Departamento de Matemática
Mestrado em Engenharia Matemática

Simulação e Análise de Probabilidades nos Jogos de Bingo: Aspectos Estatísticos e Modelagem

Análise Numérica e Simulação - M4076

Beatriz Carvalho (202107558)
Rita Chamusca (202106828)
João Amaral (202105032)
Mariana Pereira (202107839)

Dezembro 2024/2025

Índice

1 Introdução

1.1 Contextualização do Bingo

O bingo é um dos exemplos mais conhecidos de jogos de sorte (ou azar). A sua divulgação remonta aos Estados Unidos e a Edwin Lowe, um fabricante de brinquedos.

Inicialmente, o jogo chamava-se Beano, mantendo a ideia de cartões com números e da retirada aleatória de discos numerados de um saco. A maior diferença prende-se à forma de alcançar a vitória: no Beano, seria apenas necessário completar uma linha horizontal ou diagonal, e anunciar em voz alta "Beano!". Uma possível inspiração do Beano seria os jogos de lotaria europeus, com origens que remontam à Itália, ao ano de 1530, onde surgiam sob o nome "Giuocco del Lotto d´Italia", ou *jogo de lotaria de Itália*. Edwin Lowe espalhou o jogo aos seus familiares e conhecidos, começando, eventualmente, a comercializar cartões de jogo, ao qual chamou "Bingo".

As mecânicas do jogo mantêm-se relativamente constantes, mesmo que os cartões de jogo possam variar conforme os países ou os contextos de jogo. A forma de jogar não se altera, o cerne do jogo continua a estar sobre a realização do "Bingo!" através de números retirados, de forma aleatória e sem reposição.

Em Portugal, o bingo foi legalizado em 1982, e é mais comum em casinos ou casas especializadas de bingo.

1.2 Objetivos

O Bingo apresenta um excelente modelo, vastamente difundido, de combinações aleatórias sem reposição, e serve então como um bom suporte para estudar as probabilidades associadas à dita 'sorte no jogo'.

Para além disso, a simulação de cartões e jogos, com diferentes números de jogadores, revela-se uma boa forma de aplicar conceitos e técnicas de simulação computacional.

Temos portanto como objetivos fazer a ligação entre a teoria matemática e um jogo comum, e perceber de que forma é que este é influenciado por fatores como o número de cartões em jogo, e o número de jogadores.

Uma melhor compreensão do jogo e dos seus mecanismos levará, em princípio a um estilo de jogo mais consciente, levando à minimização da perda de dinheiro.

A aplicação de técnicas teóricas de simulação e geração de números aleatórios a algo mais familiar surge como outro objetivo, visto que a sua aplicação fará a sua compreensão mais fácil, e mais profunda.

1.3 Motivação

A motivação para este trabalho surge principalmente da interação notável entre a teoria que vamos aprendendo, com um jogo bastante comum, que todos já tinhamos jogado. Sendo o Bingo um jogo que, *idealmente*, é jogado por mais do que uma pessoa, surguem imediatamente questões sobre os números ideais de jogadores, cartões e de jogadas. Aqui temos uma oportunidade de trabalhar no campo da simulação, e de trabalhar com a geração de números aleatórios.

Esperamos conseguir reunir munição para nunca *(voltar a)* perder dinheiro numa casa de Bingo, ou então para conseguirmos ser campeões invictos dos jogos mais *caseiros*.

2 Fundamentação Teórica

2.1 As Regras do Bingo

Cada cartão de bingo tem tamanho 3×9, 4 espaços vazios por linha, e contém 15 espaços preenchidos. Os números de cada uma das 9 colunas pertencem a intervalos fixos: 1-10, 11-20, 21-30, 31-40, 41-50, 51-60, 61-70, 71-80 e 81-90, respetivamente. Os números são sorteados aleatoriamente, sem reposição da lista de números entre 1 e 90 até algum jogador completar uma linha, e posteriormente, completar o cartão para realizar Bingo.

2.2 Probabilidade de obter Linha e Bingo

Dado que a seleção dos números segue uma distribuição hipergeométrica (para mais informação sobre esta distribuição ver [4]) com parâmetros N : *'Número de elementos na população'*, K : *'Número de sucessos na população'* . A probabilidade de obter linha ou bingo na jogada n é obter $K-1$ sucessos em $n-1$ jogadas multiplicada pela probabilidade de sair o número que falta.

