Avaliação de um estimador para a duração média do jogo da Ruína do Jogador por simulações de Monte Carlo

Orientando: Gustavo Silva Garone

Orientadora: Elisabeti Kira

6 de março de 2025

1 Introdução

Em uma versão simples do Problema da Ruína do Jogador proposto por Pascal (EDWARDS, 1983), dois jogadores, A e B, competem apostando no lançamento de uma moeda honesta. Ao ganhar a aposta, o jogador A recebe um real do jogador B e vice-versa. Além da questão de Pascal sobre a probabilidade do jogador ir à ruína (perder todo seu dinheiro) dado que começou com uma quantia a, também foram estudados o tempo esperado de duração do jogo (STERN, 1975) e, pelo cálculo de equações de diferenças finitas, a variância dessa duração (ANDĚL; HUDECOVÁ, 2012).

Os resultados obtidos por esses autores funcionam apenas para ganho e perda de um real por aposta. Altere o dinheiro disputado por lançamento, introduza a possibilidade de empate ou crie relações de dependência entre todos os lançamentos e os métodos utilizados para calcular analíticamente a esperança da duração do jogo simples já não são mais adequados. (ISHIHARA, 2025)

Neste projeto conjunto com Eduardo Ishihara, propusemos um estimador para passeios aleatórios com regras mais complexas de ganho e perda e o avaliamos através do emprego de simulações. Para isto, empregamos o método de Monte Carlo, uma técnica baseada em amostragem aleatória para emular sistemas complexos, normalmente por meios computacionais (HARRISON, 2010). Dessa forma, é útil no estudo do comportamento de passeios aleatórios com parâmetros diversos. Sendo assim, foi crucial na avalição do estimador proposto, uma vez que replicar e adaptar as técnicas e cálculos usados para obter os resultados nos jogos simples nem sempre é possível com jogos mais complexos.

Empregando-se simulações, conseguimos verificar os casos nos quais estimador é ou não adequado. Exploramos os jogos em que o dinheiro em aposta por lançamento não é apenas um real, abordando jogos cujos resultados teóricos ainda não foram desenvolvidos. Finalmente, com a estrutura do estimador proposto, abrimos caminhos para estudo sobre casos com meios não-homogêneos — isto é, jogos cujo resultado de cada lançamento dependem dos anteriores — e empates.

2 Atividades Desenvolvidas

2.1 Descrevendo jogos

No problema da Ruína do Jogador, além do valor inicial a do primeiro jogador e o dinheiro total em aposta N, podemos empregar uma variável aleatória X para descrever o ganho e perda em cada jogada ao longo do processo estocástico no tempo discreto T. Logo

$$X = \begin{cases} 1, & P(X=1) = p \\ -1, & P(X=-1) = 1-p = q \end{cases}$$

Com a variável aleatória de regras X dessa forma, com ganho e perda de um real, a esperança da duração T de jogos desse tipo em termos de a e N (STERN, 1975) é dada por

$$\mu_{a,N} = E(T_{a,N}) = \begin{cases} a(N-a) & \textbf{se } p = q = 1/2 \\ \frac{a}{q-p} - \frac{N}{q-p}\left(\frac{1-(q/p)^a}{1-(q/p)^N}\right) & \textbf{se } p \neq q \end{cases} \tag{1}$$

Com esse recurso, descreveremos jogos futuros em termos de $J = J(a, N, X)$, em que a representa o valor inicial do jogador A, N representa o dinheiro total em aposta $N = a + b$, com b o saldo do jogador B, e X a variável aleatória que representa as regras de ganho e perda do jogo.

Ao empregar e adaptar as técnicas utilizadas para o desenvolvimento de (1) em outras formas de regras X mais complexas — como com valores de ganho e perda diferentes de 1; ou com mais valores possíveis — percebe-se que se torna inviável a resolução analítica de problemas desse tipo conforme a complexidade de X aumenta. Portanto, na busca de conseguir aproximar esses resultados, foi proposto um estimador para a esperança da duração de jogos desse tipo (ISHIHARA, 2025). Para avaliarmos a precisão desse estimador, empregamos simulações computacionais de jogos atraveś do método de Monte Carlo.

