Gambler's Ruin with Two People at a Time

Diego Ortega Hernandez

Fall 2018

Background

When my friends and I from high school turned 21, one of us wanted to go to a casino. I have no business gambling and didn't want to waste any money up to chance. So I proposed if we want to go to a casino, the best has to represent us. Idea: have a small tournament with chump change (20 nickels), and we trust the winner to go to the casino on our behalf. The problem is that took forever (at least 3 hours). But why?

Let there be p players, each starting with d dollars and the same probability of winning a round. Assuming they can bet any (integer) amount they have with a uniform distribution, what's the expected number of rounds we have to play before someone ends up with all $p \cdot d$ dollars?

1 Solution with two players

Solution. We'll start off with two players and induct from there. We'll use the following random walk algorithm to describe the situtation:

Assuming all players start with d dollars.
Assume each player is equally likely to win.
Fix $k \in [1, d]$.

1. Pick two players P1 and P2 at uniform randomness.
2. Player P1 bets.
   (a) If player P1 bets k and wins, then they get k money from player P2.
   (b) If player P1 bets k and loses, then they give P2 k money.
3. Update the amounts P1, P2 have at their disposal to bet.
4. Repeat steps 1,2,3
   (a) Stop condition: when one player has $p \cdot d$ dollars.

Let X be the random variable that measures the number of rounds the players play. Observe $X \geq 1$. Each round of play follows this format

$$\begin{tikzcd}(\text{Stage 1}) \arrow[r, bend left, "\text{gain money with } p=\frac{1}{2}"] & (\text{Stage 2}) \arrow[l, bend left, "\text{lose money with } 1-p=\frac{1}{2}"]\end{tikzcd}$$

If P1 or P2 has j money, then the number of rounds they will play is

$$X_j = 1 + A_j$$

where A_j is the number of rounds to be played after the first round (aka how many times we need to repeat steps 1 and 2). Since expectation is linear, the expectation of both sides yields

$$E\left[X_j\right] = E\left[1 + A_j\right] = E\left[1\right] + E\left[A_j\right] = 1 + E\left[A_j\right]$$

Now, we need to calculate the conditional expectation of A_j, which depends on if P1 wins or if P2 wins.

$$\begin{aligned} E\left[X_j\right] &= 1 + E\left[A_j\right] \\ &= 1 + E\left[A_j \mid P1 \text{ wins}\right] P\left(P1 \text{ wins}\right) + E\left[A_j \mid P2 \text{ wins}\right] P\left(P2 \text{ wins}\right) \\ &= 1 + E\left[A_j \mid P1 \text{ wins}\right]\left(\frac{1}{2}\right) + E\left[A_j \mid P2 \text{ wins}\right]\left(1 - P\left(P1 \text{ wins}\right)\right) \\ E\left[X_j\right] &= 1 + E\left[A_j \mid P1 \text{ wins}\right]\left(\frac{1}{2}\right) + E\left[A_j \mid P2 \text{ wins}\right]\left(\frac{1}{2}\right) \end{aligned}$$

What does this mean? Remember, the players are just looping around and winning/losing k dollars each round, which means starting off with $j+k$ and $2d-(j+k)$ is the same as both players having j and $2d-j$ respectively. Thus we have

$$E\left[X_j\right] = 1 + E\left[X_{j+k}\right]\left(\frac{1}{2}\right) + E\left[X_{j-k}\right]\left(\frac{1}{2}\right)$$

and a non-homogeneous linear recurrence

$$a_j = 1 + a_{j+k}\left(\frac{1}{2}\right) + a_{j-k}\left(\frac{1}{2}\right)$$

where $j \in k, ..., 2dk-k$ and $k \in [1, d]$. With boundary value conditions $a_0 = 0$ and $a_{2d} = 0$

If we play around with this and plug in values of $j = qk$ for some $q \in \mathbb{Z}$ using the boundary conditions $a_0 = 0$ and $a_{n-k} = 0$:

$$a_{j+k} = 2a_j - a_{j-k} - 2$$

j	$a_{j+k} = 2a_j - a_{j-k} - 2$	Simplify
k	$a_{2k} = 2a_k - a_0 - 2$	$a_{2k} = 2\left(a_k - 1\right)$
2k	$a_{3k} = 2a_{2k} - a_k - 2$	$a_{3k} = 3\left(a_k - 2\right)$
3k	$a_{4k} = 2a_{3k} - a_{2k} - 2$	$a_{4k} = 4\left(a_k - 3\right)$
...		
(i-1)k	$a_{ik} = 2a_{(i-1)k} - a_{(i-2)k} - 2$	$a_{ik} = i\left(a_k - (i-1)\right)$