Sendo a probabilidade dada por:

$$P(X=K) = \frac{\binom{K}{K-1}\binom{N-K}{n-(K-1)}}{\binom{N}{n-1}} \times \frac{1}{N-(n-1)}$$

.

Neste caso temos $N=90, K \in \{5,15\}$ e n é um parâmetro que varia, indicando quantos números já foram sorteados.

Por exemplo: A probabilidade de obter linha na 25ª jogada é dada por:

$$P(X=5) = \frac{\binom{5}{4}\binom{85}{21}}{\binom{90}{24}} \times \frac{1}{90-(25-1)} \approx \ 0.0002834708$$

E de obter bingo na jogada 43ª jogada é

$$P(X=15) = \frac{\binom{15}{14}\binom{75}{29}}{\binom{90}{42}} \times \frac{1}{90-(43-1)} \approx \ 1.67 \times 10^{-6}$$

2.2.1 Os Cartões de Jogo

Existem inúmeras combinações para expor os números num cartão de bingo. Por exemplo:

2			36		55		74	82
	16	23		49		62		85
9	17		39		58		76	

Figura 1: Exemplo de Cartão Válido

2	5		36		55		74	82
	16	23		49		62		85
9	17		87		58		76	

Figura 2: Exemplo de Cartão Inválido

Considerando a hipótese onde todos os quadrados são preenchidos, cada coluna seria escolhida de diferentes maneiras, o número de cartões possíveis seria $(A_3^{10})^9$, com $A_k^n = \frac{n!}{(n-k)!}$.

Contudo, temos a restrição de apenas 5 números preenchidos por linha, e um total de 15 números distintos. Para encontrar o número possível de cartões consideramos o seguinte problema: seja x o número de colunas com as 3 linhas preenchidas, y o número de colunas com 2 linhas preenchidas, e z, o número de colunas com apenas uma linha preenchida. Atendendo ao contexto do problema, temos que as restrições para o preenchimento das células dos cartões definem o seguinte plano:

$$3x + 2y + z = 15$$

que restringe a totalidade de células preenchidas a 15, que deve ser intercetada com a seguinte região:

$$5 \leq x + y + z \leq 9$$

que restringe o número de colunas preenchidas a um mínimo de 5 e a um máximo de 9. Além disso, por motivos de contexto, consideramos também as restrições $x \geq 0$, $y \geq 0$ e $z \geq 0$, obtendo as soluções abaixo representadas.

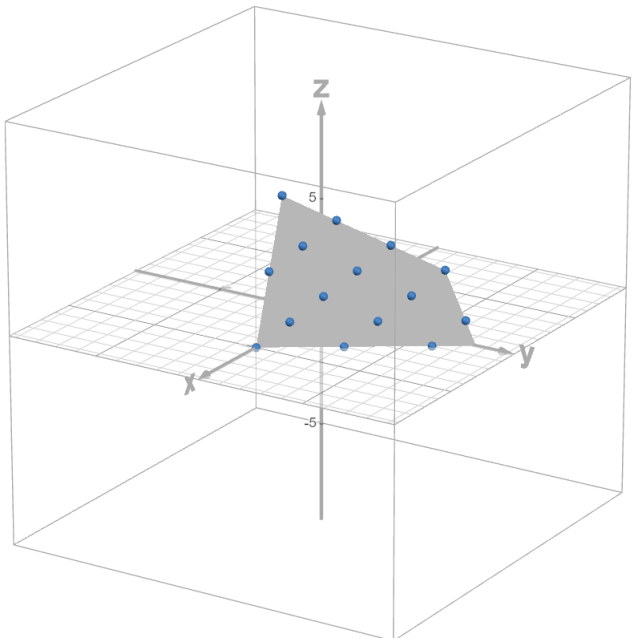

Figura 3: Visualização da região admissível e soluções inteiras

As soluções (representadas a azul) inteiras e positivas da interceção do plano $3x + 2y + z = 15$ com a região $5 \leq x + y + z \leq 9$ (representada pela região a cinzento) são:

$$(0,6,3),(0,7,1),(1,4,4),(1,5,2),(1,6,0),(2,2,5),(2,3,3),(2,4,1),(3,0,6),$$

$$(3,1,4),(3,2,2),(3,3,0),(4,0,3),(4,1,1),(5,0,0).$$

Contudo, este procedimento não garante um máximo de 5 elementos por linhas, portanto seria necessário estudar cada caso separadamente.

