolls a 3. Since the die is equally probable to land on any of its six faces, $\mathrm{P}(A \text{ rolls } 3) = 1/6$. After this turn, player A will take half the coins in the pot ($\lfloor 5/2 \rfloor = 2$), which will bring player A's total to 5 coins and the pot's total to 3 coins. Now assume player B rolls a 4, 5, or 6. Then $P(B \text{ rolls } 4,5,6) = 1/6 + 1/6 + 1/6 = 1/2$. After this turn, player B puts one coin in the pot, which will bring player B's total to 1 coin and the pot's total to 4 coins. Since die rolls are independent, $P(A \text{ rolls } 3 \cap B \text{ rolls } 4,5,6) = P(A \text{ rolls } 3) * P(B \text{ rolls } 4,5,6) = 1/6 * 1/2 =$

$1/12$. Therefore, this specific combination of rolls leads state **A03B02** to transition to state **A05B01** with probability 1/12 after one cycle.

Note that it is possible for several permutations of die rolls to lead to the same state. For example, if player A had rolled a 2, then player A would have taken all the coins in the pot, leaving none for player B to take upon a roll of a 2 or 3. Therefore, the end state for $(A \text{ rolls } 2 \,\cap\, B \text{ rolls } 1{,}2{,}3)$ is **A08B02** with probability $3 \,*\, 1/36 \,=\, 1/12$. The transition probabilities and resulting states for each permutation of die rolls are calculated below in **Tables 1.1** and **1.2**.

Table 1.1: Example Calculation of State Transition Probabilities

Cycle starting at State A03B02											
Player A's Turn					Player B's Turn					End of Cycle	
A's Roll	P(A)	A	B	Pot	B's Roll	P(B)	A	B	Pot	End State	P(A, B)
A rolls a 1	1/6	3	2	5	B rolls a 1	1/6	3	2	5	A03B02	1/36
					B rolls a 2	1/6	3	7	0	A03B07	1/36
					B rolls a 3	1/6	3	4	3	A03B04	1/36
					B rolls a 4,5,6	1/2	3	1	6	A03B01	1/12
A rolls a 2	1/6	8	2	0	B rolls a 1	1/6	8	2	0	A08B02	1/36
					B rolls a 2	1/6	8	2	0	A08B02	1/36
					B rolls a 3	1/6	8	2	0	A08B02	1/36
					B rolls a 4,5,6	1/2	8	1	1	A08B01	1/12
A rolls a 3	1/6	5	2	3	B rolls a 1	1/6	5	2	3	A05B02	1/36
					B rolls a 2	1/6	5	5	0	A05B05	1/36
					B rolls a 3	1/6	5	3	2	A05B03	1/36
					B rolls a 4,5,6	1/2	5	1	4	A05B01	1/12
A rolls a 4,5,6	1/2	2	2	6	B rolls a 1	1/6	2	2	6	A02B02	1/12
					B rolls a 2	1/6	2	8	0	A02B08	1/12
					B rolls a 3	1/6	2	5	3	A02B05	1/12
					B rolls a 4,5,6	1/2	2	1	7	A02B01	1/4

Table 1.2

End State	Transition Probability
A03B02	1/36
A03B07	1/36
A03B04	1/36
A03B01	1/12
A08B02	1/12
A08B01	1/12
A05B02	1/36
A05B05	1/36
A05B03	1/36
A05B01	1/12
A02B02	1/12
A02B08	1/12
A02B05	1/12
A02B01	1/4

Now consider state **A05B00**. Since player B does not have any coins, a roll of a 4, 5, or 6 would end the game. In that case, state **A05B00** would transition to state **END**. The transition probabilities are calculated in the same way as above. $P(A \text{ rolls } 1,\ldots,6 \,\cap\, B \text{ rolls } 4{,}5{,}6) = P(B \text{ rolls } 4{,}5{,}6) \,=\, 1/2$. Game play ceases once **END** is reached; therefore, **END** does not transition to any of the other states. To that end, we say **END** transitions to itself with probability 1.

Outline

Using this framework of the game as a system of states with transition probabilities, First Step Analysis will be used to analytically derive the expected value of the number of cycles played before the game ends given a specified starting state. An overview of First Step Analysis will be provided before describing the methodology used to calculate transition probabilities for each potential state in the system. This work will conclude with a simulation of the game using Python along with a variety of output analysis methods used to approximate the distribution of cycle counts.

