INSTITUTO LATINO-AMERICANO
DE CIÊNCIAS DA VIDA E DA NATUREZA
(ILACVN)

ENGENHARIA FÍSICA

Modelagem Computacional e Análise Probabilística em Jogos de Azar: Estimativa de Lucro e Explicitação de Vantagens do Cassino em *Roulette* e *Blackjack*

FERNANDO JOSÉ ZARDINELLO BATISTTI

Foz do Iguaçu
2024

INSTITUTO LATINO-AMERICANO DE CIÊNCIAS DA VIDA E DA NATUREZA (ILACVN)

ENGENHARIA FÍSICA

Modelagem Computacional e Análise Probabilística em Jogos de Azar: Estimativa de Lucro e Explicitação de Vantagens do Cassino em *Roulette* e *Blackjack*

FERNANDO JOSÉ ZARDINELLO BATISTTI

Trabalho de Conclusão de Curso apresentado ao Instituto Latino-Americano de Ciências da Vida e da Natureza da Universidade Federal da Integração Latino-Americana, como requisito parcial à obtenção do título de Bacharel em Engenharia Física.

Orientador: Prof. Dr. André Jacomel Torii

Foz do Iguaçu

2024

FERNANDO JOSÉ ZARDINELLO BATISTTI

Modelagem Computacional e Análise Probabilística em Jogos de Azar: Estimativa de Lucro e Explicitação de Vantagens do Cassino em *Roulette* e *Blackjack*

Trabalho de Conclusão de Curso apresentado ao Instituto Latino-Americano de Ciências da Vida e da Natureza da Universidade Federal da Integração Latino-Americana, como requisito parcial à obtenção do título de Bacharel em Engenharia Física.

BANCA EXAMINADORA

_____

Orientador: Prof. Dr. André Jacomel Torii
UNILA

_____

Prof. Dr. Luciano Calheiros Lapas
UNILA

_____

Prof. Dr. Marcelo Gonçalves Honnicke
UNILA

Foz do Iguaçu, _____ de _____ de _____

Dedico este trabalho aos meus irmãos de sangue Gustavo Zardinello Batistti e Marcelo Gustavo Zardinello Batistti, que me permitiram perceber a mortalidade em cada dia.

AGRADECIMENTOS

Em primeiro lugar, agradeço ao meu orientador, Prof. Dr. André Jacomel Torii, por proporcionar diversas oportunidades para meu desenvolvimento pessoal e acadêmico. Sua orientação constante, ensinamentos e, acima de tudo, a confiança depositada em mim foram fundamentais para a realização deste trabalho.

À minha família, que me proporcionou toda a estrutura necessária para que eu pudesse chegar até aqui. Sem o apoio incondicional deles, nada disso seria possível. Sou imensamente grato por todo o suporte e incentivo que esse sim, só eles sabem o quanto disseram as palavras “estude filho”.

Às minhas queridas amigas Paula Chaves e Patrícia Chaves, cujo apoio foi vital para minha trajetória, suas amizades e suporte emocional foram inestimáveis. Agradeço também à Sra. Noeli Tenutti Perego Chaves e ao Sr. Laudir Chaves, que me ensinaram valiosas lições sobre amor, humildade e o verdadeiro significado de família. Devo-lhes muito.

Aos meus amigos de longa data, Renato Souza e Yuri Gonçalvez Tomazolli, e aos novos amigos, Dylan Rafael Sanchez Salinas e Esdras Rebecchi Almeida, minha gratidão eterna, pois sem vocês eu faria parte da estatística desistente do curso, com toda certeza vocês me ensinaram muito, me ensinaram coisas que não se aprende nos livros, até porque pro livro é tudo trivial, não sei como que pode, meu deus do céu.

À Universidade Federal da Integração Latino-Americana (UNILA), que proporcionou o local, o ambiente e os profissionais necessários para o desenvolvimento deste trabalho. A todos os membros dessa instituição que tiveram papel incalculável na minha formação, meu sincero agradecimento.

Por fim, agradeço a todos que, de alguma forma, contribuíram para que este trabalho e este momento se tornassem possíveis. Aos amigos, familiares, colegas e todos aqueles que ofereceram apoio, incentivo e sabedoria ao longo dessa jornada, deixo aqui meu reconhecimento e gratidão. Cada gesto, por menor que tenha sido, foi fundamental para minha trajetória e a conclusão deste trabalho.

“Você tem que aprender as regras do jogo, e então,

jogar melhor do que todos.”

Albert Einstein

RESUMO

Este trabalho explora como os conceitos de probabilidade e estatística são aplicados aos jogos de azar, com foco nos jogos *Blackjack* e na *Roulette*, para garantir a lucratividade dos cassinos a longo prazo. O objetivo foi analisar e demonstrar como os cassinos utilizam estratégias probabilísticas e psicológicas para manter a vantagem sobre os jogadores, mesmo diante da aleatoriedade dos jogos. A metodologia envolveu o uso de simulações de Monte Carlo para modelar o comportamento do cassino e dos jogadores em diversos cenários. As simulações foram desenvolvidas com o intuito de calcular a probabilidade de vitória do jogador e o retorno ao jogador (RTP) em diferentes situações de jogo, além de estimar o número mínimo de jogos necessários para que o cassino obtenha lucro com confiabilidade. Propostas para equilibrar o jogo do *Blackjack* foram testadas, demonstrando que ajustes nos multiplicadores de pagamento podem revelar onde estão as vantagens do cassino se tratando do RTP. Também foi aplicada a técnica do caminhante aleatório para estimar o lucro do cassino a longo prazo, tanto dentro quanto fora da LGN, confirmando a inevitabilidade da vantagem do cassino em todos os cenários. Os resultados confirmam que, mesmo com estratégias variadas, o cassino mantém uma vantagem matemática em todos os jogos analisados. Este trabalho contribui com novas propostas que podem revelar as vantagens do cassino sob o RTP tanto no *Blackjack* quanto na *Roulette*, e também destaca a importância do uso de ferramentas probabilísticas e computacionais para prever os resultados em jogos de azar seja na probabilidade de vencer seja no RTP.

Palavras-chave: *Blackjack*; *Roulette*; Simulação Monte Carlo; Probabilidade; Confiabilidade.

RESUMEN

Este trabajo explora cómo los conceptos de probabilidad y estadística se aplican a los juegos de azar, enfocándose en los juegos de Blackjack y Ruleta, para asegurar la rentabilidad de los casinos a largo plazo. El objetivo fue analizar y demostrar cómo los casinos utilizan estrategias probabilísticas y psicológicas para mantener la ventaja sobre los jugadores, incluso frente a la aleatoriedad de los juegos. La metodología involucró el uso de simulaciones de Monte Carlo para modelar el comportamiento del casino y de los jugadores en diversos escenarios. Las simulaciones se desarrollaron con el objetivo de calcular la probabilidad de victoria del jugador y el Retorno al Jugador (RTP) en diferentes situaciones de juego, además de estimar el número mínimo de juegos necesarios para que el casino obtenga una ganancia de manera confiable. Se probaron propuestas para equilibrar el juego de Blackjack, demostrando que los ajustes en los multiplicadores de pago pueden revelar dónde están las ventajas del casino en términos de RTP. También se aplicó la técnica del paseo aleatorio para estimar la ganancia del casino a largo plazo, tanto dentro como fuera de la Ley de los Grandes Números, confirmando la inevitabilidad de la ventaja del casino en todos los escenarios. Los resultados confirman que, incluso con estrategias variadas, el casino mantiene una ventaja matemática en todos los juegos analizados. Este trabajo contribuye con nuevas propuestas que pueden revelar las ventajas del casino en términos de RTP tanto en el Blackjack como en la Ruleta, y también destaca la importancia del uso de herramientas probabilísticas y computacionales para prever los resultados en juegos de azar, ya sea en la probabilidad de ganar o en el RTP.

Palabras clave: Blackjack, Ruleta, Simulación Monte Carlo, Probabilidad, Fiabilidad.

ABSTRACT

This work explores how probability and statistics concepts are applied to gambling, focusing on Blackjack and Roulette games, to ensure the long-term profitability of casinos. The objective was to analyze and demonstrate how casinos use probabilistic and psychological strategies to maintain an advantage over players, even in the face of game randomness. The methodology involved using Monte Carlo simulations to model the behavior of the casino and players in various scenarios. The simulations were developed to calculate the player's probability of winning and the Return to Player (RTP) in different gaming situations, as well as to estimate the minimum number of games required for the casino to reliably make a profit. Proposals for balancing the Blackjack game were tested, demonstrating that adjustments in payout multipliers can reveal where the casino's advantages lie concerning RTP. The random walk technique was also applied to estimate the casino's long-term profit, both within and outside the Law of Large Numbers, confirming the inevitability of the casino's advantage in all scenarios. The results confirm that, even with varied strategies, the casino maintains a mathematical edge in all analyzed games. This work contributes new proposals that can reveal the casino's advantages regarding RTP in both Blackjack and Roulette and also highlights the importance of using probabilistic and computational tools to predict outcomes in games of chance, whether in terms of winning probability or RTP.

Key words: Blackjack, Roulette, Monte Carlo Simulation, Probability, Casino profit

LISTA DE GRÁFICOS

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

RTP	*Return To Player* (Retorno ao jogador)
BJ	*Blackjack* (Combinação de cartas que resultam 21 pontos)
LGN	Lei dos Grandes Números
MSMC	Método de Simulação Monte Carlo

SUMÁRIO

1 INTRODUÇÃO

A palavra jogo tem sua origem no termo latino *jocus*, que significa gracejo, brincadeira ou divertimento [1], e historicamente é reconhecida como uma atividade cuja natureza ou finalidade é a diversão e entretenimento [1], pouco depois que o *Homo erectus* ganhou ascendência, voltou sua atenção para as abstrações de ordem superior e também inventou um conceito que desde então foi variadamente visto como um vício, um crime, um negócio, um prazer, um tipo de magia, uma doença, uma loucura, uma fraqueza, uma forma de substituição sexual, uma expressão do instinto humano, ele inventou o jogo de azar [2]. Huzinga propõe que o jogo é uma atividade ou ocupação voluntária, exercida dentro de certos e determinados limites de tempo e de espaço, segundo regras livremente consentidas, mas absolutamente obrigatórias, dotado de um fim em si mesmo, acompanhado de um sentimento de tensão e de alegria e de uma consciência de ser diferente da “vida cotidiana” [3].

Arqueólogos que se debruçaram sobre sítios pré-históricos descobriram um grande número de ossos em forma de cubo, chamados de *astragalia*, que aparentemente eram usados em jogos há milhares de anos. Se nossos ancestrais da Idade da Pedra lançaram esses objetos para profecia ou diversão ou simplesmente para ganhar o machado de pedra do vizinho, eles deram início a um costume que sobreviveu à evolução e à revolução [2]. Embora praticamente todas as culturas tenham se envolvido em alguma forma de jogo de dados, séculos se passaram antes que se pensasse na "justiça" do lançamento de dados ou na probabilidade igual com que cada face cai ou deveria cair. A ligação entre a matemática e os jogos de azar permaneceu insuspeitada por muito tempo [2].

No mundo real, não há um "caminho fácil" para garantir um lucro financeiro nos jogos reconhecidos de azar ou habilidade; se houvesse, as regras do jogo logo seriam rapidamente alteradas ou haveria suspensão de tal jogo ou atividade [2]. No entanto, compreender a matemática por trás de cada jogo pode resultar em uma experiência altamente recompensadora. Afinal, há algo de satisfatório em racionalizar que prefere-se perder de forma inteligente do que ganhar sem entender o porquê.

1.1 Objetivo

1.1.1 Objetivo Geral

Analisar e demonstrar como os conceitos fundamentais de probabilidade e estatística são aplicados nos jogos de azar e unir tais demonstrações com conhecimentos da área da

psicologia e da neurociência, garantindo a lucratividade dos cassinos e influenciando o comportamento dos jogadores a longo prazo. Com base nisso, prever a rentabilidade dos jogos de *Roulette* e *Blackjack* por meio de simulação computacional, relacionando o programa ao problema do caminhante aleatório.

Dentro do jogo *Blackjack*, apresentar sob a principal estratégia comparadas com outras duas variantes, as probabilidades e o retorno ao jogador que é popularmente conhecido por RTP que significa "*Return To Player"*, ou seja, a porcentem teórica de retorno ao jogador. Encontrar uma proposta de multiplicadores do pagamento das apostas para que o RTP tenda à 100%, o que revelará a vantagem que o cassino determina em cada jogo.

1.1.2 Objetivos Específicos

- Explicar a aplicação da Lei dos Grandes Números (LGN) em jogos de azar, demonstrando sua função na estabilidade, previsibilidade e confiança dos resultados ao longo do tempo.
- Realizar uma análise detalhada dos conceitos de valor esperado, desvio padrão e confiança e limiar da LGN dentro do contexto dos jogos de azar, mostrando sua importância para a estratégia dos cassinos.
- Explorar a Teoria do Prospecto e a neurociência e suas influências no comportamento dos jogadores, especialmente na tomada de decisões sob risco e sob como um vício nasce.
- Desenvolver e aplicar um código baseado no Método de Simulação Monte Carlo (MSMC) para calcular as probabilidades dos principais eventos de interesse nos jogos de azar, expectativa de RTP e manipular os multiplicadores para estimar o RTP de 100% revelando quais multiplicadores estão favorecendo o cassino.
- Realizar uma análise analítica dos eventos de interesse, explicitando cálculos e comparando-os com os resultados obtidos através da simulação Monte Carlo.
- Estudar os jogos de *Roulette* e *Blackjack*, apresentando as probabilidades e estatísticas que sustentam as margens de lucro dos cassinos.

1.2 Hipótese

Parte-se da hipótese de que as estratégias dos cassinos, fundamentadas em conceitos probabilísticos e estatísticos, garantem um lucro consistente a longo prazo, mesmo diante da aleatoriedade dos jogos de azar e ainda por cima, muitas vezes contando com decisões irracionais por parte dos jogadores. Acredita-se que, ao aplicar a LGN [4] e analisar a

probabilidade de vitória dos jogadores através de simulações Monte Carlo, seja possível prever com precisão a expectativa de lucro dos cassinos nos jogos *Roulette* e *Blackjack*.

Além disso, supõe-se que o modelo de estratégia fixa do crupiê[1], quando comparado a diversas estratégias possíveis do jogador, confirma que, independentemente da estratégia adotada pelo jogador, a probabilidade de vitória permanecerá sempre inferior a 50%, o que juntamente com o RTP, precisa ser assegurado para que a LGN traga retorno e assim, assegurando a vantagem do cassino. A Teoria do Prospecto [1], ao descrever como os indivíduos tomam decisões sob risco, reforça essa hipótese ao demonstrar que os jogadores tendem a comportamentos "irracionais", aumentando ainda mais a eficácia das estratégias de cassino.

1.3 Organização de Trabalho

No Capítulo 2, apresenta-se a fundamentação teórica, que abrange um aprofundamento na definição de probabilidade e seus princípios, incluindo probabilidades racionais, cálculo de probabilidades, axiomas, e corolários que sustentam o desenvolvimento das simulações. São abordadas também as permutações e combinações e as diferentes distribuições de probabilidade, junto à expectativa matemática, para sustentar a análise estatística do estudo. Além disso, o capítulo inclui tópicos sobre variância, a LGN, conceitos de confiança e estimativa, que ajudam a interpretar os resultados de forma mais robusta. A simulação Monte Carlo é detalhada com explicações sobre sua metodologia, e a abordagem do Caminhante Aleatório é explorada como simulador de sobrevivência do jogador baseado no seu montante inicial e desempenho do mesmo dentro do jogo. A Teoria do Prospecto é incluída para examinar o comportamento em situações de risco e tomada de decisão. O capítulo termina com uma análise histórica e regras dos jogos de cassino, como *Roulette* e *Blackjack*, oferecendo um contexto prático para a aplicação das teorias discutidas.

No Capítulo 3 é descrita a implementação do código e as simulações computacionais em *Python*, com foco na simulação Monte Carlo. Nesse capítulo, é feito um detalhamento dos cálculos probabilísticos tanto para o jogo de *Roulette* quanto para o *Blackjack*, incluindo as diversas apostas e suas probabilidades de ocorrência. São discutidos também os cálculos das apostas laterais do *Blackjack* e a implementação de estratégias fixas para análise de vitória do jogador. A simulação Monte Carlo é desenvolvida para estimar tanto as probabilidades de vitória com estratégias específicas, quanto para o retorno ao jogador ou da mesma forma, o

[1] empregado que dirige o jogo nos cassinos, pagando e recolhendo o dinheiro das apostas.

lucro do cassino em um cenário de longo prazo, aplicando a LGN.

No Capítulo 4 são apresentados os resultados obtidos das simulações, destacando as probabilidades de sucesso associadas às diferentes estratégias em *Blackjack* e *Roulette*, tanto para apostas padrão quanto para apostas laterais. Também são mostradas comparações entre o uso de diferentes números de baralhos e uma análise do impacto dessas variáveis no Retorno ao Jogador (RTP). Este capítulo inclui propostas de ajuste para o equilíbrio de RTP em ambos os jogos e uma estimativa de lucro para o cassino com base nas probabilidades simuladas.

Por fim, no Capítulo 5, são apresentadas as conclusões acerca dos resultados e o impacto das estratégias de jogo analisadas, fornecendo um resumo das contribuições e observações sobre a eficácia das simulações Monte Carlo no estudo da probabilidade e desempenho do jogador em jogos de azar.

1.4 Prévia da Conclusão

Os objetivos propostos foram plenamente alcançados, e as hipóteses confirmadas, evidenciando que as estratégias utilizadas pelos cassinos, baseadas em conceitos probabilísticos, asseguram sua lucratividade a longo prazo. O MSMC corroborou os cálculos analíticos, comprovando a eficácia dessas estratégias e a vantagem matemática inevitável do cassino nos jogos estudados. As análises realizadas para o *Blackjack* foram eficazes em aproximar os resultados teóricos da realidade observada, tanto nas simulações computacionais quanto na prática. Concluiu-se que, embora os cassinos fundamentem suas operações em princípios probabilísticos, seus lucros são obtidos de forma determinística.

Além disso, foram propostas revisões dos multiplicadores de pagamento para os jogos de *Blackjack* e *Roulette*, detalhando especificamente como o cassino estabelece e assegura seu lucro em cada tipo de aposta. Com base em parâmetros unitários, foram determinadas estimativas de lucro dos cassinos de acordo com a LGN, confirmando a previsibilidade dos ganhos para o cassino no longo prazo.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo serão apresentados os fundamentos teóricos indispensáveis para a compreensão dos cálculos de probabilidade, rentabilidade e estatística aplicados ao contexto deste trabalho. A base conceitual será amplamente fundamentada em dois textos de referência: [1] *The Theory of Gambling and Statistical Logic* de Richard A. Epstein, que oferece uma análise profunda da lógica estatística aplicada aos jogos de azar, e [4] *Estatística Aplicada e Probabilidade para Engenheiros* de Douglas C. Montgomery e George C. Runger, que fornece uma abordagem detalhada e aplicada dos conceitos probabilísticos e estatísticos. Esses textos fornecerão o embasamento necessário para a análise e desenvolvimento das simulações e modelos apresentados nas seções seguintes.

2.1 Definição de Probabilidade, Probabilidade Racional

A palavra “probabilidade” origina-se do latim *probabilis*, que significa “semelhante à verdade”, o que, por si só, sugere uma certa imprecisão semântica. Para o contexto deste estudo, que se concentra na análise de fenômenos relacionados a jogos, adota-se uma definição de probabilidade mais operacional e objetiva, chamada de "probabilidade racional." Esta abordagem busca evitar tanto debates filosóficos quanto ambiguidades semânticas, focando-se em uma aplicação prática. A probabilidade racional é tida como a única descrição matemática precisa de probabilidade ou incerteza em contextos que permitem observação empírica repetida.

A probabilidade racional aplica-se especificamente a fenômenos de massa ou eventos recorrentes que possam ser avaliados ou previstos com base em experiências empíricas. Para que esta abordagem seja aplicável, é necessário dispor de uma sequência extensa e uniforme de observações no tempo ou espaço, ou poder correlacionar experiências passadas relevantes ao problema em análise.

Historicamente, três conceitos fundamentais emergiram na teoria da probabilidade: a teoria clássica, a teoria da frequência relativa, e a teoria lógica. A teoria clássica define probabilidade com base na ideia de eventos igualmente prováveis, sem definição exata. A teoria da frequência relativa se baseia em observação empírica e em um postulado matemático, e a teoria lógica define probabilidade como o grau de confirmação de uma hipótese frente a evidências.

A definição de teoria da probabilidade racional é mais consistente e completada pelo conceito de frequência relativa limitante. Se for realizado um experimento em que n tentativas do experimento produzem n_0 ocorrências de um evento específico, a razão n_0/n é denominada

frequência relativa do evento. Postula-se então a existência de um valor limite à medida que o número de tentativas aumenta indefinidamente. A probabilidade do evento específico é definida como

$$P = \lim_{n\to\infty} \frac{n_0}{n} \tag{2.1.1}$$

A teoria clássica de probabilidade considera os eventos mutuamente excludentes e casos exaustivos, definindo a probabilidade de um evento como a proporção entre o número de resultados favoráveis e o número total de resultados possíveis. Uma limitação dessa abordagem é sua completa dependência de uma análise a *priori* — um processo aplicável apenas a cenários menos complexos, em que todas as alternativas podem ser avaliadas precisamente; em situações mais amplas, muitas ocorrências não permitem atribuição de probabilidade de maneira a *priori*.

A teoria lógica oferece uma flexibilidade no exame de hipóteses e conceitos, mas seu uso tende a se restringir ao âmbito acadêmico, sendo geralmente pouco aplicável a problemas práticos de jogos. O conceito de probabilidade lógica, ou "grau de confirmação", fundamenta-se de forma analítica, ou seja, não se apoia em fatos observáveis, mas em um processo lógico que analisa os significados, sem necessidade de observação empírica.

Apesar dessas variações conceituais, há um consenso amplo sobre a estrutura matemática que define a teoria da probabilidade, e é essa estrutura — e não questões filosóficas ou nuances semânticas — que sustenta a lógica estatística e as estratégias de análise de jogos. No contexto dos fenômenos de jogo, a interpretação de probabilidade adotada neste estudo é elaborada para corresponder à realidade prática dos eventos, baseando-se em observações e experiências acumuladas empiricamente. Um exemplo disso é o caso de um dado: além de suas propriedades físicas mensuráveis, como massa, calor específico e resistência elétrica, ele apresenta uma probabilidade de que uma face específica, como o número "6", apareça voltada para cima. Aqui, a probabilidade é tratada de modo semelhante ao conceito de massa ou energia na física. Assim, a probabilidade racional relaciona-se diretamente às conexões empíricas observáveis entre as características físicas do objeto em estudo e as probabilidades atribuídas a esses fenômenos.

2.2 Cálculo da Probabilidade

A matemática, em sua essência, não possui significado intrínseco, sendo apenas uma série de proposições do tipo "se..., então...". De maneira similar à geometria euclidiana, basta estabelecer um conjunto de axiomas consistentes para que a teoria das probabilidades seja considerada um campo rigoroso dentro da matemática pura. Ao abordar a probabilidade sob

uma perspectiva geométrica, cada resultado possível de um experimento pode ser visualizado como a posição de um ponto em uma linha, e cada repetição do experimento se traduz na coordenada desse ponto em uma nova dimensão. Nesse contexto, a probabilidade é interpretada como uma medida — análoga à medição geométrica de volume. Com isso, os problemas probabilísticos são analisados como investigações geométricas de pontos em um espaço de múltiplas dimensões.

2.3 Axiomas e Corolários

Um evento aleatório é um experimento cujo resultado não é conhecido a *priori.* Pode-se dizer:

Axioma I: Para todos os eventos aleatório $\boldsymbol{A}$ corresponde à um número $P(\boldsymbol{A})$, referido como a probabilidade de $\boldsymbol{A}$, que satisfaz a desigualdade

$$0 \leq P(A) \leq 1 \tag{2.3.1}$$

Então a probabilidade é um número não negativo real no intervalo entre 0 e 1.

Agora considere um experimento cujo resultado é certo ou cujos resultados são indistinguíveis (jogar uma moeda de duas caras). Para caracterizar tal experimento, deixe $\boldsymbol{E}$ representar a coleção de todos os resultados possíveis, portanto,

Axioma II: A probabilidade do evento certo é a unidade. Ou seja, a frequência relativa do evento é

$$P(\boldsymbol{E}) = 1$$

Por fim, é necessário um axioma que caracterize a natureza de eventos mutuamente exclusivos (se uma moeda for lançada, o resultado Cara exclui o resultado Coroa, e vice-versa; portanto, Cara e Coroa são mutuamente exclusivos). Formula-se esse axioma da seguinte forma:

Axioma III: O teorema da probabilidade total. Se os eventos $\boldsymbol{A}_1, \boldsymbol{A}_2 \ldots, \boldsymbol{A}_n$ são mutuamente excludentes, a probabilidade da alternativa desses eventos é igual à soma de duas probabilidades individuais. Matematicamente,

$$P(\boldsymbol{A}_1 \cup \boldsymbol{A}_2 \ldots \cup \boldsymbol{A}_n) = P(\boldsymbol{A}_1) + \cdots + P(\boldsymbol{A}_n)$$

Onde $\boldsymbol{A}_1 \cup \boldsymbol{A}_2$ denota a união dos eventos $\boldsymbol{A}_1$ e $\boldsymbol{A}_2$. O *axioma III* expressa a propriedade aditiva da probabilidade. Pode ser estendido para incluir eventos não mutuamente excludentes. Se $\boldsymbol{A}_1$ e $\boldsymbol{A}_2$ são tais eventos, a probabilidade total ($\boldsymbol{A}_1$ e/ou $\boldsymbol{A}_2$ ocorre) é

$$P(\boldsymbol{A}_1 \cup \boldsymbol{A}_2) = P(\boldsymbol{A}_1) + P(\boldsymbol{A}_2) + P(\boldsymbol{A}_1 \cap \boldsymbol{A}_2)$$

Onde $P(\boldsymbol{A}_1 \cap \boldsymbol{A}_2)$ é definido como a probabilidade conjunta ou composta de que $\boldsymbol{A}_1$ e $\boldsymbol{A}_2$ ocorram.

Generalizando a probabilidade de ocorrência de P_1, de ao menos um evento entre n

eventos, $\boldsymbol{A}_1, \boldsymbol{A}_2, \ldots, \boldsymbol{A}_n$ é dada por

$$P_1 = \sum P(\boldsymbol{A}_i) - \sum P\left(\boldsymbol{A}_i \cap A_j\right) + \sum P\left(\boldsymbol{A}_i \cap A_j \cap A_k\right) - \cdots \tag{2.3.5}$$

E a Eq. 2.3.5 pode ser estendida a qualquer número de eventos independentes

$$P(\boldsymbol{A}_1 \cup \boldsymbol{A}_2 \cup \ldots \cup \boldsymbol{A}_n) = \sum_{i=1}^{n} P(\boldsymbol{A}_i) - \sum_{i,j=1}^{n} P\left(\boldsymbol{A}_i \boldsymbol{A}_j\right) + \sum_{i,j,k=1}^{n} P\left(\boldsymbol{A}_i \boldsymbol{A}_j \boldsymbol{A}_k\right) - \cdots$$

Desses axiomas decorrem vários corolários lógicos.

Corolário I: A soma das probabilidades de qualquer evento $\boldsymbol{A}$ e seu complemento $\overline{\boldsymbol{A}}$ é a unidade. Isto é,

$$P(\boldsymbol{A}) + P(\overline{\boldsymbol{A}}) = 1$$

O complemento de um evento é, obviamente, a não ocorrência desse evento.

Caracterizando um evento impossível O, pode-se afirmar o seguinte

Corolário II: A probabilidade de um evento impossível é zero, ou

$$P(O) = 0$$

É importante notar que a recíproca do corolário II não é verdadeira. Dado que a probabilidade de algum evento é igual a zero, não se segue que o evento seja impossível ou ao menos não é provável. (há uma probabilidade zero de selecionar aleatoriamente um ponto pré-especificado numa reta.)

Outro conceito importante é o de "probabilidade condicional". A expressão $P(\boldsymbol{A}_2|\boldsymbol{A}_1)$ se refere a probabilidade de um evento $\boldsymbol{A}_2$, dado a ocorrência de um evento $\boldsymbol{A}_1$. Pode-se escrever para $P(\boldsymbol{A}_1) > 0$,

$$P(\boldsymbol{A}_2|\boldsymbol{A}_1) = \frac{P(\boldsymbol{A}_1 \cap \boldsymbol{A}_2)}{P(\boldsymbol{A}_1)} \tag{2.3.9}$$

Pela indução da eq. 2.3.9, pode-se obter o teorema da multiplicação (ou a lei geral da probabilidade composta)

Corolário III: A probabilidade do produto de n eventos $\boldsymbol{A}_1, \boldsymbol{A}_2 \ldots, \boldsymbol{A}_n$ é igual à probabilidade do primeiro evento vezes a probabilidade condicional do segundo evento, dada a ocorrência do primeiro evento, vezes a ocorrência condicional do terceiro evento dada a ocorrência conjunta dos dois primeiros eventos, etc. Na forma matemática tem-se,

$$P(\boldsymbol{A}_1, \boldsymbol{A}_2, \ldots, \boldsymbol{A}_n) = P(\boldsymbol{A}_1)P(\boldsymbol{A}_2|\boldsymbol{A}_1)P(\boldsymbol{A}_3|\boldsymbol{A}_1\boldsymbol{A}_2) \ldots P(\boldsymbol{A}_n|\boldsymbol{A}_1\boldsymbol{A}_2 \ldots \boldsymbol{A}_{n-1})$$

Pode ser demonstrado que a probabilidade condicional satisfaz os axiomas I, II e III. Um caso especial do corolário III ocorre quando os eventos $A_1, A_2, \ldots, A_n$ são independentes; então a lei da probabilidade composta se reduz para

$$P(\boldsymbol{A}_1, \boldsymbol{A}_2, \ldots, \boldsymbol{A}_n) = P(\boldsymbol{A}_1)P(\boldsymbol{A}_2) \ldots P(\boldsymbol{A}_{n-1})P(\boldsymbol{A}_n)$$

Assim, a condição necessária e suficiente para que n eventos sejam independentes é que a probabilidade de cada combinação n vezes que pode ser formada a partir de n eventos ou os complementos sejam fatoráveis no produto das probabilidades dos n componentes distintos.

Finalmente, o conceito de probabilidade completa (ou absoluta) é expresso por

Corolário IV: Se os eventos aleatórios $\boldsymbol{A}_1, \boldsymbol{A}_2 \ldots, \boldsymbol{A}_n$ são exclusivos aos pares, então a probabilidade $P(\boldsymbol{X})$ de qualquer evento aleatório $\boldsymbol{X}$ ocorre junto com um dos eventos $\boldsymbol{A}_i$ e é dado por

$$P(\boldsymbol{X}) = P(\boldsymbol{A}_1)P(\boldsymbol{X}|\boldsymbol{A}_1) + P(\boldsymbol{A}_2)P(\boldsymbol{X}|\boldsymbol{A}_2) + \cdots + P(\boldsymbol{A}_n)P(\boldsymbol{X}|\boldsymbol{A}_n) \tag{2.3.12}$$

$$Se\ P(\boldsymbol{A}_i) > 0\ para\ i = 1,2, \ldots, n$$

A eq. 2.3.12 é conhecida como teorema de Bayes, ou a fórmula para a *posteriori* probabilidade. Deve-se notar que nenhuma suposição quanto às probabilidades dos respectivos $\boldsymbol{A}_i$ está implícita (um erro comum é interpretar o teorema de Bayes como significado que todos os $\boldsymbol{A}_i$ são iguais).

Juntos, os axiomas e corolários anteriores constituem a base da teoria da probabilidade como um ramo distinto da matemática.

2.4 Permutações e Combinações

Os fenômenos dos jogos de azar requerem frequentemente a extensão direta dos axiomas e corolários da teoria das probabilidades para o domínio da análise permutacional e combinatória. Uma *permutação* de vários elementos é qualquer arranjo desses elementos em uma ordem definida. Uma *combinação* é uma seleção de vários elementos de uma população considerada independentemente da sua ordem. Provas matemáticas rigorosas dos teoremas de permutações e combinações estão disponíveis em muitos textos sobre teoria das probabilidades. Pela notação convencional, o número de permutações de n objetos distintos (ou elementos ou coisas) considerados r em um tempo sem repetição é representado por P_n^r. Similarmente, C_n^r representa os números de combinações de n objetos distintos considerados r de cada vez, independentemente de sua ordem. Para derivar a fórmula para P_n^r, considere que há r espaços para preencher e n objetos para escolher. O primeiro espaço pode ser preenchido com qualquer um dos n objetos (ou seja, de n maneiras). Subsequentemente, o segundo espaço pode ser preenchido com qualquer uma de $(n-1)$ objetos $(n-1)$ maneiras, o terceiro espaço de $(n-2)$ maneiras, etc. e o r-ésimo espaço pode ser preenchido com $[n-(r-1)]$ maneiras. Por isso

$$P_n^r = n(n-1)(n-2)\ldots(n-r+1) \tag{2.4.1}$$

Para o caso onde $r = n$, obtém-se

$$P_n^r = n(n-1)(n-2)\dots(1) \equiv n! \tag{2.4.2}$$

E combinando as equações 2.4.1 e 2.4.2, este último pode ser reescrito na forma

$$P_n^r = \frac{P_n^n}{P_{n-r}^{n-r}} = \frac{n!}{(n-r)!}$$

Também é possível, escrever a relação de recorrência

$$P_n^r = P_{n-1}^r + rP_{n-1}^{r-1}$$

Considerações semelhantes são válidas quando há uma preocupação com permutações de n objetos que não são todos distintos. Especificamente, deixe os n objetos serem repartidos em m tipos de elementos com n_1 elementos do primeiro tipo, n_2 elementos do segundo tipo, etc., e $n = n_1 + n_2 + \cdots + n_m$ então o número de permutações P_n objetos considerados todos juntos é dado por

$$P_n = \frac{n!}{n_1!\, n_2!\, n_3! \dots n_m!}$$

Desta forma, o número de permutações de quatro cartas selecionadas em um baralho de 52 cartas é

$$P_{52}^4 = \frac{52!}{(52-4)!} = 52 \cdot 51 \cdot 49 \cdot 48 = 6.497.400$$

Note que a ordem aqui importa, ou seja, o $A, 2, 3, 4$ de espadas, difere de $2, A, 3, 4$ de espadas. Se é desejado saber o número de permutações de um baralho distinguindo seus valores, mas não os naipes (i.e., 13 tipos distintos de objetos cada um com quatro elementos indistintos) da equação

$$P_n = \frac{n!}{n_1!\, n_2!\, n_3! \dots n_m!}$$

Tem-se

$$P_{52} = \frac{52!}{(4!)^{13}} = 9{,}203 \cdot 10^{49}$$

Até esse ponto é permitido que todos os elementos de uma população fossem permutados em todas as posições possíveis. Contudo, pode-se desejar enumerar apenas aquelas permutações que são restringidas por conjuntos prescritos de restrições nas posições dos elementos permutados. A elegância e utilidade destas análises está conformada na correspondência entre os casos possíveis e tornar a análise dos problemas inicialmente complicados em simples. Passando para r combinações de n objetos distintos, observa-se que cada combinação de r objetos distintos podem ser organizados em $r!$ maneiras – isto é $r!$ permutações. Portanto $r!$ Permutações de cada um dos C_n^r combinações produz $r!\, C_n^r$

permutações.

$$r!\, C_n^r = P_n^r = \frac{n!}{(n-r)!}$$

Dividindo por $r!$, obtém-se a expressão para o número de combinações de n objetos tirados r de cada vez

$$C_n^r \equiv \binom{n}{r} \equiv \frac{n!}{r!\,(n-r)!} \tag{2.4.10}$$

Onde $\binom{n}{r}$ é o símbolo para coeficientes binomiais, é o $(r+1)$-ésimo coeficiente da expansão $(a+b)^n$. Para valores r menor que zero e maiores que n, se define C_n^r igual a zero. Esta equação acima também pode ser estendida para incluir valores negativos de n e r; entretanto, em nosso contexto não será encontrado tais valores.) Também é evidente que o número de combinações de n objetos tomados de cada vez é idêntico ao número de combinações dos n objetos tomados $(n-r)$ de uma vez, isto é

$$\binom{n}{r} = \binom{n}{n-r}$$

Ou

$$C_n^r = C_n^{n-r}$$

Pode-se derivar essa equivalência da seguinte forma

$$C_n^r = C_{n-1}^{r-1} + C_{n-1}^r \tag{2.4.13}$$

A eq. 2.4.13 é conhecida como regra de Pascal. Por iteração, segue que

$$C_n^r = C_{n-1}^{r-1} + C_{n-2}^{r-1} + \cdots + C_{r-1}^{r-1} = C_{n-1}^r + C_{n-2}^{r-1} + \cdots + C_{n-1-r}^0 \tag{2.4.14}$$

Como exemplo, poderia haver a seguinte pergunta: “Quantas mãos de Bridge [2]de 13 cartas são possíveis de um baralho de 52 cartas?”, e desta equação se obtém

$$C_{52}^{13} = \frac{52!}{13!\,(52-13)!} \equiv \binom{52}{13} = 635.013.559.597 \tag{2.4.15}$$

Note que a ordem que os jogadores recebem as cartas, não distingue em nada, mas caso diferisse o número resultante de permutações seria maior por um fator de 13!

2.5 Distribuição de Probabilidade

Com uma população não homogênea (n objetos consistindo em m tipos de elementos com n_1 elementos do primeiro tipo, n_2 elementos do segundo tipo, etc.) o número de permutações dos n objetos considerados todos juntos são dados pela equação

[2] Bridge ou brídege é um jogo de cartas, que usa a mecânicas de leilão e de vazas, jogado por dois pares de jogadores e com as 52 cartas de um baralho

$$P_n = \frac{n!}{n_1!\, n_2!\, n_3! \ldots n_m!} \tag{2.5.1}$$

Também pode-se determinar o número de combinações possíveis através da seleção de um grupo de r elementos dos n objetos. Portanto, é viável perguntar à probabilidade $P_{k_1,k_2,\ldots,k_m}$ que se o grupo de r elementos for selecionado aleatoriamente (sem reposição e sem ordenação) o grupo irá conter exatamente $k_1 \leq n_1$ elementos do primeiro tipo, $k_2 \leq n_2$ elementos do segundo tipo, etc., e $k_m \leq n_m$ elementos do m-ésimo tipo. Especificamente pode se mostrar que

$$P_{k_1,k_2,\ldots,k_m} = \frac{\binom{n_1}{k_1}\binom{n_2}{k_2}\binom{n_3}{k_3}\ldots\binom{n_m}{k_m}}{\binom{n}{r}} \tag{2.5.2}$$

Onde,

$$n = n_1 + n_2 + \cdots + n_m$$
$$k = k_1 + k_2 + \cdots + k_m$$

Esta equação representa a *distribuição hipergeométrica* generalizada, a distribuição de probabilidade para uma amostragem sem reposição de uma população finita. Deve-se notar que, no limite (para populações arbitrariamente grandes), a amostragem com ou sem reposição levam aos mesmos resultados. Como ilustração da distribuição hipergeométrica, calcula-se a probabilidade de que uma mão de Bridge de 13 cartas consista em exatamente 5 espadas, 4 copas, 3 ouros e 1 paus. De acordo com a qe. 2.5.2 tem-se

$$P_{5,4,3,1} = \frac{\binom{13}{5}\binom{13}{4}\binom{13}{3}\binom{13}{1}}{\binom{52}{13}} = 0{,}0054 \tag{2.5.4}$$

Já que a ordem das cartas na mão de Bridge é irrelevante.