Para o caso onde temos, por exemplo, $(5,0,0)$, temos apenas 5 colunas totalmente preenchidas. Os cartões com esta configuração possível são:

$$C_5^9 (A_1^{10})^5 (A_1^9)^5 (A_1^8)^5$$

Onde C_5^9 é a escolha das 5 colunas a ocupar, das 9 possíveis, e $(A_1^{10})^5 (A_1^9)^5 (A_1^8)^5$ é a escolha dos números para as 5 colunas das primeiras, segundas e terceiras linhas, respetivamente. Portanto, existem $2.43799621632 \times 10^{16}$ cartões com 5 colunas com as 3 linhas preenchidas.

Para o caso $(4,1,1)$, temos 4 colunas com as 3 linhas preenchidas, 1 coluna com 2 linhas preenchidas e 1 coluna apenas com 1 linha preenchida. Os cartões com esta configuração possível são:

$$C_4^9 (A_1^{10})^4 (A_1^9)^4 (A_1^8)^4 \times C_1^5 (A_1^{10})(A_1^9) \times 4A_1^{10}$$

Onde C_4^9 é a escolha das 4 colunas, das 9 possíveis, para as colunas que têm as 3 linhas completas e para a escolha dos números nestas colunas existem $(A_1^{10})^4(A_1^9)^4(A_1^8)$ hipóteses: para a primeira linha de cada coluna temos A_1^{10} escolhas, para a segunda linha A_1^9 e para a terceira A_1^8. De seguida, das 5 colunas restantes, escolhe-se uma para ficar com 2 linhas preenchidas e o raciocínio para a escolha dos números mantém-se, logo temos $C_1^5(A_1^{10})(A_1^9)$ formas de escolher. Por fim, $4A_1^{10}$ representa a escolha de 1 coluna das 4 ficar com uma linha preenchida e o cartão ser válido. Temos, portanto, 6.094991×10^{17} cartões nesta condição.

Veremos ainda, para o caso (4,0,3), onde há 4 colunas com as 3 linhas preenchidas, e 3 colunas apenas com 1 linha preenchida. Os cartões com esta configuração possível são:

$$C_4^9(A_1^{10})^4(A_1^9)^4(A_1^8)^4 \times C_3^5 3!(A_1^{10})3$$

Onde C_4^9 é a escolha das 4 colunas, das 9 possíveis, para as colunas que têm as 3 linhas completas e para a escolha dos números nestas colunas existem $(A_1^{10})^4(A_1^9)^4(A_1^8)$ hipóteses: para a primeira linha de cada coluna temos A_1^{10} escolhas, para a segunda linha A_1^9 e para a terceira A_1^8. Por fim, C_3^5 é a escolha de 3 colunas das 5 restantes para ficar com 1 linha preenchida, 3! são as possíveis escolhas da linha, pois, se, por exemplo, numa coluna vazia preencher a primeira linha, na próxima coluna não é possível preencher a mesma linha, caso contrário o cartão seria inválido. A escolha para os números nas linhas segue a mesma linha de pensamento. Assim, existem 2.031664×10^{18}.

Calcular todos os casos possíveis resultaria num trabalho bastante extenso, e não sendo o objetivo deste projeto, não o iremos fazer. Todavia, podemos fazer uma estimativa do mínimo de cartões possíveis e do máximo de cartões possíveis para uma maior perceção da grandeza do problema.

Para o caso mais geral, ou seja, para o máximo de cartões temos:

$$(C_5^9)^3 \times (A_1^{10})^{15}$$

Onde para cada linha escolhemos 5 colunas para preencher e para cada espaço temos sempre 10 escolhas, resultando em 2.000376×10^{21}

Por outro lado, o número mínimo de cartões é:

$$(C_5^9)^3(A_1^{10})^5(A_8^9)^5(A_1^8)^5$$

Onde consideramos que a escolha do número colocado depende no valor escolhido anteriormente. Há, então, no mínimo 3.870563×10^{20}.

Daqui concluímos que o número possível de cartões válidos pertence ao intervalo $[3.870563 \times 10^{20}, 2.000376 \times 10^{21}]$

2.2.2 A Primeira Vitória

O jogo recai sobre uma retirada aleatória de números entre 1 e 90, sem reposição, até alguém completar o seu cartão.