First Step Analysis

Overview

First Step Analysis refers to a broad set of techniques developed to analytically study Markov processes, or systems with states that transition to and from one another where a transition at any specific time step depends only on the system state at that point in time [1]. A Markov process with n states can be captured by the transition probability matrix

$$\boldsymbol{P} = \begin{pmatrix} p_{1,1} & \cdots & p_{1,n} \\ \vdots & \ddots & \vdots \\ p_{n,1} & \cdots & p_{n,n} \end{pmatrix}$$

where $p_{i,j}$ is the probability that state i transitions to state j in one time step [1]. Note that $\sum_{j=1}^{n} p_{i,j} = 1 \text{ for } i = 1,2,\ldots,n$. In other words, entries in each row add to 1. Row i represents an 'absorbing' state if $p_{i,j} = 1 \text{ when } i = j \text{ and } p_{i,j} = 0 \text{ when } i \neq j$, i.e., absorbing states can only transition to themselves [2]. A stochastic matrix $\boldsymbol{P}$ represents an absorbing Markov process if (1) at least one of the rows represents an absorbing state, and (2) each of the other states can transition to at least one of the absorbing states in a finite number of time steps [2]. These non-absorbing states are called 'transient' states. Note that these conditions are met by the coin game: (1) there is one absorbing state, the **END** state, and (2) it is possible for each of the other states to reach the **END** state by one player repeatedly rolling a 4, 5, or 6, thereby losing one coin each cycle. Though this scenario is not exactly *probable*, it illustrates the possibility. Therefore, the coin game is an absorbing Markov process. Now consider a stochastic matrix $\boldsymbol{P}$ with r absorbing states and t transient states. Then $\boldsymbol{P}$ can be partitioned such that

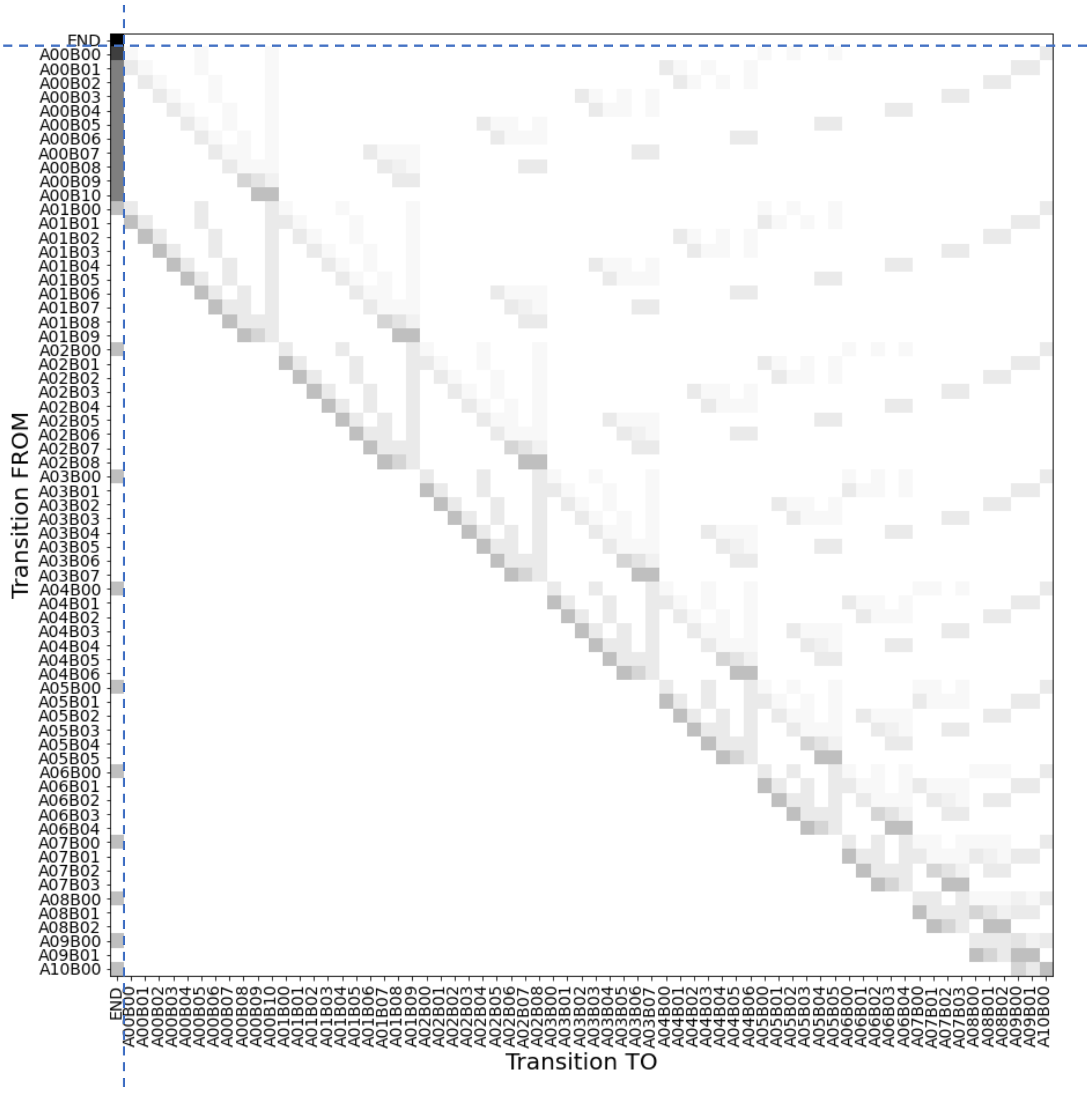
$$\boldsymbol{P} = \begin{pmatrix} \boldsymbol{I}_r & \boldsymbol{0} \\ \boldsymbol{R} & \boldsymbol{Q} \end{pmatrix}$$

