Mining for Gold Coins

B. "I put the Tran in FORTRAN" Parazin
Lord Blunderbuss' School of the Arcane
New Valentis
Occupied Dawnlands, H3A 0G4

***Abstract*—Table-top role-playing games are ubiquitous in culture today, but player outcomes in these games are still largely governed by luck and random chance. To address this shortcoming of the medium, I develop a novel piece of code which allows one to, for a short period of time, control the outcomes they will receive when digitally rolling dice, provided they are using NumPy's random library. Furthermore, through an interdisciplinary process of performing interviews with self-professed "Dungeon Masters" and statistical methods, I am able to reframe this abstract benefit of foresight as a concrete amount of in-game currency, gold. I find that players would gain, on average, 182,930,844,029 gold pieces by utilizing my code. Finally, by considering that value and the computational cost needed to find it, I find conversions between gold pieces and other currencies.**

1. Introduction

As society crumbles around us, table-top roleplaying games have become a popular form of escapism, allowing people to pretend they live in a world where the impossible is real. In this fantasy wonderland, people regularly make a decent living freelancing, get 8 hours of unbroken sleep a night, and, through the power of teamwork and friendship, take down the evil robber barons that run their local kingdom. The ability to live out these dreams, however, is wholly dependent on the whims of the "dice gods." In the course of gaming, players have to make many skill checks, attack rolls and saving throws, which use a die or series of dice modified by some static value(s) compared to a threshold to determine if a player succeeds in attempting uncertain actions.

In recent years, however, many gamers find digital dice, which use pseudo-random methods to generate a random number in a given range, to be favourable. Their reasons for this preference vary: some don't want to worry about losing dice, others have limited bag space better spent on painted miniatures, and still others have given into the siren call of their polyhedra and, in the spirit of Chronos, they consumed their dice whole.

The growth of these digital dice rollers also opens up new opportunities for players interested in tipping the scales of luck in their favor. Previously players turned towards methods of witchcraft or superstition to control their luck and stave off "Dice Curses." Common methods included talking to dice, blowing on dice, placing misbehaving dice in dice jail, or sacrificing a goat or other mammal to appease and soothe any angry dice.

Fifty percent of the time, these methods work every time [1], but with the power of modern science and mathematics, there is an opportunity to improve upon previous methods of controlling one's fate. By utilizing the pseudo-random nature of digital dice rollers, players can now set a seed that will determine the series of results they get when rolling dice. With an adequate choice of seed, a player can therefore be confident that luck will finally be on their side, the only question is, what seed to choose?

In this paper, I describe the process of choosing the optimal seed for a common pseudo-random number generator, and some of the consequences of this choice. Section 2 introduces this seed, describes the methods used to determine this seed, and highlights some other pseudo-random seeds of interest to fans of d20 tabletop games. Section 3 then details my research to quantify the significance of this seed, showcasing some potential results of using it in game. Section 4 expands on the gold amount found in section 3 by exploring alternative uses of the computational power used in this assignment, ultimately finding two further foreign exchange conversion factors for d20 fantasy gold. Section 5 discusses future research paths.

2. Pseudo-random seed exploration

In a d20 fantasy game, the best thing one can roll on a dice-based check is a 20 on the dice. This is a so-called "Natural 20," and is always a cause for celebration around the table. It therefore follows that the optimal pseudo-random seed for a dice roller is one that generates the most natural 20s in a row from the start. To find this, there are two methods one can use. For an open-source pseudo-random number generator like the one used in this study, *numpy.random.Generator.integers*, one could take a look under the hood and strive to understand the numerical methods used in generating these numbers. This intrepid researcher could then take that understanding and produce an inverse model, one that takes in a string of integers and

1. Except for the goat sacrifice, which has 36% effectiveness

returns the corresponding seeds that could produce that, in a stunning synthesis of mathematics and computer science.

Alternatively, one can simply guess-and-check seeds, recording those that produce the longest streaks of the same number, in an equally stunning synthesis of stubbornness and laziness. In this paper I undertake the second method.

2.1. Guess-and-Check code

The heart of the code which performs this guess-and-check method is reproduced in part below.

```
import numpy as np
...
rng = np.random.default_rng(seed)
streak_number = rng.integers(low=1,
    high=21)
current_roll = streak_number
streak_length = 0
while current_roll == streak_number:
    streak_length+=1
    current_roll = rng.integers(low=1,
        high=21)
...
```

For a given seed, this algorithm sees how many times in a row the first random number that seeds rolls is generated, saving the resulting seed and streak length to a file. This code was loaded onto a Raspberry Pi 4, and allowed to run for 111 days straight, checking random seeds 0 through 85,500,000,000. The calls to rng.integers are from low=1 to high=21 since the function produces random intergers in the range [low, high).

2.2. Streak length results

The results of the first 85.5 billion seeds checked are presented in table 1 as Roll, Streak Length and seed, meaning, for example, the NumPy random.default_rng seed 216240055 produces 9 12s as the first 9 integers generated when generating in the range [1, 20]. Most germane to this line of research is the seed 75364316682, which produces 8 natural 20 rolls in a row. Finding a streak this long is pushing the upper limits of what I can find with this guess-and-check method, as I will need to, on average, run this code for another 666 days or to find a streak of 9 natural 20s in a row, and another 36.46817 years to find a streak of 10 natural 20s.