O bingo necessita de três linhas completas, com um total de 15 números distintos. Como cada cartão tem 15 números diferentes, no melhor dos casos, a vitória acontece na 15ª jogada. A probabilidade de termos 2 cartões a ganhar ao mesmo tempo é muito pequena, dado que a proporção do número de jogadores e do número de cartões é muito baixa, portanto, vamos explorar a probabilidade em função do número de jogadores. Tendo isso em conta, a probabilidade de um jogo acabar na 15ª jogada é dada por:

$$P(X = 15) = \frac{\binom{15}{14}}{\binom{90}{14}} \times \frac{1}{90 - (15 - 1)} \approx \ 2.18 \times 10^{-17}$$

Então, a probabilidade de ninguém ganhar na 15ª jogada é $(1 - 2.18 \times 10^{-17})^m$, sendo m o número de jogadores naquela ronda. Portanto, a probabilidade de existir vencedor é dada por $(1 - (1 - 2.18 \times 10^{-17})^m)$. [1]

Podemos agora procurar responder a quantos jogadores são necessários para que esta probabilidade seja pelo menos 0.01.

Para isto, basta resolver para m,

$$1 - (1 - 2.18 \times 10^{-17})^m \geq 0.1$$

Devido ao facto de 2.18×10^{-17} ser menor que o ϵ -*máquina* do computador, tivemos de recorrer ao *WolframAlpha* e determinar esse número e obtemos um resultado de aproximadamente 5×10^{14}.

3 Simulação

A abordagem implementada envolve a geração de cartões de bingo seguindo as regras tradicionais, com números distribuídos em colunas delimitadas por intervalos específicos. Em seguida, realiza-se o sorteio aleatório de um conjunto de 90 números, sem reposição, simulando o processo de chamada dos números em um jogo real.

Cada cartão é monitorizado para identificar o momento em que uma linha ou o cartão completo é preenchido, registando-se o número de jogadas necessárias para cada. Finalmente, a simulação é repetida 1000 vezes para registar a variabilidade dos resultados e estimar distribuições de frequência, médias e desvios-padrões.

3.1 O Algorítmo de Fisher - Yates

Para realizar um jogo de Bingo, há normalmente um responsável por retirar os números, seja do típico bombo de Bingo, ou de outro qualquer recipiente, e posteriormente voltar a misturar os restantes números, e assim sucessivamente até o jogo terminar.

Para a realização desta simulação, poderíamos, simplesmente, ter utilizado uma das imensas funções possíveis que realizariam uma (ou várias) permutações entre os números, de 1 a 90.

Contudo, isto não seria uma simulação representativa do que realmente acontece nos jogos de bingo conhecidos. Por isso decidimos implementar o algoritmo de Fisher-Yates [2].

Pondo em pratica as ideias principais do algoritmo, conseguimos gerar um código (Código 1) que, após cada retirada de um número, baralhe novamente dos os números antes de retirar o próximo, o que melhor reflete o jogo do Bingo. Assim, geramos todos os números a sortear, a partir de uma $U \sim U(0,1)$.

```
        #criar o algoritmo de Fisher Yates
start_time = time.time()
def fisher_yates(lista):
    runif = random.uniform(0,1)

    for i in range(len(lista)-1,0,-1):
        j=math.floor(len(lista)*random.uniform(0,1))
        lista[i],lista[j]= lista[j], lista[i]

    return lista


def numeros(n,k):
```

```
    lista = list(range(1, n+1))
    count=0
    output=[]
    while count<k:
        fisher_yates(lista)
        output.append(lista[0])
        lista=lista[1:]
        count=count + 1
    return output
```

Código 1: Algoritmo de Fisher-Yates e Geração de Números

3.2 Simulação dos cartões

Para simular n cartões de bingo para cada simulação, sendo n o número de apostadores em jogo, construimos o seguinte código Python (Código 2), que cria os cartões com as regras específicas.

```
    # Listas com os intervalos de valores para cada coluna
    listas = [
                list(range(1, 11)),
                list(range(11, 21)),
                list(range(21, 31)),
                list(range(31, 41)),
                list(range(41, 51)),
                list(range(51, 61)),
                list(range(61, 71)),
                list(range(71, 81)),
                list(range(81, 91)),
                ]

#Cria um cartao de bingo com as regras especificas
def criar_jogo_bingo(listas):
colunas_ocupadas = []
for _ in range(3):
    colunas = sorted(random.sample(range(1, 10), k=5))
    colunas_ocupadas.extend(colunas)

n_de_linhas_ocupadas_na_coluna = [colunas_ocupadas.count(i)
                                  for i in range(1, 10)]


numeros_na_coluna = [
    random.sample(lista, n_de_linhas_ocupadas_na_coluna[idx])
```

```
    for idx, lista in enumerate(listas)
]

for n in range(1, 10):
    for i in range(3):
        if n not in colunas_ocupadas[i * 5:(i + 1) * 5]:
        numeros_na_coluna[n - 1].insert(i, 0)

return [[numeros_na_coluna[j][i] for j in range(9)]
                                 for i in range(3)]
```

Código 2: Criação dos cartões de Bingo

Para garantir que o o cartão segue os padrões pretendidos, começamos por criar as listas com as condições de números que podem aparecer em cada coluna.