where $\boldsymbol{I}_r$ is an (r x r) identity matrix, $\boldsymbol{0}$ is an (r x t) matrix of 0's, $\boldsymbol{R}$ is a (t x r) matrix representing the probabilities that a transient state transitions to an absorbing state, and $\boldsymbol{Q}$ is a (t x t) matrix representing the probabilities that a transient state transitions to another transient state [2]. It can be shown that such a process will always end in an absorbing state in a finite number of steps [2]. It can also be shown that $\boldsymbol{Q}_{i,j}^{k}$ represents the probability of transitioning from state i to state j in exactly k time steps [2]. By summing over all possible values of k (from 1 to ∞), we arrive at the fundamental matrix $\boldsymbol{N} = \sum_{k=0}^{\infty} \boldsymbol{Q}^k = (\boldsymbol{I}_t - \boldsymbol{Q})^{-1}$ where $\boldsymbol{I}_t$ is a (t x t) identity matrix. $\boldsymbol{N}_{i,j}$ represents the expected number of times the system will transition through state j before terminating in an absorbing state (starting from state i) [2]. Finally, the expected number of steps the system will take before landing in an absorbing state (starting from state i) is given by $\boldsymbol{t}_i$ where $\boldsymbol{t} = \boldsymbol{N}\boldsymbol{1}$ with $\boldsymbol{1}$ being a (t x 1) column vector of 1's [2].

Methodology

First, a list of all possible states was generated using Python. These range from **A00B00** to **A10B00** and **A00B10** under the constraint that the sum of the two numbers be less than or equal to 10. Transition probabilities were calculated by iterating over all the possible permutations of die rolls and aggregating the possible outcome states along with their total transition probabilities for each possible initial state. This information was stored in a 2-dimensional NumPy array. The state transition matrix is represented as an image below in **Figure 1** where the magnitude of each probability is indicated by cell brightness (a probability of 0 is displayed white and a probability of 1 is displayed black). Dotted lines are overlayed to represent the partitioning of $\boldsymbol{P}$ into sub-matrices $\boldsymbol{I}_r$, $\boldsymbol{0}$, $\boldsymbol{R}$, and $\boldsymbol{Q}$.

Figure 1: Partitioned State Transition Matrix $\boldsymbol{P}$



Note that since there is only one absorbing state in the system, $\boldsymbol{I}_r$ is the scalar value of 1 in the top left partition (the black cell), $\boldsymbol{0}$ is the white row vector along the top, $\boldsymbol{R}$ is the column vector along the left with non-zero (grey) probabilities corresponding to states with at least one player having 0 coins, and $\boldsymbol{Q}$ is the square matrix filling out the majority of the image.