2.2 Introduzindo o estimador

O estimador sob avaliação proposto por Ishihara para a esperança da duração um jogo J é da seguinte forma

$$\widehat{E}(T_J) = \begin{cases} \frac{N-a}{E(X)} & \textbf{se } E(X) > 0 \\ \left|\frac{a}{E(X)}\right| & \textbf{se } E(X) < 0 \end{cases} \tag{2}$$

Consideraremos um passeio $J = J(a = 30, N = 100, X)$, com a variável de regras X dada por

$$X = \begin{cases} 1, & P(X=1) = 0.4 \\ -1, & P(X=-1) = 0.6 \end{cases}$$

De (1), temos que o valor teórico do tempo médio de duração do jogo é

$$\mu_J = \frac{30}{0.2} - \frac{100}{0.2}\left(\frac{1-(0.6/0.4)^{30}}{1-(0.6/0.4)^{100}}\right) = 150$$

De (2), temos que o valor estimado para a esperança da duração de J é $\widehat{E}(T_J) = \frac{30}{0.2} = 150$, o que condiz com o resultado téorico. Isso nos deu certa segurança para continuar explorando o estimador em jogos mais complexos.

2.2.1 Aplicações em casos com regras diferentes

Exploramos casos com uma maior complexidade da variável de regras X. Um destes foi uma simples variação do jogo anterior, mantendo o valor inicial dos jogadores A, $a = 30$ e B, $b = 70$, $N = a + b = 100$, com a variação de que o apostador ganha dois reais por vitória e perde apenas um por derrota. Nesse caso, a variável de regras X é dada por

$$X = \begin{cases} 2, & P(X = 2) = 0.4 \\ -1, & P(X = -1) = 0.6 \end{cases}$$

Denotamos esse novo jogo por $J = (a = 30, N = 100, X)$

Uma vez que (1) não seria apropriada para fornecer a esperança da duração do jogo devido à nova forma de X, impossibilitando uma comparação com o estimador, desenvolvemos algoritmos de simulação baseados no método de Monte Carlo para avaliarmos o estimador (2) da esperança da duração do jogo. A primeira versão do algoritmo, implementada em Python, não satisfez as demandas de tempo e complexidade computacional que desejávamos. Para solucionar essa limitação, migramos para a linguagem Julia, com velocidade computacional competitiva com o C (GODOY et al., 2023), conhecida por sua alta eficiência característica da operação em baixo nível. Assim, conseguimos realizar um número maior de simulações em tempo razoável, melhorando a acurácia dos resultados (RITTER et al., 2011).

Denotamos por $\bar{\psi}_{M,J}$ a média das durações dos M jogos J simulados. Pela Lei dos Grandes Números, temos que $\bar{\psi}_{M,J} \overset{M\to\infty}{\to} \mu_J$. Isto é, a média dos valores simulados se aproxima da esperança de duração conforme o número de simulações M cresce.

Pelo Apêndice A, que contém o código da simulação de $M = 1000000$ jogos J e uma comparação entre as linguagens Julia e Python, obtivemos a média da duração dos jogos simulados $\bar{\psi}_{M,J} = 350.29$.

Aplicando o estimador (2), temos que

$$\widehat{E}(T_J) = \frac{70}{0.2} = 350$$

Não se tratou de um resultado isolado. Sob diferentes regras, o estimador (2), majoritariamente, mostrou forte concordância com os valores simulados.