3. Dice Rolls to Gold Piece conversion

There is obviously a strong benefit to being able to know what the results of your next eight rolls in a d20 fantasy game, doubly so if one knows that they will be the best possible outcomes, but this benefit is abstract and unquantified. To remedy this shortcoming of my research, I undertook interview with 21 self-proclaimed "Dungeon Masters," where I outlined the scenario and asked for their expert opinion on how many gold pieces one could gain.

Roll	Streak Length	NumPy seed
1	8	68840775906
2	8	80686509533
3	9	74955903520
4	9	83606999174
5	8	77136117593
6	8	44827294402
7	8	52740367896
8	8	50104951153
9	8	78235588018
10	8	24243953919
11	8	80127612848
12	9	216240055
13	8	1382968756
14	8	43181918036
15	8	73192838540
16	8	81406914160
17	8	84529065864
18	8	48692406026
19	8	67772198982
20	8	75364316682

TABLE 1. NUMPY.RANDOM.DEFAULT_RNG SEEDS THAT PRODUCE THE LONGEST STREAKS WHEN USED TO SIMULATE ROLLING A 20-SIDED DICE

Player number	Gold pieces aquired	Notes
1	1,000	
2	1,000	Achieved through high-stakes gambling
3	6400	
4	8,000	
5	10,000	Achieved by robbing a vault full of treasure
6	20,000	
7	40,000	
8	100,000	
9	100,000	
10	500,000	
11	1,000,000	Achieved by buying Onion Futures
12	30,000,000	Achieved through high-stakes gambling
13	1,000,000,000	
14	2,560,000,000,000	Achieved through high-stakes gambling

TABLE 2. POTENTIAL PROFITS, AND METHODS OF ACHIEVING THEM (IF SHARED) BY INTERVIEWEES

Of the respondents, 6 were excluded for giving non-numeric answers, saying that they could get "infinity gold pieces [2]." A 7th leather-clad interviewee was excluded when it became clear they were a different type of dungeon master.

Of the remaining 14 respondents, there was considerable variation in responses. The potential gold pieces varied from as low as 1,000 to as high as 2.56×10^{12} coins, with a statistical mean of 1.97×10^{11}, and a median of 70,000 gold pieces. For future analysis I will use the mean value, as it is funnier. Players also had the option to describe the plan they would undertake to get that much money, with most respondents turning to some form of gambling to find their profits. Table 2 illustrates all 14 valid responses.

2. It will be a cold day in hell before I recognize IEEE 754

4. Further Currency Conversions

4.1. Canadian Toonies

By considering the opportunity cost of performing this analysis, it also becomes possible to find direct currency conversions between d20 fantasy gold pieces and other forms of money. The easiest comparison is found when considering the energy cost of performing this analysis, and allows for a conversion to the coinage of the boreal kingdom of Canada, the toonie. The computer performing this analysis was running for 111 days, and used up 9.425 kilowatt-hours to do that. Thanks to Hydro Quebec's generous rate of 0.0325 toonies per kilowatt-hour, I spent 0.305 toonies to perform the analysis and generated 1.97 $\times 10^{11}$ gold pieces. This leads to a conversion factor of 149,300,829,941.41 toonies per gold coin. Furthermore, at the gold spot price of 47.86 toonies per gram, it follows that each gold coin must weigh 3,119,424.70 kilograms and have a volume of 161.77071 cubic meters [3].

4.2. Bitcoin

Instead of considering the opportunity cost of performing the analysis, I can also consider the opportunity cost of not using the computational resources of my Raspberry Pi on other things, such as mining Bitcoin. Fully clocked, a raspberry pi can achieve approximately 100 Hashes per second. This can be compared to the total number of hashes per second spent on mining Bitcoin to estimate the average amount of hashes before I would expect to mine a block. At time of writing, there are approximately 6.15 $\times 10^{8}$ hashes per second of computational power spent on guessing-and-checking algorithms to mine Bitcoin. This means I would expect to have a 1.62698 $\times 10^{-7}$ chance of mining any given Bitcoin block. At an average rate of 144 blocks per day for 111 days, I would have expected to have gained 0.002577143 Bitcoin if I dedicated my computational power to that instead. This produces the exchange rate of 1 d20 gold piece being equal to 7.0982E+13 Bitcoins [4]. Finally, by comparing the results from these two sections, one easily see that 1 Bitcoin is worth approximately 0.00210 toonies, indicating that the present value of 47489.075 toonies per Bitcoin is slightly inflated [5].

5. Conclusion

This work has powerful implications for world economies everywhere. The powerful exchange rate of d20 gold coins to real-world money could prove a boon to all d20 fantasy tabletop role-playing game fans if any stock exchanges offer trading services. This work is not without shortcomings, however. An important point of consideration is that this purely economic calculation missed out on some of the non-tangible benefits of playing d20 fantasy games. Unquantified are the human connections forged over rolling dice, doing silly voices and telling meaningful stories, and raises future research questions such as "What if the real fortune is the friends we made along the way?"

3. For reference, an Olympic-sized swimming pool has a volume of 2500 cubic meters

4. For reference, there will only ever exist 2.1 $\times 10^{7}$ Bitcoins

5. This is not financial advice

Acknowledgments

I would like to thank Tai Withers for their peer-review comments on this manuscript, as well as all of the people who've played TTRPGS with me over the years. I would also like to thank Pine trees and blackbirds for all of the joy they've brought into my life. Finally I would like to thank fungi, carrion beetles and other decomposers for the critical role they play in ecosystems everywhere.