The fundamental matrix $\boldsymbol{N}$ was calculated by the transformation $(\boldsymbol{I}_t - \boldsymbol{Q})^{-1}$. Similarly, this matrix is represented as an image below in **Figure 2**. The vector $\boldsymbol{t}$ was calculated by multiplying $\boldsymbol{N}$ with a column vector of 1's. This vector represents the expected number of cycles a game will last indexed by the starting state. The 10 states with longest expected cycle counts are listed in descending order in **Table 2**.

Figure 2: Fundamental Matrix $\boldsymbol{N}$

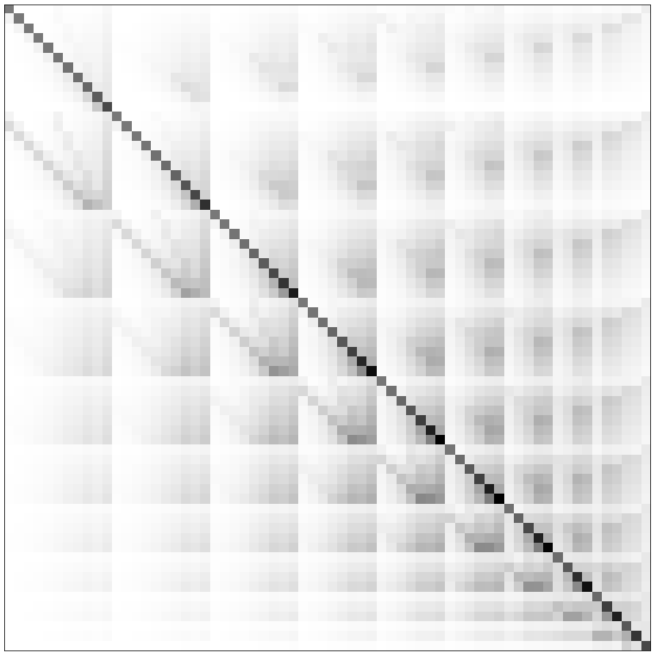
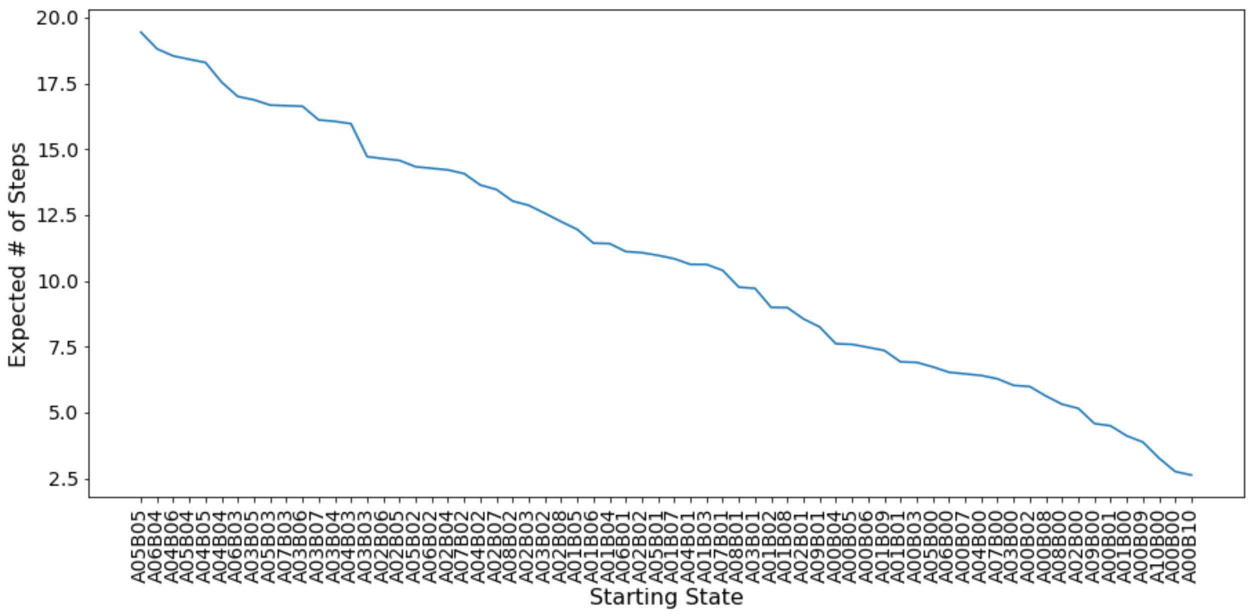