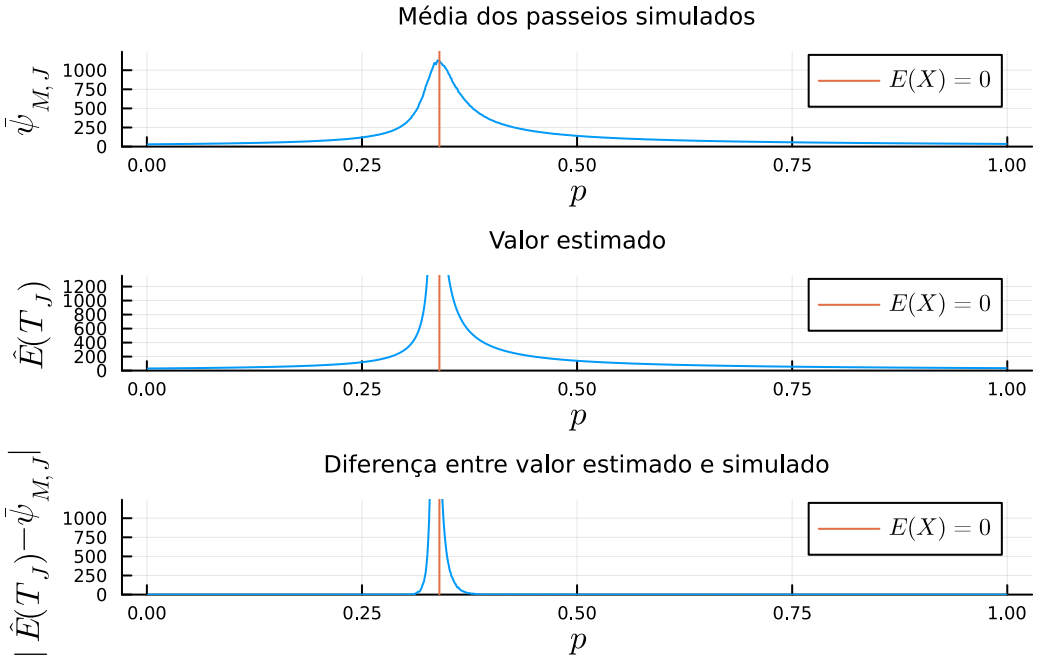
Tabela 1: Resultados de diferentes regras X com $a = 50$ e $N = 100$

$E(X)$	$\widehat{E}(T_J)$	$\bar{\psi}_{M,J}$
4	12.5	13.09
2	25	25.77
1	50	51.39
0.5	100	100.95
-1	50	50.49
-3	16.66	16.66

A Tabela 1 destaca que o estimador consegue, com certa precisão, fornecer uma aproximação da esperança da duração de um jogo com regras diversas, o que, até onde encontramos na literatura, não foi descrito apenas com desenvolvimento analítico.

Não obstante, vale notar que, quanto menor $E(X)$, mais impreciso mostra-se o estimador. Para explorar esse fenômeno, fixamos a, N e os valores que X pode assumir, variando a probabilidade p do jogador A ganhar em uma rodada e, por consequência, alterando $E(X)$.

Figura 1: Comparação entre o estimador e média de simulações
M=10000,a=10,N=100,X={2, p; -1, 1$-p$}, p variando 0.001 em (0,1)



Estes gráficos confirmam a inacurácia de (2) conforme $E(X)$ aproxima-se de 0 para um jogo com outras regras fixadas. Isto ocorre uma vez que, conforme $E(X)$ se aproxima de 0, sua presença no denominador provoca divergências quando comparado com o valor simulado. Dessa forma, o estimador não é adequado para jogos com $E(X)$ próximo ou igual a 0, o que

pode indicar a necessidade de desenvolver, como em (1), uma forma específica para esses casos ou um método de correção generalizada.

3 Conclusão e Passos Seguintes

Os resultados obtidos indicam que o estimador proposto possui grande flexibilidade para diferentes configurações do jogo e mostra-se útil, uma vez que as técnicas analíticas atuais não fornecem um resultado exato para as configurações de jogos propostas. Ademais, o estimador fornece estimativas mais precisas quando há maior circulação de dinheiro entre os jogadores, isto é, valores de a, N e X que promovam um jogo mais duradouro. No entanto, observamos que o estimador tende a divergir rapidamente dos resultados simulados quando a esperança da variável aleatória que rege as regras do jogo aproxima-se de zero.