Outra distribuição de consequência em aplicações práticas da teoria da probabilidade é a *distribuição binomial*. Se um evento tem dois resultados alternativos $\boldsymbol{A}_1$ e $\boldsymbol{A}_2$ então $P(\boldsymbol{A}_1) + P(\boldsymbol{A}_2) = 1$, e a probabilidade de ocorrência para uma tentativa individual é constante, o número de ocorrências r do resultado $\boldsymbol{A}_1$ obtida em n tentativas independentes é uma *variável aleatória* discreta, que pode assumir qualquer um dos valores possíveis $0, 1, 2, \ldots, n$. Uma *variável aleatória* é simplesmente uma quantidade variável cujos valores dependem do acaso e para a qual existe uma *função de distribuição*. Neste caso, o número r é uma *variável binomial*, e sua distribuição $P(r)$ define a distribuição binomial. Definindo $p = P(\boldsymbol{A}_1)$ e $q = P(A_2) = 1 - p$ pode-se facilmente derivar a expressão

$$P(r) = \binom{n}{r} p^r q^{n-r}, \quad r = 0, 1, 2, \dots, n \tag{2.4.5}$$

A equação 2.4.5 também é chamada de distribuição de Bernoulli, já que foi derivada pela primeira vez por Jacob Bernoulli em *Ars. Conjectandi*.

Pode-se generalizar prontamente a partir de dois resultados possíveis de um evento para o caso de m resultados mutuamente exclusivos $\boldsymbol{A}_1, \boldsymbol{A}_2, \boldsymbol{A}_3, \dots, \boldsymbol{A}_m$ cada $\boldsymbol{A}_i$ ocorrência com probabilidade $p_i = P(\boldsymbol{A}_i) \geq 0$ para cada tentativa individual. A probabilidade de obter r_1 ocorrências de $\boldsymbol{A}_1$, r_2 ocorrências de $\boldsymbol{A}_2$, etc. com n tentativas independentes é conhecida como *distribuição multinomial* e é determinada de maneira semelhantes à 2.4.5

$$P(r_1, r_2, \dots, r_m) = \frac{n!}{r_1!\, r_2! \dots r_m!} p_1^{r_1} p_2^{r_2} \dots p_m^{r_m} \tag{2.4.6}$$

Onde

$$p_1 + p_2 + \cdots + p_m = 1$$

E

$$r_1 + r_2 + \cdots + r_m = n$$

Esta equação representa a *distribuição composta* ou *conjunta* do número de resultados para cada $\boldsymbol{A}_i$.

Quando um evento casual pode ocorrer com probabilidade constate p em qualquer tentativa, então o número de tentativas r necessárias para sua primeira ocorrência é uma variável aleatória discreta que pode assumir qualquer um dos valores $1, 2, 3, \dots, \infty$. A distribuição desta variável é chamada de distribuição geométrica. Se a primeira ocorrência do evento for r-ésima tentativa, as primeiras $(r-1)$ tentativas devem abranger não ocorrências do evento (cada um com probabilidade $q = 1 - p$). A probabilidade composta de $(r-1)$ não ocorrências é q^{r-1}; portanto, $(r-1)$ não ocorrências seguidas pelo evento tem probabilidade $q^{r-1}p$, consequentemente, a distribuição geométrica é definida por

$$p(r) = pq^{r-1}, \quad r = 1, 2, 3, \dots \tag{2.4.9}$$

Pela implicação dessa distribuição, o *valor esperado* de tentativas é igual ao inverso da probabilidade constante p.

2.6 Expectativa Matemática

Um parâmetro de valor fundamental na avaliação de jogos de azar é a *expectativa matemática*. Sua definição é direta: Se uma variável aleatória X pode assumir n valores $x_1, x_2, \dots, x_n$ com a respectiva probabilidade $p_1, p_2, \dots, p_n$ a expectativa matemática de X é $E(X)$ e é expressa por

$$E(X) = x_1p_1 + x_2p_2 + \cdots + x_np_n = \sum_{i=1}^{n} x_ip_i \tag{2.5.1}$$

A expectativa matemática para o valor que um dado honesto apresentará em sua face voltada para cima é

$$E(X) = \frac{1}{6}(1) + \frac{1}{6}(2) + \frac{1}{6}(3) + \frac{1}{6}(4) + \frac{1}{6}(5) + \frac{1}{6}(6) = 3{,}5 \tag{2.5.2}$$

Os valores possíveis de X, o número mostrado, são $1, 2, 3, 4, 5$ e 6, cada um com probabilidade $1/6$. Agora para a soma de dois dados honestos, X são todos números inteiros entre 2 e 12 com as probabilidades variadas

$$\begin{aligned} E(X) = {} & \frac{1}{36}(2) + \frac{2}{36}(3) + \frac{3}{36}(4) + \frac{4}{36}(5) + \frac{5}{36}(6) + \frac{6}{36}(7) \\ & + \frac{5}{36}(8) + \frac{4}{36}(9) + \frac{3}{36}(10) + \frac{2}{36}(11) + \frac{1}{36}(12) \\ & = 7 \end{aligned} \tag{2.5.3}$$

Um teorema útil afirma que a expectativa matemática da soma de várias variáveis aleatórias $X_1, X_2, \ldots, X_n$ é igual ao produto da expectativa matemática delas, isto é

$$E(X_1 + X_2 + \cdots + X_n) = E(X_1) + E(X_2) + \cdots + E(X_n) \tag{2.5.4}$$

Ou

$$E\left[\sum_{i=1}^{n} X_i\right] = \sum_{i=1}^{n} E[X_i]$$

Assim a expectativa do total mostrado na soma de n dados honestos é $3{,}5 \cdot n$.

Da mesma forma, pode-se provar o teorema de que a expectativa matemática do produto de diversas variáveis independentes $X_1, X_2, \ldots, X_n$ é igual ao produto de duas expectativas, isto é

$$E(X_1X_2 \ldots X_n) = E(X_1)E(X_2) \ldots E(X_n)$$

Ou

$$E\left[\prod_{i=1}^{n} X_i\right] = \prod_{i=1}^{n} E[X_i]$$

Também pode ser mostrado que

$$E(aX + b) = aE(X) + b \tag{2.5.8}$$

Onde $a, b \in \mathbb{R}$.

A descrição condensada da teoria da probabilidade aqui apresentada não pode,

evidentemente, incluir todas as ferramentas necessárias para a solução dos problemas do jogo. Contudo, a maior parte da informação necessária foi abordada brevemente e tem valor propedêutico.

2.7 Estatística

Estatística e teoria da probabilidade não podem sempre ser divididas em áreas completamente separadas. A estatística pode ser vista como uma extensão natural da teoria da probabilidade, pois constrói sobre suas bases e expande seu campo de aplicação. A teoria da probabilidade permite prever a composição provável de uma amostra com base na composição da população original; a estatística, por outro lado, inverte esse raciocínio ao estimar a composição da população a partir da análise de uma amostra bem escolhida. Muitas vezes, no entanto, o propósito de um estudo estatístico não é apenas descritivo. A descrição é usada para comparar diferentes conjuntos de dados por meio de suas características particulares ou para formular estimativas sobre as propriedades que poderiam ser observadas em outros conjuntos de dados semelhantes. Em qualquer caso, fica claro que a descrição é apenas um primeiro passo, enquanto a análise e interpretação constituem o objetivo principal.

Assim como na teoria da probabilidade, o propósito final de um estudo estatístico é gerar um conjunto de conclusões. Não se antecipa o evento individual, mas considera-se o conjunto das possíveis ocorrências e calcula-se a frequência dos eventos particulares. Além disso, tal qual a teoria da probabilidade, a estatística é uma criação humana e não uma propriedade intrínseca da natureza. Seu significado repousa, em última análise, em dados empíricos, sustentados por uma base axiomática rigorosa. Acreditar que a estatística possui uma capacidade inata para descrever leis universais pode levar a conclusões equivocadas e interpretativas engenhosas, porém enganosas.

2.8 Média e Variância

A análise dos jogos de azar convencionais envolve apenas os aspectos mais elementares da teoria estatística, principalmente aqueles relacionados aos conceitos de variância, estimativa, teste de hipóteses e limites de confiança. Nosso tratamento do assunto é, portanto, correspondentemente limitado.

A expectativa matemática de uma variável aleatória X é conhecida como *média* ou *valor esperado* de X. Geralmente é representado com o símbolo μ, isto é $\mu = E(X)$. Portanto,

$E(x-\mu)=0$. Considerando uma constante c em vez da média μ, o valor esperado de $X-c$ (ou seja, $E(X-c)$) é chamado de *primeiro momento* de X tomado em torno de c. A média (ou centro de massa da função de probabilidade) representa o resultado médio de longo prazo para um experimento realizado um número arbitrariamente grande de vezes. Esse tipo de média refere-se à média aritmética de uma distribuição, definida conforme a equação de $E(x)$ e não deve ser confundida com a $moda$ (valor da distribuição que possui a maior frequência e, portanto, o valor mais provável da distribuição), a *média ponderada* (em que cada valor da variável aleatória X é multiplicado por um coeficiente de ponderação antes do processo de média aritmética), a *mediana* (a soma das frequências de ocorrência dos valores de X acima e abaixo da mediana são iguais; para distribuições simétricas, a média e a mediana são idênticas), ou a *média geométrica* (a n-ésima raiz positiva do produto de n variáveis aleatórias), entre outros.

Em adição a média, outro parâmetro é necessário para descrever a distribuição dos valores: uma medida de dispersão ou variabilidade comparando vários resultados do experimento. A medida de dispersão mais conveniente usada é a variância. Seja X uma variável aleatória assumindo qualquer um dos m valores x_i $(i=1,2,\dots,m)$ com probabilidades correspondentes $p(x_i)$ e com média $\mu=E(X)$. A variância $Var(X)=$, então é definida por

$$Var(X)=E[(X-\mu)^2]=\sum_{i=1}^{m}(x_i-\mu)^2p(x_i)$$

Observe que as unidades de $Var(X)$ são quadrados das unidades de X. Portanto, para recuperar as unidades originais, o *desvio padrão* de X, $\sigma(X)$, é definido como

$$\sigma(X)=\sqrt{Var(X)}=\sqrt{E[(X-\mu)^2]}$$

Um teorema inestimável formula a variância de X como a média do quadrado de X menos o quadrado da média de X, isto é

$$\sigma^2(X)=E(X^2)-[E(X)]^2 \qquad (2.7.3)$$

Ilustrativamente, o valor médio da face superior de um dado, conforme calculado anteriormente é igual a $7/2$. A média deste valor ao quadrado é

$$E(X^2)=\frac{1}{6}(1)^2+\frac{1}{6}(2)^2+\frac{1}{6}(3)^2+\frac{1}{6}(4)^2+\frac{1}{6}(5)^2+\frac{1}{6}(6)^2=\frac{91}{6}$$

Portanto, a variância para o número mostrado no dado é

$$\sigma^2(X)=\frac{91}{6}-\left(\frac{7}{2}\right)^2=\frac{35}{2}\Rightarrow\sigma(X)=1{,}71$$

Para dois dados, $E(X)=7$ e $E(X^2)=329/6$. Então a variância da soma do número

dos dois dados é

$$\sigma^2(X) = \frac{329}{6} - (7)^2 = \frac{35}{6} \Rightarrow \sigma(X) = 2{,}415$$

A exibição total esperada em dois dados é ocasionalmente declarada como $7 \pm 2{,}415$.

Pode-se calcular prontamente a média e a variância para cada distribuição de probabilidade bem conhecida. A distribuição hipergeométrica com apenas dois tipos de elementos simplifica para

$$P_k = \frac{\binom{n_1}{k}\binom{n-n_1}{r-k}}{\binom{n}{r}}, para\ k \leq r\ ou\ n_1 \tag{2.7.7}$$

Considere um baralho de n cartas, n_1 vermelhas e $n - n_1$ pretas. Então, se r cartas forem sorteadas sem reposição, a probabilidade de o número X de cartas vermelhas serem sorteadas é exatamente k que é dado pela equação acima, o valor esperado de X é expresso por

$$\mu = E(X) = \sum_{k=0}^{r} \frac{k\binom{n_1}{k}\binom{n-n_1}{n-k}}{\binom{n}{r}} \tag{2.7.8}$$

E o valor esperado de X^2 é

$$E(X^2) = \frac{n_1(n_1-1)r(r-1)}{n(n-1)} + \frac{n_1 r}{n}$$

Agora a variância é dada por

$$\sigma^2 = \frac{n_1 r(n-n_1)(n-r)}{n^2(n-1)} \tag{2.7.10}$$

Para a distribuição binomial, o valor esperado é

$$\mu = E(X) = np$$

Perceba que p e q são as probabilidades dos dois resultados possíveis resultados em cada n ensaio. E o quadrado do valor esperado (X^2) é

$$E(X^2) = np + n(n-1)p^2 \tag{2.7.12}$$

Portanto, a variância associada com a distribuição binomial é

$$\sigma^2 = np + n(n-1)p^2 - (np)^2 = np(1-p) = npq \tag{2.7.13}$$

2.9 A Lei Dos Grandes Números

Outro teorema altamente significante pode ser deduzido da definição de variância separando os valores da variável aleatória X naquelas que estão dentro do intervalo $\mu - k\sigma$ até $\mu + k\sigma$ e as que não estão. A soma das probabilidades atribuídas aos valores de X fora do intervalo $\mu \pm k\sigma$ é igual a probabilidade de que X seja maior que $k\sigma$ da média μ e seja menor

ou igual e, portanto

$$\frac{1}{k^2} \geq P(|X - \mu| > k\sigma) \tag{2.8.1}$$

Esta expressão é conhecida como Teorema de Tchebychev, afirma que não mais do que a fração da probabilidade total de uma variável aleatória se desvia do valor esperado por mais de k desvios padrão.

Uma aplicação notável da desigualdade de Tchebychev está na determinação do ponto de *convergência estocástica*, ou seja, a convergência de uma probabilidade amostral para seu valor esperado. Se, na equação acima, for substituído a variável aleatória X pela probabilidade de amostra p' (a razão entre o número de ocorrências de um evento e o número de tentativas) e a média μ pela probabilidade de sucesso em uma única tentativa p, a desigualdade de Tchebychev se torna

$$P\left[|p' - p| > k\sqrt{\frac{pq}{n}}\right] \leq \frac{1}{k^2} \tag{2.8.2}$$

Onde, $\sigma = \sqrt{\frac{pq}{n}}$. Especificando o valor de k como $k = \frac{\epsilon}{\sqrt{\frac{pq}{n}}}$, onde ϵ é alguma fração maior que zero, e esta equação logo acima assume a forma

$$P[|p' - p| > \epsilon] \leq \frac{pq}{n\epsilon^2} \tag{2.8.3}$$

que é a *lei dos grandes números*. Ela declara que não importa quão pequeno seja o ϵ especificado, a probabilidade P de que a probabilidade da amostra seja diferente da probabilidade de sucesso em um único teste em mais de ϵ pode se tornar arbitrariamente pequena aumentando suficientemente o número de testes n. Assim, para uma moeda imparcial, a probabilidade de a proporção de cara (ou coroa) em relação ao número total de tentativas diferir de n por mais do que uma quantidade específica se aproxima de zero como um limite. Convencionalmente, expressa-se esse fato com a afirmação de que a probabilidade da amostra converge estocasticamente para $1/2$.

A LGN tem sido frequentemente (e erroneamente) citada como a garantia de um eventual equilíbrio entre a cara e coroa. Na verdade, na forma coloquial, a lei proclama que a diferença entre o número de caras e o número de coroas lançados pode aumentar indefinidamente com um número crescente de tentativas, embora em proporções decrescentes. Seu princípio operacional é “inundação” em vez de “compensação”.

2.10 Confiança

Além de uma medida da dispersão ou variabilidade de um experimento repetido, é desejável expressar o grau de confiança que se tem de que o parâmetro pertinente ou o resultado experimental específico estará dentro de certos limites. Considere que a variável aleatória X possua uma distribuição conhecida e então considere uma amostra de tamanho $r(x_1, x_2, \dots, x_n)$ com a qual será estimado algum parâmetro θ. Então, se θ_1 e θ_2 forem duas estimativas estatísticas de θ, a probabilidade ξ de que θ esteja dentro do intervalo de θ_1 e θ_2 é chamada de *nível de confiança* - ou seja,

$$P(\theta_1 \leq \theta \leq \theta_2) = \xi \tag{2.9.1}$$

O parâmetro θ, deve-se notar, não é uma variável aleatória. Ele representa um número finito, embora desconhecido. Entretanto, θ_1 e θ_2 são variáveis aleatórias, uma vez que seus valores dependem de amostras aleatórias.

Como um exemplo, considere uma variável aleatória X que segue uma distribuição normal com o desvio padrão σ conhecido e valor esperado μ desconhecido. Pode-se perguntar o tamanho dos valores de μ que confere 95% nível de confiança de que μ está dentro deste intervalo. Pela definição de distribuição normal, a probabilidade de que μ esteja entre

$$\theta_1 = \bar{x} - \frac{y\sigma}{\sqrt{r}} \tag{2.9.2}$$

e

$$\theta_2 = \bar{x} + \frac{y\sigma}{\sqrt{r}} \tag{2.9.2}$$

é dado por

$$P\left(\bar{x} - \frac{y\sigma}{\sqrt{r}} \leq \mu \leq \bar{x} + \frac{y\sigma}{\sqrt{r}}\right) = \frac{1}{\sqrt{2\pi}} \int_{-y}^{y} \exp\left[-\frac{x^2}{2}\right] dx \tag{2.9.3}$$

Nas tabelas da integral de probabilidade normal verifica-se que esta probabilidade é igual a 95% para $y = 1{,}96$. Portanto, para uma estimativa $\bar{x}$ de μ está incluído no intervalo $\bar{x} \pm \frac{1{,}96\sigma}{\sqrt{r}}$.

Postula-se um jogador de Bridge que, tendo recebido dez mãos consecutivas de Bridge, nove das quais não contém nenhum Ás, reclama que a situação é atribuível a um embaralhamento deficiente. Que nível de confiança pode ser atribuído a esta afirmação? A probabilidade de uma mão de 13 cartas selecionadas aleatoriamente conter pelo menos um Ás é

$$P(1\ Ás) = 1 - \frac{\binom{48}{13}}{\binom{52}{13}} = 0{,}696$$

A distribuição binomial fornece a probabilidade de que, de dez mãos, apenas uma contenha pelo menos um Ás

$$P(1\ Ás\ em\ 10\ mãos) = \binom{10}{1}(0{,}696)^1(1-0{,}696)^9 = 0{,}00015$$

Portanto com um nível de confiança de 99,985%, o jogador de Bridge pode justamente denunciar a falta de aleatoriedade e sustentar que, como consequência, a hipótese nula $P(1\ Ás) = 0{,}696$ não é válida neste jogo.

2.11 Estimativa

No campo da economia, um aspecto da estatística matemática amplamente aplicada é o da estimativa a partir de dados estocásticos. A inferência estatística é um modelo de estudar as características da população bom base em amostras de informações observadas. Por exemplo, pode-se desconhecer a probabilidade p de que o lançamento de uma determinada moeda resulta em cara; em vez disso, poder-se-ia saber em 100 tentativas, 55 foram Caras obtidas, e deseja-se obter, a partir desse resultado, uma estimativa de p. Em geral o parâmetro a ser estimado pode ser uma probabilidade, uma média, uma variância etc.

Existem várias estimativas "boas" de parâmetros desconhecidos oferecidas pela disciplina de estatística. Um procedimento comum é o método dos momentos. Para estimar k parâmetros desconhecidos, este procedimento determina que os primeiros k momentos amostrais sejam computados e igualados aos primeiros k momentos da distribuição. Como os k momentos da distribuição podem ser expressos em termos dos k parâmetros, pode-se determinar esses parâmetros como funções dos momentos amostrais, que são eles próprios funções dos valores amostrais.

Certamente, a medida mais amplamente utilizada é a estimativa *máxima verossimilhança* – obtida expressando a função de distribuição conjunta das observações amostrais em termos dos parâmetros a seres estimados e, em seguida, maximizando a função de distribuição em relação aos parâmetros desconhecidos. As soluções da equação de maximização produzem as funções de estimativa como relações entre os parâmetros estimados e as observações. Estimativas de máxima verossimilhança são aquelas que atribuem valores aos parâmetros desconhecidos de forma a maximizar a probabilidade da amostra observada.

2.12 Método de Simulação Monte Carlo (MSMC)

O Método de Simulação Monte Carlo é uma técnica estatística amplamente utilizada para resolver problemas que envolvem incerteza ou variáveis aleatórias. O princípio central do método é a geração de amostras aleatórias para simular um grande número de experimentos e observar os resultados. Essas simulações repetidas permitem estimar probabilidades e prever comportamentos de sistemas complexos, onde soluções analíticas seriam difíceis ou impossíveis de se obter [19].

No contexto jogos de azar, o Método de Simulação Monte Carlo será utilizado da seguinte forma:

1. Modelagem do Problema: Primeiro, define-se um modelo matemático que descreve o sistema ou processo que se deseja estudar. Esse modelo deve incluir todas as variáveis aleatórias que influenciam o resultado final.
2. Geração de Amostras Aleatórias: A seguir, amostras aleatórias são geradas para cada uma das variáveis do modelo. Cada conjunto de amostras simula um experimento individual ou uma iteração do processo.
3. Avaliação do Modelo: Em cada iteração, o modelo matemático é avaliado com base nas amostras geradas, e o resultado do experimento é registrado.
4. Repetição: O processo é repetido um grande número de vezes. Quanto mais simulações forem realizadas, mais precisos serão os resultados.
5. Análise Estatística: Após a repetição das simulações, os resultados são analisados estatisticamente para estimar as probabilidades dos eventos de interesse e compreender melhor o comportamento do sistema.

Matematicamente, a probabilidade de um evento p ser observado pode ser estimada pela equação

$$p \approx \frac{n}{N}$$

onde, n é o número de vezes que o evento ocorreu e N é o número total de simulações realizadas. Observe que quanto maior o valor de N, mais preciso será o valor estimado para p, já que o erro diminui com o aumento de simulações.

Um exemplo claro da aplicação do MSMC está no jogo de *Blackjack*. De maneira resumida, o *Blackjack* é um jogo de cartas no qual o objetivo do jogador é derrotar o crupiê acumulando uma mão com valor superior ao dele, sem ultrapassar 21. Como o jogo envolve

múltiplas decisões estratégicas e combinações de cartas, e como o baralho se modifica no decorrer do jogo, as probabilidades não permanecem fixas, sendo alteradas conforme as decisões e o progresso do jogo. Estimar as probabilidades e identificar a melhor estratégia não é trivial de se calcular analiticamente.

No *Blackjack*, o jogador pode tomar várias decisões com base nas cartas que tem e nas cartas do crupiê. A eficácia dessas decisões depende de uma série de fatores, incluindo a combinação das cartas e as regras do cassino. Com o método de Monte Carlo, é possível simular milhões de rodadas de *Blackjack* para estimar as probabilidades de vencer com diferentes estratégias.

O principal benefício do MSMC em jogos como *Blackjack* é a capacidade de testar diversas estratégias de forma eficiente e rápida. Em vez de calcular as probabilidades de forma analítica, o que pode ser extremamente difícil devido à complexidade do jogo senão impossível, Monte Carlo permite que se explorem diferentes cenários, levando em consideração todas as variáveis envolvidas. Além disso, a técnica é adaptável a diferentes tipos de jogos e pode ser aplicada em qualquer jogo de azar que envolva probabilidades.

2.13 Caminhante Aleatório Em Uma Dimensão

Para formalizar a situação de jogos de azar práticos, pode-se descrevê-los como sequências de tentativas de Bernoulli, onde o jogador ganha uma unidade para cada sucesso e perde uma unidade para cada falha, com probabilidades respectivas de p e $q = 1 - p$. O capital do jogador, representado por x, é uma posição na linha real. Assim, a partir de um capital inicial $x = x_0$, o jogador move-se ao longo da linha a cada jogada, indo para $x_0 + 1$ (com probabilidade p) ou $x_0 - 1$ (com probabilidade q). Após um número indefinido de jogadas, o passeio aleatório termina em $x = 0$ ou $x = C$, onde C é o capital do adversário. Com um cassino como adversário, C é efetivamente infinito. Essa construção descreve o clássico passeio aleatório e leva ao teorema da ruína do jogador.

Para determinar a probabilidade de que o capital do jogador retorne ao seu valor inicial x_0, observa-se que esse evento só pode ocorrer em jogadas de número par. Após $2n$ jogadas, é necessário e suficiente que o jogador tenha experimentado n vitórias e n derrotas. Portanto,

$$Probabilidade\ (n\ vitórias\ e\ n\ derrotas) = P_{2n} = \binom{2n}{n} p^n q^n \tag{2.11.1}$$

Utilizando a aproximação de Stirling

$$n! \approx n^n e^{-n} \sqrt{2\pi n} \tag{2.11.2}$$

É possível mostrar que

$$\binom{2n}{n} p^n q^n \approx \frac{(4pq)^n}{\sqrt{\pi n}} \tag{2.11.3}$$

Para $p = q = 0{,}5$,

$$P_{2n} \approx \frac{1}{\sqrt{\pi n}} \tag{2.11.4}$$

Observe que a série $\sum P_{2n}$ diverge, mas a probabilidade $P_{2n} \to 0$ conforme n aumenta. Consequentemente, *o retorno ao capital inicial x_0 é um evento recorrente certo, enquanto o tempo médio de recorrência é infinito.*

Definindo r_j como a probabilidade de que um retorno a x_0 ocorra na *j*-ésima jogada (não necessariamente pela primeira vez), então o evento recorrente $x = x_0$ é incerto se e somente se

$$R = \sum_{j=0}^{\infty} r_j \tag{2.11.5}$$

for finito. Neste caso, a probabilidade ρ de que $x = x_0$ recorra alguma vez é dada por

$$\rho = \frac{R - 1}{R} \tag{2.11.6}$$

Para $p \neq 0{,}5$, tem-se que

$$R = \frac{1}{\sqrt{1 - 4pq}} = \frac{1}{|p - q|} \tag{2.11.7}$$

Portanto, a probabilidade de que o número acumulado de sucessos e falhas se torne igual alguma vez é expressa por $1 - |p - q|$. Esta equação também representa a probabilidade de pelo menos um retorno a x_0.

2.14 Lógica Determinística e Lógica Probabilística

Lógica determinista, termo introduzido por Moritz Schlick [15], é a visão de que uma proposição sobre o futuro, é necessariamente verdadeira, ou a sua negação é necessariamente verdadeira. Isso significa que o estado atual do mundo determina todos os eventos futuros, eliminando a ideia de incerteza ou indeterminação quanto ao que virá a ocorrer. Uma analogia prática para ilustrar essa noção pode ser encontrada nas máquinas de pegar prêmios que estão presentes em diversos locais, como shoppings, bares e aeroportos que são chamadas de grua [12][15].

Essas máquinas, que oferecem brinquedos de pelúcia, celulares ou outros prêmios, funcionam aparentemente como um jogo de azar, onde o jogador coloca uma ficha e utiliza uma garra mecânica para tentar pegar um dos prêmios. À primeira vista, o sucesso ou fracasso parece

depender inteiramente da habilidade ou sorte do jogador. No entanto, muitas dessas máquinas são programadas para liberar o prêmio apenas em momentos específicos, após um determinado número de tentativas, criando um sistema que exemplifica claramente a lógica determinista [12][15].

A programação da máquina segue um padrão pré-estabelecido, onde, por exemplo, somente após a 100ª jogada o prêmio será liberado. Assim, o estado presente da máquina (quantas jogadas já foram feitas até o momento) define necessariamente o futuro: o prêmio será entregue apenas na jogada programada, independentemente das ações individuais dos jogadores que vierem antes. Não importa o quanto um jogador seja habilidoso ou sortudo, se ele jogar na 99ª tentativa, não ganhará o prêmio. A 100ª jogada, por outro lado, resultará inevitavelmente no sucesso, pois o futuro foi determinado pela programação da máquina [12][15].

Essa dinâmica reflete diretamente o conceito de determinismo: o estado atual — neste caso, o número de jogadas já realizadas — contém a verdade sobre o que acontecerá em seguida. Para a máquina, o "*truthmaker*" que torna a proposição "O próximo jogador ganhará o prêmio" verdadeira ou falsa é a quantidade de tentativas já realizadas [12][15][18].

A lógica probabilística é uma abordagem que lida com a incerteza, em contraste direto com a lógica determinista, que considera os eventos futuros como necessariamente verdadeiros ou falsos. Na lógica probabilística, os eventos são definidos em termos de chances ou probabilidades, e o futuro não é fixo, mas sim moldado por uma série de resultados possíveis, cada um com uma probabilidade associada. Um exemplo clássico que ilustra esse conceito é o jogo de *Roulette* em cassinos, onde as probabilidades desempenham um papel central na determinação dos resultados, por exemplo, ao fazer uma aposta, o jogador não pode prever com certeza onde a bola vai parar, pois o resultado de cada rodada da *Roulette* é regido por eventos suficientemente aleatórios, ou seja, não existe tecnologia capaz de prever. No entanto, a lógica probabilística nos permite calcular as chances de cada possível resultado com base nas regras do jogo [12][16].

Em uma *Roulette*, que tem 37 números (de 1 a 36, mais o zero), a probabilidade de a bola cair em um número específico é de 1 em 37, ou aproximadamente 2,7%. Por outro lado, apostar que o número será de uma determinada cor, como vermelho, envolve 18 números, o que resulta em uma probabilidade de 18/37, ou cerca de 48,6%. Esse exemplo demonstra a essência da lógica probabilística: não se pode prever o resultado de uma rodada individual, mas pode-se determinar as chances de cada resultado ocorrer. No longo prazo, com um grande número de rodadas, as frequências observadas dos resultados tenderão a se aproximar das probabilidades teóricas — um princípio essencial conhecido como LGN. Isso significa que,

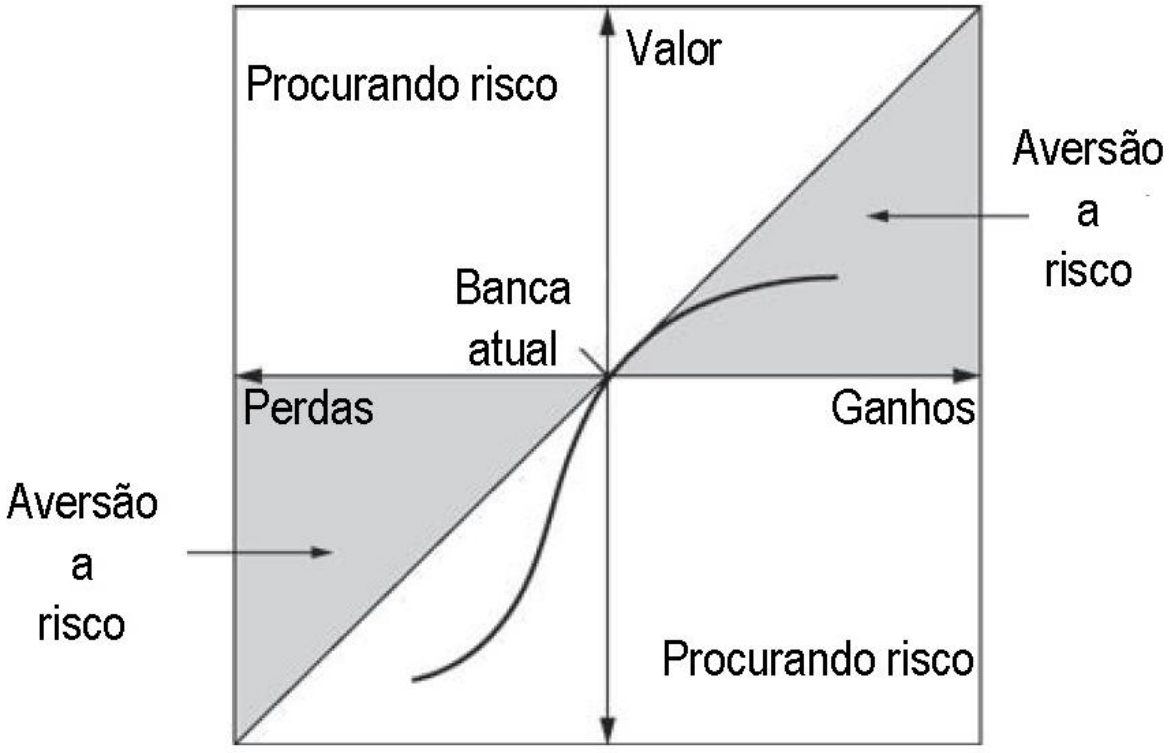
embora um jogador possa ganhar ou perder em qualquer rodada isolada, em uma sequência muito longa de rodadas, os resultados se distribuirão de acordo com as probabilidades calculadas [12][16].

2.15 Teoria dos Prospectos

Desenvolvida pelos psicólogos cognitivos Daniel Kahneman e Amos Tversky, a Teoria dos Prospectos descreve como os indivíduos avaliam ganhos e perdas na realidade psicológica, em oposição ao modelo racional de von Neumann-Morgenstern, que prescreve a tomada de decisões sob condições de incerteza (abordagens descritivas versus prescritivas). Assim, a Teoria dos Prospectos abrange anomalias fora da teoria econômica convencional (às vezes designadas como comportamento "irracional").

Primeiro, as escolhas são ordenadas de acordo com uma heurística específica. A partir daí, valores (subjetivos) são atribuídos a ganhos e perdas em vez de ativos finais. A função de valor, definida por desvios de um ponto de referência, é normalmente côncava para ganhos ("aversão ao risco"), convexa para perdas ("busca de risco") e mais íngreme para perdas do que para ganhos ("aversão à perda"), como ilustrado no gráfico 1.

Gráfico 1: Uma função de valor da Teoria dos Prospectos



Fonte: Richard A. Epstein

Quatro princípios principais sustentam grande parte da Teoria do Prospecto:

- As pessoas são inerentemente (e irracionalmente) menos inclinadas a apostar com lucros do que com um saldo reduzido por perdas.
- Se uma proposição é enquadrada para enfatizar possíveis ganhos, as pessoas são mais propensas a aceitá-la; a mesma proposição, se enquadrada para enfatizar possíveis perdas, é mais provável de ser rejeitada.
- Uma maioria substancial da população preferiria R$ 5.000,00 a uma aposta que retorna ou R$ 10.000,00 com probabilidade de 0,6 ou zero com probabilidade de 0,4 (um valor esperado de R$ 6.000,00).
- Comportamento de aversão à perda/busca de risco: em uma escolha entre uma perda certa de R$ 4.000,00 em oposição a uma probabilidade de 5% de perder R$ 100.000,00 combinada com uma probabilidade de 95% de não perder nada (uma perda esperada de R$ 5.000,00), outra maioria significativa preferiria a segunda opção.

Nos últimos anos, a Teoria dos Prospectos adquiriu status como um bloco de construção amplamente citado e solidamente estruturado da psicologia econômica.

2.16 Neurociência

No contexto dos jogos de azar, o sistema dopaminérgico desempenha um papel central, pois está profundamente ligado ao sentimento de prazer e à motivação, funcionando como o principal mecanismo por trás do vício em jogos. A dopamina é um neurotransmissor que regula a sensação de recompensa no cérebro e influencia comportamentos que são percebidos como importantes ou motivadores. Quando uma pessoa joga e ganha, há um aumento nos níveis de dopamina, o que faz com que a atividade seja associada a uma experiência extremamente prazerosa e gratificante, reforçando o desejo de continuar jogando [14].

No entanto, até mesmo quando o jogador perde, o sistema dopaminérgico ainda é ativado devido ao fenômeno do "quase ganhar". Esse "quase" mantém o jogador motivado, pois a sensação de que a vitória está próxima é suficiente para que o cérebro libere dopamina, criando a ilusão de que continuar jogando pode resultar em ganhos. Esse ciclo de expectativas e frustrações leva a flutuações intensas de humor e, consequentemente, a uma liberação exacerbada de dopamina. Isso explica por que jogadores frequentemente se sentem

mentalmente exaustos após longas sessões, especialmente quando há altos níveis de risco envolvidos nas apostas, como no caso de grandes somas de dinheiro [14].

O cérebro é programado para dar mais atenção a atividades que ele percebe como importantes, e o que define essa importância é a quantidade de dopamina liberada. Quanto maior e mais rápida for essa liberação, maior o potencial de gerar um comportamento compulsivo. Assim, o jogador se torna viciado não apenas no ato de ganhar, mas também na emoção do jogo em si, porque a dopamina é liberada em níveis elevados, independentemente do resultado final [14].

Além disso, o esforço necessário para realizar uma atividade também influencia a motivação. Atividades que exigem mais esforço tendem a gerar menos motivação, a menos que a recompensa seja considerada muito alta. Por isso, as recompensas rápidas oferecidas pelos jogos de azar se tornam extremamente atrativas para o cérebro, pois proporcionam gratificação imediata e de alta intensidade, sem exigir grandes esforços físicos ou mentais [14].

De maneira específica para este contexto, o sistema dopaminérgico é extremamente estimulado em situações de jogos de azar, porque o cérebro reconhece essas atividades como importantes devido às grandes liberações de dopamina associadas às recompensas monetárias. O vício se desenvolve porque o cérebro associa essa atividade à necessidade de repetição para continuar recebendo doses elevadas de dopamina, levando o jogador a buscar continuamente esse prazer, mesmo quando o resultado é negativo [14].

2.17 História da *Roulette*

A *Roulette* ou simplesmente Roleta é um jogo de azar muito comum em cassinos, o termo deriva do francês *Roulette*, que significa "roda pequena". O uso da *Roulette* como elemento de jogo de azar, em configurações distintas da atual, não está documentado na entrada da Idade Média [23].

Embora sua origem seja desconhecida, sabe-se pela história que foi inventada através de um erro quando o famoso matemático e físico Blaise Pascal tentou criar uma máquina de movimento perpétuo em algum momento do século XVII. As rodas e, por extensão, as *Roulettes*, sempre tiveram uma conexão com o mundo mágico e exotérico. Assim, uma delas parte do tarô, mais precisamente dos que conhecem como arcanos maiores. A eleição de números deu um alcance mais vinculado à magia (a soma dos primeiros números [1-36] é o número mágico por excelência: 666) [23]..

2.18 História do *Blackjack*

A origem exata do *Blackjack* é bastante obscura, não foi inventado por uma única pessoa, mas aparentemente evoluiu de jogos de cartas relacionados no século XIX e ganhou popularidade durante a Primeira Guerra Mundial. O *Blackjack* (ou "21", ou *Vingt-et-Un*) exibe semelhanças estruturais com *Baccarat*, "7 1/2", Quinze, e também com *Trente et Quarante*. No entanto, a conexão com esses jogos permanece historicamente tênue.