Table 2: Top 10 Starting States

Starting State	Expected # of Steps
A05B05	19.440182
A06B04	18.807237
A04B06	18.537172
A05B04	18.409492
A04B05	18.286613
A04B04	17.540922
A06B03	16.997768
A03B05	16.869764
A05B03	16.670889
A07B03	16.647590

From **Table 2**, it seems fair to conclude that, on average, the game will last the longest when both players start with an even split of all the coins, leaving none to the pot. In order to maximize the length of the game, there appear to be two factors at play regarding starting states: (1) the number of coins left in the pot and (2) the difference in coin counts between the two players. Minimizing both factors seem to increase the expected length of the game. For example, the three starting states with the longest expected cycle counts all leave 0 coins to the pot, but the difference in coin counts between the two players is 0 for first starting state and 1 for the following two. Also, even though starting state **A04B04** leaves 2 coins to the pot, it ranks higher than starting state **A06B03** (which leaves 1 coin to the pot) since the coin counts between players are balanced (a difference of 0 compared to 2). At first glance, it almost appears that minimizing the sum of the difference in coin counts between the two players and the coins left to the pot maximizes the expected number of cycles the game will last, though this pattern and the mechanics of its underlying optimization problem would need to be verified at a later time. This pattern is displayed further in **Figure 3**, a graph of the expected cycle counts (descending) as a function of the starting state.

Figure 3: Expected Number of Cycles vs. Starting State



Simulation

A simple function was defined in Python to simulate a single game. The function accepts both players' initial coin counts, as well as the total number of coins in play as inputs. Die rolls are simulated as discrete uniform random variables using Python's *random.randint* function. Each player's coin counts are tracked throughout gameplay, and the function outputs the total number of cycles until an end state is reached. A sample of the logic used for a single player's turn is shown below in **Figure 4**.

Figure 4: Simulation Logic

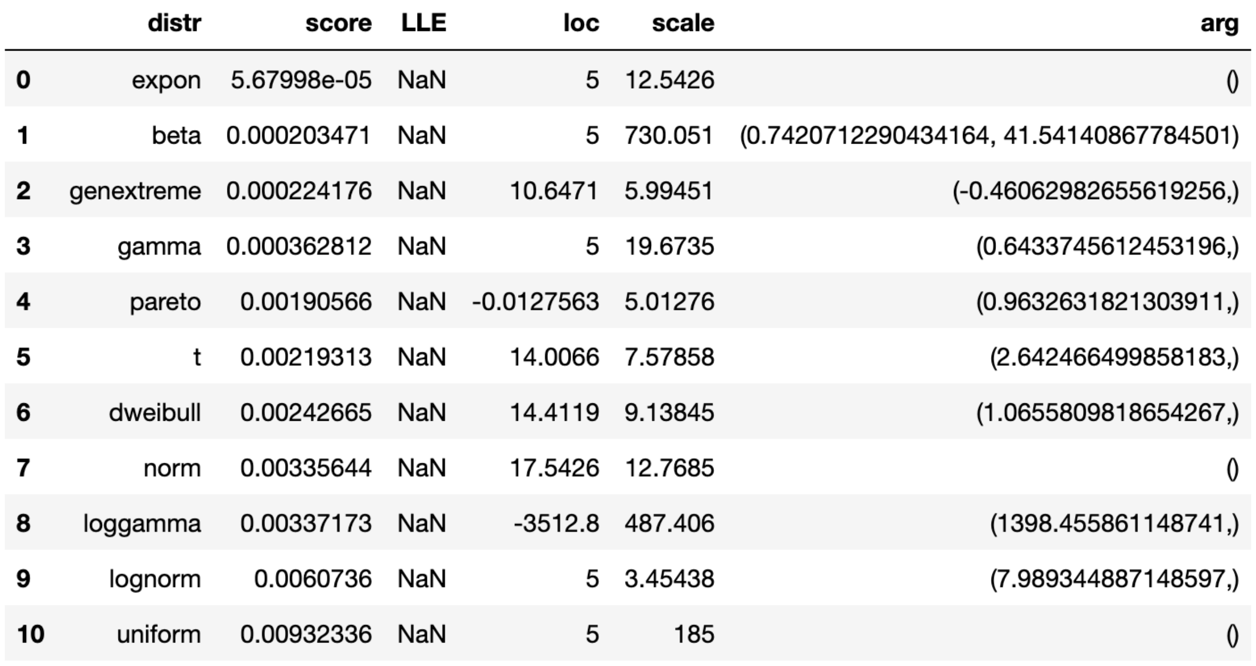
```
# A's turn

a_roll = random.randint(1,6)  # Simulate player A's die roll

if a_roll == 1: pass          # If player A rolls a 1, do nothing
elif a_roll == 2:             # If player A rolls a 2,
    a += pot                      # player A gets all the coins in the pot,
    pot = 0                       # and the pot goes to 0
elif a_roll == 3:             # If player A rolls a 3,
    a += pot // 2                 # player A gets half the coins in the pot,
    pot -= pot // 2               # and the pot loses half its coins
else:                         # If player A rolls a 4, 5, or 6
    if a == 0:                    # and player A doesn't have any coins,
        return cycles                 # end the game
    else:                         # If player A has at least 1 coin,
        a -= 1                        # player A gives one coin to the pot,
        pot += 1                      # and the pot gains 1 coin
```