De seguida, geramos aleatoriamente as 5 colunas ocupadas em cada uma das 3 linhas e contamos quantas vezes cada coluna foi ocupada nas 3 linhas. O próximo passo consiste em preencher cada coluna com números aleatórios (respeitando os intervalos) e adicionar zeros nas posições não ocupadas. Por fim, retorna o cartão de bingo como uma matriz 3×9 em que cada linha contém 5 números e 4 zeros (que representam os espaços vazios).

3.3 Simulação de Monte Carlo

A simulação de Monte Carlo é uma técnica poderosa utilizada para resolver problemas que envolvem incertezas e comportamentos estocásticos. Este método baseia-se na geração de um número suficientemente grande de amostras aleatórias para estimar os resultados esperados, permitindo visualizar detalhadamente as distribuições de probabilidade e analisar o impacto de variáveis aleatórias em sistemas complexos. [3]

Assim, apesar do método de Monte Carlo apresentar várias vantagens, como a flexibilidade em modelar problemas complexos e a facilidade em incorporar incertezas, é também importante mencionar as limitações do mesmo. A precisão depende do número de iterações, isto é, mais simulações resulta em mais precisão, o que implica num maior custo computacional, podendo tornar-se computacionalmente intensivo, dependendo do número de simulações.

Neste trabalho, a simulação de Monte Carlo é aplicada no estudo das probabilidades e padrões dos jogos de bingo. Este jogo, regido por regras simples, mas altamente influenciado pela aleatoriedade, oferece-nos um espaço para explorações estatísticas. O nosso objetivo é analisar o número médio de jogadas necessárias para completar uma linha ou atingir o bingo, considerando diferentes números de participantes em 1000 execuções repetidas.

3.3.1 Aplicação da Simulação de Monte Carlo no bingo

Ao empregar a simulação de Monte Carlo, é possível abordar perguntas-chave, como: qual é o número médio de jogadas necessárias para completar uma linha ou bingo? Como é que a quantidade de participantes no jogo influencia esses resultados? Esta metodologia permite não apenas responder a tais questões, mas também explorar a aplicabilidade de princípios estatísticos, como o Teorema do Limite Central, e reforça a relevância da simulação computacional como ferramenta de análise em sistemas estocásticos.

```
    #Gerar n cartoes por jogada num total de 1000 jogadas
# Variaveis para armazenar resultados
numeros_minimos_linha = []
numeros_minimos_bingo = []
empty_bingo = [[0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0]]
# Executar o processo 1000 jogadas
for iteracao in range(1000):
    n = 50 # Quantidade de jogos de bingo numa jogada
    lista_bingos = [criar_jogo_bingo(listas)
                    for _ in range(n)]
    numeros1 = numeros(90, 90)

    linhas_minimas = []
    bingos_minimos = []
    jogadas = 0

    for numero in numeros1:
        jogadas += 1

        for jogo in lista_bingos:
            # Verifica se o jogo inteiro esta zerado
            # Remove jogos completos
            if jogo == empty_bingo:
                lista_bingos.remove(empty_bingo)
                continue
            # Atualiza o jogo com o numero sorteado
            for linha in jogo:
                for i in range(len(linha)):
                    if linha[i] == numero:
                        linha[i] = 0

                # Verifica se a linha est  zerada
```

```
                if all(x == 0 for x in linha):
                    linhas_minimas.append(jogadas)
                if jogo == empty_bingo:
                    bingos_minimos.append(jogadas)

    # Garante que as listas n o estao vazias
    numeros_minimos_linha.append(min(linhas_minimas))
    numeros_minimos_bingo.append(min(bingos_minimos))



# Resultados
print("M nimos de jogadas para completar uma linha:",
                                numeros_minimos_linha)
print("Minimos de jogadas para completar um bingo:",
                                numeros_minimos_bingo)
```

Código 3: Processo para verificar em que jogada existe linha ou Bingo

3.3.2 Resultados

Para efeitos de estudo deste trabalho consideramos o caso em que existe apenas 1 jogador e simulamos 1000 jogos. De cada vez, contabilizamos a jogada em que ocorreu linha e Bingo. Na tabela abaixo podemos observar os resultados obtidos:

Média de nº de rodadas: Linha	Média de nº de rodadas: Bingo
64.51	85.28
Desvio Padrão do nº rodadas: Linha	Desvio Padrão do nº rodadas: Bingo
12.34	4.95

Tabela 1: Médias de 1000 jogos com 1 jogador

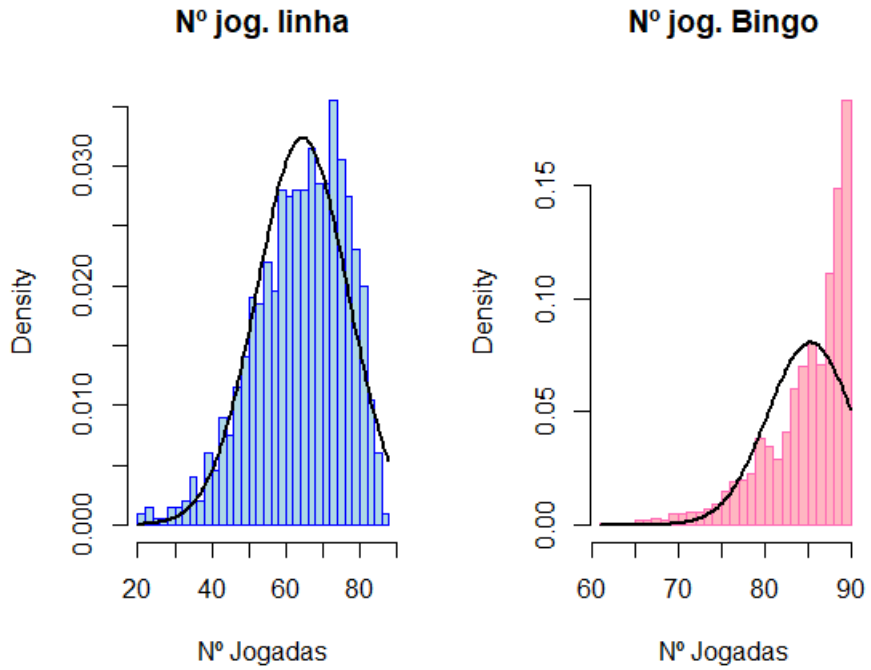

Figura 4: Histogramas de 1 jogador ao longo de 1000 jogos

Consideramos também o caso em que mantivemos o número de jogos mas agora com 50 jogadores e contabilizamos a jogada em que a primeira linha e Bingo ocorriam. Na tabela abaixo podemos observar os resultados obtidos:

Média de nº de rodadas: Linha	Média de nº de rodadas: Bingo
32.12	69.12
Desvio Padrão do nº rodadas: Linha	Desvio Padrão do nº rodadas: Bingo
6.49	4.97