So we propose

$$a_{ik} = i\left(a_k - i + 1\right)$$

for $j \in \{k, ..., 2dk-k\} = k\{1, ..., 2d-1\}$

Notice that the right hand side is only dependent on the value of a_k. Since we're shifting by k amount for each round, this is the same as shifting a multiple $1k$, which reduces the problem to looking at the recurrence for $k = 1$. Then using the inductive hypothesis that

$$a_j = j\left(a_1 - j + 1\right)$$

$j \in \{i, ..., 2d\}$, and doing some algebra

$$\begin{aligned} a_{j+1} &= 2a_j - a_{j-1} - 2 \\ &= 2\left(j\left(a_1 - j + 1\right)\right) - \left(j - 1\right)\left(a_1 - \left(j-1\right) + 1\right) - 2 \\ &= 2j\left(a_1 - j + 1\right) - j\left(a_1 - j + 2\right) + \left(a_1 - j + 2\right) - 2 \\ &= 2ja_1 - 2j^2 + 2j - ja_1 + j^2 - 2j + a_1 - j \\ &= ja_1 - j^2 + a_1 - j \\ &= a_1\left(j+1\right) - j\left(j+1\right) \\ &= \left(a_1 - j\right)\left(j+1\right) \end{aligned}$$

We get

$$a_{j+1} = \left(a_1 - j\right)\left(j+1\right)$$

Since $a_{2d} = 0$,

$$\begin{aligned}
a_{2d} &= 2d\left(a_1 - 2d + 1\right)\\
0 &= a_1 - 2d + 1\\
a_1 &= 2d - 1
\end{aligned}$$

Therefore,

$$\begin{aligned}
a_j &= j\left(2d - 1 - j + 1\right)\\
&= j\left(2d - j\right)\\
E\left[X_j\right] &= j\left(2d - j\right)
\end{aligned}$$

So the expected value with two players is the product of how much they initially started with.

2 Solution for p players

For p players, the total amount of money is

$$pd = \sum_{1}^{p} n_i$$

If we consider the number of rounds that player i plays starting with n_i, then if we consider one player a small blob with n_i going up against another blob that is the set of $p-1$ players with $2d - n_i$, the expected number of rounds between them is by our previous result is

$$E\left[X_i\right] = n_i\left(pd - n_i\right)$$

Then since the total number of rounds is the sum of all of the bouts between two small blobs within a big blob,

$$X = \sum_{1}^{p} X_i$$

We're actually doubling counting here, because order of the blobs doesn't matter (thanks Jarrett) so

$$X = \frac{1}{2}\sum_{1}^{p} X_i$$

And since the expectation of the sum is the sum of the expectations,

$$E\left[X\right] = E\left[\frac{1}{2}\sum_{1}^{p} X_i\right] = \frac{1}{2}\sum_{1}^{p} E\left[X_i\right] = \frac{1}{2}\sum_{1}^{p} n_i\left(pd - n_i\right)$$

This sum is finite so we can pass the sum over the inside and pull out the $2d$ since it's independent of the sum

Then

$$\begin{aligned}
E\left[X\right] &= \frac{1}{2}\sum_{1}^{p}\left[pdn_i - \left(n_i\right)^2\right] \\
&= \frac{1}{2}\left[\sum_{1}^{p} pdn_i - \sum_{1}^{p}\left(n_i\right)^2\right] \\
&= \frac{1}{2}\left[pd\sum_{1}^{p} n_i - \sum_{1}^{p}\left(n_i\right)^2\right] \\
&= \frac{1}{2}\left[pd\left(pd\right) - \sum_{1}^{p}\left(n_i\right)^2\right] \\
&= \frac{1}{2}\left[\left(pd\right)^2 - \sum_{1}^{p}\left(n_i\right)^2\right]
\end{aligned}$$

3 Explicit solution without scaling k

Solving the homogeneous solution:

We propose r^j is a solution. Then

$$r^j = r^{j+k}\left(\frac{1}{2}\right) + r^{j-k}\left(\frac{1}{2}\right)$$

Dividing by the lowest degree term r^{j-k} yields

$$\begin{aligned}
r^{j-j+k} &= r^{j+k-j+k}\left(\frac{1}{2}\right) + \left(\frac{1}{2}\right) \\
r^k &= r^{2k}\left(\frac{1}{2}\right) + \left(\frac{1}{2}\right)
\end{aligned}$$

