Relatório Semestral

Orientando: Gustavo Silva Garone

Orientadora: Elisabeti Kira

28 de fevereiro de 2025

1 PASSEIO ALEATÓRIO: SIMULAÇÕES E AVALIAÇÃO DE NOVAS ABORDAGENS

1.1 Introdução

Numa versão simples da original proposta 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 proposta por Pascal sobre a probabilidade do jogador ir à ruína, ou seja, 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). Todavia, os resultados obtidos por esses autores funcionam apenas para ganho e perda de um real por aposta.

Neste projeto conjunto com Eduardo Ishihara, analisamos um estimador proposto por este para passeios aleatórios com regras mais complexas de ganho e perda através do emprego dessas simulações. A descrição desses passeios e de suas diferenças quando comparados a uma forma simples do problema clássico da Ruína do Jogador será apresentada ao longo do corpo do texto.

Para isto, empregamos o método de Monte Carlo. A Simulação de Monte Carlo é 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, como elaborado por Ishihara, replicar as técnicas e cálculos para jogos mais complexos nem sempre é possível.

1.2 Atividades Desenvolvidas

1.2.1 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 essa variável aleatória de regras X, podemos obter resultados tratando da esperança da duração T de jogos desse tipo em função de a e N (STERN, 1975)

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

Ao técnicas utilizadas para o desenvolvimento de para outras formas de 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, foi proposto um estimador para avaliação atraveś de simulações de Monte Carlo.

Descreveremos jogos (passeios) futuros em termos de $J = J(a, N, X)$, em que a representa o valor inicial, N representa o valor da barreira superior (dinheiro em aposta) e X a variável aleatória que contém as regras de ganho e perda do jogo. e buscaremos $\mu_J = E(T_J)$ a esperança da duração do jogo J, com X da forma genérica

1.2.2 Introdução do estimador

O estimador sob avaliação é da seguinte forma

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

Consideraremos um passeio $J_{30,100,X}$, com (**?@eq-var-x**)

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

De (1), temos que

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

De (2), como $E(X) = -0.2$, temos $\widehat{E(T_J)} = \frac{30}{0.2} = 150$. O que condiz com o resultado téorico.

1.2.2.1 Aplicações em casos com regras diferentes

Passamos a estudar casos em que (1) deixa de funcionar devido a uma maior complexidade de X.

Um desses jogos foi uma simples variação do jogo anterior, $a = 30$ e $N = 100$, mas fizemos o apostador ganhar dois por vitória, ainda perdendo apenas um por derrota. Nesse caso, temos nossa variável de regras X

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

E o novo jogo $J_{30,100,X}$

Uma vez que (1) não seria apropriada devido a nova forma de X, desenvolvemos algoritmos de simulação baseados no método de Monte Carlo para avaliarmos o estimador. A primeira versão do algoritmo, implementada em Python, não satisfez as demandas de tempo e complexidade 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 rodar um número maior de simulações em tempo razoável para melhorar a acurácia dos resultados (RITTER et al., 2011).

Chamaremos de $\bar{\psi}_{M,J}$ a média das durações dos M passeios J simulados. Confere que $\bar{\psi}_{M,J} \overset{M\to\infty}{\to} \mu_J$. Isto é, os valores simulados se aproximam da esperança de duração conforme o número de simulações M cresce.

Com esse recurso, $\bar{\psi}_{10000,J} \approx 350$ (veja o Apêndice A para o código dessa simulação assim como uma comparação entre as linguagens Julia e Python).

Considerando que $E(X) = 0.2$, Aplicando o estimador (2), temos que

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

o estimador sob diferentes regras e, na maioria dos casos, mostrou forte concordância com os valores simulados. Isso destaca a vantagem do estimador: estimar a duração de um jogo com regras diversas, o que, até onde encontramos na literatura, não foi descrito apenas com desenvolvimento analítico.

Tabela 1: Resultados de diferentes passeios com $a = 50, N = 100$ e X variando

$E(X)$	$\widehat{E}(T_J)$	$\bar{\psi}_{M,J}$
10	12.5	13.14
5	25	25.44
3	50	51.14
0.5	100	101.12

$E(X)$	$\widehat{E}(T_J)$	$\bar{\psi}_{M,J}$
-3	50	51.14
-5	16.66	16.66

Note que, como esperado, quanto menor $E(X)$, mais impreciso tende a ser o estimador, por 1.