Um interessante caminho de abordagem que exploraremos é a conexão entre passeios aleatórios e redes (DOYLE; SNELL, 1984). Queremos estudar a existência de conexões entre o estimador proposto e resultados da física. Dentre outras abordagens para explorarmos passeios aleatórios, o emprego de equações de diferenças finitas e o estudo de martingais mostram-se promissores na viabilização do estimador nos casos em que, na forma atual, não é adequado.

No contexto de flexibilizar o estimador, estudaremos casos que envolvam empate entre os jogadores, isto é, lançamentos onde dinheiro não é circulado e o saldo a permanece o mesmo. Acreditamos que, com a forma atual do estimador, empates não contribuem informação suficiente para $E(X)$. Esta falha faz com que subestime o efeito retardante que empates provocam no tempo até o fim do jogo.

Outro desafio identificado foi a complexidade computacional envolvida na simulação de jogos com muitas regras, isto é, quando a variável aleatória pode assumir muitos valores. Essa limitação impõe a necessidade de soluções mais eficientes para lidar com cenários de alta complexidade, o que será um obstáculo a ser superado no estudo de passeios com empate e especialmente passeios não-homogêneos. Para isso, estudaremos a viabilidade da implementação de computação paralela e concorrente no contexto de simulações de Monte Carlo, assim como otimizar nossos métodos de simulação atuais.

4 Bibliografia

ANDĚL, J.; HUDECOVÁ, Š. Variance of the game duration in the gambler's ruin problem. **Statistics & Probability Letters**, v. 82, n. 9, p. 1750–1754, 2012.

DOYLE, P. G.; SNELL, J. L. **Random Walks and Electric Networks**. [s.l.] American Mathematical Soc., 1984.

EDWARDS, A. W. F. Pascal's Problem: The 'Gambler's Ruin'. **International Statistical Review / Revue Internationale de Statistique**, v. 51, n. 1, p. 73–79, 1983.

GODOY, W. F. et al. **Evaluating performance and portability of high-level programming models: Julia, Python/Numba, and Kokkos on exascale nodes**. 2023 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW). **Anais**... Em: 2023 IEEE INTERNATIONAL PARALLEL AND DISTRIBUTED PROCESSING SYMPOSIUM WORKSHOPS (IPDPSW). mai. 2023. Disponível em: <https://ieeexplore.ieee.org/abstract/document/10196600>. Acesso em: 14 fev. 2025

HARRISON, R. L. Introduction To Monte Carlo Simulation. **AIP conference proceedings**, v. 1204, p. 17–21, 5 jan. 2010.

ISHIHARA, E. **Um Estimador para a Duração do Jogo da Ruína do Jogador**. São Paulo: Universidade de São Paulo, 2025.

RITTER, F. E. et al. Determining the Number of Simulation Runs: Treating Simulations as Theories by Not Sampling Their Behavior. Em: ROTHROCK, L.; NARAYANAN, S. (Eds.). **Human-in-the-Loop Simulations: Methods and Practice**. London: Springer, 2011. p. 97–116.

STERN, F. Conditional Expectation of the Duration in the Classical Ruin Problem. **Mathematics Magazine**, v. 48, n. 4, p. 200–203, 1 set. 1975.

5 Apêndice A — Comparação de um mesmo algoritmo em Julia e Python

Para validar nossa decisão de trocarmos de linguagem, comparamos o mesmo (ingênuo) algoritmo em Julia e Python que usamos na avaliação do estimador:

5.1 Julia

```julia
using  LaTeXStrings, Random

function main()
  Random.seed!(1)
  duracoesSoma = 0
  M = 1_000_000
  p = 0.4
  teto = 100
  for i in 1:M
    saldo = 30
    duracao = 0
    while saldo > 0 && saldo < teto
      if p > rand()
        saldo += 2
      else
        saldo -= 1
      end
      duracao += 1
    end
    duracoesSoma += duracao
  end
  display(LaTeXString("Média da duração de \$$M\$ passeios:
                       \$$(round(duracoesSoma/M, digits = 2))\$"))
end

tempo = @elapsed main()
display(LaTeXString("Tempo de execução: $tempo segundos"))
```