Tabela 2: Médias de 1000 jogos com 50 jogadores

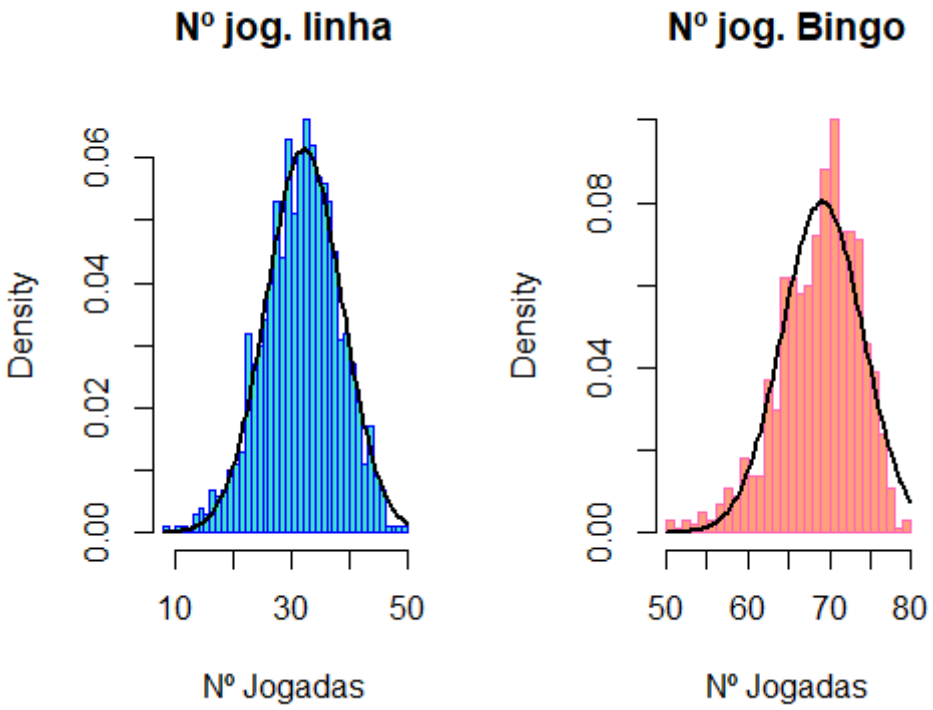

Figura 5: Histogramas de 50 jogador ao longo de 1000 jogos

Por último temos o caso em que temos 1000 jogadores e 1000 jogos e voltamos a contabilizar a jogada em a primeira linha e Bingo ocorriam. Na tabela abaixo podemos observar os resultados obtidos:

Média de nº de rodadas: Linha	Média de nº de rodadas: Bingo
18.80	58.16
Desvio Padrão do nº rodadas: Linha	Desvio Padrão do nº rodadas: Bingo
3.79	3.99

Tabela 3: Médias de 1000 jogos com 1000 jogadores

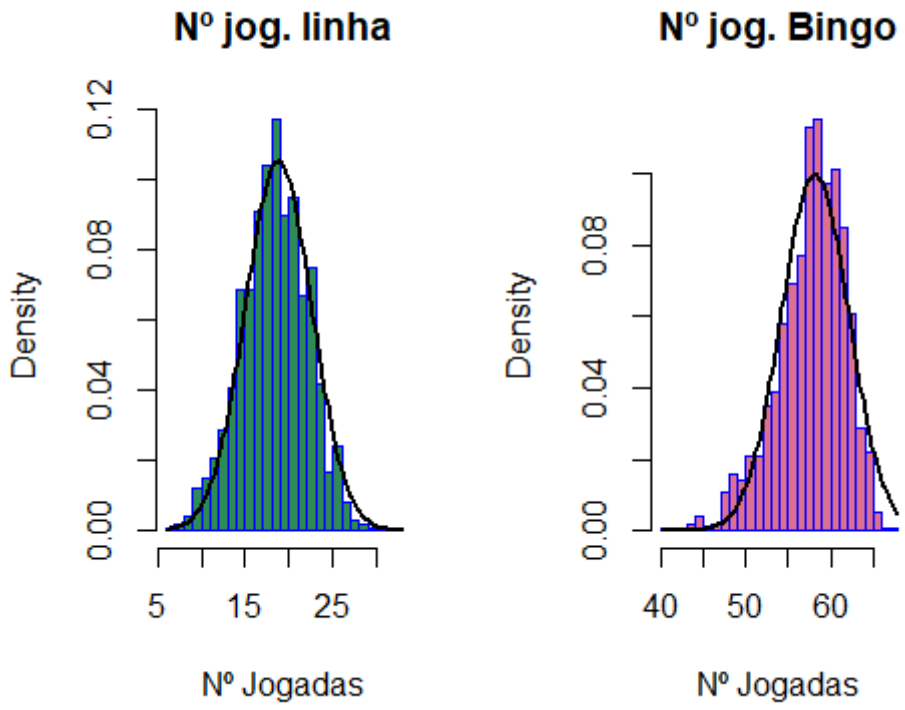

Figura 6: Histogramas de 1000 jogador ao longo de 1000 jogos

Com o código apresentado e depois de realizadas as simulações obtemos os seguintes tempos de execução:

- Para 1 pessoa: O tempo de execução foi de aproximadamente 2.236 segundos
- Para 50 pessoas: O tempo de execução foi de aproximadamente 36.527 segundos
- Para 1000 pessoas: O tempo de execução foi de aproximadamente 573.153 segundos

3.4 Conclusões das Simulações

Após realizar várias simulações e observar os histogramas, é fácil verificar que o número de jogadas necessárias para obter linha e bingo vai diminuindo à medida que o número de jogadores aumenta, o que é um resultado esperado, visto que estão em jogo mais cartões, portanto é de esperar que um desses atinja linha ou bingo mais rápido. Também é notável que, dado o elevado número de cartões, é altamente improvável que haja a eventualidade de dois jogadores ganharem ao mesmo tempo.