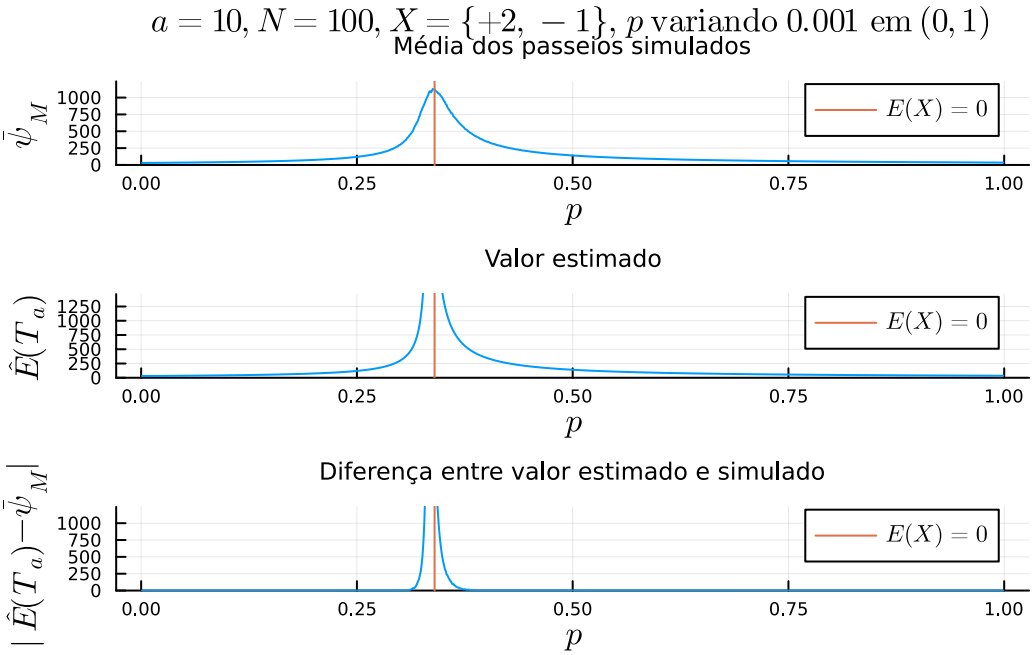

Figura 1: Comparação entre o estimador e simulações a=10,N=100,X={+2,−1},p variando 0.001 em (0,1)

Estes gráficos ilustram essa inacurácia conforme $E(X)$ se aproxima de 0 para um jogo com outras regras fixadas. Isto é restrições do uso do estimador residem nas regras que levam a $E(X)$ próximo de 0, uma vez que, conforme $E(X)$ se aproxima de 0, sua presença no denominador provoca divergências quando comparado com o valor simulado.

Estudamos também como simplificar a variável aleatória de regras X através de médias quando esta pode assumir mais que dois valores.

1.3 Conclusão e Passos Seguintes

Os resultados obtidos indicam que o estimador proposto possui grande flexibilidade para diferentes configurações do jogo e mostra-se especialmente útil, pois as técnicas atuais não são capazes de fornecer um resultado teórico exato para as configurações de jogos propostas. Ademais, o estimador fornece estimativas mais precisas quando há maior circulação de dinheiro entre os jogadores. 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 devido ao denominador.

Outro desafio identificado foi a complexidade computacional envolvida na simulação de jogos com muitas regras, isto é, quando a variável aleatória pode assumir um grande número de 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 do caso não homogêneo.

Um interessante caminho de abordagem que buscaremos explorar é a conexão entre passeios aleatórios e redes elétricas como descrito por DOYLE; SNELL (1984). Queremos descobrir se existem conexões entre o estimador proposto e fórmulas da física, além de outras abordagens para estudarmos passeios aleatórios, como o emprego de equações de diferenças finitas e o estudo de martingais para viabilização do estimador nesses casos.

1.4 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.

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.

1.5 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 para avaliarmos o estimador:

1.5.1 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 em segundos: $tempo"))
```

Média da duração de 1000000 passeios: 350.29

Tempo de execução em segundos: 0.590661401

1.5.2 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: 37.268186881 segundos

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

1.5.3 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 de GODOY et al. (2023)

1.6 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 = nothing)
  if !isnothing(semente)
    Random.seed!(5)
  end
  duracoesSoma = 0

  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("Média da duração de \$$M\$ passeios:
                       \$$(round(duracoesSoma/M, digits = 2))\$"))
  return duracoesSoma / M
end
simulador(X=[[8, 0.6],[-2, 0.5]], semente = 1) # 4
simulador(X=[[6, 0.5],[4, 0.25], [-8, 0.25]], semente = 2) # 2
simulador(X=[[2, 0.5],[6, 0.2], [-4, 0.3]], semente = 3) # 1
simulador(X=[[1, 0.5],[3, 0.2], [-2, 0.3]], semente = 4) # 0.5
simulador(X=[[-2, 0.5],[-6, 0.2], [4, 0.3]], semente = 5) # -1
simulador(X=[[5, 0.2],[-5, 0.8]], semente = 6) # -3
```

Média da duração de 10000 passeios: 13.14

Média da duração de 10000 passeios: 25.44

Média da duração de 10000 passeios: 51.14

Média da duração de 10000 passeios: 101.12

Média da duração de 10000 passeios: 51.14

Média da duração de 10000 passeios: 16.66

```
16.6648
```