Da França para a América, inicialmente foi rejeitado. Para aumentar o interesse, os cassinos ofereciam bônus especiais, um dos quais era um pagamento de 10:1 se a mão do jogador incluísse o Ás de Espadas e qualquer Valete preto. Embora o bônus geralmente tenha sido descontinuado, o nome permanece - embora "21" seja um nome mais lógico.

O *Blackjack* não constitui um jogo no sentido usual da teoria dos jogos, uma vez que a linha de jogo de um jogador é fixada a *priori* - o *dealer* não pode exercer julgamento ou inteligência de forma independente (pode-se referir a isso como um "jogo de uma pessoa"). Seu apelo especial reside em sua propriedade de ser a única forma de jogo público onde o jogador conhecedor pode obter uma expectativa matemática positiva. Ao contrário de *Craps* [3]e *Roulette*, o *Blackjack* possui uma memória (a interdependência das cartas) e uma consciência (jogadas inferiores serão inevitavelmente penalizadas) e não é democrático (a agilidade mental e a capacidade de retenção do jogador são fatores significativos). Esses fatos foram realizados de maneira qualitativa por muitos jogadores profissionais por quase um século. Na década de 1950, análises numéricas da estratégia de *Blackjack* foram realizadas simultaneamente por engenheiros e matemáticos em várias empresas de engenharia nos Estados Unidos [1]. Resultados quantitativos através de programação de computadores foram relatados pelo autor em palestras na UCLA, *Princeton* e outras universidades em 1955. A primeira estratégia publicada (Ref. *Baldwin et al*.) apareceu em 1956. Allan Wilson, da *San Diego State University*, programou uma simulação de várias centenas de milhares de mãos de *Blackjack* em 1957, obtendo os resultados mais precisos até aquela data [1].

Então, em 1962, Edward O. Thorp (Ref. 1962) publicou "*Beat the Dealer*", uma obra imediatamente aclamada como a bíblia dos adeptos do *Blackjack*. Aqui, pela primeira vez, estava um tratamento matemático abrangente de toda a gama de estratégias de *Blackjack* e suas probabilidades subjacentes, além da primeira síntese quantitativa de sistemas de contagem de cartas [1].

[3] É um jogo de dados em que os jogadores apostam nos resultados do lançamento de um par de dados

Desde então, o *Blackjack* tornou-se o passatempo favorito nos empórios de jogos de Nevada e nos cassinos resort da área do Caribe. Também é destaque em cassinos em toda a América do Sul, Austrália e Extremo Oriente. Na Europa, o jogo às vezes recebe um status inferior ao de primeira classe (embora seja destaque no Casino *Municipale di Venezia* sob uma placa anunciando "*Gioco del Blak Jak o 21*").

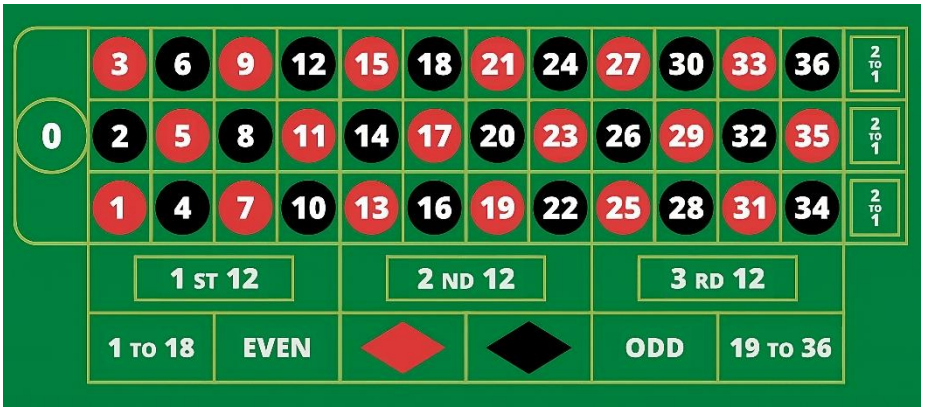
2.19 Regras da *Roulette*

Este capítulo apresenta as regras do jogo *Roulette*, conforme encontrado em cassinos em geral, tomando como referência, para este trabalho, o cassino online Blaze [21].

2.19.1 Objetivo do Jogo

O objetivo em *Roulette* é prever o número em que a bolinha irá cair, através de uma ou mais apostas que cubram esse número específico. A *Roulette* inclui os números de 1 a 36, mas um único 0 (zero) como ilustrado na figura 1.

Figura 1: Mesa de apostas da *Roulette*



Fonte: Vecteezy [25].

Após o tempo de aposta encerrar, uma pequena esfera é jogada dentro da *Roulette*. No final, ela cairá em uma das caçapas numeradas da *Roulette*. O jogador vence se tiver feito uma aposta que cubra o número sorteado.

2.19.2 Tipos de Aposta

O jogador pode fazer vários tipos de apostas diferentes na mesa de *Roulette*. As apostas podem cobrir um único número ou um determinado conjunto de números e cada tipo de aposta

tem sua própria taxa de pagamento.

As apostas feitas nos espaços numerados no ponto de apostas, ou nas linhas entre elas, são chamadas Apostas Internas, enquanto as apostas feitas nas caixas especiais abaixo e ao lado da placa principal são chamadas Apostas Externas.

2.19.3 Apostas Internas

- Direta - coloque sua ficha diretamente sobre qualquer número (inclusive o zero).
- Apostas dividida - Coloque sua ficha na linha entre dois números quaisquer na vertical ou horizontal.
- Aposta em Rua - Coloque sua ficha no final de qualquer linha de números. Uma aposta em rua cobre três números.
- Aposta em Canto - Coloque sua ficha no canto (interseção central) onde quatro números se encontram. Todos os quatro números estarão cobertos.
- Aposta em Linha - coloque sua ficha no final de duas linhas, bem na interseção entre as duas linhas. Uma aposta em linha cobre os números em ambas as linhas, para um total de seis números.

2.19.4 Apostas Externas

- Aposta em coluna - coloque sua ficha em uma das caixas com a marca "2 para 1" no final da coluna para cobrir todos os 12 números naquela coluna. O zero não é coberto por nenhuma aposta em coluna.
- Aposta em dúzia - coloque sua ficha em uma das três caixas marcadas "1º 12", "2º 12" ou "3º 12" para cobrir os 12 números junto à caixa.
- Vermelho/Preto - coloque sua ficha na caixa Vermelha ou Preta para cobrir os 12 números vermelhos ou os 18 pretos. O zero não é coberto por nenhuma aposta.
- Even/Odd (Par/ímpar) - coloque sua ficha em uma dessas caixas para cobrir 18 números pares ou os 18 ímpares. O zero não é coberto por nenhuma dessas apostas.
- 1-18/19-36 - coloque sua ficha em uma dessas caixas para cobrir o primeiro ou segundo conjunto de 18 números. O zero não é coberto por nenhuma dessas apostas.

2.19.5 Pagamento

Seu pagamento depende do tipo de aposta feita.

2.19.5.1 Apostas Internas

Tabela 1: Tipos de apostas e pagamentos para o jogo *Roulette*.

Tipo de aposta	Pagamento
Direta	35:1
Dividir	17:1
Rua	11:1
Canto	8:1
Linha	5:1

Fonte: Blaze [21].

2.19.5.2 Apostas Externas

Tabela 2: Tipos de apostas e pagamentos para o jogo *Roulette*.

Tipo de aposta	Pagamento
Coluna	2:1
Dúzia	2:1
Preto/Vermelho	1:1
Par/Ímpar	1:1
1-18/19-36	1:1

Fonte: Blaze [21].

2.19.6 Retorno ao Jogador

A porcentagem ideal teórica de pagamento é: 97,30%

2.19.7 Limites de Aposta

Tabela 3: Limites de apostas.

Aposta	Limite (R$)
Direta	2,50 – 5.000
Dividida	2,50 – 10.000
Rua	2,50 – 15.000
Canto	2,50 – 20.000

Linha	2,50 – 30.000
Coluna	2,50 – 60.000
Dúzia	2,50 – 60.000
Vermelho/Preto	2,50 – 100.000
Par/Ímpar	2,50 – 100.000
1-18/19-36	2,50 – 100.000

Fonte: Blaze [21].

2.20 Regras do *Blackjack*

Este capítulo apresenta as regras do jogo *Blackjack*, conforme encontrado em cassinos em geral, tomando como referência, para este trabalho, o cassino online Blaze [21].

2.20.1 Objetivo do Jogo

No *Blackjack*, o objetivo é ter uma contagem de cartas maior que a do crupiê sem ultrapassar 21 pontos e é jogado com 8 baralhos comuns de 52 cartas. Os valores das cartas no *Blackjack* são os seguintes: As cartas de 2 a 10 valem seu valor de face, as cartas de figura (Valetes, Rainha e reis) valem 10 cada e Ases valem 1 ou 11, o que for mais favorável para a mão. Observe que uma mão que inclui um ás com valor de 11 é chamada de *soft* enquanto todas as outras são chamadas de *hard*.

A melhor mão é o *Blackjack*, onde as duas primeiras cartas somam exatamente 21. É jogado somente contra o crupiê, e não contra outros jogadores. O crupiê tem uma estratégia fixa e determinada que é a parar em um *soft* 17.

É permitido dobrar a aposta em quaisquer duas cartas iniciais, mas não após dividir.

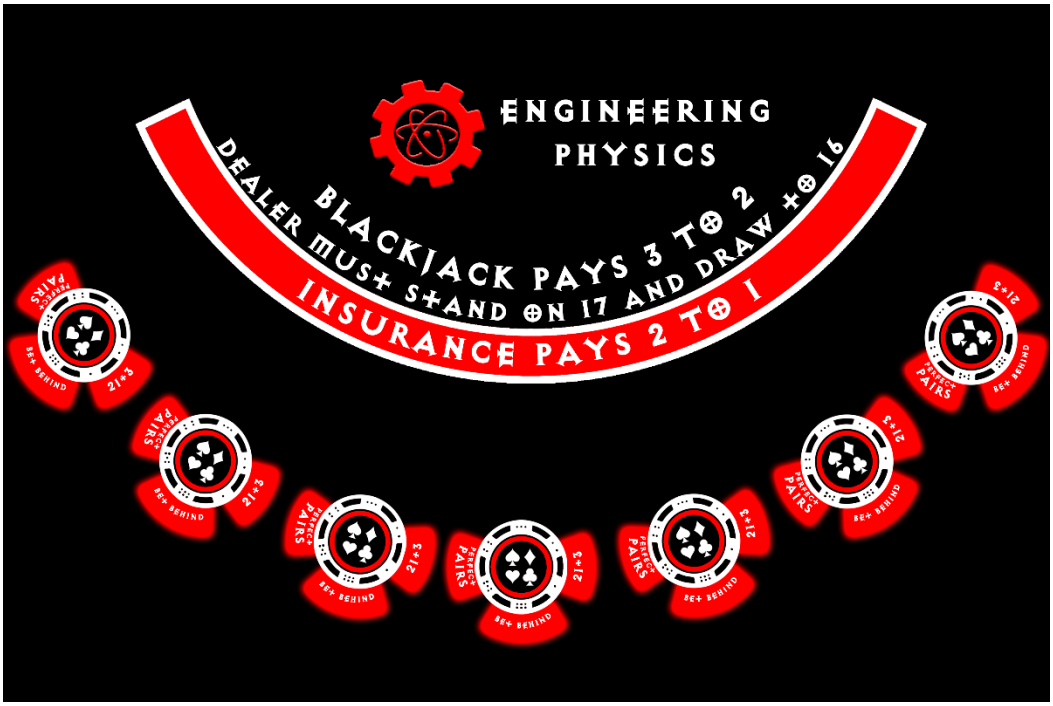
Cartas de valor igual podem ser divididas apenas uma vez por mão, salvo o caso especial em que se divide duas cartas Ás onde o jogador recebe somente uma carta para cada Ás dividido.

Quando o crupiê mostra um Ás, é oferecido o seguro, que paga 2 para 1. O *Blackjack* paga 3 para 2, e um empate ocorre quando as mãos têm o mesmo valor.

O jogo é comandado por um crupiê e permite até 7 jogadores sentados na mesa de *Blackjack* como ilustrado na figura 2.19.1. Inicialmente há um tempo para realizar as apostas e após o tempo de aposta expirar, o crupiê distribui uma carta variada para cima a cada jogador,

a distribuição começa pelo primeiro jogador à esquerda do crupiê e segue no sentido horário até terminar com o próprio crupiê, logo depois, o crupiê então distribui uma segunda carta virada para cima para cada jogador, mas a segunda carta do crupiê é distribuída virada para baixo.

Figura 2: Mesa de apostas de *Blackjack*.



Fonte: Autor.

2.20.2 Blackjack

Se o valor da mão original do jogador de duas cartas for exatamente 21, o jogador tem um *Blackjack*!

2.20.3 Seguro

Se a carta virada para cima do crupiê for um Ás, o jogador tem a opção de adquirir um seguro para compensar o risco de o crupiê ter um *Blackjack*, a não ser que o jogador tenha um *Blackjack*. A quantidade do seguro é igual à metade de sua aposta principal, e a aposta do seguro é feita separadamente de aposta em sua mão. O crupiê então verifica o valor de sua carta virada para baixo para verificar se tem um *Blackjack*. Se o crupiê não tiver *Blackjack*, a rodada continua, se o crupiê tiver um *Blackjack* e o jogador não, o crupiê vence, se tanto o jogador quanto o crupiê tiverem um *Blackjack*, o jogo termina empatado e a aposta do jogador é

devolvida.

Observe que, quando a carta do crupiê virada para cima for um 10 ou uma carta de figura, o jogador não terá a opção de adquirir seguro, e o crupiê não irá verificar se sua carta virada para baixo é um *Blackjack*.

2.20.4 Dobrar, Sortear ou Ficar

Quando o crupiê não tiver um *Blackjack* ao verificar suas duas cartas iniciais, os jogadores têm a chance de melhorar os valores de suas mãos na rodada. Para habilitar isso, o crupiê se move no sentido horário ao redor da mesa oferecendo a distribuição de outras cartas para as mãos dos jogadores. Se o valor da mão inicial do jogador não for 21, o jogador pode decidir dobrar. Nesse caso, o jogador irá dobrar sua posta e será dada somente uma carta adicional para a sua mão. Em alternativa, o jogador pode decidir sortear para receber uma carta adicional a acrescentar ao valor de sua mão. O jogador pode sortear mais de uma vez para receber cartas adicionais, antes de decidir ficar quando estiver satisfeito com o valor de sua mão.

2.20.5 Dividir

Se a mão inicial do jogador for um par de cartas de valor igual, o jogador pode dividir o par, para fazer duas mãos separadas, cada uma com uma aposta separada igual à sua aposta principal. Após uma segunda carta ser distribuída as duas mãos do jogador, o mesmo pode melhorar o valor dessas duas mãos decidindo sortear. Outra vez, o jogador pode escolher ficar, uma vez que estiver satisfeito com o valor de suas duas mãos. Entretanto, se o jogador dividir um par de Ases inicial, o jogador somente receberá uma carta adicional por mão sem opção de sortear.

2.20.6 Apostas Laterais

Esse jogo de *Blackjack* também contempla apostas laterais opcionais e são elas: Pares perfeitos e 21+3. O jogador pode fazer apostas laterais, em combinação com a sua principal aposta *Blackjack*. O jogador tem a chance de ganhar em qualquer aposta lateral mesmo que mais tarde ganhe ou não em sua aposta *Blackjack*.

2.20.6.1 Pares Perfeitos

A aposta lateral pares perfeitos *Perfect Pairs* dá ao jogador a chance de ganhar se suas primeiras duas cartas tiverem qualquer par - por exemplo, duas Damas, dois Ases, ou dois 3s. Existem três tipos de pares, cada um com um pagamento diferente:

- Par perfeito: Mesmo naipe, por exemplo dois Ases de espadas.
- Par colorido: Naipes diferentes da mesma cor, por exemplo 2 de ouros e 2 de copas.
- Par misto: Naipes diferentes de uma cor diferente, por exemplo 10 de copas e 10 de paus.

2.20.6.2 21+3

A aposta lateral 21+3 dá ao jogador a chance de ganhar se as suas duas primeiras cartas mais a carta virada para cima do crupiê compreende qualquer das seguintes combinações vencedoras (similares às do pôquer), cada um com um prêmio diferente:

- *Suited Trips* - Um trio idêntico, por exemplo 3 Damas de Copas
- *Straight Flush* - Uma sequência numérica de mesmo naipe, por exemplo 10, Valete e Dama de ouros.
- *Three of a Kind* - mesmo valor, mas naipes diferentes, por exemplo, quaisquer três Reis diferentes.
- *Straight* - Uma sequência numérica, mas de naipes diferentes 2 de espadas, 3 de paus e 4 de copas.
- *Flush* - cartas não sequenciais no mesmo naipe, por exemplo 2, 6 e 10 de paus.

2.20.7 Bet Behind

O recurso *Bet Behind* (apostar na jogada de outro jogador) permite apostar em uma mão dada por outro jogador. Por exemplo, um jogador irá compartilhar o resultado da mão do outro jogador. Os pagamentos para *Bet Behind* são iguais àqueles para apostas comuns.

O jogador pode fazer uma aposta *Bet Behind* estando ou não sentado à mesa do *Blackjack* com o intuito de jogar sua própria mão. Contudo, um jogador não pode fazer uma aposta *Bet Behind* da sua própria mão.

Faça uma aposta em qualquer local ativo de *Bet Behind* no lugar de outro jogador enquanto a aposta estiver aberta.

Se fizer uma aposta *Bet Behind* e o jogador nesse lugar decidir não participar na rodada, sua aposta é reembolsada imediatamente.

O jogador pode decidir se faz ou não uma aposta de seguro quando a carta do crupiê for um Ás. O jogador também pode decidir previamente se deseja ou não dobrar sua aposta quando o outro jogador atrás do qual um jogador está fazendo a *Bet Behind* opta por Dobrar ou Dividir. Se o jogador decidir o pagamento, a mesma decisão será adotada para a mão do jogador.

2.20.8 Resultado

Se a soma de sua mão exceder 21, o jogador está eliminado e perde sua aposta nessa mão. Quando todos os jogadores tiverem tido sua vez, o crupiê revela o valor da carta virada para baixo. O crupiê deve sortear em uma mão de 16, ou menor, e deve ficar em uma mão *soft* de 17 ou mais. Observe que uma "mão *soft*" inclui um Ás com valor de 11.

O jogador ganha quando o valor de sua mão final for o mais próximo de 21, comparada com a mão do crupiê, ou quando ele for eliminado. Se o valor de sua mão for o mesmo do crupiê e menor que 21 ou ambos tiverem *Blackjack*, a rodada do jogo termina em um empate e a sua aposta é retornada.

O *Blackjack* somente é possível usando as duas cartas na distribuição inicial de duas cartas. Uma mão de 21 resultante de um par dividido não é considerada *Blackjack*. O *Blackjack* também bate qualquer mão de 21 com mais de duas cartas e também um 21 resultante de um par dividido.

2.20.9 Pagamento

2.20.9.1 Blackjack

Tabela 4: Pagamentos das apostas em *Blackjack*.

Tipo de aposta	Pagamento
Mão com *Blackjack*	3:2
Mão vencedora	1:1
Seguro se o crupiê tiver *Blackjack*	2:1

Fonte: Blaze [21].

2.20.9.2 Pares perfeitos

Tabela 5: Pagamentos das apostas laterais Pares Perfeitos.

Mão	Pagamento
Par Perfeito	25:1
Par Colorido	12:1
Par Misturado	6:1

Fonte: Blaze [21].

2.20.9.3 21+3

Tabela 6: Pagamento da aposta lateral 21+3.

Mão	Pagamento
Suited Trips	100:1
Straight Flush	40:1
Three Of A Kind	30:1
Straight	10:1
Flush	5:1

Fonte: Blaze [21].

2.20.10 Retorno ao Jogador

A porcentagem ideal teórica de pagamento é:

- *Blackjack* - 99,29% (Baseado apenas na primeira mão)
- Aposta Pares Perfeitos: 95,90%
- Aposta 21+3: 96,80%

2.20.11 Limites de Apostas

O painel de limites de apostas exibe os limites de apostas mínimo e máximo permitidos na mesa, que podem variar de mesa para mesa

Tabela 7: Limites de apostas.

Aposta	Limite (R$)
Blackjack	25 – 50.000
Perfect Pairs	0,01 – 5.000

21+3	0,01 – 2.500

Fonte: Blaze [21].

2.21 Definição de Cassinos

Os cassinos surgiram há muito tempo, ainda no século XVII, em Itália, na cidade de Veneza. No entanto, como ainda estavam nos primórdios desta prática, o termo utilizado para identificar os jogos em Veneza ainda não era o atual. A palavra casino vem do italiano para definir clube social ou vila pequena, portanto, não é uma definição direta para jogos. Como já dito pelo historiador Afonso Ribeiro *"A ideia dos primeiros cassinos era juntar as pessoas para entretenimento e conversa. Os jogos não eram a atração principal"* [7].

Hoje em dia, um casino (português europeu) ou cassino (português brasileiro) é uma Casa de Jogos de azar onde os clientes, frequentemente atraídos por um ambiente luxuoso e festivo, apostam dinheiro em diversas modalidades de jogos [10]. Os frequentadores geralmente buscam, mas também a chance de grandes ganhos, mas também entretenimento, o que os torna vulneráveis às estratégias dos cassinos que se baseiam no estudo matemático e probabilístico que os favoreçam [1].

Os cassinos, por sua vez, assim como qualquer outra empresa, são projetados para maximizar lucros, oferecendo jogos de azar como *Roulette*, *Blackjack*, *Poker*, entre outros, todos projetados para garantir vantagem à casa [12]. Estes estabelecimentos são, muitas vezes localizados em regiões estratégicas, como áreas turísticas ou locais próximos a fronteiras, onde a legislação pode ser mais permissiva ou ambígua, facilitando sua operação [12]. O ambiente interno é meticulosamente planejado para que os apostadores percam a noção do tempo, sem relógios ou janelas, e com luzes e sons que incentivam a continuidade do jogo. Além disso, mulheres atraentes com roupas provocantes trabalham no local, ajudando a criar um clima de descontração e distração, o que colabora para que os jogadores não estejam sempre em pleno juízo [13].

2.22 Cassinos na Internet

Com o surgimento da internet, o conceito de cassino se transformou, possibilitando que jogadores façam apostas online de qualquer lugar, o que tornou os cassinos virtuais uma das formas de entretenimento mais populares e lucrativas em todo o mundo. [7]. Cassinos na internet, também conhecidos como cassinos online ou virtuais, permitem que qualquer pessoa jogue jogos de cassino através da internet. Alguns oferecem uma variedade de jogos, enquanto

outros se especializam em apenas um tipo. No Brasil, os jogos *Crash* são os mais populares, seguidos pelos caça-níqueis e pela *Roulette* [10].

Não se sabe ao certo se os cassinos online oferecem probabilidades mais favoráveis aos jogadores em comparação com os cassinos tradicionais. O grande desafio é a dificuldade em estabelecer confiança, já que muitos operam em jurisdições onde sua atividade não é devidamente regulamentada. [10].

3 DESENVOLVIMENTO

3.1 Explicação do Código Envolvendo o Caminhante Aleatório

O código tem como objetivo simular o desempenho de vários jogadores em um jogo com probabilidade definida, utilizando uma abordagem probabilística para modelar ganhos e perdas ao longo de múltiplas rodadas. Para visualização, um gráfico é gerado para representar quantos jogadores estão ganhando e quantos estão perdendo dinheiro ao longo do tempo.

Os parâmetros necessários para a execução da simulação são: quantidade de jogadores; número de jogos; saldo inicial dos jogadores; valor constante de aposta; probabilidade de o jogador ganhar cada partida.

A lógica do código é simples: a biblioteca GSL [22] é utilizada para realizar sorteios de números aleatórios e equiprováveis uniformemente distribuídos no intervalo [0, 1], simulando a aleatoriedade esperada em jogos de azar (como um baralho bem embaralhado ou um dado não viciado). Para cada jogador, um número aleatório é sorteado e comparado com o valor da probabilidade de vitória predefinida. Se o número sorteado for menor ou igual à probabilidade de vitória, o jogador ganha a aposta, que é multiplicada por uma constante chamada ODD (que representa o multiplicador da aposta). O saldo total do jogador é atualizado de acordo com o resultado. Caso contrário, o valor apostado é subtraído do saldo e esta lógica é repetida para todos os jogadores em cada rodada de jogo.

O comportamento do código é comparado ao problema do Caminhante Aleatório, em que a probabilidade de um passo para a direita é igual à probabilidade de o jogador vencer (p), enquanto a probabilidade de um passo para a esquerda é a chance de perder (q), sendo que $p + q = 1$. Ao simular várias rodadas (ou passos), o valor esperado para o saldo final dos jogadores é determinado, de forma semelhante à posição final do caminhante aleatório.

Além disso, ao aplicar a LGN, é possível prever um comportamento determinístico quando o número de jogos tende ao infinito. Por exemplo, se $p = q$, o saldo esperado dos jogadores é zero, ou seja, nenhum lucro ou prejuízo líquido, como no caso de uma moeda honesta lançada infinitas vezes, onde o número de caras e coroas se equipara. Então de maneira óbvia, um jogo com probabilidade de vencer menor que a probabilidade de perder, é determinístico. Quanto maior o número de jogos, mais o cassino ganha seja qual for o jogo.

3.2 Cálculos Probabilísticos do *Roulette*

De forma geral, existem dois tipos de aposta no jogo *Roulette*, e são classificadas em:

aposta interna ou externa, para os cálculos probabilísticos será considerado cada um dos tipos de apostadores. Para o cálculo, será considerado que o jogador aposte \$1,00 em cada aposta, de forma que os resultados sejam normalizados em relação à unidade, e considerando a ordem como descrito no capítulo das regras do jogo, existem as apostas:

3.2.1 Direta

Para este caso, a aposta vencedora é premiada com +\$35,00 e ela é capaz de cobrir 1 número, então quaisquer apostas diretas têm probabilidade de ser sorteada como sendo $1/37$ ou $2{,}70\%$. Se o jogador apostar 37 vezes \$1,00 a expectativa teórica é que ele obtenha uma vitória sendo premiado no total com +\$35,00 e nesta aposta onde ele ganha, o valor apostado retorna, então o jogador recupera estes \$1,00 e ainda o prêmio, totalizando \$36,00 e comparando com o valor inicial, ele tem um prejuízo de \$1,00 o que caracteriza um prejuízo aproximado de \$0,0270 por jogo, ou seja, em outras palavras, para este tipo de aposta o retorno ao jogador é obtido pela relação de completeza

$$1 - 0{,}0270 \approx 97{,}3\%$$

O valor esperado para este tipo de aposta é

$$E[X] = \sum_{i=1}^{2} x_i p_i = \left(35 \cdot \frac{1}{37}\right) + \left(-1 \cdot \frac{36}{37}\right) = -\frac{1}{37} = -0{,}027$$

O desvio padrão é

$$\sigma(X) = \sqrt{E[X^2] - (E[X])^2}$$

$$= \sqrt{\left(\left((35)^2 \cdot \frac{1}{37}\right) + \left((-1)^2 \cdot \frac{36}{37}\right) - \left(-\frac{1}{37}\right)^2\right)} = 5{,}84$$

Olhando para a LGN na eq. (2.8.3) é de interesse saber qual o valor de n jogos que devem acontecer para que o valor das apostas convirja para o valor esperado com um determinado grau de confiança. Considerando o intervalo de confiança como sendo 95%, no fundo significa que o valor esperado observado esteja dentro de

$$E[X] \pm z_{\left(\frac{\alpha}{2}\right)} \cdot \frac{\sigma}{\sqrt{n}},$$

onde, $z_{\left(\frac{\alpha}{2}\right)}$ é o valor crítico da distribuição normal para um nível de confiança 95%, e para a distribuição normal padrão o valor é $1{,}96$ (da tabela de distribuição normal). O intervalo de confiança para o valor esperado é dado por

$$z_{\left(\frac{\alpha}{2}\right)} \cdot \frac{\sigma}{\sqrt{n}} = \epsilon$$

Perceba que ϵ é a margem de erro desejada, ou seja, o quanto que a média observada está próxima do valor esperado descrito pelo intervalo $[E[X] - \epsilon, E[X] + \epsilon]$. Então como o valor esperado é da ordem de 10^{-2}, um ϵ suficiente é $\epsilon = 0{,}01$

$$n = \left(\frac{1{,}96 \cdot 5{,}8378}{0{,}01}\right)^2 \approx 1.309.230$$

3.2.2 Dividir

Para este caso a aposta vencedora é premiada com +$17,00 e ela é capaz de cobrir 2 números, então quaisquer apostas divididas têm probabilidade de ser sorteada como sendo $2/37$ ou $5{,}40\%$. Se o jogador apostar 37 vezes $1,00 a expectativa teórica é que ele obtenha duas vitórias sendo premiado no total com +$34,00 e nestas apostas onde ele ganha, o valor apostado retorna, então como foram duas apostas vencedoras, o jogador recupera estes $2,00 e ainda o prêmio, totalizando $36,00 e comparando com o valor inicial, ele tem um prejuízo de $1,00 o que caracteriza um prejuízo aproximado de $0,0270 por jogo, em outras palavras, para este tipo de aposta o retorno ao jogador é de $1 - 0{,}0270 \approx 97{,}3\%$

O valor esperado para este tipo de aposta é

$$E[X] = \sum_{i=1}^{2} x_i p_i = \left(17 \cdot \frac{2}{37}\right) + \left(-1 \cdot \frac{35}{37}\right) = -\frac{1}{37} = -0{,}027$$

O desvio padrão é

$$\sigma(X) = \sqrt{E[X^2] - (E[X])^2}$$

$$= \sqrt{\left(\left((17)^2 \cdot \frac{1}{37}\right) + \left((-1)^2 \cdot \frac{35}{37}\right) - \left(-\frac{1}{37}\right)^2\right)} = 4{,}07$$

Seguindo a mesma análise e determinando o valor de n jogos necessários,

$$n = \left(\frac{1{,}96 \cdot 4{,}0702379}{0{,}01}\right)^2 \approx 636.432$$

3.2.3 Rua

Para este caso a aposta vencedora é premiada com +$11,00 e ela é capaz de cobrir 3 números, então quaisquer apostas em rua têm probabilidade de ser sorteada como sendo $3/37$ ou $8{,}11\%$. Se o jogador apostar 37 vezes $1,00 a expectativa teórica é que ele obtenha três vitórias sendo premiado no total com +$33,00 e nestas apostas onde ele ganha, o valor apostado retorna, então como foram três apostas vencedoras, o jogador recupera estes R$3,00 e ainda o

prêmio, totalizando \$36,00 e comparando com o valor inicial, ele tem um prejuízo de \$1,00 o que caracteriza um prejuízo aproximado de \$0,0270 por jogo, em outras palavras, para este tipo de aposta o retorno ao jogador é de $1 - 0{,}0270 \approx 97{,}3\%$

O valor esperado para este tipo de aposta é

$$E[X] = \sum_{i=1}^{2} x_i p_i = \left(11 \cdot \frac{3}{37}\right) + \left(-1 \cdot \frac{34}{37}\right) = -\frac{1}{37} = -0{,}027$$

O desvio padrão é

$$\sigma(X) = \sqrt{E[X^2] - (E[X])^2}$$

$$= \sqrt{\left(\left((11)^2 \cdot \frac{3}{37}\right) + \left((-1)^2 \cdot \frac{34}{37}\right) - \left(-\frac{1}{37}\right)^2\right)} = 3{,}27$$

Seguindo a mesma análise e determinando o valor de n jogos necessários,

$$n = \left(\frac{1{,}96 \cdot 3{,}27551511}{0{,}01}\right)^2 \approx 412.165$$

3.2.4 Canto

Para este caso a aposta vencedora é premiada com +\$8,00 e ela é capaz de cobrir 4 números, então quaisquer apostas em canto têm probabilidade de ser sorteada como sendo $4/37$ ou $10{,}81\%$. Se o jogador apostar 37 vezes \$1,00 a expectativa teórica é que ele obtenha quatro vitórias sendo premiado no total com +\$32,00 e nestas apostas onde ele ganha, o valor apostado retorna, então como foram duas apostas vencedoras, o jogador recupera estes \$4,00 e ainda o prêmio, totalizando \$36,00 e comparando com o valor inicial, ele tem um prejuízo de \$1,00 o que caracteriza um prejuízo aproximado de \$0,0270 por jogo, em outras palavras, para este tipo de aposta o retorno ao jogador é de $1 - 0{,}0270 \approx 97{,}3\%$

O valor esperado para este tipo de aposta é

$$E[X] = \sum_{i=1}^{2} x_i p_i = \left(8 \cdot \frac{4}{37}\right) + \left(-1 \cdot \frac{33}{37}\right) = -\frac{1}{37} = -0{,}027$$

O desvio padrão é

$$\sigma(X) = \sqrt{E[X^2] - (E[X])^2}$$

$$= \sqrt{\left(\left((8)^2 \cdot \frac{4}{37}\right) + \left((-1)^2 \cdot \frac{33}{37}\right) - \left(-\frac{1}{37}\right)^2\right)} = 2{,}73$$

Seguindo a mesma análise e determinando o valor de n jogos necessários,

$$n = \left(\frac{1{,}96 \cdot 2{,}731067}{0{,}01}\right)^2 \approx 300.032$$

3.2.5 Linha

Para este caso a aposta vencedora é premiada com +$5,00 e ela é capaz de cobrir 6 números, então quaisquer apostas em linha têm probabilidade de ser sorteada como sendo $6/37$ ou $16{,}21\%$. Se o jogador apostar 37 vezes $1,00 a expectativa teórica é que ele obtenha duas vitórias sendo premiado no total com +$30,00 e nestas apostas onde ele ganha, o valor apostado retorna, então como foram seis apostas vencedoras, o jogador recupera estes $6,00 e ainda o prêmio, totalizando $36,00 e comparando com o valor inicial, ele tem um prejuízo de $1,00 o que caracteriza um prejuízo aproximado de $0,0270 por jogo, em outras palavras, para este tipo de aposta o retorno ao jogador é de $1 - 0{,}0270 \approx 97{,}3\%$

O valor esperado para este tipo de aposta é

$$E[X] = \sum_{i=1}^{2} x_i p_i = \left(5 \cdot \frac{6}{37}\right) + \left(-1 \cdot \frac{31}{37}\right) = -\frac{1}{37} = -0{,}027$$

O desvio padrão é

$$\sigma(X) = \sqrt{E[X^2] - (E[X])^2}$$

$$= \sqrt{\left((5)^2 \cdot \frac{6}{37}\right) + \left((-1)^2 \cdot \frac{31}{37}\right) - \left(-\frac{1}{37}\right)^2} = 2{,}21$$

Seguindo a mesma análise e determinando o valor de n jogos necessários,

$$n = \left(\frac{1{,}96 \cdot 2{,}2115970}{0{,}01}\right)^2 \approx 187.899$$

3.2.6 Coluna e Dúzia

Para este caso a aposta vencedora é premiada com +$2,00 e ela é capaz de cobrir 12 números, então quaisquer apostas em coluna ou dúzia, têm probabilidade de ser sorteada como sendo $12/37$ ou $32{,}43\%$. Se o jogador apostar 37 vezes $1,00 a expectativa teórica é que ele obtenha doze vitórias sendo premiado no total com +$24,00 e nestas apostas onde ele ganha, o valor apostado retorna, então como foram doze apostas vencedoras, o jogador recupera estes $12,00 e ainda o prêmio, totalizando $36,00 e comparando com o valor inicial, ele tem um prejuízo de $1,00 o que caracteriza um prejuízo aproximado de $0,0270 por jogo, em outras palavras, para este tipo de aposta o retorno ao jogador é de $1 - 0{,}0270 \approx 97{,}3\%$

O valor esperado para este tipo de aposta é

$$E[X] = \sum_{i=1}^{2} x_i p_i = \left(2 \cdot \frac{12}{37}\right) + \left(-1 \cdot \frac{25}{37}\right) = -\frac{1}{37} = -0{,}027$$

O desvio padrão é

$$\sigma(X) = \sqrt{E[X^2] - (E[X])^2}$$

$$= \sqrt{\left(\left((2)^2 \cdot \frac{12}{37}\right) + \left((-1)^2 \cdot \frac{25}{37}\right) - \left(-\frac{1}{37}\right)^2\right)} = 1{,}40$$

Seguindo a mesma análise e determinando o valor de n jogos necessários,

$$n = \left(\frac{1{,}96 \cdot 1{,}4043655}{0{,}01}\right)^2 \approx 75.766$$

3.2.7 Vermelho/Preto, Par ou Ímpar e 1-18/19-36

Para este caso a aposta vencedora é premiada com +$1,00 e ela é capaz de cobrir 18 números, então quaisquer apostas divididas têm probabilidade de ser sorteada como sendo $18/37$ ou $48{,}65\%$. Se o jogador apostar 37 vezes $1,00 a expectativa teórica é que ele obtenha dezoito vitórias sendo premiado no total com +$18,00 e nestas apostas onde ele ganha, o valor apostado retorna, então como foram doze apostas vencedoras, o jogador recupera estes $18,00 e ainda o prêmio, totalizando $36,00 e comparando com o valor inicial, ele tem um prejuízo de $1,00 o que caracteriza um prejuízo aproximado de $0,0270 por jogo, em outras palavras, para este tipo de aposta o retorno ao jogador é de $1 - 0{,}0270 \approx 97{,}3\%$

O valor esperado para este tipo de aposta é

$$E[X] = \sum_{i=1}^{2} x_i p_i = \left(1 \cdot \frac{18}{37}\right) + \left(-1 \cdot \frac{19}{37}\right) = -\frac{1}{37} = -0{,}027$$

O desvio padrão é

$$\sigma(X) = \sqrt{E[X^2] - (E[X])^2}$$

$$= \sqrt{\left(\left((1)^2 \cdot \frac{18}{37}\right) + \left((-1)^2 \cdot \frac{19}{37}\right) - \left(-\frac{1}{37}\right)^2\right)} = 1$$

Seguindo a mesma análise e determinando o valor de n jogos necessários,

$$n = \left(\frac{1{,}96 \cdot 0{,}99963470}{0{,}01}\right)^2 \approx 38.388$$

3.3 Cálculos Probabilísticos do *Blackjack*

Os cálculos probabilísticos no *Blackjack* apresentam uma complexidade considerável, devido à interdependência das cartas e às múltiplas variáveis envolvidas. Diferente de outros jogos, como a *Roulette*, o *Blackjack* não se enquadra na teoria dos jogos tradicional, uma vez que o *dealer* não toma decisões independentes, seguindo regras predefinidas. Isso o torna, de certa forma, um "jogo de uma pessoa", onde as decisões do jogador influenciam diretamente o resultado.

O grande apelo do *Blackjack* está no fato de ser um dos poucos jogos em que o jogador pode obter uma expectativa matemática positiva, especialmente quando estratégias específicas, como a contagem de cartas, são aplicadas. Ao contrário de jogos puramente aleatórios, como a *Roulette*, o *Blackjack* tem "memória", já que as cartas removidas do baralho influenciam as próximas rodadas, e "consciência", pois jogadas mal executadas podem ser penalizadas severamente. Além disso, a habilidade mental e a capacidade de retenção do jogador afetam o desempenho, tornando o *Blackjack* um jogo onde a habilidade pode prevalecer.