Then the characteristic polynomial is

$$0 = r^{2k} - 2r^k + 1$$

Let $r^k = u$. Then

$$\begin{aligned}
0 &= u^2 - 2u + 1 \\
&= \left(u-1\right)^2 \\
1 &= u \\
1 &= r^k
\end{aligned}$$

then

$$r = \begin{cases} \pm 1 & k \in 2\mathbb{Z} \\ 1 & k \in 2\mathbb{Z}+1 \end{cases} = 1$$

(with multiplicity m) since the expected number of rounds can't be negative because $X \geq 1$ and is strictly increasing (you cannot "unplay" a round). Then

$$h_m = \sum_{1}^{m} \alpha_i \left(1\right)^m$$

by our initial condition, $\sum_1^m \alpha_i = 0$, which means $h_m = 0$.

Solving the inhomogeneous solution:

Recall if $b_n \sim f(n)$ satisfies a_n, then it's a particular solution.

We propose $b_j = c$ is a particular solution to our linear recurrence $a_j = 1 + a_{j+k}\left(\frac{1}{2}\right) + a_{j-k}\left(\frac{1}{2}\right)$. Then

$$\begin{aligned} c &= 1 + \left(\frac{1}{2}\right)c + \left(\frac{1}{2}\right)c \\ c &= 1 + c \\ 0 &= 1 \end{aligned}$$

which is a contradiction. Now, we propose $b_j = cj + m$. Then

$$\begin{aligned} cj + m &= 1 + \left[c(j+k) + m\right]\left(\frac{1}{2}\right) + \left[c(j-k) + m\right]\left(\frac{1}{2}\right) \\ cj + m &= 1 + c(j+k)\left(\frac{1}{2}\right) + m\left(\frac{1}{2}\right) + c\left(\frac{1}{2}\right)(j-k) + m\left(\frac{1}{2}\right) \\ cj + m &= 1 + cj + m \\ 0 &= 1 \end{aligned}$$

which is another contradiction. Next, we propose $b_j = cj^2 + mj + e$. Then

$$\begin{aligned} cj^2 + mj + e &= 1 + \left[c(j+k)^2 + m(j+k) + e\right]\left(\frac{1}{2}\right) + \left[c(j-k)^2 + m(j-k) + e\right]\left(\frac{1}{2}\right) \\ cj^2 + mj + e &= 1 + c\left(j^2 + 2jk + k^2\right)\left(\frac{1}{2}\right) + mj + c\left(j^2 - 2jk + k^2\right)\left(\frac{1}{2}\right) + e \\ cj^2 &= 1 + cj^2 + ck^2 \\ -ck^2 &= 1 \\ c &= -\frac{1}{k^2} \end{aligned}$$

So $b_j = -\frac{1}{k^2}j^2 + mj + e$. Since b_j is a proposed solution, then it satisfies our boundary value conditions. Namely,

$$0 = b_0 = 0 + 0 + e$$

$$\begin{aligned} 0 = b_{2d} &= -\frac{1}{k^2}4d^2 + 2md \\ 4\frac{d^2}{k^2} &= 2md \\ \frac{2d}{k^2} &= m \end{aligned}$$

so

$$\begin{aligned} a_j = h_j + b_j &= 0 - \frac{1}{k^2}j^2 + mj + e \\ E[X]_{k\geq 1} &= \frac{j}{k^2}(2d - j) \end{aligned}$$

This means the expected number of rounds (for two players) for a fixed bet larger than 1 is ridiculously smaller than the expected number of rounds with $k = 1$ because of the squared scalar (see Simulations).

Using the same trick from before,

$$E[X]_{k\geq 1} = \frac{1}{2}\sum_{1}^{p}\frac{n_i}{k^2}(pd - n_i) = \frac{1}{2k^2}\left[(pd)^2 - \sum_{1}^{p}(n_i)^2\right]$$

4 Simulations

We present two different simulations of a random walk to empirically calculate the expected value using the algorithm described in Section 1 with Python's random package. We store the counts of the number of rounds and bets (if they change). We believe 10000 is a reasonable number of simulations because it roughly gets the same result as 100000 but in less time. We chose $d = 20$ to simulate what actually happened the night this idea came about. We also chose $k = 1$ and $k = 5$ to get a sense of scale. I think we made a huge mistake setting $n_i = d$ to compute an upper bound, so the comparisons may be misleading. We also couldn't run $k = 1, p = 10$ very easily on my computer.