To estimate the distribution of cycle counts, one million games were simulated with starting state **A04B04**. Each game's cycle count was recorded in a list. The sample mean was found to be 17.542582. This matches closely with the expected number of cycles for initial state **A04B04** found from the First Step Analysis (17.540922). The Python library *distfit* was then used to perform parametric probability density estimation on the samples. The function *distfit* estimates parameters for multiple probability density functions and returns the pdf that best fits the data [3]. To estimate parameters for each tested pdf, *distfit* finds the parameter values that minimize the residual sum of squares (RSS) between the empirical distribution and the estimated pdf: $RSS_\theta = \sum_{i=1}^{n}(y_i - f_\theta(x_i))^2$ where θ represents the set of parameters for pdf $f(x)$ [3]. The sum is indexed over n bins from the empirical histogram. For example, by setting *distfit's bin* parameter to 50, a histogram with 50 bins will be created from the empirical data. The counts from each bin, the $y_i's$, will be compared to the predicted values of corresponding bins from the theoretical pdf with parameters θ, the $f_\theta(x_i)'s$. Once parameters have been estimated for each tested pdf, the best fitting distribution is selected by the smallest RSS [3]. The simulated game data was tested against 11 distributions: Normal, Exponential, Pareto, Weibull, Student's t, Generalized Extreme Value, Gamma, Log Normal, Beta, Uniform, and Log Gamma. The summary of the fitting is shown below in **Table 3**.

Table 3: Distribution Fit Summary



	distr	score	LLE	loc	scale	arg
0	expon	5.67998e-05	NaN	5	12.5426	()
1	beta	0.000203471	NaN	5	730.051	(0.7420712290434164, 41.54140867784501)
2	genextreme	0.000224176	NaN	10.6471	5.99451	(-0.46062982655619256,)
3	gamma	0.000362812	NaN	5	19.6735	(0.6433745612453196,)
4	pareto	0.00190566	NaN	-0.0127563	5.01276	(0.9632631821303911,)
5	t	0.00219313	NaN	14.0066	7.57858	(2.642466499858183,)
6	dweibull	0.00242665	NaN	14.4119	9.13845	(1.0655809818654267,)
7	norm	0.00335644	NaN	17.5426	12.7685	()
8	loggamma	0.00337173	NaN	-3512.8	487.406	(1398.455861148741,)
9	lognorm	0.0060736	NaN	5	3.45438	(7.989344887148597,)
10	uniform	0.00932336	NaN	5	185	()