Para simplificar os cálculos iniciais, será considerado a hipótese de que todas as cartas ainda estão no baralho, tratando-o como um baralho infinito, onde a probabilidade de cada carta ser sorteada é de 1/13, pode-se pensar também que sortear uma carta seja o mesmo evento que lançar um dado justo com 13 lados. Essa aproximação é válida ao calcular as probabilidades iniciais do jogo, antes de qualquer carta ser retirada. No entanto, no contexto de um jogo com 8 baralhos, as probabilidades variam conforme as cartas são removidas, tornando necessária a simulação Monte Carlo para cálculos mais precisos. Contudo, inicialmente serão apresentados cálculos com a hipótese do baralho infinito e cartas equiprováveis.

Primeiramente, é interessante determinar qual é a probabilidade de obter uma pontuação inicial específica. Todos os possíveis resultados situam-se no intervalo discreto de 4 a 21. O fato de o Ás valer tanto 1 quanto 11, e as cartas 10, *J*, *Q* e *K* valerem 10, cria um cenário onde a análise combinatória se torna menos trivial, exigindo a contagem cuidadosa de cada caso. Ainda assim, uma constante permanece: a combinação de duas cartas, entre 13 cartas equiprováveis, gera um total de 13*13 = 169. Portanto, sabe-se de antemão que o espaço amostral é de 169, e as probabilidades podem ser calculadas a partir disso.

Para o valor 4, há 1 possibilidade, que é [2,2], e a probabilidade como sendo

$$P(4) = \frac{1}{13 \cdot 13} = 0{,}592\%$$

Para o valor 5, há 2 possibilidades, que são [2,3] e [3,2], e a probabilidade como sendo

$$P(5) = \frac{2}{13 \cdot 13} = 1{,}18\%$$

Para o valor 6, há 3 possibilidades, que são [4,2], [2,4] e [3,3], e a probabilidade como sendo

$$P(6) = \frac{3}{13 \cdot 13} = 1{,}77\%$$

Para o valor 7, há 4 possibilidades, que são [2,5], [5,2], [3,4] e [4,3], e a probabilidade como sendo

$$P(7) = \frac{4}{13 \cdot 13} = 2{,}36\%$$

Para o valor 8, há 5 possibilidades, que são [4,4], [2,6], [6,2], [5,3] e [3,5], e a probabilidade como sendo

$$P(8) = \frac{5}{13 \cdot 13} = 2{,}96\%$$

Para o valor 9, há 6 possibilidades, que são [2,7], [7,2], [3,6], [6,3], [4,5] e [5,4], e a probabilidade como sendo

$$P(9) = \frac{6}{13 \cdot 13} = 3{,}55\%$$

Para o valor 10, há 7 possibilidades, que são [5,5], [4,6], [6,4], [3,7], [7,3], [2,8] e [8,2], e a probabilidade como sendo

$$P(10) = \frac{7}{13 \cdot 13} = 4{,}14\%$$

Para o valor 11, há 8 possibilidades, que são [5,6], [6,5], [4,7], [7,4], [3,8], [8,3], [2,9] e [9,2], e a probabilidade como sendo

$$P(11) = \frac{8}{13 \cdot 13} = 4{,}73\%$$

Para o valor 12, há 16 possibilidades, que são [6,6], [5,7], [7,5], [8,4], [4,8], [3,9], [9,3], [A,A], [10,2], [2,10], [2,J], [J,2], [Q,2], [2,Q], [K,2] e [2,K], e a probabilidade como sendo

$$P(12) = \frac{16}{13 \cdot 13} = 9{,}47\%$$

Para o valor 13, há 16 possibilidades, que são [A,2], [2,A], [3,10], [10,3], [J,3], [3,J], [Q,3], [3,Q], [K,3], [3,K], [4,9], [9,4], [5,8], [8,5], [6,7] e [7,6], e a probabilidade como sendo

$$P(13) = \frac{16}{13 \cdot 13} = 9{,}47\%$$

Para o valor 14, há 15 possibilidades, que são [A,3], [3,A], [4,10], [10,4], [J,4], [4,J], [Q,4], [4,Q], [K,4], [4,K], [5,9], [9,5], [6,8], [8,6] e [7,7], e a probabilidade como sendo

$$P(14) = \frac{15}{13 \cdot 13} = 8{,}87\%$$

Para o valor 15, há 14 possibilidades, que são [A,4], [4,A], [5,10], [10,5], [J,5], [5,J], [Q,5], [5,Q], [K,5], [5,K], [6,9], [9,6], [7,8] e [8,7], e a probabilidade como sendo

$$P(15) = \frac{14}{13 \cdot 13} = 8{,}28\%$$

Para o valor 16, há 13 possibilidades, que são [A,5], [5,A], [6,10], [10,6], [J,6], [6,J], [Q,6], [6,Q], [K,6], [6,K], [7,9], [9,7] e [8,8], e a probabilidade como sendo

$$P(16) = \frac{13}{13 \cdot 13} = 7{,}69\%$$

Para o valor 17, há 12 possibilidades, que são [A,6], [6,A], [7,10], [10,7], [J,7], [7,J], [Q,7], [7,Q], [K,7], [7,K], [8,9] e [9,8], e a probabilidade como sendo

$$P(17) = \frac{12}{13 \cdot 13} = 7{,}10\%$$

Para o valor 18, há 11 possibilidades, que são [A,7], [7,A], [8,10], [10,8], [J,8], [8,J], [Q,8], [8,Q], [K,8], [8,K] e [9,9], e a probabilidade como sendo

$$P(18) = \frac{11}{13 \cdot 13} = 6{,}51\%$$

Para o valor 19, há 12 possibilidades, que são [A,8], [8,A], [9,10], [10,9], [J,9], [9,J], [Q,9], [9,Q], [K,9] e [9,K], e a probabilidade como sendo

$$P(19) = \frac{12}{13 \cdot 13} = 7{,}10\%$$

Para o valor 20, há 12 possibilidades, que são [A,9], [9,A], [10,10], [J,J], [Q,Q], [K,K], [10,J], [J,10], [10,Q], [Q,10], [K,10], [10,K], [J,Q], [Q,J], [J,K], [K,J], [Q,K] e [K,Q], e a probabilidade como sendo

$$P(20) = \frac{12}{13 \cdot 13} = 7{,}10\%$$

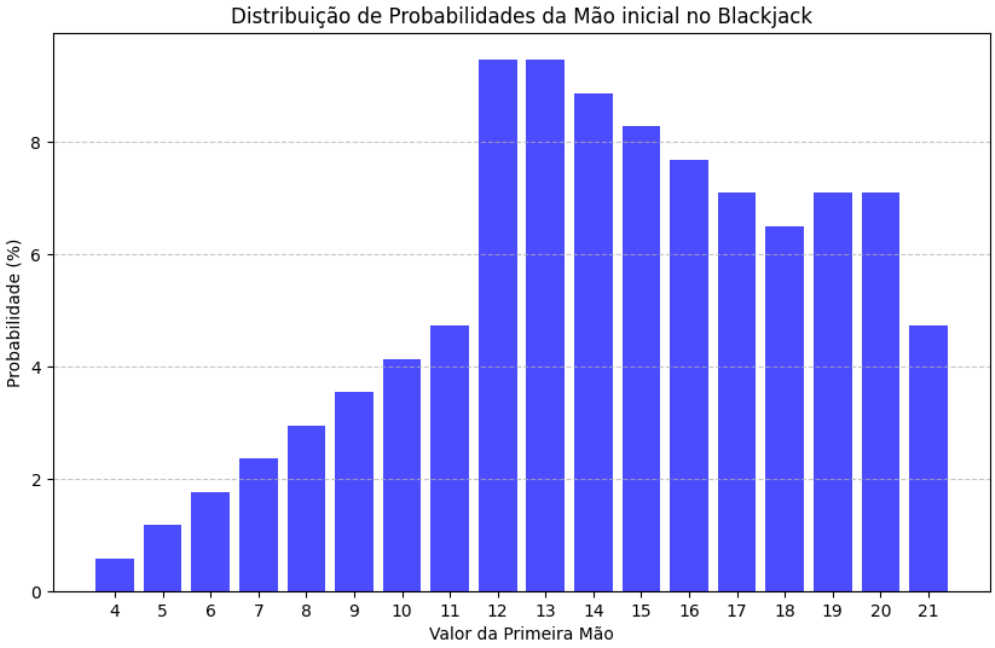
Para o valor 21, há 8 possibilidades, que são [A,10], [10,A], [A,J], [J,A], [A,Q], [Q,A], [A,K] e [K,A], e a probabilidade como sendo

$$P(21) = \frac{8}{13 \cdot 13} = 4{,}73\%$$

A probabilidade de a carta virada para cima do crupiê ser qualquer uma das 13 possíveis é sempre 1/13, ou 7,69%. No entanto, como há quatro cartas que representam o valor 10 (10, *J*, *Q*, *K*), a probabilidade de o jogador enfrentar uma dessas cartas é de 4/13, ou 30,8%.

A distribuição de probabilidade também é expressa no gráfico 3.

Gráfico 2: Distribuição de probabilidades da mão inicial de *Blackjack*.



Fonte: Autor.

Ao considerar a probabilidade de o jogador ou o crupiê "estourar" (exceder 21 pontos), faz-se necessário limitar a análise aos casos em que a mão possui 12 pontos ou mais. Para uma mão com 12 pontos, a probabilidade de exceder 21 ao solicitar mais uma carta é de 4/13; para 13 pontos, essa probabilidade sobe para 5/13, e assim sucessivamente. Dessa forma, para um valor x de pontos, a probabilidade de estourar é dada por

$$P(estourar\ com\ x) = \frac{x-8}{13}$$

Para $12 \geq x \geq 20$.

Tabela 8: Probabilidade de estourar.

Pontos da mão	Probabilidade de estourar (%)
12	30,8
13	38,5
14	46,1
15	53,8
16	61,5
17	69,2

18	76,9
19	84,6
20	92,3

Fonte: Autor

Agora para o crupiê quando tem a sua carta virada para cima com valor 6 ou mais, a probabilidade de ele obter um valor maior ou igual a 17 revelando apenas a sua outra carta é

Tabela 9: Probabilidade de crupiê ter no mínimo 17 na primeira mão.

Carta do crupiê	Probabilidade de obter 17 ou mais (%)
6	7,69
7	38,5
8	46,1
9	53,8
10, J, Q ou K	61,5
A	69,2

Fonte: Autor

Para os casos onde há valores de mãos com pontuações *soft*, é diferente, então encarando a probabilidade de obter um valor entre o intervalo $17 \leq x \leq 21$

Tabela 10: Probabilidade de obter um valor entre 17 e 21 com uma mão *soft*.

Cartas da mão	Probabilidade de obter $17 \leq x \leq 21$ (%)
A-2	38,5
A-3	38,5
A-4	38,5
A-5	38,5
A-6	30,8
A-7	23,1
A-8	15,4
A-9	7,69

Fonte: Autor

Encarando a probabilidade de transformar uma mão com pontuação *soft* em *hard*, ou seja, obter um valor maior que 21.

Tabela 11: Probabilidade de transformar uma mão *soft* em *hard*.

Cartas da mão	Probabilidade de obter $x > 21$ (%)
A-2	38,5
A-3	46,1
A-4	53,8
A-5	61,5
A-6	69,2
A-7	76,9
A-8	84,6
A-9	92,3

Fonte: Autor

Esses resultados são válidos sob a hipótese de cartas equiprováveis em um baralho infinito, e oferecem uma boa base para compreender as dinâmicas do jogo.

3.4 Cálculos Probabilísticos das Apostas Laterais

Para o cálculo probabilístico analítico, será considerado apenas o primeiro jogo, ou seja, a primeira mão do primeiro jogador. Isso porque, a cada nova rodada, ao retirar uma carta do baralho, o tamanho do espaço amostral (composto por 8 baralhos completos) é alterado. Essa mudança cria uma dependência clara entre a probabilidade das cartas e a quantidade e quais cartas ainda restam no baralho, já que as cartas utilizadas são descartadas e não reaparecem até o momento do reembaralhamento.

3.4.1 Pares Perfeitos

Para calcular a probabilidade P de um evento desejado, utiliza-se a relação de probabilidade básica

$$P = \frac{N_{cf}}{N_{cp}}$$

onde, N_{cf} é o número de casos favoráveis e N_{cp} é o número de casos possíveis. No contexto deste cálculo, considera-se um conjunto de 8 baralhos de 52 cartas cada, totalizando 416 cartas.

O número total de combinações possíveis ao retirar duas cartas é dado por

$$C_n^r = \frac{n!}{r!\,(n-r)!} \equiv \binom{n}{r}$$

onde C_n^r representa o número de combinações de n objetos tirados r de cada vez, para duas cartas ($r = 2$) retiradas de 416 cartas, o cálculo resulta em

$$C_{416}^2 = \frac{416!}{2!\,(416-2)!} \equiv \binom{416}{2} = 86.320\ combinações$$

Para o cálculo dos casos favoráveis, considera-se que em cada baralho há 4 naipes e 13 valores diferentes (Ás, 2, 3, ..., K). No caso dos "Pares Perfeitos", apenas as combinações de cartas do mesmo valor e naipe são levadas em conta. Como há 4 naipes diferentes (Copas, Ouros, Paus e Espadas), existem 4 combinações possíveis de pares de mesmo naipe:

- (Copas, Copas)
- (Ouros, Ouros)
- (Paus, Paus)
- (Espadas, Espadas)

Dado que há 8 baralhos, existem 8 cartas de cada valor e naipe. Para uma combinação específica, como duas cartas de Ás de Copas, o número de maneiras de combinar 2 cartas de 8 é dado por

$$\binom{8}{2} = \frac{8!}{2!\,(8-2)!} = \frac{8 \cdot 7}{2!} = 28$$

Assim, para cada valor de carta (Á s, 2, ..., K), há 4 combinações de pares de naipes iguais, e para cada uma dessas combinações há 28 maneiras de selecionar as cartas. O número total de casos favoráveis é então

$$13 \cdot 4 \cdot 28 = 1.456$$

Finalmente, a probabilidade de obter um par perfeito na primeira rodada de *Blackjack*

$$P = \frac{1.456}{86.320} \approx 1{,}69\%$$

Portanto, a probabilidade de formar um par perfeito em uma jogada de *Blackjack* é de aproximadamente 1,69% e em termos de frequência, o jogador tende a ter um par perfeito aproximadamente a cada 60 jogos.

3.4.2 Pares Coloridos

Para os pares coloridos apenas os naipes com cores diferentes são considerados, e observe que os naipes Copas e Ouros possuem a coloração vermelha enquanto Paus e Espadas

possuem a coloração preta, sendo assim, existem obviamente 2 combinações de cores iguais possíveis:

- (Ouros, Copas)
- (Paus, Espadas)

E como há 8 baralhos, há 8 cartas de cada valor e naipe, por exemplo, há 8 Ás de Copas, 8 Ás de Ouros, 8 Ás de Paus e 8 Ás de Espadas. Agora considerando que em uma combinação específica de cores, por exemplo, Copas e Paus. Para um valor específico como Ás, há 8 Ás de Copas e 8 Ás de Paus e o número de maneiras de ser combinado 1 Ás de Copas e 1 Ás de Paus são $8 \cdot 8 = 64$ e agora com todas essas considerações feitas, há

$$13 \cdot 2 \cdot 64 = 1.664$$

E sendo assim a probabilidade de se obter um Par Colorido é

$$\frac{1.664}{86.320} = 1{,}93\%$$

3.4.3 Pares Mistos

Para determinar a probabilidade de formar um par misto, utiliza-se a relação de probabilidade básica já citada, e o número de casos favoráveis. Já foi calculado que o número total de combinações possíveis ao retirar duas cartas de 416 cartas (correspondentes a 8 baralhos) é 86.320 combinações. Para o cálculo das situações favoráveis é considerado que há 8 baralhos, cada um com 52 cartas, totalizando 416 cartas. Em cada baralho, há 4 naipes e cada naipe possui 13 valores diferentes, porém note que neste caso específico, apenas os naipes com cores diferentes são considerados, e observe que os naipes Copas e Ouros possuem a coloração vermelha enquanto Paus e Espadas possuem a coloração preta, sendo assim, existem obviamente 4 combinações de preto e vermelho possíveis:

- (Ouros, Paus)
- (Ouros, Espadas)
- (Copas, Paus)
- (Copas, Espadas)

Portanto, para cada valor de carta, Ás, 2, ..., K, há 4 combinações de cores diferentes.

E como há 8 baralhos, há 8 cartas de cada valor e naipe, por exemplo, há 8 Ás de Copas, 8 Ás de Ouros, 8 Ás de Paus e 8 Ás de Espadas. Agora vamos considerar que em uma combinação específica de cores, por exemplo, Copas e Paus. Para um valor específico como Ás, há 8 Ás de Copas e 8 Ás de Paus e o número de maneiras de combinar 1 Ás de Copas

e 1 Ás de Paus são

$$8 \cdot 8 = 64$$

E agora com todas essas considerações feitas, há

$$13 \cdot 4 \cdot 64 = 3.328$$

E sendo assim a probabilidade de se obter um Par Misto é

$$\frac{3.328}{86.320} = 3{,}85\%$$

3.4.4 Flush

Uma combinação Flush ocorre quando as duas cartas do jogador, juntamente com a carta aberta do crupiê, possuem naipes idênticos, mas números distintos. Para calcular a probabilidade dessa combinação, iniciando da definição do espaço amostral total para um jogo com 8 baralhos, que é dado por,

$$C_{416}^{3} = \frac{416!}{3!\,(416-3)!} \equiv \binom{416}{3} = 11.912.160\ combinações$$

Nosso objetivo é encontrar, dentro desse conjunto, as combinações que resultam em um Flush.

Para que haja um Flush, as três cartas (as duas do jogador e a do crupiê) devem ser do mesmo naipe, mas sem formar uma sequência numérica (como 2, 3 e 4), que configuraria um Straight Flush. Como cada baralho contém 13 cartas de cada naipe, em um conjunto de 8 baralhos há 104 cartas de um mesmo naipe. O número de maneiras de selecionar 3 cartas do mesmo naipe em um baralho com 104 cartas é dado por,

$$\binom{104}{3} = \frac{104!}{3!\,(104-3)!} = \frac{104 \cdot 103 \cdot 102}{3!} = 182.104$$

No entanto, é necessário descontar as combinações que resultam em uma sequência (Straight Flush).

Existem 12 sequências possíveis por naipe (como A, 2, 3; 2, 3, 4; ..., Q, K, A), e com 8 baralhos, há

$$12 \cdot 8 \cdot 8 \cdot 8 = 6.144$$

Assim, o número de combinações possíveis para um Flush em um único naipe, excluindo as sequências, é

$$182.104 - 6.144 = 175.960\ combinações$$

Considerando os quatro naipes, o número total de combinações possíveis de Flush é,

$$175.960 \cdot 4 = 703.840\ combinações$$

Portanto, a probabilidade de obter um Flush com 8 baralhos é

$$\frac{703.840}{11.912.160} = 5{,}91\%$$

3.4.5 Straight

Uma combinação *Straight* ocorre quando as duas cartas do jogador, juntamente com a carta do crupiê, formam uma sequência numérica (como 2, 3, 4), com naipes diferentes entre si. Para que a combinação seja considerada um *Straight*, não é necessário que cada carta tenha um naipe distinto, mas as três cartas não podem ser do mesmo naipe, pois isso configuraria um *Straight Flush*.

Conforme discutido anteriormente, o número total de combinações possíveis de três cartas em um conjunto de 8 baralhos, contendo 416 cartas, é $11.912.160$. Nosso objetivo agora é determinar quantas dessas combinações resultam em um Straight. Para uma sequência específica, como *J*, *Q*, *K*, cada valor de carta (*J*, *Q* ou *K*) pode estar presente em um dos quatro naipes, e, com 8 baralhos, há 8 cartas de cada valor por naipe. O número total de combinações possíveis para uma única sequência em 8 baralhos é dado por:

$$(4 \cdot 8)^3 = 32.768$$

No entanto, dentro dessas combinações, algumas formarão um *Straight Flush*, que precisa ser subtraído. Como calculado anteriormente, o número de combinações de *Straight Flush* para 8 baralhos é

$$32.768 - 6.144 = 26.624\ combinações$$

Como existem 12 sequências possíveis (como A, 2, 3; 2, 3, 4; ..., *Q*, *K*, *A*), o número total de combinações que resultam em um *Straight* é

$$\frac{12 \cdot 26.624}{11.912.160} = \frac{319.488}{11.912.160} = 2{,}68\%$$

3.4.6 Three Of A Kind

Uma combinação *Three of a Kind* ocorre quando as duas cartas do jogador, combinadas com a carta do crupiê, possuem o mesmo valor numérico, mas não são todas do mesmo naipe. Não é necessário que os três naipes sejam diferentes entre si, mas as três cartas não podem ser do mesmo naipe.

Como mencionado anteriormente, o espaço amostral total para um conjunto de 8 baralhos, contendo 416 cartas, é 11.912.160 combinações. Agora, para calcular o número de combinações que resultam em um *Three of a Kind*, considerando um valor específico, por

exemplo, Ás. Em um único baralho, existem 4 cartas Ás. Com 8 baralhos, há 32 cartas Ás. O número de maneiras de selecionar três Áses entre essas 32 cartas é dado por:

$$\binom{32}{3} = \frac{32!}{3!\,(32-3!)} = \frac{32 \cdot 31 \cdot 30}{3!} = 4.960\ combinações$$

No entanto, entre essas combinações, estão incluídos os casos em que as três cartas são do mesmo naipe, o que configura um *Suited Trips*, e precisa-se subtrair essas combinações. Para calcular as combinações onde todas as três cartas são do mesmo naipe, observe que, com 8 baralhos, existem 8 cartas de cada valor para cada naipe. O número de maneiras de selecionar três cartas do mesmo valor e naipe é dado por

$$\binom{8}{3} = \frac{8!}{3!\,(8-3)!} = \frac{8 \cdot 7 \cdot 6}{3!} = 56$$

Como existem 4 naipes, o total de combinações de *Suited Trips* é

$$56 \cdot 4 = 224\ combinações$$

Subtraindo este valor das 4960 combinações, tem-se 4736 combinações. Finalmente, multiplicando esse resultado pelo número de valores de cartas (13 valores distintos, de Ás a Rei) para obter o total de combinações de *Three of a Kind*:

$$13 \cdot 4.736 = 61.568$$

E, portanto, a probabilidade de se obter um *Three Of A Kind* é

$$\frac{61.568}{11.912.160} = 0{,}517\%$$

3.4.7 Straight Flush

Uma combinação *Straight Flush* é quando suas duas cartas combinadas com a do crupiê possuem uma sequência do mesmo naipe. Como mencionado anteriormente, o número de combinações totais são $11.912.160$, e o número de casos favoráveis para um naipe é 6.144 então para 4 naipes tem-se 24.576 combinações de interesse, então a probabilidade de se obter *Straight Flush* é dada por

$$\frac{24.576}{11.912.160} = 0{,}206\%$$

3.4.8 Suited Trips

Agora note que em 8 baralhos existem 8 cartas de cada tipo, por exemplo, existem 8 cartas Ás de Copas, e para formar a combinação *Suited Trips* escolhendo um valor, por exemplo o que já foi citado, Ás de Copas, e como há 8 cartas dessa no monte, precisa-se escolher 3 delas,

o número de escolher 3 delas sendo que existem 8, é dado por

$$\binom{8}{3} = \frac{8!}{3!\,(8-3)!} = 56\ combinações$$

Para cada valor de cada naipe, agora existem 13 valores diferentes, *A*, 2, 3, ..., *J*, *Q*, *K* e existem 4 naipes, portanto o total de combinações de *Suited Trips* é

$$13 \cdot 4 \cdot 56 = 2.912\ combinações$$

E dito isso, obtém-se então a relação

$$\frac{2.912}{11.912.160} = 0{,}0244\%$$

3.5 Implementação Computacional em Linguagem *Python* da Simulação Monte Carlo Para Determinar a Probabilidade de Vitória do Jogador com uma Estratégia Fixa e Determinada

A função *criar_baralho()* é responsável por criar o baralho para o jogo de *Blackjack*, essa função é essencial para garantir que o ambiente de jogo reflita as condições reais encontradas nos cassinos, onde múltiplos baralhos são utilizados para aumentar a complexidade e a aleatoriedade, e neste caso específico, o ato de gerar números aleatórios que é necessário para que a simulação Monte Carlo aconteça, é feita pelo embaralhamento do baralho.

Inicialmente, a função define a quantidade de baralhos a serem criados, o que é representado por uma variável que, no exemplo, define oito baralhos. Esse valor é típico de partidas de *Blackjack* tanto em cassinos online quanto presencial, o que tem como objetivo dificultar a contagem de cartas e aumentar a imprevisibilidade do jogo.

A criação do baralho considera dois aspectos fundamentais: os valores das cartas e seus respectivos naipes. Os valores incluem números que vão de 2 a 10, além das figuras (Valete, Dama e Rei) e do Ás, totalizando as cartas convencionais usadas no jogo de *Blackjack*. Essas cartas são associadas aos quatro naipes (Ouros, Copas, Espadas e Paus), resultando na formação de todas as combinações possíveis de valor e naipe.

Para garantir que a simulação utilize múltiplos baralhos, o código replica cada carta de cada naipe pelo número de baralhos definido inicialmente, formando assim um grande conjunto de cartas. Após a criação do baralho, é aplicado um processo de embaralhamento utilizando a função *shuffle*, que embaralha as cartas de maneira aleatória. Esse embaralhamento é absolutamente necessário, pois busca simular a aleatoriedade encontrada em um ambiente de cassino real, onde as cartas são constantemente misturadas antes de serem distribuídas aos jogadores.

A função *cortar_baralho(baralho)* é responsável por calcular a posição do corte no baralho simula uma prática comum em cassinos, onde o baralho é cortado antes da distribuição das cartas para aumentar a imprevisibilidade do jogo. Esse processo de corte, normalmente feito pelo profissional responsável pelo embaralhamento, ou até mesmo pelo crupiê ou em cassinos físicos, pelo jogador, é parte essencial do *Blackjack* e visa introduzir uma camada extra de aleatoriedade no jogo, dificultando qualquer tentativa de prever a ordem das cartas.

Para simular esse procedimento de forma precisa, a função se baseia em uma distribuição normal (gaussiana). O corte é calculado a partir da posição central do baralho, o que faz sentido dado que, em situações reais, o corte geralmente ocorre próximo ao centro, mas nunca de maneira exata. Para representar essa variação, a função usa a posição média do baralho como referência, mas aplica um desvio padronizado para que o corte seja feito em um ponto levemente aleatório, mas ainda dentro de uma faixa razoável ao redor do centro.

O uso da distribuição gaussiana permite simular esse comportamento natural do corte, oferecendo uma variação controlada em torno da metade do baralho. A escolha de um desvio padrão pequeno (nesse caso, 2) faz com que o corte raramente ocorra muito longe do centro, mas ainda possibilita alguma aleatoriedade para imitar o que acontece em situações práticas. Isso é fulcral para a simulação, pois garante que o embaralhamento das cartas, seguido pelo corte, contribua para um maior nível de imprevisibilidade no jogo.

Além disso, a função inclui uma verificação para garantir que a posição do corte esteja sempre dentro dos limites válidos do baralho. Isso evita que o corte seja realizado fora do intervalo de cartas existente, um cuidado importante para garantir a consistência e precisão da simulação. Sem esse ajuste, o corte poderia ocorrer em posições inválidas, comprometendo a validade da simulação.

Ao final, a função retorna à posição exata onde o corte será feito, assegurando que o processo seja conduzido de forma controlada, porém suficientemente aleatória para refletir a realidade de uma mesa de *Blackjack*. A introdução do corte no baralho antes de cada rodada de simulação é um aspecto importante do método de Monte Carlo, pois reforça o nível de aleatoriedade necessário para que as simulações capturem a complexidade do jogo real. Essa abordagem ajuda a gerar resultados mais confiáveis e representativos das probabilidades envolvidas nas estratégias de jogo.

A função *valor_mao(mao)* é responsável por avaliar o valor de uma mão de *Blackjack*, levando em consideração a soma total das cartas e o tratamento especial dado ao Ás. No *Blackjack*, o Ás pode valer 1 ou 11. A função calcula a soma inicial das cartas, atribuindo 11 ao Ás, e verifica se a soma ultrapassa 21. Se isso ocorrer, o valor do Ás é reduzido para 1, evitando

que a mão perca automaticamente. Esse ajuste garante que o jogador aproveite ao máximo a flexibilidade oferecida pelo Ás no jogo, otimizando suas chances de vitória. Além disso, existe uma variável que é responsável por contar o número de Áses presentes em uma mão, quando o número de Áses da mão é maior que 1, e o jogador não excede 21 o Ás vale 11, se o valor da mão exceder 21 e o número de Áses for maior que 1, então o valor do Ás passa a ser 1 e o número de Áses é reduzido em 1.

Além disso, outra função *num_ases(mao)* é responsável por contar o número de Áses presentes em uma mão, o que é essencial para estratégias mais avançadas no *Blackjack*. Essa contagem, junto com o cálculo do valor da mão, auxilia na decisão do jogador de pedir ou não mais cartas, dependendo do número de Áses e do valor atual da mão.

A função *verificar_Blackjack(mao)* é responsável por verificar se o jogador tem um *Blackjack* natural, ou seja, uma combinação de um Ás com uma carta de valor 10 (10, Valete, Dama ou Rei) nas duas primeiras cartas. O *Blackjack* natural resulta em uma vitória automática, a menos que o crupiê também tenha *Blackjack*, o que resultaria em um empate. A verificação de *Blackjack* é um passo cabal na simulação, pois influencia diretamente o resultado da rodada e, consequentemente, as estatísticas das simulações.

A função *verificar_pares_perfeitos(mao)* que verifica a presença de pares perfeitos em uma mão compara as duas cartas iniciais do jogador entre si, se tiverem números e naipes iguais, o jogador tem um par perfeito, se os números forem iguais, naipes diferentes e cores dos naipes iguais, o jogador tem um par colorido e se o jogador possui duas cartas com números iguais, naipes diferentes e cores dos naipes diferentes, o jogador tem um par colorido e se o jogador tem números iguais, naipes diferentes e de cores diferentes ele tem um par misto.

Outro aspecto coberto no código é a verificação de mãos especiais conhecidas como "21+3", uma aposta lateral comum no *Blackjack*. As funções que realizam essa verificação são *valor_numerico(carta)*, *valor_numerico(valor)* e *verificar_21_mais_3(mao, carta_crupie)* avalia se a mão do jogador, combinada com a carta virada do crupiê, forma uma das combinações possíveis, como "*Suited Trips*", "*Straight Flush*", "*Three of a Kind*", "*Straight*" ou "*Flush*".

A função principal *jogar_Blackjack(jogador, crupie, baralho, resultados, ks)* que conduz o fluxo de uma rodada completa do jogo é fundamental para determinar os resultados de cada partida. Essa função recebe como parâmetros as mãos iniciais do jogador e do crupiê, o baralho utilizado, uma estrutura que armazena os resultados das partidas e um conjunto de parâmetros estratégicos, chamados de "ks", que guiam as decisões do jogador.

Inicialmente, a função calcula o valor das mãos do jogador e do crupiê, bem como

verifica se há combinações que satisfaçam as apostas laterais, como "pares perfeitos" ou a aposta "21+3". Essas verificações são importantes para contabilizar dados relevantes para as estatísticas da simulação referentes às apostas laterais.

Em seguida, a função verifica se o crupiê ou o jogador têm *Blackjack* logo no início da partida. Se o crupiê tiver um Ás virado para cima, o resultado pode ser determinístico, e o jogo é interrompido se for constatado que apenas um dos participantes possui *Blackjack*, garantindo uma vitória automática. Caso ambos, crupiê e jogador, obtenham *Blackjack* simultaneamente, ocorre um empate. No entanto, se o crupiê apresentar um 10, *J*, *Q* ou *K* virado para cima, o jogo prossegue normalmente, permitindo que o jogador. Somente após o término das jogadas, quando todas as cartas do crupiê são reveladas, é verificado se ele tem *Blackjack*, o que poderá resultar em uma vitória ou empate, dependendo da mão do jogador.

Se nenhum *Blackjack* for detectado, a estratégia de decisão do jogador entra em ação. Com base na primeira carta virada para cima do crupiê, é determinado um valor de "k" para *soft hand* ou *hard hand*, que define o limite até o qual o jogador pode pedir mais cartas. O algoritmo implementa essa lógica, permitindo que o jogador continue comprando cartas enquanto o valor da mão estiver abaixo do limite estratégico definido. Em paralelo, o crupiê segue uma regra fixa de continuar comprando cartas até atingir pelo menos 17 pontos.

A partir desse ponto, o resultado da rodada é avaliado por meio de diferentes cenários. O jogo pode terminar em empate se o valor das mãos do jogador e do crupiê forem iguais e menores ou iguais a 21. Caso contrário, a função verifica se o crupiê obteve um *Blackjack* "tardio", ou seja, formou *Blackjack* após a primeira rodada de compra de cartas. Esse resultado também leva à vitória imediata do crupiê.

Em outros cenários, a função verifica se o jogador "estourou", ou seja, ultrapassou o valor de 21, resultando em uma vitória automática do crupiê. Alternativamente, se o crupiê estourar sem que o jogador estoure, o jogador vence. Se ambos permanecerem com valores válidos abaixo de 21, a vitória é decidida pela comparação direta entre os valores das mãos: o jogador vence se tiver um valor maior que o do crupiê, e o crupiê vence se seu valor for maior.

Esse ciclo de decisões e verificações proporciona uma simulação completa da rodada de *Blackjack*, capturando todas as possíveis interações entre jogador e crupiê, e alimentando as estatísticas que posteriormente serão usadas para analisar a eficácia das estratégias simuladas.

A função *simular_jogos(numero_de_simulacoes, ks)* é responsável por realizar uma série de simulações de partidas de *Blackjack*, contabilizando e registrando diversos resultados estatísticos das partidas. Ela começa inicializando um dicionário chamado resultados, onde são armazenados dados sobre vitórias, derrotas, empates, "*Blackjacks*" (tanto do jogador quanto do

crupiê), e apostas laterais como "*Perfect Pairs*" e "21+3".

O baralho é criado e cortado, simulando o processo real de um cassino. Para cada simulação, o jogador e o crupiê recebem duas cartas iniciais, e verifica-se se há necessidade de reembaralhar as cartas caso o tamanho do baralho seja menor que o tamanho do corte.

A primeira carta do crupiê é usada como chave para selecionar uma estratégia de jogo específica baseada na tabela de estratégias fornecida. Dependendo da carta inicial do crupiê, a função é chamada, simulando uma rodada de *Blackjack* e atualizando os resultados correspondentes àquela carta inicial do crupiê.

Ao final das simulações, a função calcula e exibe os resultados acumulados para cada carta inicial do crupiê, apresentando dados como porcentagens de vitórias do jogador, crupiê, empates, frequência de "*Blackjacks*" e quantidades de cada uma das apostas laterais bem-sucedidas (como "*Suited Trips*" ou "*Perfect Pairs*"). Além disso, a função também calcula as estatísticas globais, como a porcentagem total de vitórias do jogador e do crupiê, além de normalizar esses valores excluindo os empates.

Esse processo possibilita a análise detalhada do comportamento das probabilidades do jogo com base em diferentes estratégias e cartas iniciais do crupiê, fornecendo uma visão abrangente do desempenho do jogador e do crupiê ao longo de várias simulações com uma estratégia fixa e determinada a *priori*.

No final da implementação, são definidos parâmetros importantes para a simulação dos jogos de *Blackjack*, com o objetivo de obter probabilidades de vitória com base nas mãos do jogador e do crupiê. Primeiramente, o número de simulações a serem realizadas é estabelecido com a variável *numero_de_simulacoes*, neste caso, configurada para 100 milhões. Essa quantidade define quantas rodadas serão simuladas, o que é fundamental para gerar uma amostra estatisticamente relevante, aumentando a confiabilidade dos resultados.

Em seguida, é criado um dicionário *ks*, que contém os valores de "*k*" correspondentes para cada valor de carta do crupiê. O "*k*" representa o limite de paradas para o jogador, ou seja, o momento em que ele deve parar de pedir para sortear cartas. Cada valor de carta do crupiê, de 2 a Ás, recebe um valor específico de "*k*", o que permite que o comportamento do jogador varie de acordo com a carta inicial, seguindo as regras do jogo.

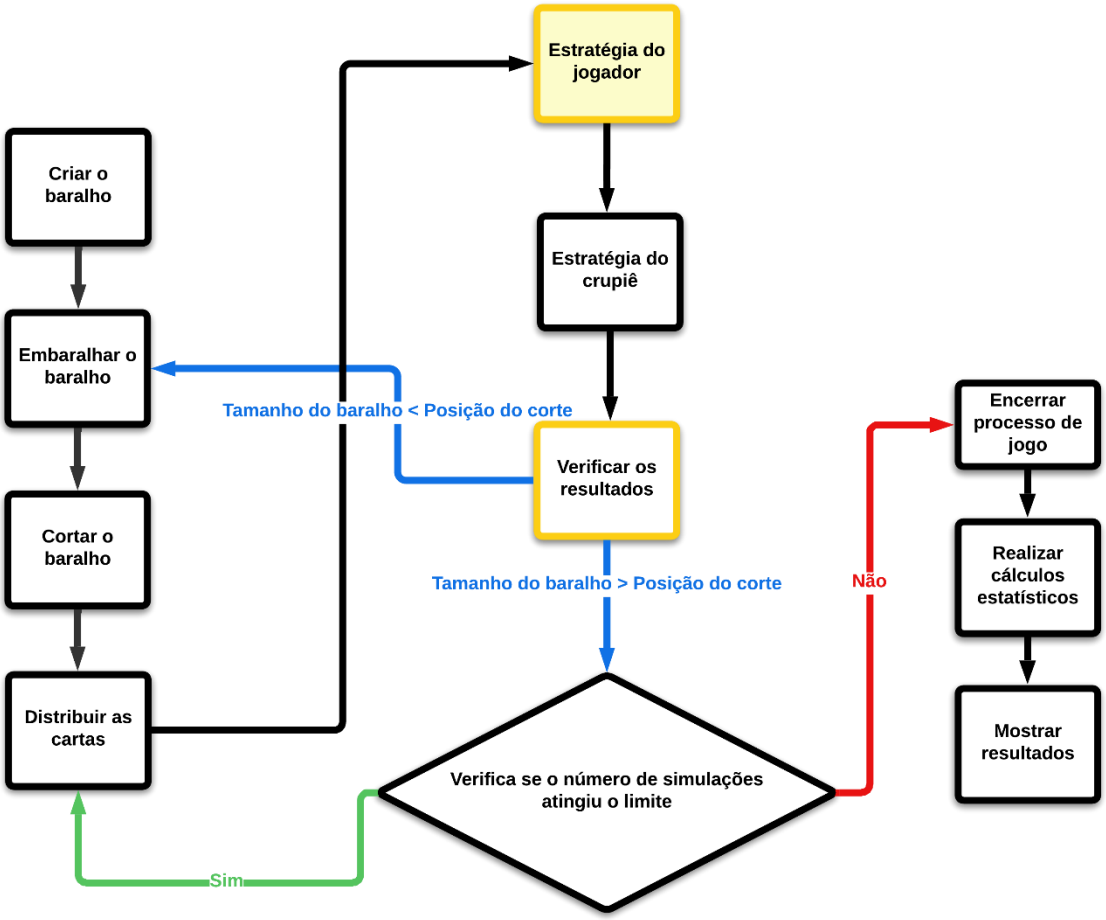
É definido outro dicionário chamado *kssoft*, que atribui valores de "*k*" para mãos *soft*, ou seja, aquelas que envolvem um Ás que pode valer 1 ou 11. Nesse cenário, os valores de "*k*" também variam de acordo com a carta inicial do crupiê, refletindo a flexibilidade que o Ás traz ao jogo.