Média da duração de 1000000 passeios: 350.29

Tempo de execução: 1.927357837 segundos

5.2 Python

```python
import random
import time


def main():
    random.seed(1)
    duracoes = 0
    M = 1_000_000
    p = 0.4
    teto = 100
    for i in range(M):
        saldo = 30
        duracao = 0
        while saldo > 0 and saldo < teto:
            if p > random.random():
                saldo += 2
            else:
                saldo -= 1
            duracao += 1
        duracoes += duracao
    media = duracoes / M
    print(f"Média da duração de {M} passeios: {media:.2f}")


start_time = time.time()
main()
print(f"Tempo de execução: {time.time() - start_time} segundos")
```

Média da duração de 1000000 passeios: 350.45

Tempo de execução: 66.370362508 segundos

Para a simulação em Python nesse artigo, foi utilizado a biblioteca PyCall que nos permite criar um ambiente Python no Julia, possibilitando executar código Python nessa linguagem. Isso foi necessário para inclusão de sua saída exata no relatório. O desempenho pode ter sido afetado minimamente nos tempos de chamada ao Kernel de execução do Python.

5.3 Decisão de mudança

Desconsiderando o curto tempo de compilação do tempo total do código em Julia, que só precisa ser feito uma vez, temos que essa linguagem costuma ser consideravelmente mais rápida do que o Python, o que foi crucial nessas simulações e justificou nossas mudanças como previsto por resultados anteriores (GODOY et al., 2023).

6 Apêndice B — Código de simulação com regras mais complexas

```
using Random, LaTeXStrings

function simulador(;a = 50, N = 100,
                   X=[[1, 0.5],[-1, 0.5]],
                   M=10_000, semente = 0)
  Random.seed!(semente)
  duracoesSoma = 0
  edex = 0
  for x in X
    edex += x[1] * x[2]
  end
  edex = round(edex, digits=2)

  for i in 1:M
    saldo = a
    duracao = 0
    while saldo > 0 && saldo < N
      p = rand()
      for x in X
        p -= x[2]
        if p < 0
          saldo += x[1]
          break
        end
      end
      duracao += 1
    end
    duracoesSoma += duracao
  end
  display(LaTeXString("A média da duração de \$$M\$ passeios do jogo
                       com \$a = $a, N=$a\$ e \$E(X)=$edex\$ é
                       \$\\bar{\\psi}_{M,J}=
                       $(round(duracoesSoma/M, digits = 2))\$"))
end
simulador(X=[[8, 0.6],[-2, 0.4]], semente = 0) # 4
simulador(X=[[6, 0.5],[4, 0.25], [-8, 0.25]], semente = 1) # 2
simulador(X=[[2, 0.5],[6, 0.2], [-4, 0.3]], semente = 2) # 1
simulador(X=[[1, 0.5],[3, 0.2], [-2, 0.3]], semente = 3) # 0.5
simulador(X=[[-2, 0.5],[-6, 0.2], [4, 0.3]], semente = 4) # -1
simulador(X=[[5, 0.2],[-5, 0.8]], semente = 5) # -3
```

A média da duração de 10000 passeios do jogo com $a = 50, N = 50$ e $E(X) = 4.0$ é $\bar{\psi}_{M,J} = 13.09$

A média da duração de 10000 passeios do jogo com $a = 50, N = 50$ e $E(X) = 2.0$ é $\bar{\psi}_{M,J} = 25.77$

A média da duração de 10000 passeios do jogo com $a = 50, N = 50$ e $E(X) = 1.0$ é $\bar{\psi}_{M,J} = 51.39$

A média da duração de 10000 passeios do jogo com $a = 50, N = 50$ e $E(X) = 0.5$ é $\bar{\psi}_{M,J} = 100.95$

A média da duração de 10000 passeios do jogo com $a = 50, N = 50$ e $E(X) = -1.0$ é $\bar{\psi}_{M,J} = 50.49$

A média da duração de 10000 passeios do jogo com $a = 50, N = 50$ e $E(X) = -3.0$ é $\bar{\psi}_{M,J} = 16.66$