The distributions are listed in ascending order by RSS (score). Thus, for this data (starting state **A04B04)**, the distribution of cycle counts is best fit by an Exponential distribution with estimated parameters $\hat{\mu} = 5$ (loc) and $\hat{\beta} = 12.54$ (scale). The RSS is around 5.68e-05. Note that the two-parameter Exponential distribution differs slightly from the usual singe-parameter distribution. The two-parameter pdf is given by: $f(x; \mu, \beta) = (1/\beta)e^{-(x-\mu)/\beta}$ for $x \geq \mu, \beta > 0$ [4]. Here, the inverse of the scale parameter β is equivalent to the usual rate λ, and the two-parameter pdf is shifted to the right of the single-parameter pdf by μ units [4]. Thus, the expected value of a two-parameter Exponential distribution can be expressed as $(1/\hat{\lambda}) + \hat{\mu} = \hat{\beta} + \hat{\mu} = 5 + 12.54 = 17.54$ which matches the sample mean of the cycle counts, as expected. The shift in location is explained by the fact that for starting state **A04B04**, a game cannot end in under 5 cycles since a player can only lose one coin per cycle, whereas the standard exponential distribution starts at 0. Graphs and histograms displaying the empirical distribution compared against the estimated Exponential distribution are shown below in **Figures 5**, **6.1**, and **6.2**.