E no fim, a função *simular_jogos* é chamada, utilizando o número de simulações e os

dicionários *ks* e *kssoft*. A função executa as rodadas de *Blackjack* conforme as estratégias definidas, considerando as cartas do crupiê e do jogador, e gera os resultados das simulações. Isso permite avaliar estatisticamente as probabilidades de vitória para diferentes situações de jogo, utilizando um modelo probabilístico baseado na simulação de Monte Carlo.

Essa abordagem possibilita a análise de diversas estratégias de *Blackjack*, oferecendo uma nova intuição e mais próxima da real, sobre as chances de sucesso com diferentes combinações de cartas e estratégias adotadas pelo jogador.

Figura 3: Fluxograma das implementações referentes ao *Blackjack*



Fonte: Autor.

Na figura 3, as principais diferenças entre os dois códigos estão nas seções "Estratégia do jogador" e "Verificação dos resultados". Quando o objetivo é calcular as probabilidades com base em uma estratégia fixa, não é necessário implementar as decisões de dividir ou dobrar, já que essas ações afetam apenas os ganhos financeiros e não as probabilidades de vitória do jogador. Nesse caso, foram desenvolvidos dois códigos: um voltado para calcular apenas a probabilidade de vitória, considerando uma estratégia fixa e

constante, e outro para analisar o impacto das decisões de dividir e dobrar no saldo do jogador.

O código que foca nas probabilidades considera a distinção entre mãos "*soft*" e "*hard*", já que essas categorias influenciam as decisões tomadas e, consequentemente, os resultados obtidos. Essa diferenciação é refletida nos resultados apresentados. Já no código que avalia o desempenho financeiro do jogador, levando em conta uma banca e uma aposta fixas, as decisões de dividir e dobrar são fundamentais. Nessas situações, o impacto sobre os ganhos ou perdas do jogador é diretamente influenciado por essas escolhas quando a probabilidade de vitória é favorável.

A forma de contabilizar os resultados difere nos dois casos. No cálculo das probabilidades, considera-se apenas a chance de vitória do jogador em determinada situação. No entanto, ao analisar o retorno financeiro, é indispensável considerar os multiplicadores envolvidos nas apostas, que têm grande impacto nos eventos registrados. Essas distinções são evidenciadas tanto nas estratégias implementadas quanto na contagem dos resultados. Foram testadas várias estratégias, e as três que apresentaram os melhores desempenhos (ou menores perdas) são destacadas neste trabalho.

O código apresenta diferentes estratégias dependendo da primeira carta do crupiê. Abaixo estão algumas das tabelas de decisões que orientam as ações do jogador com base nas possíveis mãos do crupiê:

Estratégias:

Valores de "k" para cada mão possível do crupiê (mãos "hard"):

ks = { 2: 13, 3: 13, 4: 12, 5: 12, 6: 12, 7: 17, 8: 17, 9: 17, 10: 17, 'J': 17, 'Q': 17, 'K': 17, 'A': 17 }

Valores de "k" para cada mão possível do crupiê (mãos "soft"):

kssoft = { 2: 17, 3: 18, 4: 18, 5: 18, 6: 18, 7: 17, 8: 17, 9: 18, 10: 18, 'J': 18, 'Q': 18, 'K': 18, 'A': 18 }

Além disso, o código contém listas de estratégias que determinam quando o jogador deve optar por dividir, dobrar ou pedir uma nova carta ("*hit*"), dependendo da mão do crupiê. Por exemplo:

Dividir (*split*): Ações recomendadas caso o crupiê tenha cartas específicas, como:

split_2 = [['A', 'A'], [2, 2], [3, 3], [6, 6], [7, 7], [8, 8], [9, 9]] (crupiê com 2)

split_3 = [['A', 'A'], [2, 2], [3, 3], [6, 6], [7, 7], [8, 8], [9, 9]] (crupiê com 3)

E assim por diante para outras cartas do crupiê.

A verificação dos resultados no código depende da mão final do crupiê e do jogador. Se ambos tiverem *Blackjack*, o resultado é um empate. Outras condições incluem o *Blackjack*

do crupiê ou do jogador e as diversas combinações de resultados baseadas nas mãos divididas ou dobradas. As regras específicas para contar os eventos, como *bust* do jogador (quando a mão ultrapassa 21 pontos) ou do crupiê, variam conforme o código, pois o objetivo é registrar não só a vitória ou derrota, mas também o impacto financeiro das decisões tomadas. Um exemplo do funcionamento do código:

Se o crupiê tiver um 2 como carta inicial, o código verifica se o jogador deve dividir ou dobrar com base em sua mão. Caso contrário, o jogador pode optar por pedir novas cartas ("*hit*") até que sua mão alcance um valor adequado ou até que estoure (ultrapasse 21). Se o jogador tomar decisões corretas conforme a estratégia implementada, os resultados são registrados adequadamente.

De forma geral, as estratégias de dividir, dobrar ou pedir cartas são implementadas no código de acordo com as probabilidades calculadas e os impactos financeiros que essas decisões podem trazer, sendo ambos os cenários avaliados separadamente.

3.6 Implementação Computacional em Linguagem *Python* da Simulação Monte Carlo Baseado no Caminhante Aleatório Para Determinação de Números de Jogadores Necessários Para a Lei dos Grandes Números Prevalecer e Para Estimar o Lucro de Cassino Para Jogos Determinados

O código que está representado pelo fluxograma, representa a implementação uma simulação de apostas em um cassino, com foco no comportamento financeiro dos jogadores e no lucro do cassino ao longo de diversas partidas, além de ser possível confirmar o número de jogos com um índice de confiança necessários para que o cassino garanta o lucro. A simulação utiliza a biblioteca *NumPy* para manipulação de vetores e é parametrizada por valores como a quantidade de jogadores, o número máximo de partidas, o saldo inicial dos jogadores e as probabilidades de vitória e derrota.

O código define uma série de parâmetros que influenciam diretamente o comportamento da simulação:

QUANTIDADE_JOGADORES: Define quantos jogadores participarão da simulação. Neste caso, é configurado como 1.

QUANTIDADE_MAXIMA_PARTIDAS: Define o número máximo de partidas que serão simuladas, no caso, 3500.

SALDO_INICIAL_JOGADORES: Estabelece o saldo inicial de cada jogador, configurado como 3500 unidades monetárias.

VALOR_APOSTA_JOGADORES: O valor fixo apostado por cada jogador em cada rodada, que neste caso é de 1 unidade monetária.

CHANCE_GANHAR_JOGO: A probabilidade de vitória de cada jogador, expressa como uma chance percentual de 2,70%.

MULTIPLICADOR_VITORIA: Define o valor multiplicado sobre a aposta quando o jogador vence. Neste caso, o multiplicador é 35, refletindo uma odd de 35:1.

São criados vetores para armazenar o saldo dos jogadores ao longo das partidas, além de variáveis para acompanhar o número de partidas, o lucro do cassino, e as quantidades de vitórias e derrotas. As principais variáveis são: *saldoJogador* que é um vetor que armazena o saldo de cada jogador em cada partida, *partidaAtual* que é um contador para rastrear o número de partidas já jogadas, *lucroCassino* que representa uma variável que acumula o lucro do cassino ao longo da simulação, e *quantidadeVitorias* e *quantidadeDerrotas* que são contadores para rastrear o número de vitórias e derrotas dos jogadores.

A função *atualizarSaldoJogadores()* atualiza o saldo dos jogadores após cada partida, levando em consideração se o jogador venceu ou perdeu. O funcionamento da função pode ser descrito da seguinte forma: Para cada jogador, o saldo é verificado. Se o jogador tiver saldo suficiente para apostar, ele participa da rodada. Caso contrário, ele aposta o valor restante de seu saldo.

Um número aleatório que segue uma distribuição uniforme e pertence ao intervalo entre 0 e 100 é gerado. Se o número sorteado for menor que a probabilidade de vitória, o jogador ganha, e seu saldo é atualizado com o valor da vitória multiplicado pela aposta. Nesse caso, o cassino registra um prejuízo correspondente à vitória do jogador. Se o jogador perde, o saldo é reduzido pelo valor da aposta, e o cassino acumula lucro equivalente ao valor perdido pelo jogador. Ao final de cada rodada, a variável *partidaAtual* é incrementada.

A função *rodarSimulacao()* é a principal, ela que controla a simulação e utiliza um loop while que continua chamando a função *atualizarSaldoJogadores()* até que o número máximo de partidas seja alcançado. Ao final da simulação, são exibidos os resultados:

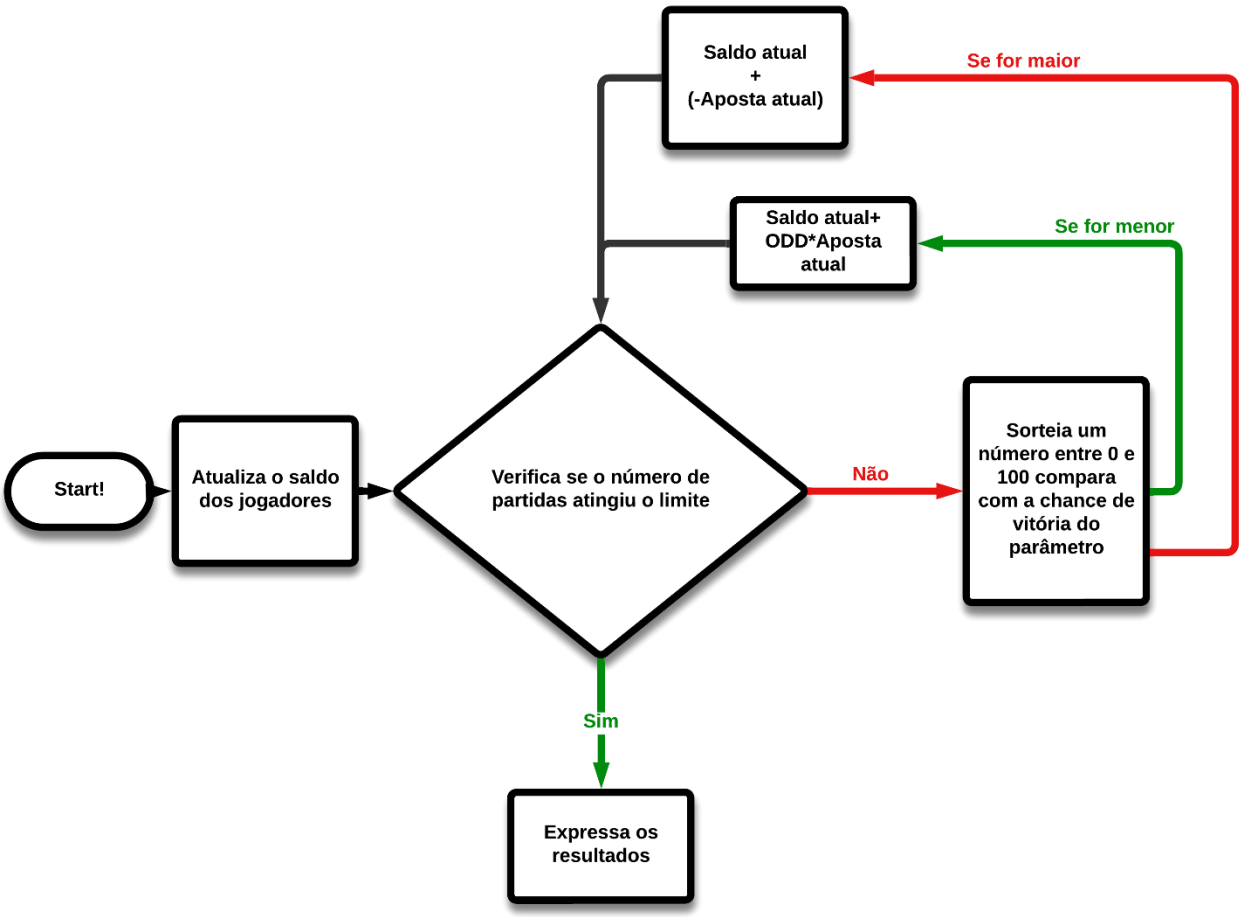
Lucro do cassino: O montante total acumulado pelo cassino.

Quantidade de partidas jogadas: O número total de rodadas.

Quantidade de vitórias e derrotas: O número de vezes que os jogadores venceram ou perderam.

O código principal da simulação é executado dentro do bloco *if __name__ == "__main__"*, garantindo que a simulação seja rodada quando o script é executado diretamente.

Figura 4: Fluxograma das implementações referente às estimativas de lucro do cassino e confirmação do número de jogos necessários para a LGN prevalecer.



Fonte: Autor.

4 RESULTADOS

4.1 Explicação das Estratégias Variantes II, e III

Nesta seção, são apresentados os resultados das simulações computacionais com variações nas três estratégias selecionadas (detalhadas no apêndice), para avaliar as diferenças nas decisões em cada cenário. No contexto desta análise, "caso parcial" se refere à carta visível do crupiê, enquanto o "caso geral" diz respeito ao resultado final da rodada. Simplificadamente, as estratégias II e III refletem ajustes no ponto de decisão do jogador. A estratégia II implica em parar com um valor de ponto inferior ao ideal (exceto quando o valor é 11, onde sempre deve-se pedir uma nova carta), enquanto a estratégia III envolve parar com um valor acima do indicado pela estratégia I.

Se a proporção de vitórias do jogador e do crupiê em uma determinada situação se aproxima ou se altera de forma relevante com essas variações, é possível avaliar o impacto das decisões mais conservadoras ou agressivas adotadas pelo jogador, comparando-as à estratégia ideal.

4.2 Estratégia I, II e III Para 8 baralhos

Os quadros a seguir apresentam os resultados das simulações de 100 milhões de jogos com 8 baralhos, incluindo dados gerais, parciais, apostas laterais e o retorno ao jogador (RTP).

Tabela 12: Resultados gerais da estratégia I para 8 baralhos.

Vitórias Jogador (%)	Vitórias Crupiê (%)	Empate (%)	BJ do Jogador (%)	BJ do Crupiê (%)	Jogador *Bust* (%)	Crupiê *Bust* (%)	Vitórias * Jogador (%)	Vitórias * Crupiê (%)
45,022	46,540	8,438	4,738	4,739	15,576	20,338	49,171	50,829

Vitórias* jogador = Vitórias do jogador excluindo empates

Vitórias* crupiê = Vitórias do crupiê excluindo empates

Fonte: Autor.

Na estratégia I, o jogador venceu 45,02% das rodadas, enquanto o crupiê venceu 46,53%, com 8,44% de empates. Excluindo os empates, o jogador obteve 49,17% das vitórias,

e o crupiê 50,83%, confirmando a vantagem para o crupiê.

Tabela 13: Resultados parciais da estratégia I para 8 baralhos.

Carta do crupiê	Vitórias Jogador (%)	Vitórias Crupiê (%)	Empate (%)	BJ do Jogador (%)	BJ do Crupiê (%)	Jogador *Bust* (%)	Crupiê *Bust* (%)
2	53,468	40,970	5,562	4,759	0,000	3,275	28,921
3	54,380	40,076	5,545	4,764	0,000	3,318	30,580
4	55,633	39,456	4,911	4,764	0,000	0,000	33,630
5	56,778	38,461	4,762	4,765	0,000	0,000	35,548
6	57,465	37,731	4,804	4,764	0,000	0,000	35,961
7	51,544	38,638	9,818	4,763	0,000	26,551	15,338
8	47,063	42,995	9,942	4,762	0,000	26,545	14,312
9	41,795	47,790	10,415	4,768	0,000	26,543	13,427
10	34,833	54,310	11,214	4,384	7,371	24,536	12,500
J	34,857	54,307	11,192	4,378	7,390	24,548	12,492
Q	34,832	54,325	11,204	4,390	7,367	24,571	12,479
K	34,846	54,321	11,191	4,378	7,385	24,566	12,486
A	28,608	63,365	9,458	3,231	29,827	18,753	6,903

Fonte: Autor.

Ao analisar os resultados parciais, observa-se que o jogador tem vantagem quando o crupiê exibe cartas baixas (como 4, 5 e 6), vencendo mais de 55% das rodadas nesses casos. Conforme a carta do crupiê aumenta, essa vantagem se inverte, com o crupiê vencendo mais de 54% das rodadas quando a carta visível é 10, J, Q, K ou Ás. As apostas laterais também mostraram que a frequência de combinações não foi favorável ao jogador.

Tabela 14: Resultados gerais para 8 baralhos.

Pares perfeitos (%)	Par perfeito (%)	Par colorido (%)	Par misto (%)	21+3 (%)	*Flush* (%)	*Straight* (%)	*Three of a kind* (%)	*Straight flush* (%)	*Suited trips* (%)
7,473	1,687	1,926	3,860	9,306	5,739	2,836	0,517	0,189	0,024

Fonte: Autor.

Os resultados das apostas laterais indicam que a frequência de combinações é para o jogador uma ruína ainda maior que o próprio *Blackjack* em si.

Tabela 15: Resultados gerais da estratégia II para 8 baralhos.

Vitórias Jogador (%)	Vitórias Crupiê (%)	Empate (%)	BJ do Jogador (%)	BJ do Crupiê (%)	Jogador *Bust* (%)	Crupiê *Bust* (%)	Vitórias * Jogador (%)	Vitórias * Crupiê (%)
45,157	46,925	7,918	4,740	4,743	10,837	21,500	49,040	50,960

Fonte: Autor.

Quando é comparado a estratégia I com a II, o número de empates diminui, mas as partidas que deixam de empatar favorecem mais o crupiê.

Tabela 16: Resultados parciais da estratégia II para 8 baralhos.

Carta do crupiê	Vitórias Jogador (%)	Vitórias Crupiê (%)	Empate (%)	BJ do Jogador (%)	BJ do Crupiê (%)	Jogador *Bust* (%)	Crupiê *Bust* (%)
2	53,557	41,431	5,012	4,756	0,000	0,000	30,081
3	54,635	40,397	4,968	4,760	0,000	0,000	31,836
4	55,849	39,317	4,834	4,753	0,000	0,000	33,673
5	57,008	38,306	4,686	4,779	0,000	0,000	35,550
6	57,721	37,486	4,793	4,763	0,000	0,000	35,971
7	51,391	39,507	9,102	4,770	0,000	19,104	17,302
8	46,917	43,845	9,238	4,760	0,000	19,089	16,103
9	41,910	48,433	9,657	4,762	0,000	19,115	15,140
10	35,160	54,708	10,486	4,391	7,379	17,649	14,075
J	35,145	54,687	10,523	4,379	7,381	17,664	14,084
Q	35,137	54,711	10,512	4,385	7,383	17,652	14,076
K	35,141	54,718	10,496	4,389	7,387	17,653	14,070
A	28,299	64,225	8,922	3,241	29,856	13,465	7,776

Fonte: Autor.

O jogador tende a estourar menos frequentemente quando o crupiê exibe cartas

diferentes de 4, 5 ou 6. Contudo, a redistribuição dessas partidas em que o jogador não estoura aumenta a frequência de vitórias do crupiê, o que indica que a estratégia II apresenta um desempenho inferior em relação à estratégia I.

Tabela 17: Resultados parciais da estratégia III para 8 baralhos.

Vitórias Jogador (%)	Vitórias Crupiê (%)	Empate (%)	BJ do Jogador (%)	BJ do Crupiê (%)	Jogador *Bust* (%)	Crupiê *Bust* (%)	Vitórias * Jogador (%)	Vitórias * Crupiê (%)
44,561	47,615	7,823	4,738	4,737	21,921	18,663	48,343	51,657

Fonte: Autor.

Embora a quantidade de empates diminua na comparação das estratégias I e III, observa-se novamente que as partidas que não resultam mais em empate são redistribuídas de maneira que beneficiam o crupiê mais do que o jogador. Em particular, essa estratégia consegue ser ainda pior que a estratégia II de forma geral.

Tabela 18: Resultados parciais da estratégia III para 8 baralhos.

Carta do crupiê	Vitórias Jogador (%)	Vitórias Crupiê (%)	Empate (%)	BJ do Jogador (%)	BJ do Crupiê (%)	Jogador *Bust* (%)	Crupiê *Bust* (%)
2	53,060	40,769	6,172	4,753	0,000	7,508	27,459
3	53,779	40,087	6,135	4,755	0,000	7,486	29,036
4	55,188	39,336	5,477	4,766	0,000	3,308	32,348
5	56,109	38,557	5,333	4,755	0,000	3,315	34,193
6	56,771	37,874	5,355	4,759	0,000	3,323	34,574
7	51,412	43,344	5,245	4,754	0,000	35,156	13,103
8	46,829	44,205	8,966	4,757	0,000	35,250	12,154
9	41,442	49,057	9,501	4,771	0,000	35,220	11,416
10	34,212	55,760	10,383	4,376	7,378	32,600	10,660
J	34,209	55,741	10,404	4,399	7,362	32,615	10,656
Q	34,176	55,796	10,386	4,393	7,384	32,615	10,648
K	34,175	55,781	10,400	4,382	7,381	32,627	10,657
A	28,755	64,460	8,227	3,237	29,802	24,861	5,882

Fonte: Autor.

Ao comparar a estratégia I com a III, observa-se que o jogador estoura com maior frequência em todos os cenários. A quantidade de empates também diminui, mas, mais uma vez, essa redistribuição de partidas favorece o crupiê, resultando em um maior número de vitórias para ele, independentemente da carta visível.

Tabela 19: Retorno ao jogador para 8 baralhos.

Estratégia	Total apostado em *Blackjack*	Total apostado em Perfect Pairs	Total apostado em 21+3	RTP em *Blackjack* (%)	RTP em Perfect Pairs (%)	RTP em 21+3 (%)
I	110461017	100000000	100000000	98,576	95,878	92,820
II	110385378	100000000	100000000	98,011	95,868	92,897
III	110942291	100000000	100000000	97,531	95,861	92,883

Fonte: Autor.

Os resultados sugerem que, embora a estratégia considerada ideal (estratégia I) seja, de fato, a melhor entre as três, mesmo com ela, a expectativa matemática do jogador continua negativa ao longo de um grande número de rodadas. Isso se reflete na simulação, onde o retorno ao jogador (RTP) calculado foi de 98,576%, comparado ao valor de 99,29% anunciado pelos cassinos. A pequena vantagem observada na primeira mão está relacionada à finitude dos baralhos, o que permite uma estimativa mais precisa das probabilidades das cartas remanescentes (contagem de cartas). Contudo, a análise de Epstein demonstra que essa vantagem é insignificante com 8 baralhos, não conferindo ao jogador um benefício relevante.

4.3 Estratégia I, II e III Para 6 baralhos

Os quadros a seguir apresentam os resultados das simulações de 100 milhões de jogos com 6 baralhos, incluindo dados gerais, parciais, apostas laterais e o retorno ao jogador (RTP).

Tabela 20: Resultados gerais da estratégia I para 6 baralhos.

Vitórias Jogador	Vitórias Crupiê	Empate (%)	BJ do Jogador	BJ do Crupiê	Jogador *Bust*	Crupiê *Bust*	Vitórias *	Vitórias *

(%)	(%)		(%)	(%)	(%)	(%)	Jogador (%)	Crupiê (%)
45,029	46,548	8,423	4,742	4,741	15,582	20,350	49,171	50,829

Fonte: Autor.

Na estratégia I, o jogador venceu 45,03% das rodadas, enquanto o crupiê venceu 46,55%, com 8,42% de empates. Excluindo os empates, o jogador obteve 49,17% das vitórias, e o crupiê 50,83%, confirmando a mesma vantagem para o crupiê como quando haviam 8 baralhos.

Tabela 21: Resultados parciais da estratégia I para 6 baralhos.

Carta do crupiê	Vitórias Jogador (%)	Vitórias Crupiê (%)	Empate (%)	BJ do Jogador (%)	BJ do Crupiê (%)	Jogador *Bust* (%)	Crupiê *Bust* (%)
2	53,481	40,940	5,579	4,766	0,000	3,260	28,934
3	54,417	40,034	5,549	4,783	0,000	3,305	30,624
4	55,667	39,412	4,922	4,771	0,000	0,000	33,674
5	56,814	38,423	4,764	4,774	0,000	0,000	35,593
6	57,472	37,729	4,799	4,776	0,000	0,000	35,953
7	51,550	38,647	9,802	4,775	0,000	26,563	15,325
8	47,024	43,017	9,959	4,784	0,000	26,567	14,264
9	41,773	47,840	10,388	4,774	0,000	26,573	13,426
10	34,841	54,361	11,150	4,375	7,377	24,579	12,499
J	34,850	54,350	11,151	4,385	7,384	24,568	12,490
Q	34,846	54,350	11,157	4,384	7,393	24,572	12,507
K	34,860	54,341	11,152	4,379	7,383	24,560	12,503
A	28,553	63,421	9,448	3,216	29,877	18,760	6,913

Fonte: Autor.

Além das mesmas conclusões para o caso de 8 baralhos, observa-se que o em grande maior parte dos casos parciais, o crupiê tem uma porcentagem de vitória menor

Tabela 22: Resultados gerais para 6 baralhos.

Pares	Par	Par	Par	21+3	Flush	Straig	Three	Straig	Suited

perfeitos (%)	perfeito (%)	colorido (%)	misto (%)	(%)	(%)	ht (%)	of a kind (%)	ht flush (%)	trips (%)
7,393	1,608	1,928	3,857	9,263	5,702	2,845	0,505	0,190	0,021

Fonte: Autor.

Observa-se que o em grande maior parte dos casos de apostas laterais, também ao comparar com 8 baralhos, ocorrem com menor frequência pares perfeitos e combinações 21+3.

Tabela 23: Resultados gerais da estratégia II para 6 baralhos.

Vitórias Jogador (%)	Vitórias Crupiê (%)	Empate (%)	BJ do Jogador (%)	BJ do Crupiê (%)	Jogador *Bust* (%)	Crupiê *Bust* (%)	Vitórias * Jogador (%)	Vitórias * Crupiê (%)
45,186	46,911	7,903	4,743	4,736	10,845	21,504	49,064	50,937

Fonte: Autor.

Tabela 24: Resultados parciais da estratégia II para 6 baralhos.

Carta do crupiê	Vitórias Jogador (%)	Vitórias Crupiê (%)	Empate (%)	BJ do Jogador (%)	BJ do Crupiê (%)	Jogador *Bust* (%)	Crupiê *Bust* (%)
2	53,577	41,417	5,005	4,778	0,000	0,000	30,062
3	54,690	40,336	4,974	4,757	0,000	0,000	31,855
4	55,848	39,308	4,844	4,777	0,000	0,000	33,656
5	57,102	38,208	4,691	4,774	0,000	0,000	35,587
6	57,745	37,455	4,801	4,776	0,000	0,000	35,970
7	51,339	39,558	9,103	4,774	0,000	19,155	17,274
8	46,924	43,870	9,206	4,770	0,000	19,136	16,090
9	41,937	48,424	9,639	4,781	0,000	19,079	15,140
10	35,203	54,674	10,474	4,396	7,382	17,667	14,098
J	35,206	54,681	10,467	4,381	7,369	17,659	14,092
Q	35,182	54,696	10,475	4,387	7,366	17,654	14,074
K	35,203	54,705	10,444	4,390	7,385	17,630	14,087

A	28,257	64,264	8,907	3,215	29,841	13,516	7,790

Fonte: Autor.

Tabela 25: Resultados gerais da estratégia III para 6 baralhos.

Vitórias Jogador (%)	Vitórias Crupiê (%)	Empate (%)	BJ do Jogador (%)	BJ do Crupiê (%)	Jogador *Bust* (%)	Crupiê *Bust* (%)	Vitórias * Jogador (%)	Vitórias * Crupiê (%)
44,568	47,620	7,812	4,742	4,741	21,917	18,661	48,345	51,655

Fonte: Autor.

Tabela 26: Resultados parciais da estratégia III para 6 baralhos.

Carta do crupiê	Vitórias Jogador (%)	Vitórias Crupiê (%)	Empate (%)	BJ do Jogador (%)	BJ do Crupiê (%)	Jogador *Bust* (%)	Crupiê *Bust* (%)
2	53,055	40,771	6,175	4,770	0,000	7,505	27,408
3	53,788	40,088	6,123	4,766	0,000	7,497	29,045
4	55,186	39,328	5,486	4,764	0,000	3,312	32,346
5	56,138	38,530	5,332	4,779	0,000	3,315	34,199
6	56,796	37,839	5,366	4,787	0,000	3,313	34,561
7	51,408	43,361	5,232	4,772	0,000	35,163	13,102
8	46,884	44,168	8,948	4,777	0,000	35,236	12,166
9	41,421	49,083	9,496	4,778	0,000	35,243	11,457
10	34,222	55,753	10,377	4,404	7,375	32,580	10,643
J	34,176	55,815	10,364	4,380	7,373	32,634	10,643
Q	34,218	55,766	10,366	4,389	7,377	32,590	10,637
K	34,217	55,780	10,357	4,375	7,395	32,577	10,634
A	28,652	64,560	8,216	3,205	29,878	24,913	5,871

Fonte: Autor.

Tabela 27: Retorno ao jogador para 6 baralhos.

Estratégia	Total apostado em	Total apostado em Perfect	Total apostado em 21+3	RTP em *Blackjack*	RTP em Perfect Pairs	RTP em 21+3

	Blackjack	Pairs				
I	110437209	100000000	100000000	98,593	93,839	91,880
II	110357479	100000000	100000000	98,025	93,883	91,896
III	110911242	100000000	100000000	97,554	93,928	91,887

Fonte: Autor.

Os resultados são extremamente semelhantes para o caso dos 8 baralhos, para o *Blackjack*, portanto se atribui as mesmas análises, porém quando se trata das apostas laterais, o RTP é muito inferior ao caso de 8 baralhos.

4.4 Estratégia I, II e III Para 4 baralhos

Tabela 28: Resultados gerais da estratégia I para 4 baralhos.

Vitórias Jogador (%)	Vitórias Crupiê (%)	Empate (%)	BJ do Jogador (%)	BJ do Crupiê (%)	Jogador *Bust* (%)	Crupiê *Bust* (%)	Vitórias * Jogador (%)	Vitórias * Crupiê (%)
45,054	46,549	8,397	4,742	4,744	15,579	20,344	49,184	50,816

Fonte: Autor

Tabela 29: Resultados parciais da estratégia I para 4 baralhos.

Carta do crupiê	Vitórias Jogador (%)	Vitórias Crupiê (%)	Empate (%)	BJ do Jogador (%)	BJ do Crupiê (%)	Jogador *Bust* (%)	Crupiê *Bust* (%)
2	53,540	40,902	5,558	4,792	0,000	3,233	28,935
3	54,393	40,036	5,571	4,788	0,000	3,315	30,568
4	55,698	39,382	4,920	4,783	0,000	0,000	33,687
5	56,888	38,343	4,770	4,802	0,000	0,000	35,620
6	57,473	37,715	4,811	4,770	0,000	0,000	35,906
7	51,544	38,678	9,778	4,794	0,000	26,619	15,292
8	47,013	43,052	9,935	4,777	0,000	26,601	14,219
9	41,764	47,845	10,391	4,787	0,000	26,559	13,443
10	34,930	54,340	11,077	4,387	7,391	24,537	12,487
J	34,920	54,341	11,085	4,395	7,381	24,544	12,507

Q	34,907	54,361	11,078	4,380	7,412	24,537	12,493
K	34,915	54,342	11,087	4,381	7,397	24,535	12,502
A	28,443	63,530	9,423	3,160	29,928	18,825	6,923

Fonte: Autor

Tabela 30: Resultados gerais para 4 baralhos.

Pares perfeitos (%)	Par perfeito (%)	Par colorido (%)	Par misto (%)	21+3 (%)	Flush (%)	Straight (%)	Three of a kind (%)	Straight flush (%)	Suited trips (%)
7,246	3,866	1,931	1,449	9,171	5,630	2,858	0,478	0,191	0,014

Fonte: Autor

Tabela 31: Resultados gerais da estratégia II para 4 baralhos.

Vitórias Jogador (%)	Vitórias Crupiê (%)	Empate (%)	BJ do Jogador (%)	BJ do Crupiê (%)	Jogador *Bust* (%)	Crupiê *Bust* (%)	Vitórias * Jogador (%)	Vitórias * Crupiê (%)
45,199	46,922	7,879	4,743	4,745	10,843	21,504	49,065	50,935

Fonte: Autor

Tabela 32: Resultados parciais da estratégia II para 4 baralhos.

Carta do crupiê	Vitórias Jogador (%)	Vitórias Crupiê (%)	Empate (%)	BJ do Jogador (%)	BJ do Crupiê (%)	Jogador *Bust* (%)	Crupiê *Bust* (%)
2	53,588	41,396	5,017	4,791	0,000	0,000	30,045
3	54,681	40,332	4,987	4,783	0,000	0,000	31,815
4	55,907	39,245	4,848	4,782	0,000	0,000	33,687
5	57,157	38,162	4,681	4,798	0,000	0,000	35,621
6	57,717	37,454	4,829	4,781	0,000	0,000	35,897
7	51,388	39,535	9,078	4,788	0,000	19,156	17,239
8	46,858	43,929	9,214	4,790	0,000	19,164	16,037
9	41,890	48,462	9,648	4,792	0,000	19,095	15,160

10	35,249	54,699	10,397	4,385	7,403	17,606	14,099
J	35,256	54,708	10,383	4,395	7,392	17,644	14,091
Q	35,228	54,735	10,383	4,375	7,400	17,649	14,093
K	35,215	54,737	10,394	4,381	7,415	17,663	14,085
A	28,163	64,360	8,875	3,167	29,920	13,561	7,813

Fonte: Autor

Tabela 33: Resultados gerais da estratégia III para 4 baralhos.

Vitórias Jogador (%)	Vitórias Crupiê (%)	Empate (%)	BJ do Jogador (%)	BJ do Crupiê (%)	Jogador *Bust* (%)	Crupiê *Bust* (%)	Vitórias * Jogador (%)	Vitórias * Crupiê (%)
44,598	47,612	7,790	4,743	4,742	21,926	18,670	48,366	51,634

Fonte: Autor

Tabela 34: Resultados parciais da estratégia III para 4 baralhos.

Carta do crupiê	Vitórias Jogador (%)	Vitórias Crupiê (%)	Empate (%)	BJ do Jogador (%)	BJ do Crupiê (%)	Jogador *Bust* (%)	Crupiê *Bust* (%)
2	53,085	40,740	6,175	4,782	0,000	7,492	27,427
3	53,868	39,992	6,140	4,792	0,000	7,455	29,076
4	55,271	39,249	5,480	4,785	0,000	3,319	32,388
5	56,252	38,415	5,333	4,788	0,000	3,320	34,275
6	56,759	37,864	5,377	4,789	0,000	3,322	34,494
7	51,425	43,331	5,244	4,782	0,000	35,135	13,062
8	46,852	44,190	8,959	4,781	0,000	35,267	12,145
9	41,403	49,104	9,493	4,795	0,000	35,255	11,466
10	34,270	55,788	10,288	4,394	7,383	32,611	10,639
J	34,243	55,798	10,304	4,381	7,392	32,614	10,638
Q	34,243	55,813	10,288	4,378	7,403	32,616	10,646
K	34,255	55,806	10,289	4,393	7,400	32,596	10,644
A	28,565	64,651	8,185	3,160	29,909	25,023	5,891

Fonte: Autor

Tabela 35: Retorno ao jogador para 4 baralhos.

Estratégia	Total apostado em *Blackjack*	Total apostado em Perfect Pairs	Total apostado em 21+3	RTP em *Blackjack*	RTP em Perfect Pairs	RTP em 21+3
I	110391093	100000000	100000000	98,623	89,787	90,159
II	110308556	100000000	100000000	98,055	89,815	90,171
III	110858879	100000000	100000000	97,593	89,872	90,144

Fonte: Autor

Os resultados aqui começam a diferir bastante para os casos de 6 e 8 baralhos para o *Blackjack*, portanto percebe-se uma variação do RTP em função do número de baralhos, e quando se trata das apostas laterais, o RTP é muito inferior ao caso de 8 e 6 baralhos.

4.5 Estratégia I, II e III Para 2 Baralhos

Tabela 36: Resultados gerais da estratégia I para 2 baralhos.

Vitórias Jogador (%)	Vitórias Crupiê (%)	Empate (%)	BJ do Jogador (%)	BJ do Crupiê (%)	Jogador *Bust* (%)	Crupiê *Bust* (%)	Vitórias * Jogador (%)	Vitórias * Crupiê (%)
45,134	46,537	8,328	4,755	4,754	15,582	20,367	49,235	50,765

Fonte: Autor

Tabela 37: Resultados parciais da estratégia I para 2 baralhos.

Carta do crupiê	Vitórias Jogador (%)	Vitórias Crupiê (%)	Empate (%)	BJ do Jogador (%)	BJ do Crupiê (%)	Jogador *Bust* (%)	Crupiê *Bust* (%)
2	53,552	40,871	5,578	4,848	0,000	3,181	28,883
3	54,488	39,911	5,601	4,851	0,000	3,329	30,557
4	55,906	39,162	4,932	4,851	0,000	0,000	33,803
5	57,199	38,011	4,790	4,854	0,000	0,000	35,802
6	57,484	37,660	4,856	4,853	0,000	0,000	35,744
7	51,579	38,676	9,745	4,840	0,000	26,686	15,201

8	46,984	43,109	9,907	4,848	0,000	26,639	14,088
9	41,737	47,918	10,345	4,859	0,000	26,549	13,503
10	35,039	54,413	10,872	4,389	7,458	24,516	12,522
J	35,048	54,389	10,886	4,381	7,442	24,489	12,548
Q	35,037	54,401	10,890	4,380	7,451	24,489	12,540
K	35,039	54,398	10,888	4,371	7,429	24,533	12,530
A	28,186	63,809	9,320	2,989	30,070	19,061	7,007

Fonte: Autor

Tabela 38: Resultados gerais para 2 baralhos.

Pares perfeitos (%)	Par perfeito (%)	Par colorido (%)	Par misto (%)	21+3 (%)	Flush (%)	Straight (%)	Three of a kind (%)	Straight flush (%)	Suited trips (%)
6,795	0,969	1,943	3,883	8,910	5,420	2,897	0,399	0,193	0,000

Fonte: Autor

Tabela 39: Resultados gerais da estratégia II para 2 baralhos.

Vitórias Jogador (%)	Vitórias Crupiê (%)	Empate (%)	BJ do Jogador (%)	BJ do Crupiê (%)	Jogador *Bust* (%)	Crupiê *Bust* (%)	Vitórias * Jogador (%)	Vitórias * Crupiê (%)
45,268	46,922	7,810	4,757	4,750	10,867	21,512	49,103	50,897

Fonte: Autor

Tabela 40: Resultados parciais da estratégia II para 2 baralhos.