Além do número de jogadores influenciar drasticamente os resultados, é também a única variável ao longo dos jogos - não controlamos qual será o nosso cartão, nem qual o ordem de saída dos números - é de notar que, individualmente, quantos mais jogadores, menor é a probabilidade de ganhar, por isso devemos escolher bem quando, e onde, colocar o nosso dinheiro. Por exemplo, como os cartões são independentes uns dos outros, podemos esperar que em 100 jogadores, ao longo de 100 rondas, vamos ganhar pelo menos uma vez, algo que obviamente pode não acontecer. Assim, caso o retorno seja um pouco menor que o número de jogadores, podemos, tendo em conta que a longo prazo vamos perder dinheiro, apostar. Isto é muito mais rentável para nós quando comparamos com a situação de ser um prémio muito maior, para por exemplo, 1000 pessoas, visto que, em contra partida, teremos de apostar mais e jogar mais.

Um outro caso interessante é a situação de estarmos numa sala de bingo com apenas um outro jogador, e não estarmos particularmente interessados no dinheiro que vamos ganhar ou perder, mas sim apenas na vitória. Já vimos que o número de jogadas para obter a vitoria é bem alta, portanto o primeiro jogador ao comprar 48 cartões para competir com o seu adversário, estará à espera de ganhar mais vezes devido ao maior número de cartões em sua posse.

De uma forma geral, não existe nenhum processo que amplie a probabilidade de ganhar, o que torna o jogo muito mais interessante e viciante, dado que estamos verdadeiramente entregues à sorte.

4 Conclusão

Neste trabalho, exploramos primariamente a aplicação de técnicas como o método de Monte Carlo e o algoritmo de Fisher-Yates. O método de Monte Carlo, permitiu a modelação eficiente do comportamento estocástico dos sorteios de bingo, reproduzindo a aleatoriedade dos eventos de jogo. A realização da simulação de vários jogos permitiu uma estimativa das distribuições das probabilidades associadas ao jogo, nomeadamente a "linha"e o "Bingo". Este método permite também a variação nos parâmetros, neste caso, o número de jogadores, e auxilia na perceção de como é que essa variação afeta o decorrer do jogo.

O algoritmo de Fisher-Yates foi utilizado de forma a que a seleção dos números fosse o mais aleatório possível, utilizando apenas a geração de uma uniforme entre 0 e 1, baralhando os números de forma eficiente para garantir uma retirada justa e imprevisível, o que é uma reflexão adequada dos eventos de um jogo real. Isto serviu como garantia de que, independentemente do número de simulações, a retirada de números se mantivesse consistentemente justa.

O uso combinado do método de Monte Carlo e do algoritmo de Fisher-Yates leva-nos à nossa motivação inicial, visto que permite aprofundar a compreensão das dinâmicas do jogo e a otimização das estratégias vencedoras.

A utilização destas metodologias não só auxiliou de forma significativa o estudo das probabilidades associadas a jogos de sorte, como permitiu uma maior compreensão do funcionamento das mecânicas destes mesmos jogos.

Com a contagem do número de cartões, percebemos que um simples processo de apenas contar números em certas posições, pode revelar-se bastante complicado, devido ao elevado número de casos possíveis e da dependência entre as várias colunas, o que nos levou a pensar fora do âmbito da simulação e do contexto probabilístico.

Outro aspeto que tivemos de ter bastante em conta no nosso trabalho foi a otimização do código. Nós escrevemos código de forma pouco eficiente e implementamos partes que podiam ter sido retiradas, contudo foi feito de forma propositada, para dar uma experiência mais parecida ao que acontece no mundo real. Contudo, o nosso objetivo primário foi realizar as simulações e entender o que acontece e, mais tarde otimizar o código, tendo consciência de que nesse aspeto ainda nos resta algum trabalho.

Lista de Figuras

Lista de Tabelas

Lista de Códigos

Referências

[1] Gerald G Brown and Herbert C Rutemiller. Some probability problems concerning the game of bingo. *The Mathematics Teacher*, 66(5):403–406, 1973.

[2] Manuel Eberl. Fisher-yates shuffle. *Arch. Formal Proofs*, 2016:19, 2016.

[3] Dirk P Kroese, Tim Brereton, Thomas Taimre, and Zdravko I Botev. Why the monte carlo method is so important today. *Wiley Interdisciplinary Reviews: Computational Statistics*, 6(6):386–392, 2014.

[4] Sheldon M. Ross. *Introduction to Probability Models*. Academic Press, San Diego, CA, USA, tenth edition, 2009.