Figure 5: Empirical vs. Exponential Distribution

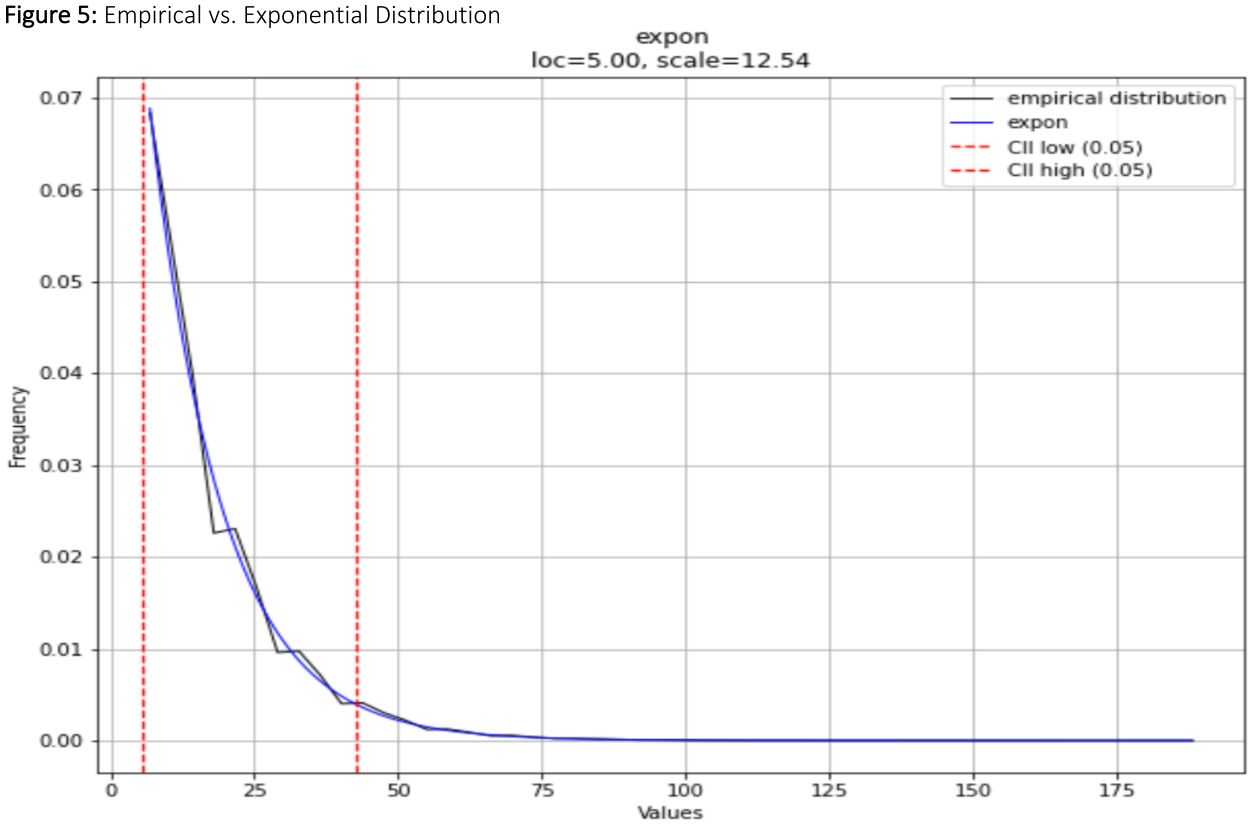


Figure 6.1: Empirical Histogram

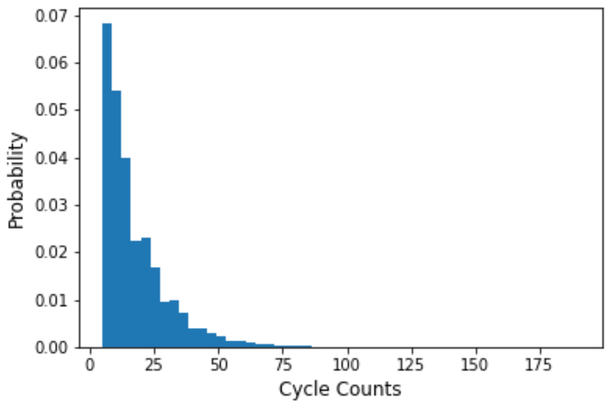
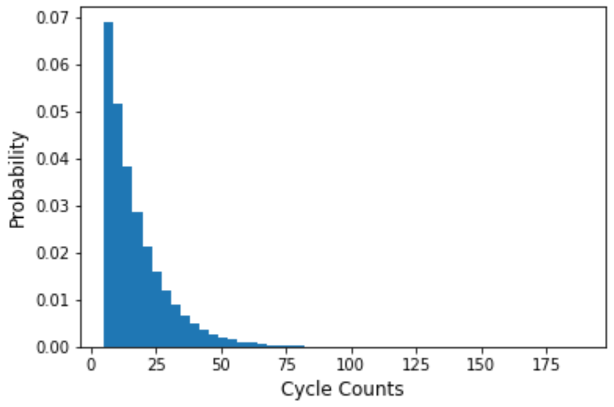


Figure 6.2: Exponential (5, 12.54) Histogram



Conclusion

Summary

A simple game of chance was framed as an absorbing Markov process made up of states with transition probabilities. Analytical derivations of the processes' properties followed once the states were clearly defined, and the probabilities calculated. Namely, given the game setup and rules previously outlined, the expected length of the game in cycles was calculated for each potential initial state. A game starting with both players having four coins each with two coins in the pot will last approximately 18 cycles, on average. It was also found that in order to maximize the length of the game, the players should split the total number of coins evenly, leaving none in the pot. Next, the game was repeatedly simulated to create an empirical distribution of cycle counts. A parametric probability density estimation method using residual sum of squared error as a goodness-of-fit metric found the distribution to be approximately Exponential with mean 5 and scale 12.54.

Future Work

The following questions are left open for exploration: Is there a reliable heuristic to determine which initial states have higher (or lower) expected cycle counts before ending? For example, given states **A03B06** and **A04B07**, can the state with the higher expected length be determined without using First Step Analysis? The sum of the difference in coin counts between the two players and the coins left to the pot for each of the two states is $3 + 1 = 4$ and $4 + 1 = 4$, respectively. The previously mentioned heuristic of selecting the state that minimizes this value fails here. If such a heuristic is found, can it be formalized into an optimization problem?

The expected cycle length was found for each initial starting state, but the estimated distribution of cycle lengths was only found for one. What do the distributions look like for the other initial states? Do they only differ by location (found to be the shortest possible game given an initial state, i.e., the smallest number of coins in a player's initial hand plus 1), or do they also differ by scale? What if the game was not limited to 10 coins? What if the rules were changed to make it more likely to gain coins than to lose coins?

Finally, the distribution of cycle counts is discrete, yet it was estimated with continuous parametric distributions. What are the underlying processes that enable a discrete system to closely follow a continuous distribution? Is there a way to derive the exact discrete distribution through the discretization of a continuous distribution? Or more generally, is there a way to analytically derive the discrete distribution with similar methods used in First Step Analysis?

Works Cited

[1] N. Privault, Understanding Markov chains: Examples and applications, Singapore, Heidelberg, New York, Dordrecht, London: Springer, 2013.

[2] J. G. Kemeny and J. L. Snell, Finite Markov chains, New York, Berlin, Heidelberg, Tokyo: Springer-Verlag, 1976.

[3] E. Taskesen, "distfit," 2019. [Online]. Available: https://github.com/erdogant/distfit.

[4] NIST/SEMATECH, "e-Handbook of Statistical Methods," 1 June 2003. [Online]. Available: https://itl.nist.gov/div898/handbook/eda/section3/eda3667.htm.