Carta do crupiê	Vitórias Jogador (%)	Vitórias Crupiê (%)	Empate (%)	BJ do Jogador (%)	BJ do Crupiê (%)	Jogador *Bust* (%)	Crupiê *Bust* (%)
2	53,644	41,314	5,043	4,846	0,000	0,000	29,992
3	54,751	40,245	5,004	4,851	0,000	0,000	31,774
4	56,096	39,031	4,873	4,846	0,000	0,000	33,789
5	57,447	37,854	4,699	4,854	0,000	0,000	35,797

6	57,736	37,413	4,851	4,849	0,000	0,000	35,760
7	51,320	39,624	9,057	4,848	0,000	19,310	17,105
8	46,832	44,006	9,163	4,859	0,000	19,223	15,866
9	41,825	48,578	9,598	4,855	0,000	19,135	15,182
10	35,377	54,752	10,197	4,391	7,430	17,611	14,097
J	35,357	54,777	10,194	4,392	7,427	17,632	14,125
Q	35,368	54,783	10,176	4,388	7,427	17,644	14,120
K	35,380	54,764	10,180	4,376	7,446	17,649	14,140
A	27,871	64,624	8,819	2,988	30,063	13,758	7,887

Fonte: Autor

Tabela 41: Resultados gerais da estratégia III para 2 baralhos.

Vitórias Jogador (%)	Vitórias Crupiê (%)	Empate (%)	BJ do Jogador (%)	BJ do Crupiê (%)	Jogador *Bust* (%)	Crupiê *Bust* (%)	Vitórias * Jogador (%)	Vitórias * Crupiê (%)
44,672	47,606	7,723	4,756	4,755	21,928	18,676	48,410	51,590

Fonte: Autor

Tabela 42: Resultados parciais da estratégia III para 2 baralhos.

Carta do crupiê	Vitórias Jogador (%)	Vitórias Crupiê (%)	Empate (%)	BJ do Jogador (%)	BJ do Crupiê (%)	Jogador *Bust* (%)	Crupiê *Bust* (%)
2	53,116	40,694	6,190	4,849	0,000	7,469	27,371
3	53,913	39,933	6,154	4,839	0,000	7,411	29,072
4	55,444	39,050	5,506	4,863	0,000	3,345	32,488
5	56,490	38,174	5,337	4,846	0,000	3,339	34,408
6	56,788	37,784	5,428	4,859	0,000	3,350	34,350
7	51,516	43,252	5,232	4,847	0,000	35,051	13,029
8	46,846	44,244	8,911	4,840	0,000	35,348	11,983
9	41,341	49,207	9,452	4,853	0,000	35,300	11,480
10	34,378	55,846	10,101	4,380	7,458	32,563	10,650
J	34,409	55,820	10,104	4,383	7,438	32,568	10,655

Q	34,368	55,861	10,096	4,384	7,433	32,590	10,645
K	34,388	55,833	10,103	4,394	7,457	32,563	10,633
A	28,248	64,992	8,074	2,985	30,087	25,285	5,944

Fonte: Autor

Tabela 43: Retorno ao jogador para 2 baralhos.

Estratégia	Total apostado em *Blackjack*	Total apostado em Perfect Pairs	Total apostado em 21+3	RTP em *Blackjack*	RTP em Perfect Pairs	RTP em 21+3
I	110234859	100000000	100000000	98,785	77,731	85,374
II	110157204	100000000	100000000	98,202	77,672	85,266
III	110678821	100000000	100000000	97,742	77,649	85,242

Fonte: Autor

Os resultados de RTP aqui começam a diferir ainda para o *Blackjack*, portanto percebe-se novamente uma variação do RTP em função do número de baralhos, e quando se trata das apostas laterais já não tem mais sentido avaliar, pois combinações como Suited Trips não são mais possíveis com 2 baralhos, então obviamente o RTP é muito inferior.

4.6 Estratégia I, II e III Para 1 Baralho

Tabela 44: Resultados gerais da estratégia I para 1 baralho.

Vitórias Jogador (%)	Vitórias Crupiê (%)	Empate (%)	BJ do Jogador (%)	BJ do Crupiê (%)	Jogador *Bust* (%)	Crupiê *Bust* (%)	Vitórias * Jogador (%)	Vitórias * Crupiê (%)
45,274	46,540	8,186	4,778	4,781	15,594	20,379	49,310	50,690

Fonte: Autor

Tabela 45: Resultados parciais da estratégia I para 1 baralho.

Carta do crupiê	Vitórias Jogador (%)	Vitórias Crupiê (%)	Empate (%)	BJ do Jogador (%)	BJ do Crupiê (%)	Jogador *Bust* (%)	Crupiê *Bust* (%)
2	53,657	40,771	5,573	4,970	0,000	3,049	28,812

3	54,625	39,727	5,648	4,978	0,000	3,341	30,476
4	56,214	38,838	4,948	4,963	0,000	0,000	33,957
5	57,734	37,450	4,816	4,970	0,000	0,000	36,127
6	57,496	37,568	4,936	4,964	0,000	0,000	35,459
7	51,671	38,673	9,656	4,964	0,000	26,841	15,010
8	46,923	43,234	9,843	4,958	0,000	26,754	13,809
9	41,648	48,079	10,273	4,963	0,000	26,608	13,587
10	35,302	54,501	10,479	4,387	7,544	24,469	12,577
J	35,307	54,513	10,463	4,390	7,525	24,465	12,562
Q	35,304	54,502	10,476	4,394	7,535	24,423	12,551
K	35,289	54,519	10,479	4,396	7,541	24,455	12,546
A	27,567	64,405	9,180	2,621	30,449	19,429	7,129

Fonte: Autor

Tabela 46: Resultados gerais para 1 baralho.

Pares perfeitos (%)	Par perfeito (%)	Par colorido (%)	Par misto (%)	21+3 (%)	Flush (%)	Straight (%)	Three of a kind (%)	Straight flush (%)	Suited trips (%)
5,883	0,000	1,960	3,923	8,392	4,978	2,981	0,235	0,199	0,000

Fonte: Autor

Tabela 47: Resultados gerais da estratégia II para 1 baralho.

Vitórias Jogador (%)	Vitórias Crupiê (%)	Empate (%)	BJ do Jogador (%)	BJ do Crupiê (%)	Jogador *Bust* (%)	Crupiê *Bust* (%)	Vitórias * Jogador (%)	Vitórias * Crupiê (%)
45,383	46,949	7,669	4,781	4,783	10,925	21,522	49,152	50,848

Fonte: Autor

Tabela 48: Resultados parciais da estratégia II para 1 baralho.

Carta do crupiê	Vitórias Jogador (%)	Vitórias Crupiê (%)	Empate (%)	BJ do Jogador (%)	BJ do Crupiê (%)	Jogador *Bust* (%)	Crupiê *Bust* (%)

2	53,782	41,136	5,082	4,965	0,000	0,000	29,900
3	54,901	40,038	5,061	4,979	0,000	0,000	31,729
4	56,384	38,718	4,899	4,962	0,000	0,000	33,967
5	58,007	37,264	4,729	4,979	0,000	0,000	36,125
6	57,712	37,367	4,921	4,969	0,000	0,000	35,450
7	51,296	39,717	8,987	4,959	0,000	19,522	16,864
8	46,695	44,219	9,087	4,972	0,000	19,380	15,574
9	41,762	48,743	9,495	4,976	0,000	19,302	15,267
10	35,546	54,952	9,787	4,386	7,543	17,653	14,122
J	35,586	54,922	9,779	4,393	7,556	17,623	14,158
Q	35,608	54,898	9,782	4,386	7,528	17,636	14,171
K	35,610	54,910	9,764	4,395	7,539	17,629	14,154
A	27,271	65,233	8,648	2,617	30,463	14,104	8,035

Fonte: Autor

Tabela 49: Resultados gerais da estratégia III para 1 baralho.

Vitórias Jogador (%)	Vitórias Crupiê (%)	Empate (%)	BJ do Jogador (%)	BJ do Crupiê (%)	Jogador *Bust* (%)	Crupiê *Bust* (%)	Vitórias * Jogador (%)	Vitórias * Crupiê (%)
44,808	47,608	7,584	4,783	4,780	21,957	18,678	48,485	51,515

Fonte: Autor

Tabela 50: Resultados parciais da estratégia III para 1 baralho.

Carta do crupiê	Vitórias Jogador (%)	Vitórias Crupiê (%)	Empate (%)	BJ do Jogador (%)	BJ do Crupiê (%)	Jogador *Bust* (%)	Crupiê *Bust* (%)
2	53,242	40,563	6,195	4,986	0,000	7,444	27,280
3	54,052	39,773	6,176	4,970	0,000	7,308	29,049
4	55,717	38,743	5,540	4,965	0,000	3,366	32,622
5	57,004	37,638	5,358	4,974	0,000	3,368	34,665
6	56,781	37,722	5,497	4,977	0,000	3,423	34,014
7	51,632	43,141	5,227	4,973	0,000	34,916	12,936

8	46,887	44,294	8,820	4,972	0,000	35,505	11,763
9	41,213	49,445	9,342	4,960	0,000	35,417	11,528
10	34,612	55,980	9,695	4,390	7,548	32,584	10,640
J	34,651	55,925	9,706	4,387	7,541	32,521	10,670
Q	34,613	55,968	9,703	4,401	7,534	32,585	10,631
K	34,621	55,960	9,706	4,393	7,542	32,556	10,640
A	27,659	65,570	7,920	2,624	30,414	25,796	6,034

Fonte: Autor

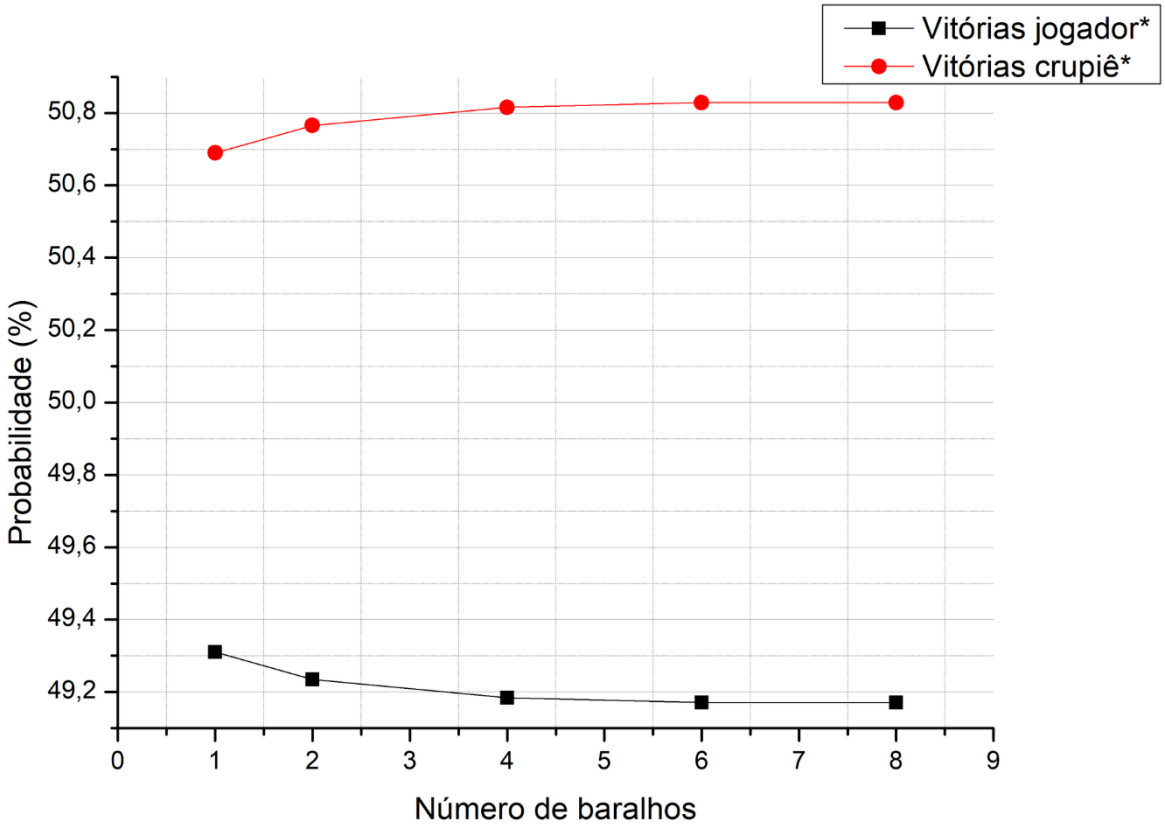
Tabela 51: Retorno ao jogador para 1 baralho.

Estratégia	Total apostado em *Blackjack*	Total apostado em Perfect Pairs	Total apostado em 21+3	RTP em *Blackjack*	RTP em Perfect Pairs	RTP em 21+3
I	109959137	100000000	100000000	99,045	52,961	78,167
II	109864640	100000000	100000000	98,405	52,958	78,104
III	110339040	100000000	100000000	98,040	52,984	78,119

Fonte: Autor

Os resultados de RTP aqui começam a diferir ainda para o *Blackjack*, portanto percebe-se novamente uma variação do RTP em função do número de baralhos, e devido a vantagem do crupiê ser tão pequena, a técnica de contagem de cartas aqui possibilita uma expectativa matemática positiva para o jogador como publicado em "*Beat the Dealer: A Winning Strategy for the Game of Twenty-One*" por Edward Thorp.

Gráfico 3: Chance de o jogador ganhar em função do número de baralhos.



Fonte: Autor.

4.7 Proposta Para Equilibrar O RTP do *Blackjack*.

Dependendo do tipo de alteração proposta pode-se mudar características do jogo, se pode propor alterações no jogo em si, ou no pagamento do jogo. Mudanças no jogo em si alteram diretamente a probabilidade de o jogador ganhar, e indiretamente o RTP, agora alterando apenas os multiplicadores, somente o RTP é alterado, as probabilidades de vencer no jogo continuam as mesmas. As únicas formas de propor alterações no jogo com intuito de torná-lo um jogo onde o retorno ao jogador seja 100%, (assumindo a melhor estratégia possível), e como em essência o jogo é o que é por ser ele em si, alterar os multiplicadores acaba se tornando um processo mais fácil computacionalmente e simples intuitivamente. Alterações nos pagamentos de *Blackjack* foram sugeridos para tornar o jogo *Blackjack* em si um jogo onde o RTP fosse justamente 100%, ou seja, na LGN o jogador obterá exatamente o valor inicial.

Tabela 52: Tipos de apostas e pagamentos para apostas laterais propostos.

Mão	Pagamento
Suited Trips	145:1

Straight Flush	70:1
Three Of A Kind	30:1
Straight	10:1
Flush	5:1
Par Perfeito	26:1
Par Colorido	13:1
Par Misto	6:1
Blackjack	2:1

Fonte: Autor.

Os resultados obtidos para 100 milhões de simulações estão representados no quadro 4.6.2

Tabela 53: Retorno ao jogador para 8 baralhos com multiplicadores propostos.

Estratégia	Total apostado em *Blackjack*	Total apostado em Perfect Pairs	Total apostado em 21+3	RTP em *Blackjack*	RTP em Perfect Pairs	RTP em 21+3
I	110461988	100000000	100000000	99,991	99,775	99,536

Fonte: Autor

4.8 Proposta de Equilíbrio Para o *Roulette*

A proposta para equilibrar o *Roulette* é trivial tanto para as probabilidades do jogo quanto para o RTP, a única vantagem do cassino nesse jogo é o número zero, então removê-lo tornaria o jogo justo e os pagamentos seriam, outra forma ou então os pagamentos poderiam serem feitos conforme a tabela abaixo

Tabela 54: Tipos de apostas e propostas de pagamentos para o jogo *Roulette*.

Tipo de aposta	Pagamento
Direta	36:1
Dividir	18:1
Rua	12:1
Canto	9:1
Linha	6:1

Coluna	2,08333:1
Dúzia	2,08333:1
Vermelho/Preto	1,0555:1
Par/Ímpar	1,0555:1
1-18/19-36	1,0555:1

Fonte: Autor.

4.9 Estimativa de Lucro Para *Roulette*

Para a estimativa de lucro do cassino, considerou-se uma base unitária de apostas (a menor aposta permitida em cada tipo de aposta) e o número máximo de jogos que um jogador pode realizar. Assim, cada jogador possui uma banca limitada, calculada multiplicando-se o número de jogos pelo valor da aposta unitária. Esse valor máximo de banca indica que o jogador poderá falir após determinado número de jogos, sendo obrigado a deixar a mesa. Essa abordagem visa simular uma situação próxima à realidade, considerando a rotatividade dos jogadores na mesa de *Roulette*, onde no caso extremo, alguns podem jogar apenas uma vez antes de sair.

Para verificar a validade das simulações e dos cálculos, utilizou-se a o número de jogos calculados anteriormente para cada tipo de aposta, assegurando que os resultados obtidos estejam alinhados com o esperado. Os resultados para diferentes tipos de aposta foram

Tabela 55: Estimativa de lucro para tipos de aposta do jogo *Roulette* na prevalência da lei dos grandes números.

Tipo de Aposta	Número de Jogadores	Banca Inicial por Jogador (R$)	Lucro do Cassino (R$)	Quantidade de Partidas	Apostas Vitoriosas	Apostas Perdedoras
Direta	100.000	35,00	78.065,00	14	34.017	1.222.333
Dividir	100.000	17,50	40.725,00	7	34.008	594.426
Rua	100.000	12,50	40.247,00	5	40.183	459.817
Canto	100.000	7,50	18.281,50	3	32.471	267.529

Linha	100.000	5,00	11.538,00	2	32.523	167.477
Coluna e Dúzia	10.000	20,00	3.950,50	8	25.921	53.440

Fonte: Autor

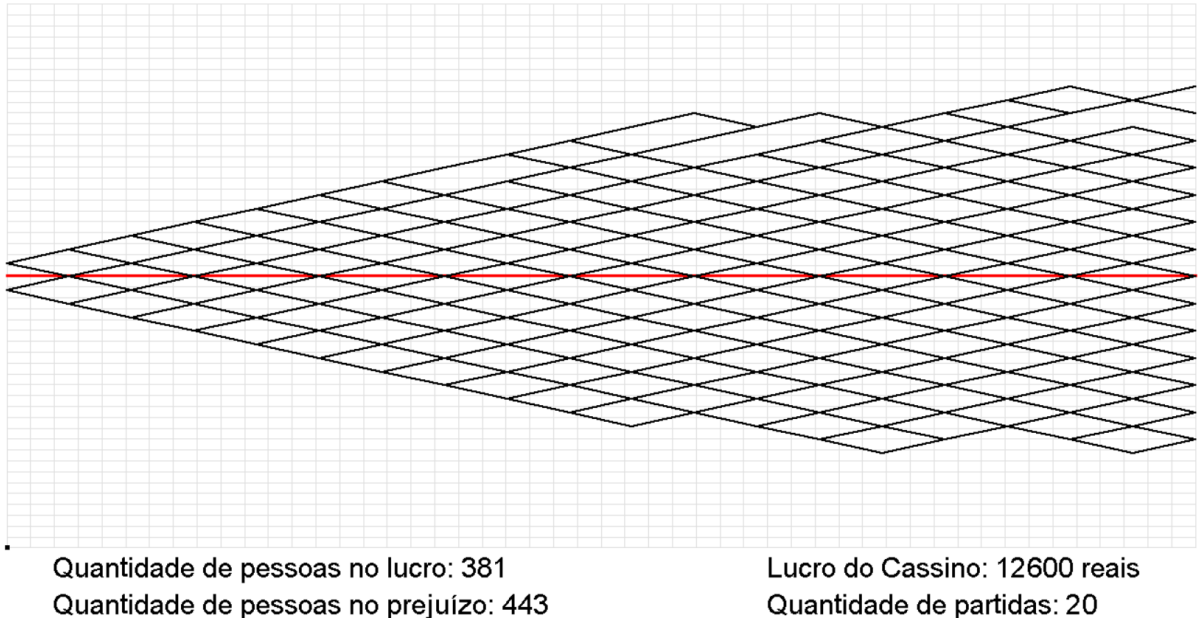
Para as apostas de Vermelho/Preto, Par/Ímpar e 1-18/19-36, utilizou-se uma situação real, sem base unitária, observando dados coletados a partir do site de apostas Blaze. A coleta foi feita em dias e horários aleatórios para garantir uma amostra representativa, considerando jogos gerados por programas de computador. Além disso, entrevistas com dealers ajudaram a esclarecer o funcionamento das mesas. Os dealers relataram que, ao final de cada turno de trabalho (8 horas), recebem relatórios detalhados sobre o número de jogos realizados. Cada jogo é contabilizado sempre que há apostas e pagamentos, independentemente do número de jogadores.

Para a estimativa, assumiu-se um fluxo reduzido de jogadores, com cerca de 30 jogadores por dia, cada um com uma banca máxima de 50 vezes o valor mínimo da aposta. Embora mesas de *Blackjack* e *Roulette* permitam apostas de até R$50.000, o valor médio mais comum é R$25,00. Portanto, foi utilizado como base a estimativa de lucro considerando uma aposta de R$25,00 por jogador.

Além dos dados de uma mesa de cassino, foram consideradas simulações para o jogo de *Roulette* com apostas populares como Vermelho/Preto, Par/Ímpar e 1-18/19-36. As simulações abrangeram situações em que a LGN prevalece e não prevalece. Realizaram-se duas simulações representando um mês, uma com o número de jogos e valores abaixo da LGN e outra acima, de modo a observar o comportamento estatístico esperado.

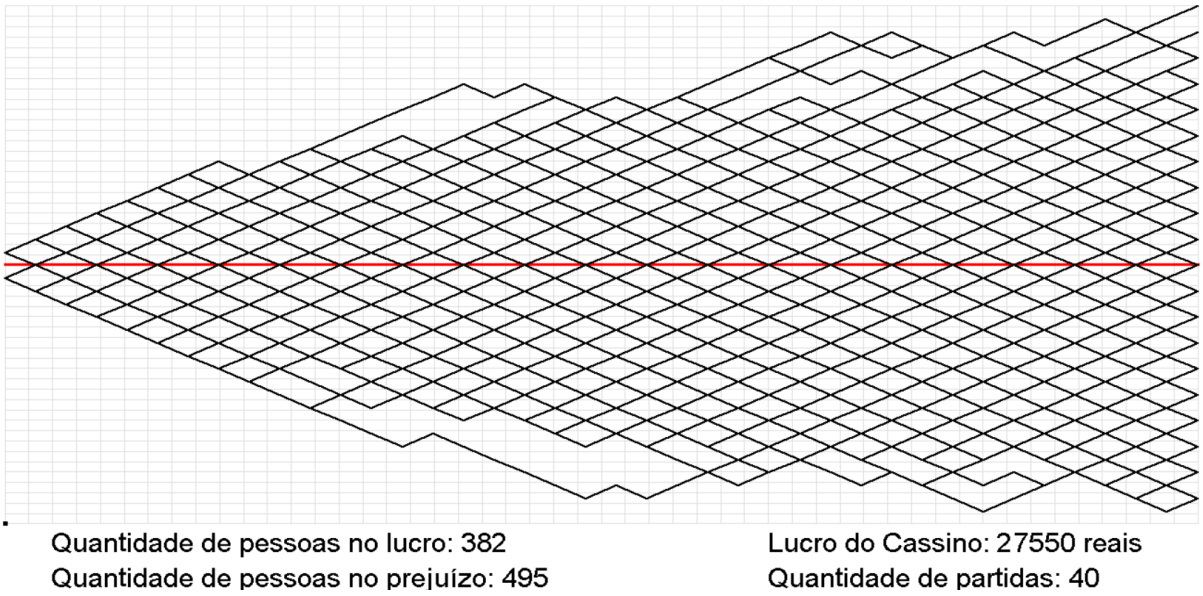
Gráfico 4: Simulação com número de jogos inferior a prevalência da lei dos grandes

números.



Fonte: Autor e José Victor Dias Pereira.

Gráfico 5: Simulação com número de jogos superior a prevalência da lei dos grandes números.



Fonte: Alterado José Victor Dias Pereira.

5 CONCLUSÃO

Este trabalho demonstrou como os cassinos, tanto físicos quanto digitais, garantem lucros consistentes a longo prazo por meio de uma combinação de estratégias probabilísticas e comportamentais. A aplicação de conceitos como a LGN e simulações de Monte Carlo validou que, apesar das flutuações de curto prazo, a vantagem do cassino se consolida, permitindo calcular o número mínimo de jogos para atingir uma rentabilidade esperada com confiabilidade determinada, o que confirma o predomínio do cassino sobre os jogadores a longo prazo.

Além disso, este estudo desenvolveu um método computacional para calcular o RTP em jogos como *Blackjack*. Cálculos analíticos e simulações foram usados para entender o impacto dos multiplicadores nos resultados dos jogos e na probabilidade de vitória. Foram propostos novos multiplicadores tanto no *Blackjack* quanto nas apostas laterais e também nas apostas de *Roulette*, com o intuito de aproximar o RTP de 100%. Essa abordagem revelou a capacidade do cassino de ajustar a margem de lucro específica para cada tipo de aposta, além de evidenciar onde o mesmo determina a intensidade e temporalidade do lucro, de acordo com a LGN e confiabilidade.

Para aprofundar a análise de lucros potenciais, uma simulação baseada no problema do caminhante aleatório foi implementada. Essa simulação possibilita estimar o lucro do cassino ao expressar cada jogo como um cenário de vitória ou derrota e ao considerar apenas um tipo de multiplicador. Esta abordagem simplificada permite uma visão prática do impacto das diferentes apostas e multiplicador na receita do cassino, mesmo considerando a complexidade de jogos como o *Blackjack*, que incluem várias possibilidades de desfecho além de simples vitórias e derrotas.

Assim, conclui-se que os cassinos utilizam uma combinação sofisticada de estratégias matemáticas para maximizar seus lucros. O trabalho não só valida a eficácia dessas estratégias, mas também aponta quais multiplicadores distorcem o jogo do equilíbrio em prol do cassino. As contribuições deste estudo abrem caminho para abordagens inovadoras que podem promover uma experiência de jogo mais transparente e ressaltam a importância de uma participação consciente nos jogos de azar.

6 TRABALHOS FUTUROS

Diversas frentes de pesquisa e desenvolvimento podem ser exploradas a fim de expandir o escopo deste estudo e aprofundar a análise de jogos de azar. Uma dessas possibilidades é a aplicação da metodologia aqui desenvolvida em outros jogos de azar populares, *Baccarat*, jogo do bicho, e também em apostas esportivas. O objetivo seria determinar o número necessário de partidas para que a LGN garanta o lucro do cassino em cada um desses jogos, utilizando modelagens matemáticas semelhantes às aplicadas no *Roulette*.

Outro passo importante é a implementação computacional de simulações para esses outros jogos, permitindo a análise detalhada das probabilidades e estratégias aplicáveis a cada um deles. Isso possibilitará a adaptação das abordagens de simulação e de cálculo de retorno, tornando o estudo mais abrangente.

Além disso, é interessante estudar a influência da estratégia de apostas *Martingale* que consiste considerar que em momentos de sucessivas perdas, o jogador dobre sua aposta na expectativa que um movimento a seu favor reverta os resultados e compense os prejuízos anteriores. Determinar uma expectativa de sobrevivência dessa estratégia nos jogos de *Roulette* e no *Blackjack*, bem como nos demais jogos futuramente implementados. A análise da expectativa matemática para os jogadores que utilizam essa estratégia pode oferecer novas ideias sobre os riscos associados e as chances reais de sucesso, considerando diferentes cenários e limites de aposta.

Outro aspecto a ser investigado é o impacto da contagem de cartas no *Blackjack* ou qualquer outro jogo de cartas. Um programa computacional poderia ser desenvolvido para estudar essa estratégia e como ela altera as probabilidades em jogos com diferentes números de baralhos (1, 2, 4, 6 e 8) como por exemplo o *Blackjack*. A implementação desse modelo seria imprescindível para avaliar a eficácia da contagem de cartas em situações reais e verificar o impacto possível para números de baralhos diferentes e determinar sua influência nas probabilidades de vitória e retorno, especialmente em cassinos que utilizam múltiplos baralhos.

REFERÊNCIAS

[1] JOGO. In: Oxford Languages and Google - Portuguese. Oxford University Press, 2023.

[2] Richard A. Epstein - **The Theory of Gambling and Statistical Logic**, Second Edition, Academic Press (2009)

[3] HUIZINGA, J. Homo Ludens - O jogo como elemento da cultura. Perspectiva S.A, 2000.

[4] Douglas C. MONTGOMERY, George C. RUNGER - **Estatística Aplicada e Probabilidade para Engenheiros**, LTC (2021)

[5] SEBBANE, M. Board Games from Canaan in the Early and Intermediate Bronze Ages and the Origin of the Egyptian Senet Game. Tel Aviv, 28(2), 213-230, Routledge, 2001.

[6] PICCIONE, P. A. The egyptian game of senet and the migration of the soul. Ancient Board Games in Perspective, 54–63, British Museum Press, 2007.

[7] **Curiosidades Sobre Os Casinos E Os Seus Principais Jogos!** (n.d.). Disponível em:https://noctulachannel.com/curiosidades-sobre-os-casinos-e-os-seus-principais-jogos. Acesso em 07/09/2024

[8] **The History of Casino Games.** (n.d.), Disponível em: http://www.librarypreservation.org/history-casino-games. Acesso em 07/09/2024

[9] **História do Casino.** (n.d.) Disponível em: https://casinoportugal.com/historia. Acesso em 07/09/2024

[10] **Casino**. (n.d.) Disponível em: https://pt.wikipedia.org/wiki/Casino. Acesso em 07/09/2024

[12] **Entenda a matemática por trás dos CASSINOS**! Disponível em: https://youtu.be/2jQuuCMRpZk?si=KMEYeiYDjxd-Ova8 Acesso em 07/09/2024

[13] **ALANA TRAUCZYNSKI** [RECALCULANDO A ROTA] | SAPIENSCAST #144 Disponível em: https://www.youtube.com/live/U1jDw5S1eTw?si=UKiTPd97Pah4Yxjb Acesso em 07/09/2024

[14] **ANDREI MAYER (NEUROCIENTISTA) - Lutz Podcast** #25 Disponível em: https://www.youtube.com/watch?v=EiU_7MSrolw. Acesso em 07/09/2024

[15] **Determinismo lógico** Disponível em: https://pt.wikipedia.org/wiki/Determinismo_lógico. Acesso em 07/09/2024

[16] **Lógica probabilística** Disponível em: https://pt.wikipedia.org/wiki/Lógica_probabilística. Acesso em 07/09/2024

[17] **Barbosa, Fábio Daniel Moreira Probabilistic propositional logic** Disponível em: https://ria.ua.pt/handle/10773/22198. Acesso em 07/09/24

[18] **Buckner and Zupko, Duns Scotus on Time and Existence: The Questions on Aristotle's 'De interpretatione'**, translated with Introduction and Commentary by Edward Buckner and Jack Zupko, Washington, DC: Catholic University of America Press, 2014, p. 318

[19] **Melchers, Beck - Structural Reliability Analysis and Prediction**, Third edition, WILEY (2018)

[20] BRIDGE. In: Oxford Languages and Google - Portuguese. Oxford University Press, 2023.

[21] **Blaze.** Disponível em: https://blaze.com/pt/. Acesso em: 11/11/2024

[22] **Biblioteca GLS** disponível em: https://www.gnu.org/software/gsl/ Acesso em: 11/11/2024

[23] **Roleta** disponível em: https://pt.wikipedia.org/wiki/Roleta Acesso em: 11/11/2024

[24] **Craps** disponível em: https://en.wikipedia.org/wiki/Craps Acesso em 11/11/2024