Here is the full code (python) for a fixed bet for each round. Each player is equally likely to win. And the number of players reduces when one or more players goes to 0.

```
'''fixes the bet for all games'''
#p players
p = 10
#d dollars
d = 20
#fixes the bet for all games
k = 5
Total = []
num_iterations = 10000
for i in range(num_iterations):
    count = 0
    L = [d]*p
    while len(L)>1:
        #pick two poeple to play from the list
        L2 = [L.pop(random.randrange(len(L))) for _ in range(2)]
        #record where those two people are
        #both players put some money in the pot (bet)
        for i in range(len(L2)):
            L2[i] = L2[i] - k
        #pick a random person to win
        index = random.choice(range(len(L2)))
        #winner collects his earnings
        L2[index] = L2[index] + 2*k
        #put those players back in
        L = L + L2
        #zero out this mofo
        while 0 in L:
            del L[L.index(0)]
        #counts the number of rounds
        count = count + 1
    Total.append(count)
#prints the average print(sum(Total)/len(Total))
```

$n_i = d = 20$	$E\left[X\right]_{k\geq 1}$	$E\left[X\right]_{k\geq 1} \leq \frac{1}{2k^2}\left[(pd)^2 - \sum_1^p (d)^2\right]$
$k = 1, p = 2$	394.7048	400
$k = 1, p = 10$	>10894.07	18000
$k = 5, p = 2$	16.0714	16
$k = 5, p = 10$	≈ 716	720
$k = 11, d = 77, p = 7$	≈ 1028	1029

Table 1: Simulation with fixed bet $k = 1, 5$ and $d = 20$ with $p = 2, p = 10$

We decided to simulate a more realistic situation where the bet changes each round, and the amount the players can bet depends on the player with the least amount of money. This led to some interesting phenomena. But first, the full code.

```
'''changes the bet for each round (more realistic)'''
#p players
p = 10
#d dollars
d = 20
Total = []
num_iterations = 10000
for i in range(num_iterations):
    count = 0
    L = [d]*p
    while len(L)>1:
        #pick two poeple to play from the list
        L2 = [L.pop(random.randrange(len(L))) for _ in range(2)]
        #bet changes (depends on the player with least amount)
                k = random.choice(1,min(L2))
        #both players put some money in the pot (bet)
        for i in range(len(L2)):
            L2[i] = L2[i] - k
        #pick a random person to win
        index = random.choice(range(len(L2)))
        #winner collects his earnings
        L2[index] = L2[index] + 2*k
        #put those players back in
        L = L + L2
        #zero out this mofo
        while 0 in L:
            del L[L.index(0)]
        #counts the number of rounds
        count = count + 1
    Total.append(count)
#prints the average print(sum(Total)/len(Total))
```

$d = 20$	$E\left[X\right]_{k>1}$	max bet	avg. bet
$p = 2$	8.1359	20	5.2013667817942695
	8.1907	20	5.2437520602634695
	8.1491	20	5.23915524413739
$p = 10$	21.6476	100	10.65217853249321
	21.2549	100	10.767535956414756
	21.7678	100	10.735618666103143

Table 2: Simulation with $d = 20$ and $p = 2, p = 10$

This result is very interesting. Since max bet is d in both cases, then that means there existed a situation where someone went all in and bet their holdings. This was not in the table, but the minimum was also 1 in both situations. This meant there was a case where at least one person had 1 and the rest had more than one. e.g.

```
>>if k == 1: print(L)
[55, 5, 5, 5, 5, 105, 5, 5, 5, 5]
[54, 4, 4, 4, 4, 104, 14, 4, 4, 4]
[53, 3, 3, 3, 3, 103, 13, 13, 3, 3]
[52, 2, 2, 2, 2, 102, 12, 12, 2, 12]
[51, 1, 1, 1, 1, 101, 11, 11, 1, 21]
[86, 91, 1, 11, 11] [109, 87, 2, 2]
[108, 86, 1, 5]
[113, 86, 1]
[194, 6]
[1, 1, 1, 1, 21, 1, 1, 1, 1, 171]
[26, 7, 167]
[28, 6, 166]
[33, 5, 162]
[46, 154]
[198, 2]
[197, 3]
[198, 2]
[197, 3]
[198, 2]
[199, 1]

>>if k == 100: print(L)
[100,100]
[100,100]
```

References

[1] Professor Zhang at SJSU, who suggested I do this project, along with the blob and scaling idea

[2] Professor Crunk at SJSU, who taught me conditional expectation

[3] Jarrett Jimeno, who reminded me about double counting

[4] Alex, Aj, Christian, Shaft, Vincent, Jarod, Michael, Louis, David, and Kevin. The knuckleheads I went to highschool with.