[25] **Vecteezy** disponível em: www.vecteezy.com/ Acesso em 11/11/2024

APÊNDICES

APÊNDICE A – AVALIAÇÃO DE ESTRATÉGIAS DE *BLACKJACK* SIMPLIFICADO UTILIZANDO SIMULAÇÃO MONTE CARLO EM LINGUAGEM *PYTHON*.

```
import random

n = 8  # número de baralhos

# Representação dos n baralhos com naipes
def criar_baralho():
    baralho = []
    naipes = ['Ouros', 'Copas', 'Espadas', 'Paus']
    valores_cartas = [2, 3, 4, 5, 6, 7, 8, 9, 10, 'J', 'Q', 'K', 'A']
    for valor in valores_cartas:
        for naipe in naipes:
            quantidade = n
            baralho.extend([(valor, naipe)] * quantidade)
    random.shuffle(baralho)
    return baralho

# Função para calcular a posição do corte no baralho
def cortar_baralho(baralho):
    meio = len(baralho) // 2
    desvio_padrao = 2
    pos_corte = round(random.gauss(meio, desvio_padrao))

    # Garantir que a posição do corte está dentro dos limites do baralho
    pos_corte = max(0, min(len(baralho) - 1, pos_corte))  # "min(len(baralho) - 1)" é o índice da última carta

    return pos_corte
```

```
def valor_mao(mao):
    valor = 0
    num_ases = 0

    for carta, naipe in mao:
        if carta == 'A':
            num_ases += 1
            valor += 11
        elif carta in ['J', 'Q', 'K']:
            valor += 10
        else:
            valor += carta

    while valor > 21 and num_ases >= 1:
        valor -= 10
        num_ases -= 1

    return valor

def num_ases(mao):
    valor = 0
    num_ases = 0

    for carta, naipe in mao:
        if carta == 'A':
            num_ases += 1
            valor += 11
        elif carta in ['J', 'Q', 'K']:
            valor += 10
        else:
            valor += carta

    while valor > 21 and num_ases >= 1:
        valor -= 10
```

```
            num_ases -= 1

        return num_ases

    def verificar_Blackjack(mao):
        valores = [carta for carta, naipe in mao]
        if 'A' in valores and any(valor in valores for valor in ['J', 'Q', 'K', 10]):
            return True
        return False

    def verificar_pares_perfeitos(mao):
        if len(mao) != 2:
            return None
        (valor1, naipe1), (valor2, naipe2) = mao
        if valor1 == valor2:
            if naipe1 == naipe2:
                return 'Par Perfeito'
            elif (naipe1 in ['Copas', 'Ouros'] and naipe2 in ['Ouros', 'Copas']) or (naipe1 in
['Paus', 'Espadas'] and naipe2 in ['Espadas', 'Paus']):
                return 'Par Colorido'
            else:
                return 'Par Misto'
        return None

    def valor_numerico(carta):
        if carta == 'A':
            return 14
        elif carta == 'K':
            return 13
        elif carta == 'Q':
            return 12
        elif carta == 'J':
            return 11
        else:
```

```
            return int(carta)

        def valor_numerico(valor):
            if valor in ['J', 'Q', 'K']:
                return {'J': 11, 'Q': 12, 'K': 13}[valor]
            elif valor == 'A':
                return 14
            return int(valor)

        def verificar_21_mais_3(mao, carta_crupie):
            mao_completa = mao + [carta_crupie]
            valores = [carta[0] for carta in mao_completa]
            naipes = [carta[1] for carta in mao_completa]

            # Converter valores das cartas de face para valores numéricos
            valores_numericos = [valor_numerico(valor) for valor in valores]
            valores_ordenados = sorted(valores_numericos)

            # Verificar Suited Trips
            if (valores_ordenados[0] == valores_ordenados[1] == valores_ordenados[2]) and
(naipes[0] == naipes[1] == naipes[2]):
                return 'Suited Trips'

            # Verificar Straight Flush
            if (naipes[0] == naipes[1] == naipes[2]):
                if valores_ordenados[2] - valores_ordenados[0] == 2 and
len(set(valores_ordenados)) == 3:
                    return 'Straight Flush'

            # Verificar Three of a Kind
            if (valores_ordenados[0] == valores_ordenados[1] == valores_ordenados[2]) and
not (naipes[0] == naipes[1] == naipes[2]):
                return 'Three of a Kind'
```

```
        # Verificar Straight
        if (valores_ordenados[2] - valores_ordenados[0] == 2 and
len(set(valores_ordenados)) == 3) and not naipes[0] == naipes[1] == naipes[2]:
            return 'Straight'

        # Verificar Flush
        if (naipes[0] == naipes[1] == naipes[2]) and not (valores_ordenados[2] -
valores_ordenados[0] == 2 and len(set(valores_ordenados))):
            return 'Flush'

        return None

    def jogar_Blackjack(jogador, crupie, baralho, resultados, ks):
        # Calcular valores das mãos
        valor_jogador = valor_mao(jogador)
        valor_crupie = valor_mao(crupie)
        primeira_carta_crupie = crupie[0]

        # Verificar as apostas laterais
        par_perfeito = verificar_pares_perfeitos(jogador)
        aposta_21_mais_3 = verificar_21_mais_3(jogador, primeira_carta_crupie)

        # Atualizar resultados das apostas laterais
        if par_perfeito:
            resultados['pares_perfeitos'][par_perfeito] += 1
        if aposta_21_mais_3:
            resultados['21_mais_3'][aposta_21_mais_3] += 1

        # Verificar se o crupiê já tem Blackjack
        crupie_Blackjack = verificar_Blackjack(crupie)
        jogador_Blackjack = verificar_Blackjack(jogador)

        if crupie_Blackjack and jogador_Blackjack:
            resultados['empate_com_Blackjack'] += 1
```

```
            return 'empate_com_Blackjack'

        elif (len(crupie) == 2 and valor_mao(crupie) == 21) and not (len(jogador) != 2 and
valor_mao(jogador) == 21):
            resultados['crupie_Blackjack'] += 1
            return 'crupie_Blackjack'

        elif (len(jogador) == 2 and valor_mao(jogador) == 21) and not (len(crupie) != 2 and
valor_mao(crupie) == 21):
            resultados['jogador_Blackjack'] += 1
            return 'jogador_Blackjack'

        # Determinar o valor de k com base na primeira carta do crupiê
        if ((num_ases(jogador) != 0) and (valor_mao(jogador) <= 21)) and
primeira_carta_crupie[0] in kssoft:
            k = kssoft[primeira_carta_crupie[0]]
            while (valor_mao(jogador) < k and (num_ases(jogador) != 0)):
                jogador.append(baralho.pop(0))
                valor_jogador = valor_mao(jogador)

        elif (valor_mao(jogador) <= 21) and primeira_carta_crupie[0] in ks:
            k = ks[primeira_carta_crupie[0]]
            while valor_mao(jogador) < k:
                jogador.append(baralho.pop(0))
                valor_jogador = valor_mao(jogador)

            while valor_mao(crupie) < 17:
                crupie.append(baralho.pop(0))
                valor_crupie = valor_mao(crupie)

        #print(f'Mão 1 do jogador depois das decisões: {jogador} com valor
{valor_mao(jogador)}')
        #print(f'Mão do crupiê depois das decisões: {crupie} com valor
{valor_mao(crupie)}')
```

```
#print(num_ases(jogador))

## Caso em que há empate
if (valor_mao(jogador) <= 21) and (valor_mao(jogador) == valor_mao(crupie)):
    resultados['empate'] += 1
    return 'empate'

## Caso onde o crupie tem um Blackjack tardio
elif len(crupie) == 2 and valor_mao(crupie) == 21 and valor_mao(jogador) <= 21:
    resultados['crupie_Blackjack'] += 1
    return 'crupie_Blackjack'

## Caso em que o jogador estoura
elif (valor_mao(jogador) > 21):
    resultados['jogador_bust'] += 1
    return 'crupie'

## Caso em que o crupiê estoura e o jogador não
elif (valor_mao(jogador) <= 21) and (valor_mao(crupie) > 21):
    resultados['crupie_bust'] += 1
    return 'jogador'

## Caso em que o jogador ganha por comparação
elif (valor_mao(jogador) <= 21) and (valor_mao(jogador) > valor_mao(crupie)):
    resultados['jogador'] += 1
    return 'jogador'

## Caso em que o crupiê ganha por comparação
elif (valor_mao(jogador) <= 21) and (valor_mao(jogador) < valor_mao(crupie)):
    resultados['crupie'] += 1
    return 'crupie'

## Caso blackjack tardio do crupiê
elif len(crupie) == 2 and (valor_mao(crupie) == 21) and (valor_mao(jogador) <=
```

```
21):
                resultados['crupie_Blackjack'] += 1
                return 'crupie_Blackjack'


        def simular_jogos(numero_de_simulacoes, ks):
            resultados = {chave: {'jogador': 0, 'crupie': 0, 'empate': 0, 'empate_com_Blackjack': 0, 'jogador_Blackjack': 0, 'crupie_Blackjack': 0, 'jogador_bust': 0, 'crupie_bust': 0, 'pares_perfeitos': {'Par Perfeito': 0, 'Par Colorido': 0, 'Par Misto': 0}, '21_mais_3': {'Suited Trips': 0, 'Straight Flush': 0, 'Three of a Kind': 0, 'Straight': 0, 'Flush': 0}} for chave in ks.keys()}
            baralho = criar_baralho()
            pos_corte = cortar_baralho(baralho)

            for i in range(numero_de_simulacoes):
                # Resetar as mãos do jogador e crupiê
                jogador = [baralho.pop(0), baralho.pop(1)]
                crupie = [baralho.pop(0), baralho.pop(0)]

                # Testar mãos fixas
                #jogador = [("A", 'Ouros'), ('J', 'Espadas')]
                #crupie = [('A', 'Espadas'), ("Q", 'Ouros')]

                # Printar a mão do jogador
                #print(f'mão do jogador: {jogador} com valor inicial {valor_mao(jogador)}')
                # Printar a mão do crupie
                #print(f'mão do crupiê: {crupie} com a primeira carta: {crupie[0][0]}')

                # Verificar se há cartas suficientes no baralho, senão reembaralhar
                if len(baralho) < pos_corte:
                    baralho = criar_baralho()
                    pos_corte = cortar_baralho(baralho)

                primeira_carta_crupie = crupie[0]
```

```
            if primeira_carta_crupie[0] in ks:
                resultado = jogar_Blackjack(jogador, crupie, baralho,
resultados[primeira_carta_crupie[0]], ks)
            else:
                resultado = jogar_Blackjack(jogador, crupie, baralho, {},
{primeira_carta_crupie[0]: 17})  # Usar uma estratégia padrão para os outros casos

        for chave in ks.keys():
            total_de_jogos_para_n_carta = (
                resultados[chave]['jogador'] +
                resultados[chave]['jogador_Blackjack'] +
                resultados[chave]['crupie_bust'] +
                resultados[chave]['crupie'] +
                resultados[chave]['crupie_Blackjack'] +
                resultados[chave]['jogador_bust'] +
                resultados[chave]['empate']
            )
            wins_jogador = (
                resultados[chave]['jogador'] +
                resultados[chave]['jogador_Blackjack'] +
                resultados[chave]['crupie_bust']
            )
            wins_crupie = (
                resultados[chave]['crupie'] +
                resultados[chave]['crupie_Blackjack'] +
                resultados[chave]['jogador_bust']
            )
            empates = (
                resultados[chave]['empate'] +
                resultados[chave]['empate_com_Blackjack']
            )
            total_Blackjack_jogador = (
                resultados[chave]['jogador_Blackjack'] +
                resultados[chave]['empate_com_Blackjack']
```

```
            )

            total_Blackjack_crupie = (
                resultados[chave]['crupie_Blackjack'] +
                resultados[chave]['empate_com_Blackjack']
            )

            # Vitórias do jogador por carta em formato de porcentagem
            wins_jogador_porcentagem = (wins_jogador / (total_de_jogos_para_n_carta)) *
100

            # Vitórias do crupiê por carta em formato de porcentagem
            wins_crupie_porcentagem = (wins_crupie / total_de_jogos_para_n_carta) * 100

            # Empates por carta em formato de porcentagem
            empates_porcentagem = (empates / total_de_jogos_para_n_carta) * 100

            # Número de vezes que o jogador teve Blackjack por carta
            Blackjacks_jogador = resultados[chave]['jogador_Blackjack']

            # Número de vezes que o crupiê teve Blackjack por carta
            Blackjacks_crupie = resultados[chave]['crupie_Blackjack']

            # Número de vezes que o jogador estourou por carta
            bust_jogador = resultados[chave]['jogador_bust']

            # Número de vezes que o crupiê estourou por carta
            bust_crupie = resultados[chave]['crupie_bust']

            # Porcentagem de jogos que o jogador teve um Blackjack por carta
            Blackjacks_jogador_porcentagem = (Blackjacks_jogador /
total_de_jogos_para_n_carta) * 100

            # Porcentagem de vezes que o crupiê teve um Blackjack por carta
```

```
            Blackjacks_crupie_porcentagem = (Blackjacks_crupie /
total_de_jogos_para_n_carta) * 100

            # Porcentagem de vezes que o jogador estourou por carta
            bust_jogador_porcentagem = (bust_jogador / total_de_jogos_para_n_carta) *
100

            # Porcentagem de vezes que o crupiê estourou por carta
            bust_crupie_porcentagem = (bust_crupie / total_de_jogos_para_n_carta) * 100

            # Número total de pares perfeitos, misto e colorido somados por carta
            pares_perfeitos = resultados[chave]['pares_perfeitos']

            # Número total apenas de pares perfeitos por carta
            par_perfeito = pares_perfeitos['Par Perfeito']

            # Número total apenas de pares coloridos por carta
            par_colorido = pares_perfeitos['Par Colorido']

            # Npumero total apenas de pares mistos por carta
            par_misto = pares_perfeitos['Par Misto']

            # Número total de 21+3 por carta
            aposta_21_mais_3 = resultados[chave]['21_mais_3']

            # Número total de suited trips por carta
            suited_trips = aposta_21_mais_3['Suited Trips']

            # Número total de straight flush por carta
            straight_flush = aposta_21_mais_3['Straight Flush']

            # Número total de three of a kind por carta
            three_of_a_kind = aposta_21_mais_3['Three of a Kind']
```

```
            # Número total de straight por carta
            straight = aposta_21_mais_3['Straight']

            # Número total de flush por carta
            flush = aposta_21_mais_3['Flush']

            # Print dos resultados por carta
            print(f"Resultados para crupiê com carta inicial {chave}:", "(Total de", total_de_jogos_para_n_carta, "jogo(s))")
            print(f"Vitórias do jogador: {wins_jogador} ({wins_jogador_porcentagem:.4f}%)")
            print(f"Vitórias do crupiê: {wins_crupie} ({wins_crupie_porcentagem:.4f}%)")
            print(f"Empates: {empates} ({empates_porcentagem:.4f}%)")
            print(f"Blackjacks do jogador: {Blackjacks_jogador} ({Blackjacks_jogador_porcentagem:.4f}%)")
            print(f"Blackjacks do crupiê: {Blackjacks_crupie} ({Blackjacks_crupie_porcentagem:.4f}%)")
            print(f"Jogador estourou: {bust_jogador} ({bust_jogador_porcentagem:.4f}%)")
            print(f"Crupiê estourou: {bust_crupie} ({bust_crupie_porcentagem:.4f}%)")
            print("-------------------------------------------------------")

        # Cálculo do total de apostas laterais (Perfect Pairs e 21+3) de todas as simulações
        total_pares_perfeitos = sum(resultados[chave]['pares_perfeitos']['Par Perfeito'] + resultados[chave]['pares_perfeitos']['Par Colorido'] + resultados[chave]['pares_perfeitos']['Par Misto'] for chave in ks.keys())
        total_pares_perfeitos_misto = sum(resultados[chave]['pares_perfeitos']['Par Misto'] for chave in ks.keys())
        total_pares_perfeitos_colorido = sum(resultados[chave]['pares_perfeitos']['Par Colorido'] for chave in ks.keys())
        total_pares_parfeitos_perfeitos = sum(resultados[chave]['pares_perfeitos']['Par Perfeito'] for chave in ks.keys())
        total_21_mais_3 = sum(resultados[chave]['21_mais_3']['Suited Trips'] + resultados[chave]['21_mais_3']['Straight Flush'] + resultados[chave]['21_mais_3']['Three of a Kind'] + resultados[chave]['21_mais_3']['Straight'] + resultados[chave]['21_mais_3']['Flush']
```

```
for chave in ks.keys())
        total_21_mais_3_suited_trips = sum(resultados[chave]['21_mais_3']['Suited Trips'] for chave in ks.keys())
        total_21_mais_3_straight_flush = sum(resultados[chave]['21_mais_3']['Straight Flush'] for chave in ks.keys())
        total_21_mais_3_three_of_a_kind = sum(resultados[chave]['21_mais_3']['Three of a Kind'] for chave in ks.keys())
        total_21_mais_3_straight = sum(resultados[chave]['21_mais_3']['Straight'] for chave in ks.keys())
        total_21_mais_3_flush = sum(resultados[chave]['21_mais_3']['Flush'] for chave in ks.keys())
        total_jogador_bust = sum(resultados[chave]['jogador_bust'] for chave in ks.keys())
        total_crupie_bust = sum(resultados[chave]['crupie_bust'] for chave in ks.keys())

        # Cálculo do total de apostas laterais (Perfect Pairs e 21+3) em porcentagem de totas as simulações
        porcentagem_total_pares_perfeitos = (total_pares_perfeitos / numero_de_simulacoes) * 100
        porcentagem_total_pares_perfeitos_misto = (total_pares_perfeitos_misto / numero_de_simulacoes) * 100
        porcentagem_total_pares_perfeitos_colorido = (total_pares_perfeitos_colorido / numero_de_simulacoes) * 100
        porcentagem_total_pares_parfeitos_perfeitos = (total_pares_parfeitos_perfeitos / numero_de_simulacoes) * 100
        porcentagem_total_21_mais_3 = (total_21_mais_3 / numero_de_simulacoes) * 100
        porcentagem_total_21_mais_3_suited_trips = (total_21_mais_3_suited_trips / numero_de_simulacoes) * 100
        porcentagem_total_21_mais_3_straight_flush = (total_21_mais_3_straight_flush / numero_de_simulacoes) * 100
        porcentagem_total_21_mais_3_three_of_a_kind = (total_21_mais_3_three_of_a_kind / numero_de_simulacoes) * 100
        porcentagem_total_21_mais_3_straight = (total_21_mais_3_straight / numero_de_simulacoes) * 100
        porcentagem_total_21_mais_3_flush = (total_21_mais_3_flush /
```

```
numero_de_simulacoes) * 100
        porcentagem_total_jogador_bust = (total_jogador_bust / numero_de_simulacoes) *
100
        porcentagem_total_crupie_bust = (total_crupie_bust / numero_de_simulacoes) *
100

        # Calcular total de vitórias do crupiê
        total_vitorias_crupie = sum(resultados[chave]['crupie'] +
resultados[chave]['crupie_Blackjack'] + resultados[chave]['jogador_bust'] for chave in
ks.keys())
        total_jogos = numero_de_simulacoes

        # Cálculo do total de vitórias do crupiê em porcentagem
        porcentagem_total_vitorias_crupie = (total_vitorias_crupie / total_jogos) * 100

        # Cálculo do total de vitórias do jogador
        total_vitorias_jogador = sum(resultados[chave]['jogador'] +
resultados[chave]['jogador_Blackjack'] + resultados[chave]['crupie_bust'] for chave in
ks.keys())

        # Cálculo total de vitórias do jogador em porcentagem
        porcentagem_total_vitorias_jogador = (total_vitorias_jogador / total_jogos) * 100

        # Cálculo total de empates
        total_empates = sum(resultados[chave]['empate'] +
resultados[chave]['empate_com_Blackjack'] for chave in ks.keys())

        # Cálculo total de empates em porcentagem
        total_empates_porcentagem = (total_empates / total_jogos) * 100

        # Cálculo total de Blackjack do jogador
        total_Blackjack_jogador = sum(resultados[chave]['jogador_Blackjack'] +
resultados[chave]['empate_com_Blackjack'] for chave in ks.keys())
```

```
        #Cálculo total de Blackjack do jogador em porcentagem
        total_Blackjack_jogador_porcentagem = (total_Blackjack_jogador / numero_de_simulacoes) * 100

        # Cálculo total de Blackjack do crupie
        total_Blackjack_crupie = sum(resultados[chave]['crupie_Blackjack'] + resultados[chave]['empate_com_Blackjack'] for chave in ks.keys())

        #Cálculo total de Blackjack do jogador em porcentagem
        total_Blackjack_crupie_porcentagem = (total_Blackjack_crupie / numero_de_simulacoes) * 100

        # Print das porcentagens
        print(f"Porcentagem total de vitórias do crupiê: {porcentagem_total_vitorias_crupie:.4f}%")
        print(f"Porcentagem total de vitórias do jogador: {porcentagem_total_vitorias_jogador:.4f}%")
        print(f"Porcentagem total de empates: {total_empates_porcentagem:.4f}%")
        print(f"Porcentagem total que o cupiê estourou: {porcentagem_total_crupie_bust:.4f}%")
        print(f"Porcentagem total que o jogador estourou: {porcentagem_total_jogador_bust:.4f}%")
        print(f"Porcentagem total que o crupiê tinha Blackjack: {total_Blackjack_crupie_porcentagem:.4f}%")
        print(f"Porcentagem total que o jogador tinha Blackjack: {total_Blackjack_jogador_porcentagem:.4f}%")

        # Normalização com a anulação dos empates
        total_jogos_sem_empates = total_jogos - total_empates

        # Cálculo do percentual de vitórias do crupiê excluindo os empates
        porcentagem_total_vitorias_crupie = (total_vitorias_crupie / total_jogos_sem_empates) * 100
```

```
        # Cálculo do percentual de vitórias do jogador excluindo os empates
        porcentagem_total_vitorias_jogador = (total_vitorias_jogador / total_jogos_sem_empates) * 100

        # Print da porcentagem excluindo os empates
        print(f"Total de vitórias do crupiê (excluindo empates): {total_vitorias_crupie} ({porcentagem_total_vitorias_crupie:.4f}%)")
        print(f"Total de vitórias do jogador (excluindo empates): {total_vitorias_jogador} ({porcentagem_total_vitorias_jogador:.4f}%)")
        print("-------------------------------------------------------")

        # Calcular e imprimir a porcentagem total de pares perfeitos
        print(f"Total de pares perfeitos de todos os tipos somados: {total_pares_perfeitos} ({porcentagem_total_pares_perfeitos:.4f}%)")
        print(f"Total de pares perfeitos (misto): {total_pares_perfeitos_misto} ({porcentagem_total_pares_perfeitos_misto:.4f}%)")
        print(f"Total de pares perfeitos (colorido): {total_pares_perfeitos_colorido} ({porcentagem_total_pares_perfeitos_colorido:.4f}%)")
        print(f"Total de pares perfeitos (perfeito): {total_pares_parfeitos_perfeitos} ({porcentagem_total_pares_parfeitos_perfeitos:.4f}%)")

        # Calcular e imprimir a porcentagem total de 21+3
        print(f"Total de 21+3: {total_21_mais_3} ({porcentagem_total_21_mais_3:.4f}%)")
        print(f"Total de 21+3 (suited three of a kind): {total_21_mais_3_suited_trips} ({porcentagem_total_21_mais_3_suited_trips:.4f}%)")
        print(f"Total de 21+3 (straight flush): {total_21_mais_3_straight_flush} ({porcentagem_total_21_mais_3_straight_flush:.4f}%)")
        print(f"Total de 21+3 (three of a kind): {total_21_mais_3_three_of_a_kind} ({porcentagem_total_21_mais_3_three_of_a_kind:.4f}%)")
        print(f"Total de 21+3 (straight): {total_21_mais_3_straight} ({porcentagem_total_21_mais_3_straight:.4f}%)")
        print(f"Total de 21+3 (flush): {total_21_mais_3_flush} ({porcentagem_total_21_mais_3_flush:.4f}%)")
```

```
    print("-----------------------------------------------------")

    return resultados


# Número de simulações
numero_de_simulacoes = 1000

# Valores de k para cada possível mão do crupiê
ks = {
    2: 13,
    3: 13,
    4: 12,
    5: 12,
    6: 12,
    7: 17,
    8: 17,
    9: 17,
    10: 17,
    'J': 17,
    'Q': 17,
    'K': 17,
    'A': 17
}

# Valores de k para cada possível mão do crupiê com mão soft
kssoft = {
    2: 17,
    3: 18,
    4: 18,
    5: 18,
    6: 18,
    7: 17,
    8: 17,
```

```
    9: 18,
    10: 18,
    'J': 18,
    'Q': 18,
    'K': 18,
    'A': 18
}

resultados = simular_jogos(numero_de_simulacoes, ks)
```

APÊNDICE B – CÓDIGO EM LINGUAGEM C++ PARA ESTIMATIVA DE LUCRO DO CASSINO.

```
#include "PIG.h"
#include <gsl/gsl_rng.h>

#define QUANTIDADE_JOGADORES          (100)
#define QUANTIDADE_MAXIMA_PARTIDAS    (1000)

#define SALDO_INICIAL_JOGADORES       (1000)
#define VALOR_APOSTA_JOGADORES        (1)

#define CHANCE_GANHAR_JOGO            (2.70270)

    gsl_rng*    gerador;
    int         timerGeral;

    int
saldoJogador[QUANTIDADE_JOGADORES][QUANTIDADE_MAXIMA_PARTIDAS];
    int         partidaAtual;

    bool        iniciou;
```

```
    int         lucroCassino;
    int         pessoasGanhandoDinheiro;
    int         pessoasPerdendoDinheiro;
    int         quantidadeVitorias;
    int         quantidadeDerrotas;
    int         fonte, fontePequena;


/// Inicialização

void inicializarSimulacao()
{
    partidaAtual           = 0;
    lucroCassino           = 0;
    pessoasGanhandoDinheiro = 0;
    pessoasPerdendoDinheiro = 0;
    quantidadeDerrotas     = 0;
    quantidadeVitorias     = 0;
    iniciou                = false;
}

void configuracoesIniciais()
{
    PIG_criarJanela("Simulação Cassino", 1440, 750);

    gsl_rng_env_setup();
    gerador = gsl_rng_alloc(gsl_rng_default);
    gsl_rng_set (gerador, time(NULL));

    timerGeral             = PIG_criarTimer();

    fonte                  = PIG_criarFonte("arial.ttf",30,PRETO,0,PRETO);
    fontePequena           = PIG_criarFonte("arial.ttf",15,PRETO,0,PRETO);
```

```
        inicializarSimulacao();
    }

    /// Atualização

    void atualizarLucroCassino()
    {
        pessoasGanhandoDinheiro = 0;
        pessoasPerdendoDinheiro = 0;
        lucroCassino = 0;
        for(int i=0; i<QUANTIDADE_JOGADORES; i++)
        {
            double prejuizoJogador = (SALDO_INICIAL_JOGADORES - saldoJogador[i][partidaAtual]);
            if(prejuizoJogador > 0)
            {
                pessoasPerdendoDinheiro++;
            }
            if(prejuizoJogador < 0)
            {
                pessoasGanhandoDinheiro++;
            }
            lucroCassino = lucroCassino + prejuizoJogador;
        }
    }

    void atualizarSaldoJogadores()
    {
        if(partidaAtual == QUANTIDADE_MAXIMA_PARTIDAS)
        {
            return;
        }

        for(int i=0; i<QUANTIDADE_JOGADORES; i++)
```

```
{
    int saldoAtual;
    if(partidaAtual == 0)
    {
        saldoAtual = SALDO_INICIAL_JOGADORES;
    }
    else
    {
        saldoAtual = saldoJogador[i][partidaAtual-1];
    }

    int apostaAtual;
    if(saldoAtual >= VALOR_APOSTA_JOGADORES)
    {
        apostaAtual = VALOR_APOSTA_JOGADORES;
    }
    else
    {
        apostaAtual = saldoAtual;
    }

    if(apostaAtual > 0)
    {
        double numeroSorteado = 100.0*gsl_rng_uniform(gerador);

        if(numeroSorteado < CHANCE_GANHAR_JOGO)
        {
            saldoJogador[i][partidaAtual] = saldoAtual + 35*apostaAtual; //Aqui
podemos mudar a ODD da aposta
            if(i == 0)
            {
                quantidadeVitorias++;
            }
        }
```

```
            else
            {
                saldoJogador[i][partidaAtual] = saldoAtual - apostaAtual;
                if(i == 0)
                {
                    quantidadeDerrotas++;
                }
            }
        }
        else
        {
            saldoJogador[i][partidaAtual] = 0;
        }
    }
    atualizarLucroCassino();
    partidaAtual++;
}

/// Desenhos

void desenharGrafico()
{
    double xInicial = 90;
    double yInicial = 90;

    double larguraGrafico = LARG_TELA-2*xInicial;
    double alturaGrafico = ALT_TELA-2*yInicial;

    double deslocamento = larguraGrafico/(partidaAtual-1);
    double alturaSaldoInicial = alturaGrafico/2.0;

    PIG_desenharGrade(xInicial, yInicial, larguraGrafico, alturaGrafico, 10, 10,
CINZA_CLARO, NAO);
    PIG_desenharLinha(xInicial, yInicial+alturaSaldoInicial, xInicial + larguraGrafico,
```

```
yInicial+alturaSaldoInicial, PRETO, 3, NAO);

        PIG_desenharPonto(xInicial, yInicial, PRETO, 5, NAO);

        char String[1000];
        sprintf(String, "%.0f\n",SALDO_INICIAL_JOGADORES);
        PIG_escreverDireita(String, xInicial-10, yInicial+alturaSaldoInicial-8,
fontePequena);

        sprintf(String, "Saldo por Partida\n");
        PIG_escreverCentralizada(String, xInicial+(larguraGrafico/2.0),
yInicial+alturaGrafico+30, fonte);

        for(int i=0; i<QUANTIDADE_JOGADORES; i++)
        {
           for(int j=0; j<partidaAtual; j++)
           {
              double valor =
((double)saldoJogador[i][j]/SALDO_INICIAL_JOGADORES)*alturaSaldoInicial;

              double x = xInicial + j*deslocamento;
              double y = yInicial + valor;

              //PIG_desenharPonto(x,y,corJogador[i],1,NAO);

              if(j > 0)
              {
                 double valorA = ((double)saldoJogador[i][j-
1]/SALDO_INICIAL_JOGADORES)*alturaSaldoInicial;
                 double xA = xInicial + (j-1)*deslocamento;
                 double yA = yInicial + valorA;

                 PIG_desenharLinha(x,y,xA,yA,VERMELHO,2,NAO);
                 //PIG_desenharLinha(x,y,xA,yA,corJogador[i],1,NAO);
```

```
            }
        }
    }


}

void desenharInformacoes()
{
    char String[1000];
    sprintf(String,"Lucro do Cassino: %d reais", lucroCassino);
    PIG_escreverEsquerda(String,950,50,fonte);

    sprintf(String,"Quantidade de partidas: %d", partidaAtual);
    PIG_escreverEsquerda(String,950,10,fonte);

    if(QUANTIDADE_JOGADORES == 1)
    {
        double qtdPartidas = quantidadeDerrotas + quantidadeVitorias;
        if(qtdPartidas == 0)
        {
            qtdPartidas = 1;
        }

        sprintf(String,"Quantidade de vitórias: %d (%.1f)", quantidadeVitorias,
100.0*quantidadeVitorias/qtdPartidas);
        PIG_escreverEsquerda(String,100,50,fonte);

        sprintf(String,"Quantidade de derrotas: %d (%.1f)", quantidadeDerrotas,
100.0*quantidadeDerrotas/qtdPartidas);
        PIG_escreverEsquerda(String,100,10,fonte);
    }
    else
    {
```

```
            sprintf(String,"Quantidade de pessoas no lucro: %d",
pessoasGanhandoDinheiro);
            PIG_escreverEsquerda(String,100,50,fonte);

            sprintf(String,"Quantidade de pessoas no prejuízo: %d",
pessoasPerdendoDinheiro);
            PIG_escreverEsquerda(String,100,10,fonte);
        }
    }

    void desenhar()
    {
        PIG_iniciarDesenho();

        PIG_desenharRetangulo(0,0,LARG_TELA,ALT_TELA,BRANCO,NAO,NAO);
        desenharGrafico();
        desenharInformacoes();

        PIG_encerrarDesenho();
    }

    /// Controle da Simulação

    void verificarTeclasApertadas()
    {
        switch(PIG_tecla)
        {
            case TECLA_ENTER:
            {
                iniciou = true;
            }   break;

            case TECLA_BARRAESPACO:
            {
```

```
            inicializarSimulacao();
        }   break;
    }
}



int main(int argc, char* args[])
{
    configuracoesIniciais();

    while(PIG_jogoRodando() == SIM)
    {
        PIG_atualizarJanela();
        verificarTeclasApertadas();

        if(PIG_tempoDecorrido(timerGeral) >= 0.01)
        {
            if(iniciou == true)
            {
                atualizarSaldoJogadores();
            }
            desenhar();

            PIG_reiniciarTimer(timerGeral);
        }
    }

    /// Desalocando e finalizando ---------------------------------
    PIG_finalizarJanela();

    return 0;
}
```

APÊNDICE C – AVALIAÇÃO DO RETORNO ECONÔMICO DE ESTRATÉGIAS DE BLACKJACK UTILIZANDO SIMULAÇÃO MONTE CARLO EM LINGUAGEM *PYTHON*.

```
import random

n = 8  # número de baralhos

# Representação dos n baralhos com naipes
def criar_baralho():
    baralho = []
    naipes = ['Ouros', 'Copas', 'Espadas', 'Paus']
    valores_cartas = [2, 3, 4, 5, 6, 7, 8, 9, 10, 'J', 'Q', 'K', 'A']
    for valor in valores_cartas:
        for naipe in naipes:
            quantidade = n
            baralho.extend([(valor, naipe)] * quantidade)
    random.shuffle(baralho)
    return baralho

# Função para calcular a posição do corte no baralho
def cortar_baralho(baralho):
    meio = len(baralho) // 2
    desvio_padrao = 2
    pos_corte = round(random.gauss(meio, desvio_padrao))

    # Garantir que a posição do corte está dentro dos limites do baralho
    pos_corte = max(0, min(len(baralho) - 1, pos_corte))  # "min(len(baralho) - 1" é o índice da última carta

    return pos_corte

# Verificar o valor da mão
```

```
def valor_mao(mao):
    valor = 0
    num_ases = 0

    for carta in mao:
        valor_carta = carta[0]
        if valor_carta == 'A':
            num_ases += 1
            valor += 11
        elif valor_carta in ['J', 'Q', 'K']:
            valor += 10
        else:
            valor += valor_carta

    while valor > 21 and num_ases >= 1:
        valor -= 10
        num_ases -= 1

    return valor

# Verificar se a mão é um Blackjack
def verificar_Blackjack(mao):
    valores = [carta for carta, naipe in mao]
    if 'A' in valores and any(valor in valores for valor in ['J', 'Q', 'K', 10]):
        return True
    return False

# Verificar pares perfeitos
def verificar_pares_perfeitos(mao1, resultados):
    if len(mao1) != 2:
        return None
    (valor1, naipe1), (valor2, naipe2) = mao1
    if valor1 == valor2:
        if naipe1 == naipe2:
```

```
                resultados['pares_perfeitos'] += 1
                resultados['Par Perfeito'] += 1
                return 'Par Perfeito'
            elif (naipe1 in ['Copas', 'Ouros'] and naipe2 in ['Ouros', 'Copas']) or (naipe1 in ['Paus', 'Espadas'] and naipe2 in ['Espadas', 'Paus']):
                resultados['pares_perfeitos'] += 1
                resultados['Par Colorido'] += 1
                return 'Par Colorido'
            else:
                resultados['pares_perfeitos'] += 1
                resultados['Par Misto'] += 1
                return 'Par Misto'
        elif valor1 != valor2:
            resultados['sem_pares_perfeitos'] += 1
        return None

    # Verificar valor numérico da carta
    def valor_numerico(carta):
        if carta == 'A':
            return 14
        elif carta == 'K':
            return 13
        elif carta == 'Q':
            return 12
        elif carta == 'J':
            return 11
        else:
            return int(carta)

    # Verificar 21 + 3
    def verificar_21_mais_3(mao1, primeira_carta_crupie, resultados):
        mao_completa = mao1 + [primeira_carta_crupie]
        valores = [carta[0] for carta in mao_completa]
        naipes = [carta[1] for carta in mao_completa]
```

```
            # Converter valores das cartas de face para valores numéricos
            valores_numericos = [valor_numerico(valor) for valor in valores]
            valores_ordenados = sorted(valores_numericos)

            # Verificar Suited Trips
            if (valores_ordenados[0] == valores_ordenados[1] == valores_ordenados[2]) and
(naipes[0] == naipes[1] == naipes[2]):
                resultados['21_mais_3'] += 1
                resultados['Suited Trips'] += 1
                return 'Suited Trips'

            # Verificar Straight Flush
            if (naipes[0] == naipes[1] == naipes[2]) and ((valores_ordenados[2] -
valores_ordenados[0] == 2) and (len(set(valores_ordenados)) == 3)):
                    resultados['21_mais_3'] += 1
                    resultados['Straight Flush'] += 1
                    return 'Straight Flush'

            # Verificar Three of a Kind
            if valores_ordenados[0] == valores_ordenados[1] == valores_ordenados[2] and not
(naipes[0] == naipes[1] == naipes[2]):
                resultados['21_mais_3'] += 1
                resultados['Three of a Kind'] += 1
                return 'Three of a Kind'

            # Verificar Straight
            if (valores_ordenados[2] - valores_ordenados[0] == 2 and
len(set(valores_ordenados)) == 3) and not (naipes[0] == naipes[1] == naipes[2]):
                resultados['21_mais_3'] += 1
                resultados['Straight'] += 1
                return 'Straight'

            # Verificar Flush
```

```
            if (naipes[0] == naipes[1] == naipes[2]) and not ((valores_ordenados[2] -
valores_ordenados[0] == 2) and (len(set(valores_ordenados)) == 3)):
                resultados['21_mais_3'] += 1
                resultados['Flush'] += 1
                return 'Flush'

            else:
                resultados['sem_21_mais_3'] += 1

            return None

        # Função para pedir uma carta adicional (hit) para a mão1
        def hit1(mao1, baralho):
            mao1.append(baralho.pop(0))
            return mao1

        # Função para pedir uma carta adicional (hit) para a mão2
        def hit2(mao2, baralho):
            mao2.append(baralho.pop(0))
            return mao2

        # Função para pedir double down para a mão1
        def double_down(mao1, maoD, baralho):
            mao1.append(baralho.pop(0))
            maoD = mao1 # A mão D é igual a mão 1
            return mao1, maoD

        # Função para dividir a mão do jogador em mão 1 e mão 2
        def split(mao1, mao2, baralho):
            mao1 = hit1(mao1, baralho) # coloca a primeira carta do baralho na mão do jogador
            mao1 = hit1(mao1, baralho) # coloca a segunda carta do baralho na mão do jogador
            mao2 = [mao1.pop(1), mao1.pop(2)] # Divide a mão do jogador em duas como no
jogo
            return mao1, mao2
```

```
# Função que determina a estratégia do jogador contra a primeira carta do crupiê
def estrategia_jogador(mao1, mao2, maoD, crupie, baralho):
    valor_mao1 = valor_mao(mao1) # Valor da mão 1 em número
    valor_mao2 = valor_mao(mao2) # Valor da mão 2 em número
    valor_maoD = valor_mao(maoD) # Valor da mão D em número
    valor_crupie = valor_mao(crupie) # Valor da mão do crupiê em número
    valores_mao1 = [carta[0] for carta in mao1] # Valores das cartas da mão 1 em lista
    valores_mao2 = [carta[0] for carta in mao2] # Valores das cartas da mão 2 em lista
    valores_maoD = [carta[0] for carta in maoD] # Valores das cartas da mão D em lista
    valores_crupie = [carta[0] for carta in crupie] # Valores das cartas do crupiê em lista

    soft_hand1 = False
    if 'A' in valores_mao1 and valor_mao1 <= 21 and valor_mao1 - 11 <= 10:
        soft_hand1 = True

    soft_hand2 = False
    if 'A' in valores_mao2 and valor_mao2 <= 21 and valor_mao2 - 11 <= 10:
        soft_hand2 = True

    soft_handD = False
    if 'A' in valores_maoD and valor_maoD <= 21 and valor_maoD - 11 <= 10:
        soft_handD = True

    # Estratégia contra cada um dos casos possíveis em relação à primeira carta do crupiê
    split_2 = [['A', 'A'], [2, 2], [3, 3], [6, 6], [7, 7], [8, 8], [9, 9]] # Estratégia para caso o crupiê tenha um 2
    split_3 = [['A', 'A'], [2, 2], [3, 3], [6, 6], [7, 7], [8, 8], [9, 9]] # Estratégia para caso o crupiê tenha um 3
    split_4 = [['A', 'A'], [2, 2], [3, 3], [6, 6], [7, 7], [8, 8], [9, 9]] # Estratégia para caso o crupiê tenha um 4
    split_5 = [['A', 'A'], [2, 2], [3, 3], [4, 4], [6, 6], [7, 7], [8, 8], [9, 9]] # Estratégia para caso o crupiê tenha um 5
```

```
        split_6 = [['A', 'A'], [2, 2], [3, 3], [4, 4], [6, 6], [7, 7], [8, 8], [9, 9]] # Estratégia para caso o crupiê tenha um 6
        split_7 = [['A', 'A'], [2, 2], [3, 3], [6, 6], [7, 7], [8, 8]] # Estratégia para caso o crupiê tenha um 7
        split_8 = [['A', 'A'], [8, 8], [9, 9]] # Estratégia para caso o crupiê tenha um 8
        split_9 = [['A', 'A'], [8, 8], [9, 9]] # Estratégia para caso o crupiê tenha um 9
        split_10 = [['A', 'A'], [8, 8]] # Estratégia para caso o crupiê tenha um 10
        split_A = [['A', 'A'], [8, 8]] # Estratégia para caso o crupiê tenha um Ás

        double_down_2 = [10, 11] # Estratégia para caso o crupiê tenha um 2
        double_down_3 = [9, 10, 11] # Estratégia para caso o crupiê tenha um 3
        double_down_4 = [9, 10, 11] # Estratégia para caso o crupiê tenha um 4
        double_down_5 = [9, 10, 11] # Estratégia para caso o crupiê tenha um 5
        double_down_6 = [9, 10, 11] # Estratégia para caso o crupiê tenha um 6
        double_down_7 = [10, 11] # Estratégia para caso o crupiê tenha um 7
        double_down_8 = [10, 11] # Estratégia para caso o crupiê tenha um 8
        double_down_9 = [10, 11] # Estratégia para caso o crupiê tenha um 9
        double_down_10 = [11] # Estratégia para caso o crupiê tenha um 10
        double_down_A = [] # Estratégia para caso o crupiê tenha um Ás

        soft_hand_double_down_2 = [] # Estratégia para caso o crupiê tenha um 2
        soft_hand_double_down_3 = [17, 18] # Estratégia para caso o crupiê tenha um 3
        soft_hand_double_down_4 = [15, 16, 17, 18] # Estratégia para caso o crupiê tenha um 4
        soft_hand_double_down_5 = [13, 14, 15, 16, 17, 18] # Estratégia para caso o crupiê tenha um 5
        soft_hand_double_down_6 = [13, 14, 15, 16, 17, 18] # Estratégia para caso o crupiê tenha um 6
        soft_hand_double_down_7 = [] # Estratégia para caso o crupiê tenha um 7
        soft_hand_double_down_8 = [] # Estratégia para caso o crupiê tenha um 8
        soft_hand_double_down_9 = [] # Estratégia para caso o crupiê tenha um 9
        soft_hand_double_down_10 = [] # Estratégia para caso o crupiê tenha um 10
        soft_hand_double_down_A = [] # Estratégia para caso o crupiê tenha um Ás
```

```
            hit_2 = [4, 5, 6, 7, 8, 9, 10, 11, 12] # Estratégia para caso o crupiê tenha um 2
            hit_3 = [4, 5, 6, 7, 8, 9, 10, 11, 12] # Estratégia para caso o crupiê tenha um 3
            hit_4 = [4, 5, 6, 7, 8, 9, 10, 11] # Estratégia para caso o crupiê tenha um 4
            hit_5 = [4, 5, 6, 7, 8, 9, 10, 11] # Estratégia para caso o crupiê tenha um 5
            hit_6 = [4, 5, 6, 7, 8, 9, 10, 11] # Estratégia para caso o crupiê tenha um 6
            hit_7 = [4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16] # Estratégia para caso o crupiê
tenha um 7
            hit_8 = [4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16] # Estratégia para caso o crupiê
tenha um 8
            hit_9 = [4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16] # Estratégia para caso o crupiê
tenha um 9
            hit_10 = [4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16] # Estratégia para caso o crupiê
tenha um 10
            hit_A = [4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16] # Estratégia para caso o crupiê
tenha um Ás

            soft_hand_hit_2 = [13, 14, 15, 16, 17] # Estratégia para caso o crupiê tenha um 2
            soft_hand_hit_3 = [13, 14, 15, 16] # Estratégia para caso o crupiê tenha um 3
            soft_hand_hit_4 = [13, 14] # Estratégia para caso o crupiê tenha um 4
            soft_hand_hit_5 = [] # Estratégia para caso o crupiê tenha um 5
            soft_hand_hit_6 = [] # Estratégia para caso o crupiê tenha um 6
            soft_hand_hit_7 = [13, 14, 15, 16, 17] # Estratégia para caso o crupiê tenha um 7
            soft_hand_hit_8 = [13, 14, 15, 16, 17] # Estratégia para caso o crupiê tenha um 8
            soft_hand_hit_9 = [13, 14, 15, 16, 17, 18] # Estratégia para caso o crupiê tenha um
9
            soft_hand_hit_10 = [13, 14, 15, 16, 17, 18] # Estratégia para caso o crupiê tenha um
10
            soft_hand_hit_A = [13, 14, 15, 16, 17, 18] # Estratégia para caso o crupiê tenha um
Ás

            # Estratégia para caso o crupiê tenha um 2
            if valores_crupie[0] == 2:
               if (len(mao1) == 2 and maoD == [] and mao2 == []) and valor_mao(mao1) in
split_2:
```

```
                mao1, mao2 = split(mao1, mao2, baralho)
                valor_mao1 = valor_mao(mao1)
                valor_mao2 = valor_mao(mao2)

                if mao1[0] not in [('A', 'Espadas'), ('A', 'Paus'), ('A', 'Copas'), ('A', 'Ouros')]:
                    while valor_mao(mao1) in hit_2:
                        mao1 = hit1(mao1, baralho)
                        if valor_mao1 not in hit_2:
                            return mao1, mao2, maoD

                    while valor_mao(mao2) in hit_2:
                        mao2 = hit2(mao2, baralho)
                        if valor_mao2 not in hit_2:
                            return mao1, mao2, maoD
                else:
                    return mao1, mao2, maoD

            elif (len(mao1) == 2 and maoD == [] and mao2 == []) and valor_mao(mao1) in
double_down_2:
                mao1, maoD = double_down(mao1, maoD, baralho)
                valores_mao1 = valor_mao(mao1)
                valores_maoD = valor_mao(maoD)
                return mao1, mao2, maoD

            elif (len(mao1) == 2 and maoD == [] and mao2 == []) and soft_hand1 and
soft_handD and (valor_mao(mao1) in soft_hand_double_down_2):
                mao1, maoD = double_down(mao1, maoD, baralho)
                valores_mao1 = valor_mao(mao1)
                valores_maoD = valor_mao(maoD)
                return mao1, mao2, maoD

            while (valores_mao1 not in split_2) and valor_mao(mao1) in hit_2:
                mao1 = hit1(mao1, baralho)
                valor_mao1 = valor_mao(mao1)
```

```
                if valor_mao1 not in hit_2:
                    return mao1, mao2, maoD
                else:
                    return mao1, mao2, maoD

            # Estratégia para caso o crupiê tenha um 3
            elif valores_crupie[0] == 3:
                if (len(mao1) == 2 and maoD == [] and mao2 == []) and valores_mao1 in
split_3:
                    mao1, mao2 = split(mao1, mao2, baralho)
                    valor_mao1 = valor_mao(mao1)
                    valor_mao2 = valor_mao(mao2)

                    if mao1[0] not in [('A', 'Espadas'), ('A', 'Paus'), ('A', 'Copas'), ('A', 'Ouros')]:
                        while valor_mao(mao1) in hit_3:
                            mao1 = hit1(mao1, baralho)
                            if valor_mao1 not in hit_3:
                                return mao1, mao2, maoD

                        while valor_mao(mao2) in hit_3:
                            mao2 = hit2(mao2, baralho)
                            if valor_mao2 not in hit_3:
                                return mao1, mao2, maoD
                    else:
                        return mao1, mao2, maoD

                elif (len(mao1) == 2 and maoD == [] and mao2 == []) and valor_mao1 in
double_down_3:
                    mao1, maoD = double_down(mao1, maoD, baralho)
                    valores_mao1 = valor_mao(mao1)
                    valores_maoD = valor_mao(maoD)
                    return mao1, mao2, maoD

                elif (len(mao1) == 2 and maoD == [] and mao2 == []) and soft_hand1 and
```

```
soft_handD and (valor_mao1 in soft_hand_double_down_3):
                    mao1, maoD = double_down(mao1, maoD, baralho)
                    valores_mao1 = valor_mao(mao1)
                    valores_maoD = valor_mao(maoD)
                    return mao1, mao2, maoD

                while (valores_mao1 not in split_3) and valor_mao1 in hit_3:
                    mao1 = hit1(mao1, baralho)
                    valor_mao1 = valor_mao(mao1)
                    if valor_mao1 not in hit_3:
                        return mao1, mao2, maoD
                else:
                    return mao1, mao2, maoD

            # Estratégia para caso o crupiê tenha um 4
            elif valores_crupie[0] == 4:
                if (len(mao1) == 2 and maoD == [] and mao2 == []) and valores_mao1 in
split_4:
                    mao1, mao2 = split(mao1, mao2, baralho)
                    valor_mao1 = valor_mao(mao1)
                    valor_mao2 = valor_mao(mao2)

                    if mao1[0] not in [('A', 'Espadas'), ('A', 'Paus'), ('A', 'Copas'), ('A', 'Ouros')]:
                        while valor_mao(mao1) in hit_4:
                            mao1 = hit1(mao1, baralho)
                            if valor_mao1 not in hit_4:
                                return mao1, mao2, maoD

                        while valor_mao(mao2) in hit_4:
                            mao2 = hit2(mao2, baralho)
                            if valor_mao2 not in hit_4:
                                return mao1, mao2, maoD
                    else:
                        return mao1, mao2, maoD
```

```
                elif (len(mao1) == 2 and maoD == [] and mao2 == []) and valor_mao1 in
double_down_4:
                    mao1, maoD = double_down(mao1, maoD, baralho)
                    valores_mao1 = valor_mao(mao1)
                    valores_maoD = valor_mao(maoD)
                    return mao1, mao2, maoD

                elif (len(mao1) == 2 and maoD == [] and mao2 == []) and soft_hand1 and
(valor_mao1 in soft_hand_double_down_4):
                    mao1, maoD = double_down(mao1, maoD, baralho)
                    valores_mao1 = valor_mao(mao1)
                    valores_maoD = valor_mao(maoD)
                    return mao1, mao2, maoD

                while (valores_mao1 not in split_4) and valor_mao1 in hit_4:
                    mao1 = hit1(mao1, baralho)
                    valor_mao1 = valor_mao(mao1)
                    if valor_mao1 not in hit_4:
                        return mao1, mao2, maoD
                else:
                    return mao1, mao2, maoD

            # Estratégia para caso o crupiê tenha um 5
            elif valores_crupie[0] == 5:
                if (len(mao1) == 2 and maoD == [] and mao2 == []) and valores_mao1 in
split_5:
                    mao1, mao2 = split(mao1, mao2, baralho)
                    valor_mao1 = valor_mao(mao1)
                    valor_mao2 = valor_mao(mao2)

                    if mao1[0] not in [('A', 'Espadas'), ('A', 'Paus'), ('A', 'Copas'), ('A', 'Ouros')]:
                        while valor_mao(mao1) in hit_5:
                            mao1 = hit1(mao1, baralho)
```

```
                    if valor_mao1 not in hit_5:
                        return mao1, mao2, maoD

                    while valor_mao(mao2) in hit_5:
                        mao2 = hit2(mao2, baralho)
                        if valor_mao2 not in hit_5:
                            return mao1, mao2, maoD
                    else:
                        return mao1, mao2, maoD

                elif (len(mao1) == 2 and maoD == [] and mao2 == []) and valor_mao1 in double_down_5:
                    mao1, maoD = double_down(mao1, maoD, baralho)
                    valores_mao1 = valor_mao(mao1)
                    valores_maoD = valor_mao(maoD)
                    return mao1, mao2, maoD

                elif (len(mao1) == 2 and maoD == [] and mao2 == []) and soft_hand1 and soft_handD and (valor_mao1 in soft_hand_double_down_5):
                    mao1, maoD = double_down(mao1, maoD, baralho)
                    valores_mao1 = valor_mao(mao1)
                    valores_maoD = valor_mao(maoD)
                    return mao1, mao2, maoD

                while (valores_mao1 not in split_5) and valor_mao1 in hit_5:
                    mao1 = hit1(mao1, baralho)
                    valor_mao1 = valor_mao(mao1)
                    if valor_mao1 not in hit_5:
                        return mao1, mao2, maoD
                else:
                    return mao1, mao2, maoD

            # Estratégia para caso o crupiê tenha um 3
            elif valores_crupie[0] == 6:
```

```
                if (len(mao1) == 2 and maoD == [] and mao2 == []) and valores_mao1 in
split_6:
                    mao1, mao2 = split(mao1, mao2, baralho)
                    valor_mao1 = valor_mao(mao1)
                    valor_mao2 = valor_mao(mao2)

                    if mao1[0] not in [('A', 'Espadas'), ('A', 'Paus'), ('A', 'Copas'), ('A', 'Ouros')]:
                        while valor_mao(mao1) in hit_6:
                            mao1 = hit1(mao1, baralho)
                            if valor_mao1 not in hit_6:
                                return mao1, mao2, maoD

                        while valor_mao(mao2) in hit_6:
                            mao2 = hit2(mao2, baralho)
                            if valor_mao2 not in hit_6:
                                return mao1, mao2, maoD
                    else:
                        return mao1, mao2, maoD

                elif (len(mao1) == 2 and maoD == [] and mao2 == []) and valor_mao1 in
double_down_6:
                    mao1, maoD = double_down(mao1, maoD, baralho)
                    valores_mao1 = valor_mao(mao1)
                    valores_maoD = valor_mao(maoD)
                    return mao1, mao2, maoD

                elif (len(mao1) == 2 and maoD == [] and mao2 == []) and soft_hand1 and
soft_handD and (valor_mao1 in soft_hand_double_down_6):
                    mao1, maoD = double_down(mao1, maoD, baralho)
                    valores_mao1 = valor_mao(mao1)
                    valores_maoD = valor_mao(maoD)
                    return mao1, mao2, maoD

                while (valores_mao1 not in split_6) and valor_mao1 in hit_6:
```

```
                    mao1 = hit1(mao1, baralho)
                    valor_mao1 = valor_mao(mao1)
                    if valor_mao1 not in hit_6:
                        return mao1, mao2, maoD
                else:
                    return mao1, mao2, maoD

            # Estratégia para caso o crupiê tenha um 7
            elif valores_crupie[0] == 7:
                if (len(mao1) == 2 and maoD == [] and mao2 == []) and valores_mao1 in
split_7:
                    mao1, mao2 = split(mao1, mao2, baralho)
                    valor_mao1 = valor_mao(mao1)
                    valor_mao2 = valor_mao(mao2)

                    if mao1[0] not in [('A', 'Espadas'), ('A', 'Paus'), ('A', 'Copas'), ('A', 'Ouros')]:
                        while valor_mao(mao1) in hit_7:
                            mao1 = hit1(mao1, baralho)
                            if valor_mao1 not in hit_7:
                                return mao1, mao2, maoD

                        while valor_mao(mao2) in hit_7:
                            mao2 = hit2(mao2, baralho)
                            if valor_mao2 not in hit_7:
                                return mao1, mao2, maoD
                    else:
                        return mao1, mao2, maoD

                elif (len(mao1) == 2 and maoD == [] and mao2 == []) and valor_mao1 in
double_down_7:
                    mao1, maoD = double_down(mao1, maoD, baralho)
                    valores_mao1 = valor_mao(mao1)
                    valores_maoD = valor_mao(maoD)
                    return mao1, mao2, maoD
```

```
                elif (len(mao1) == 2 and maoD == [] and mao2 == []) and soft_hand1 and
soft_handD and (valor_mao1 in soft_hand_double_down_7):
                    mao1, maoD = double_down(mao1, maoD, baralho)
                    valores_mao1 = valor_mao(mao1)
                    valores_maoD = valor_mao(maoD)
                    return mao1, mao2, maoD

                while (valores_mao1 not in split_7) and valor_mao1 in hit_7:
                    mao1 = hit1(mao1, baralho)
                    valor_mao1 = valor_mao(mao1)
                    if valor_mao1 not in hit_7:
                        return mao1, mao2, maoD
                else:
                    return mao1, mao2, maoD

            # Estratégia para caso o crupiê tenha um 8
            elif valores_crupie[0] == 8:
                if (len(mao1) == 2 and maoD == [] and mao2 == []) and valores_mao1 in
split_8:
                    mao1, mao2 = split(mao1, mao2, baralho)
                    valor_mao1 = valor_mao(mao1)
                    valor_mao2 = valor_mao(mao2)

                    if mao1[0] not in [('A', 'Espadas'), ('A', 'Paus'), ('A', 'Copas'), ('A', 'Ouros')]:
                        while valor_mao(mao1) in hit_8:
                            mao1 = hit1(mao1, baralho)
                            if valor_mao(mao1) not in hit_8:
                                return mao1, mao2, maoD

                        while valor_mao(mao2) in hit_8:
                            mao2 = hit2(mao2, baralho)
                            if valor_mao(mao2) not in hit_8:
                                return mao1, mao2, maoD
```

```
                else:
                    return mao1, mao2, maoD

            elif (len(mao1) == 2 and maoD == [] and mao2 == []) and valor_mao1 in
double_down_8:
                mao1, maoD = double_down(mao1, maoD, baralho)
                valores_mao1 = valor_mao(mao1)
                valores_maoD = valor_mao(maoD)
                return mao1, mao2, maoD

            elif (len(mao1) == 2 and maoD == [] and mao2 == []) and soft_hand1 and
soft_handD and (valor_mao1 in soft_hand_double_down_8):
                mao1, maoD = double_down(mao1, maoD, baralho)
                valores_mao1 = valor_mao(mao1)
                valores_maoD = valor_mao(maoD)
                return mao1, mao2, maoD

            while (valores_mao1 not in split_8) and valor_mao1 in hit_8:
                mao1 = hit1(mao1, baralho)
                valor_mao1 = valor_mao(mao1)
                if valor_mao1 not in hit_8:
                    return mao1, mao2, maoD
            else:
                return mao1, mao2, maoD

        # Estratégia para caso o crupiê tenha um 9
        elif valores_crupie[0] == 9:
            if (len(mao1) == 2 and maoD == [] and mao2 == []) and valores_mao1 in
split_9:
                mao1, mao2 = split(mao1, mao2, baralho)
                valor_mao1 = valor_mao(mao1)
                valor_mao2 = valor_mao(mao2)

                if mao1[0] not in [('A', 'Espadas'), ('A', 'Paus'), ('A', 'Copas'), ('A', 'Ouros')]:
```

```
                while valor_mao(mao1) in hit_9:
                    mao1 = hit1(mao1, baralho)
                    if valor_mao1 not in hit_9:
                        return mao1, mao2, maoD

                while valor_mao(mao2) in hit_9:
                    mao2 = hit2(mao2, baralho)
                    if valor_mao2 not in hit_9:
                        return mao1, mao2, maoD
            else:
                return mao1, mao2, maoD

        elif (len(mao1) == 2 and maoD == [] and mao2 == []) and valor_mao1 in
double_down_9:
            mao1, maoD = double_down(mao1, maoD, baralho)
            valores_mao1 = valor_mao(mao1)
            valores_maoD = valor_mao(maoD)
            return mao1, mao2, maoD

        elif (len(mao1) == 2 and maoD == [] and mao2 == []) and soft_hand1 and
soft_handD and (valor_mao1 in soft_hand_double_down_9):
            mao1, maoD = double_down(mao1, maoD, baralho)
            valores_mao1 = valor_mao(mao1)
            valores_maoD = valor_mao(maoD)
            return mao1, mao2, maoD

        while (valores_mao1 not in split_9) and valor_mao1 in hit_9:
            mao1 = hit1(mao1, baralho)
            valor_mao1 = valor_mao(mao1)
            if valor_mao1 not in hit_9:
                return mao1, mao2, maoD
        else:
            return mao1, mao2, maoD
```

```
        # Estratégia para caso o crupiê tenha um 10, J, Q ou K
        elif valores_crupie[0] in [10, 'J', 'Q', 'K']:
            if (len(mao1) == 2 and maoD == [] and mao2 == []) and valores_mao1 in
split_10:
                mao1, mao2 = split(mao1, mao2, baralho)
                valor_mao1 = valor_mao(mao1)
                valor_mao2 = valor_mao(mao2)

                if mao1[0] not in [('A', 'Espadas'), ('A', 'Paus'), ('A', 'Copas'), ('A', 'Ouros')]:
                    while valor_mao(mao1) in hit_10:
                        mao1 = hit1(mao1, baralho)
                        if valor_mao1 not in hit_10:
                            return mao1, mao2, maoD

                    while valor_mao(mao2) in hit_10:
                        mao2 = hit2(mao2, baralho)
                        if valor_mao2 not in hit_10:
                            return mao1, mao2, maoD
                else:
                    return mao1, mao2, maoD

            elif (len(mao1) == 2 and maoD == [] and mao2 == []) and valor_mao1 in
double_down_10:
                mao1, maoD = double_down(mao1, maoD, baralho)
                valores_mao1 = valor_mao(mao1)
                valores_maoD = valor_mao(maoD)
                return mao1, mao2, maoD

            elif (len(mao1) == 2 and maoD == [] and mao2 == []) and soft_hand1 and
soft_handD and (valor_mao1 in soft_hand_double_down_10):
                mao1, maoD = double_down(mao1, maoD, baralho)
                valores_mao1 = valor_mao(mao1)
                valores_maoD = valor_mao(maoD)
                return mao1, mao2, maoD
```

```
                while (valores_mao1 not in split_10) and valor_mao1 in hit_10:
                    mao1 = hit1(mao1, baralho)
                    valor_mao1 = valor_mao(mao1)
                    if valor_mao1 not in hit_10:
                        return mao1, mao2, maoD
                else:
                    return mao1, mao2, maoD

            # Estratégia para caso o crupiê tenha um Ás
            elif valores_crupie[0] == 'A':
                if (len(mao1) == 2 and maoD == [] and mao2 == []) and valores_mao1 in
split_A:
                    mao1, mao2 = split(mao1, mao2, baralho)
                    valor_mao1 = valor_mao(mao1)
                    valor_mao2 = valor_mao(mao2)

                    if mao1[0] not in [('A', 'Espadas'), ('A', 'Paus'), ('A', 'Copas'), ('A', 'Ouros')]:
                        while valor_mao(mao1) in hit_A:
                            mao1 = hit1(mao1, baralho)
                            if valor_mao1 not in hit_A:
                                return mao1, mao2, maoD

                        while valor_mao(mao2) in hit_A:
                            mao2 = hit2(mao2, baralho)
                            if valor_mao2 not in hit_A:
                                return mao1, mao2, maoD
                    else:
                        return mao1, mao2, maoD

                elif (len(mao1) == 2 and maoD == [] and mao2 == []) and valor_mao(mao1)
in double_down_A:
                    mao1, maoD = double_down(mao1, maoD, baralho)
                    valores_mao1 = valor_mao(mao1)
```

```
                valores_maoD = valor_mao(maoD)
                return mao1, mao2, maoD

            elif (len(mao1) == 2 and maoD == [] and mao2 == []) and soft_hand1 and
soft_handD and (valor_mao1 in soft_hand_double_down_A):
                mao1, maoD = double_down(mao1, maoD, baralho)
                valores_mao1 = valor_mao(mao1)
                valores_maoD = valor_mao(maoD)
                return mao1, mao2, maoD

            while (valores_mao1 not in split_A) and valor_mao1 in hit_A:
                mao1 = hit1(mao1, baralho)
                valor_mao1 = valor_mao(mao1)
                if valor_mao1 not in hit_A:
                    return mao1, mao2, maoD
            else:
                return mao1, mao2, maoD
        return mao1, mao2, maoD

    # Função para jogar Blackjack
    def jogar_Blackjack(mao1, mao2, maoD, crupie, baralho, resultados):
        mao1 = [baralho.pop(0), baralho.pop(1)]
        crupie = [baralho.pop(0), baralho.pop(0)]
        mao2 = []
        maoD = []
        #print(f'Mão do jogador: {mao1} com valor {valor_mao(mao1)}')
        #print(f'Mão do crupiê: {crupie[0]}')
        valor_mao1 = valor_mao(mao1) # Valor da mão 1 em número
        valor_mao2 = valor_mao(mao2) # Valor da mão 2 em número
        valor_maoD = valor_mao(maoD) # Valor da mão D em número
        valor_crupie = valor_mao(crupie) # Valor da mão do crupiê em número
        valores_mao1 = [carta[0] for carta in mao1] # Valores das cartas da mão 1 em lista
        valores_mao2 = [carta[0] for carta in mao2] # Valores das cartas da mão 2 em lista
        valores_maoD = [carta[0] for carta in maoD] # Valores das cartas da mão D em lista
```

```
            valores_crupie = [carta[0] for carta in crupie] # Valores das cartas do crupiê em lista

            # Declarar a primeira carta do crupiê
            primeira_carta_crupie = crupie[0]

            # Verificar apostas laterais
            par_perfeito = verificar_pares_perfeitos(mao1, resultados)
            aposta_21_mais_3 = verificar_21_mais_3(mao1, primeira_carta_crupie, resultados)

            # Verificar se há Blackjack
            crupie_Blackjack = verificar_Blackjack(crupie)
            jogador_Blackjack = verificar_Blackjack(mao1)

            if crupie_Blackjack and jogador_Blackjack:
                resultados['empate'] += 1
                return 'empate'

            elif (len(crupie) == 2 and valor_mao(crupie) == 21) and not (len(mao1) != 2 and valor_mao(mao1) == 21):
                resultados['crupie_Blackjack'] += 1
                return 'crupie_Blackjack'

            elif (len(mao1) == 2 and valor_mao(mao1) == 21) and not (len(crupie) != 2 and valor_mao(crupie) == 21):
                resultados['jogador_Blackjack'] += 1
                return 'jogador_Blackjack'

            # Estratégia do jogador
            mao1, mao2, maoD = estrategia_jogador(mao1, mao2, maoD, crupie, baralho)

            # Estratégia do crupiê
            if (valor_mao(mao1) <= 21) or (valor_mao(mao2) <= 21 and mao2 != []):
                while valor_mao(crupie) < 17:
```

```
            crupie.append(baralho.pop(0))
            valor_crupie = valor_mao(crupie)

        #print(f'Mão 1 do jogador: {mao1} com valor {valor_mao(mao1)}')
        #print(f'Mão 2 do jogador: {mao2} com valor {valor_mao(mao2)}')
        #print(f'Mão do crupiê: {crupie} com valor {valor_mao(crupie)}')

        ## Primeiro os casos de split
        if (mao1 != [] and mao2 != [] and maoD == []):
            # Caso onde a mão 1 perde por (CRUPIÊ_BLACKJACK_TARDIO e ESTOURANDO) e mão 2 perde por (CRUPIÊ_BLACKJACK_TARDIO e ESTOURANDO)
            if crupie_Blackjack and (valor_mao(mao1) > 21) and (valor_mao(mao2) > 21):
                resultados['jogador_bust'] += 2
                #resultados['crupie_Blackjack'] += 2 #(pode ser contado como crupie_Blackjack, mas como estamos avaliando somente o que acontece com o valor da banca, tanto faz a maneira que o jogador seja derrotado, apenas importa que seja quando for.)
                return 'crupie'

            # Caso onde a mão 1 perde por (CRUPIÊ_BLACKJACK_TARDIO e ESTOURANDO) e mão 2 perde por (CRUPIÊ_BLACKJACK_TARDIO e NÃO ESTOURANDO)
            elif crupie_Blackjack and (valor_mao(mao1) > 21) and (valor_mao(mao2) <= 21):
                resultados['crupie_Blackjack'] += 1
                resultados['jogador_bust'] += 1
                return 'crupie'

            # Caso onde a mão 1 perde por (CRUPIÊ_BLACKJACK_TARDIO e NÃO ESTOURANDO) e mão 2 perde por (CRUPIÊ_BLACKJACK_TARDIO e ESTOURANDO)
            elif crupie_Blackjack and (valor_mao(mao1) <= 21) and (valor_mao(mao2) > 21):
                resultados['crupie_Blackjack'] += 1
                resultados['jogador_bust'] += 1
                return 'crupie'
```

```
            # Caso onde a mão 1 perde por (CRUPIÊ_BLACKJACK_TARDIO e NÃO ESTOURANDO) e mão 2 perde por (CRUPIÊ_BLACKJACK_TARDIO e NÃO ESTOURANDO)
            elif crupie_Blackjack and (valor_mao(mao1) <= 21) and (valor_mao(mao2) <= 21):
                resultados['crupie_Blackjack'] += 2
                return 'crupie'

            # Caso onde a mão 1 ganha e a mão 2 ganha (POR COMPARAÇÃO DE MÃOS)
            elif (valor_mao(mao1) <= 21 and valor_mao(mao2) <= 21) and (valor_mao(mao1) > valor_mao(crupie)) and (valor_mao(mao2) > valor_mao(crupie)):
                resultados['jogador'] += 2
                return 'jogador'

            #Caso onde a mão 1 ganha (CRUPIE ESTOURANDO) e a mão 2 ganha (CRUPIÊ ESTOURANDO)
            elif ((valor_mao(mao1) <= 21) and (valor_mao(mao2) <= 21) and (valor_mao(crupie) > 21)):
                resultados['crupie_bust'] += 2
                return 'jogador'

            # Caso onde a mão 1 ganha e a mão 2 perde (POR COMPARAÇÃO DE MÃOS)
            elif (((valor_mao(mao1) <= 21) and (valor_mao(mao1) > valor_mao(crupie))) and ((valor_mao(mao2) <= 21) and (valor_mao(mao2) < valor_mao(crupie)))):
                resultados['jogador'] += 1
                resultados['crupie'] += 1
                return 'jogador'

            # Caso onde a mão 1 ganha (POR COMPARAÇÃO DE MÃOS) e a mão 2 perde (ESTOURANDO)
            elif ((valor_mao(mao1) <= 21) and (valor_mao(mao1) > valor_mao(crupie))) and (valor_mao(mao2) > 21):
                resultados['jogador'] += 1
                resultados['jogador_bust'] += 1
```

```
            return 'jogador'

        # Caso onde a mão 1 ganha (CRUPIÊ ESTOURANDO) e a mão 2 perde (ESTOURANDO)
        elif ((valor_mao(mao1) <= 21) and (valor_mao(crupie) > 21)) and (valor_mao(mao2) > 21):
            resultados['crupie_bust'] += 1
            resultados['jogador_bust'] += 1
            return 'jogador'

        # Caso onde a mão 1 perde e a mão 2 ganha (POR COMPARAÇÃO DE MÃOS)
        elif (((valor_mao(mao2) <= 21) and (valor_mao(mao2) > valor_mao(crupie))) and ((valor_mao(mao1) <= 21) and (valor_mao(mao1) < valor_mao(crupie)))):
            resultados['jogador'] += 1
            resultados['crupie'] += 1
            return 'jogador'

        # Caso onde a mão 1 perde (ESTOURANDO) e a mão 2 ganha (POR COMPARAÇÃO DE MÃOS)
        elif ((valor_mao(mao2) <= 21) and (valor_mao(mao2) > valor_mao(crupie))) and (valor_mao(mao1) > 21):
            resultados['jogador'] += 1
            resultados['jogador_bust'] += 1
            return 'jogador'

        # Caso onde a mão 1 perde (ESTOURANDO) e a mão 2 ganha (CRUPIÊ ESTOURANDO)
        elif ((valor_mao(mao2) <= 21) and (valor_mao(crupie) > 21)) and (valor_mao(mao1) > 21):
            resultados['crupie_bust'] += 1
            resultados['jogador_bust'] += 1
            return 'jogador'

        # Caso onde a mão 1 perde e a mão 2 perde (POR COMPARAÇÃO DE MÃOS)
```

```
            elif ((valor_mao(mao1) <= 21) and (valor_mao(mao1) < valor_mao(crupie)) and
((valor_mao(mao2) <= 21) and valor_mao(mao2) < valor_mao(crupie))):
                resultados['crupie'] += 2
                return 'crupie'

            # Caso onde a mão 1 perde (ESTOURANDO) e a mão 2 perde (POR
COMPARAÇÃO DE MÃOS)
            elif ((valor_mao(mao2) <= 21) and (valor_mao(mao2) < valor_mao(crupie)) and
(valor_mao(mao1) > 21)):
                resultados['crupie'] += 1
                resultados['jogador_bust'] += 1
                return 'crupie'

            # Caso onde a mão 1 perde (POR COMPARAÇÃO DE MÃOS) e a mão 2 perde
(ESTOURANDO)
            elif ((valor_mao(mao1) <= 21) and (valor_mao(mao1) < valor_mao(crupie)) and
(valor_mao(mao2) > 21)):
                resultados['crupie'] += 1
                resultados['jogador_bust'] += 1
                return 'crupie'

            # Caso onde a mão 1 perde (ESTOURANDO) e a mão 2 perde (ESTOURANDO)
            elif ((valor_mao(mao1) > 21) and (valor_mao(mao2) > 21)):
                resultados['jogador_bust'] += 2
                return 'crupie'

            # Caso onde a mão 1 empata e a mão 2 empata
            elif ((valor_mao(mao1) <= 21) and valor_mao(mao1) == valor_mao(crupie)) and
((valor_mao(mao2) <= 21) and valor_mao(mao2) == valor_mao(crupie)):
                resultados['empate'] += 2
                return 'empate'

            # Caso onde a mão 1 ganha (POR COMPARAÇÃO) e a mão 2 empata
            elif ((valor_mao(mao1) <= 21) and (valor_mao(mao1) > valor_mao(crupie)) and
```

```
((valor_mao(mao2) <= 21) and valor_mao(mao2) == valor_mao(crupie))):
                    resultados['jogador'] += 1
                    resultados['empate'] += 1
                    return 'jogador'

                # Caso onde a mão 1 perde (POR COMPARAÇÃO) e a mão 2 empata
                elif (((valor_mao(mao1) <= 21) and (valor_mao(mao1) < valor_mao(crupie)))
and ((valor_mao(mao2) <= 21) and (valor_mao(mao2) == valor_mao(crupie)))):
                    resultados['crupie'] += 1
                    resultados['empate'] += 1
                    return 'crupie'

                # Caso a mão 1 empata e a mão 2 ganhe (POR COMPARAÇÃO DE MÃOS)
                elif ((valor_mao(mao1) <= 21) and (valor_mao(mao1) == valor_mao(crupie)))
and ((valor_mao(mao2) <= 21) and (valor_mao(mao2) > valor_mao(crupie))):
                    resultados['empate'] += 1
                    resultados['jogador'] += 1
                    return 'jogador'

                # Caso onde a mão 1 empata e a mão 2 perde (POR COMPARAÇÃO DE MÃOS)
                elif (((valor_mao(mao2) <= 21) and (valor_mao(mao2) < valor_mao(crupie)))
and ((valor_mao(mao1) <= 21) and (valor_mao(mao1) == valor_mao(crupie)))):
                    resultados['crupie'] += 1
                    resultados['empate'] += 1
                    return 'crupie'

                # Caso onde a mão 1 empata e a mão 2 perde (ESTOURANDO)
                elif ((valor_mao(mao2) > 21) and ((valor_mao(mao1) <= 21) and
(valor_mao(mao1) == valor_mao(crupie)))):
                    resultados['jogador_bust'] += 1
                    resultados['empate'] += 1
                    return 'crupie'

            ## Casos de double down
```

```
elif (mao1 != [] and mao2 == [] and maoD != []):

    if len(crupie) == 2 and valor_mao(crupie) == 21 and valor_mao(mao1) <= 21:
        resultados['crupie_Blackjack'] += 2
        return 'crupie_Blackjack'

    # Caso em que o jogador estoura
    elif valor_mao(mao1) > 21:
        resultados['jogador_bust'] += 2
        return 'crupie'

    # Caso em que o crupiê estoura e o jogador não
    elif valor_mao(mao1) <= 21 and valor_mao(crupie) > 21:
        resultados['crupie_bust'] += 2
        return 'jogador'

    # Caso em que o jogador ganha por comparação
    elif (valor_mao(mao1) <= 21) and (valor_mao(mao1) > valor_mao(crupie)):
        resultados['jogador'] += 2
        return 'jogador'

    # Caso em que o crupiê ganha por comparação
    elif (valor_mao(mao1) <= 21) and (valor_mao(mao1) < valor_mao(crupie)):
        resultados['crupie'] += 2
        return 'crupie'

    # Caso em que há empate
    elif (valor_mao(mao1) <= 21) and (valor_mao(mao1) == valor_mao(crupie)):
        resultados['empate'] += 2
        return 'empate'

## Casos de hit simples
## Caso em que há empate
elif (mao2 == [] and maoD == []) and (valor_mao(mao1) <= 21) and
```

```
(valor_mao(mao1) == valor_mao(crupie)):
                resultados['empate'] += 1
                return 'empate'

            ## Caso onde o crupie tem um Blackjack tardio
            elif len(crupie) == 2 and valor_mao(crupie) == 21 and valor_mao(mao1) <= 21:
                resultados['crupie_Blackjack'] += 1
                return 'crupie_Blackjack'

            ## Caso em que o jogador estoura
            elif (mao2 == [] and maoD == []) and (valor_mao(mao1) > 21):
                resultados['jogador_bust'] += 1
                return 'crupie'

            ## Caso em que o crupiê estoura e o jogador não
            elif (mao2 == [] and maoD == []) and (valor_mao(mao1) <= 21) and
(valor_mao(crupie) > 21):
                resultados['crupie_bust'] += 1
                return 'jogador'

            ## Caso em que o jogador ganha por comparação
            elif (mao2 == [] and maoD == []) and (valor_mao(mao1) <= 21) and
(valor_mao(mao1) > valor_mao(crupie)):
                resultados['jogador'] += 1
                return 'jogador'

            ## Caso em que o crupiê ganha por comparação
            elif (mao2 == [] and maoD == []) and (valor_mao(mao1) <= 21) and
(valor_mao(mao1) < valor_mao(crupie)):
                resultados['crupie'] += 1
                return 'crupie'

            ## Caso blackjack tardio do crupiê
            elif (mao2 == [] and maoD == [] and len(crupie) == 2) and (valor_mao(crupie) ==
```

```
21) and (valor_mao(mao1) <= 21):
                resultados['crupie_Blackjack'] += 1
                return 'crupie_Blackjack'

            return resultados

        # Função para simular jogos
        def simular_jogos(numero_de_simulacoes):
            resultados = {
                'jogador': 0, 'crupie': 0, 'empate': 0,
                'jogador_Blackjack': 0, 'crupie_Blackjack': 0,
                'jogador_bust': 0, 'crupie_bust': 0,
                'pares_perfeitos': 0, 'Par Perfeito': 0, 'Par Colorido': 0, 'Par Misto': 0,
'sem_pares_perfeitos': 0,
                '21_mais_3': 0, 'Suited Trips': 0, 'Straight Flush': 0, 'Three of a Kind': 0, 'Straight':
0, 'Flush': 0, 'sem_21_mais_3': 0
                }

            baralho = criar_baralho()
            pos_corte = cortar_baralho(baralho)

            for i in range(numero_de_simulacoes):
                # Resetar as mãos do jogador e crupiê
                mao1 = [baralho.pop(0), baralho.pop(1)]
                crupie = [baralho.pop(0), baralho.pop(0)]
                mao2 = []
                maoD = []

                # Verificar se há cartas suficientes no baralho, senão reembaralhar
                if len(baralho) < pos_corte:
                    baralho = criar_baralho()
                    pos_corte = cortar_baralho(baralho)

                resultado = jogar_Blackjack(mao1, mao2, maoD, crupie, baralho, resultados)
```

```
    return resultados

numero_de_simulacoes = 10000000
resultados = simular_jogos(numero_de_simulacoes) # Exemplo de uso
print(resultados)
# Daqui em diante, será cálculo estatístico para verificar a porcentagem de retorno da estratégia com base em apostas unitárias

ganhos_jogador_Blackjack = 1.5*resultados['jogador_Blackjack'] + resultados['jogador'] + resultados['crupie_bust']
ganhos_jogador_aposta_lateral_21_mais_3 = 100*resultados['Suited Trips'] + 40*resultados['Straight Flush'] + 30*resultados['Three of a Kind'] + 10*resultados['Straight'] + 5*resultados['Flush']
ganhos_jogador_aposta_lateral_pares_perfeitos = 25*resultados['Par Perfeito'] + 12*resultados['Par Colorido'] + 6*resultados['Par Misto']
perdas_jogador_Blackjack = resultados['crupie_Blackjack'] + resultados['crupie'] + resultados['jogador_bust']
perdas_jogador_aposta_lateral_21_mais_3 = resultados['sem_21_mais_3']
perdas_jogador_aposta_lateral_pares_perfeitos = resultados['sem_pares_perfeitos']
total_em_apostas_Blackjack = (resultados['jogador'] + resultados['crupie'] + resultados['empate'] + resultados['jogador_Blackjack']) + resultados['crupie_Blackjack'] + resultados['jogador_bust'] + resultados['crupie_bust']
total_em_apostas_21_mais_3 = resultados['Suited Trips'] + resultados['Straight Flush'] + resultados['Three of a Kind'] + resultados['Straight'] + resultados['Flush'] + resultados['sem_21_mais_3']
total_em_apostas_pares_perfeitos = resultados['Par Perfeito'] + resultados['Par Colorido'] + resultados['Par Misto'] + resultados['sem_pares_perfeitos']

retorno_Blackjack = (ganhos_jogador_Blackjack - perdas_jogador_Blackjack)/total_em_apostas_Blackjack
retorno_aposta_lateral_21_mais_3 = (ganhos_jogador_aposta_lateral_21_mais_3 - perdas_jogador_aposta_lateral_21_mais_3)/total_em_apostas_21_mais_3
retorno_aposta_lateral_pares_perfeitos = (ganhos_jogador_aposta_lateral_pares_perfeitos -
```

```
perdas_jogador_aposta_lateral_pares_perfeitos)/total_em_apostas_pares_perfeitos
        print(f'O total de apostas em Blackjack foi {total_em_apostas_Blackjack}')
        print(f'O total de apostas em 21+3 foi {total_em_apostas_21_mais_3}')
        print(f'O total de apostas em pares perfeitos foi {total_em_apostas_pares_perfeitos}')
        print(f'O retorno esperado da estratégia de Blackjack é de {retorno_Blackjack:.5f} por
jogo')
        print(f'O retorno esperado da estratégia de pares perfeitos é de
{retorno_aposta_lateral_pares_perfeitos:.5f} por jogo')
        print(f'O retorno esperado da estratégia de 21+3 é de
{retorno_aposta_lateral_21_mais_3:.5f} por jogo')
```

APÊNDICE D – AVALIAÇÃO DO NÚMERO DE JOGOS NECESSÁRIOS PARA PREVALECER A LEI DOS GRANDES NÚMEROS PARA UM DADO JOGO.

```
        import numpy as np

        # Parâmetros de simulação
        QUANTIDADE_JOGADORES = 1          # Quantidade de jogadores
        QUANTIDADE_MAXIMA_PARTIDAS = 3500   # Quantidade de partidas
        SALDO_INICIAL_JOGADORES = 3500     # Saldo inicial de cada jogador
        VALOR_APOSTA_JOGADORES = 1         # Aposta dos jogadores fixa
        CHANCE_GANHAR_JOGO = 2.70          # Chance de vitória
        MULTIPLICADOR_VITORIA = 35         # ODD da aposta

        # Inicialização
        saldoJogador = np.zeros((QUANTIDADE_JOGADORES,
QUANTIDADE_MAXIMA_PARTIDAS), dtype=int)
        partidaAtual = 0
        lucroCassino = 0
        quantidadeVitorias = 0
        quantidadeDerrotas = 0
```

```
        # Função para atualizar o saldo dos jogadores e o lucro do cassino
        def atualizarSaldoJogadores():
            global partidaAtual, lucroCassino, quantidadeVitorias, quantidadeDerrotas

            if partidaAtual == QUANTIDADE_MAXIMA_PARTIDAS:
                return

            for i in range(QUANTIDADE_JOGADORES):
                saldoAtual = SALDO_INICIAL_JOGADORES if partidaAtual == 0 else saldoJogador[i][partidaAtual-1]

                apostaAtual = VALOR_APOSTA_JOGADORES if saldoAtual >= VALOR_APOSTA_JOGADORES else saldoAtual

                if apostaAtual > 0:
                    numeroSorteado = np.random.uniform(0, 100)

                    if numeroSorteado < CHANCE_GANHAR_JOGO:
                        # Jogador ganha
                        saldoJogador[i][partidaAtual] = saldoAtual + MULTIPLICADOR_VITORIA * apostaAtual
                        quantidadeVitorias += 1
                        lucroCassino -= (MULTIPLICADOR_VITORIA * apostaAtual)  # Prejuízo do cassino
                    else:
                        # Jogador perde
                        saldoJogador[i][partidaAtual] = saldoAtual - apostaAtual
                        quantidadeDerrotas += 1
                        lucroCassino += apostaAtual  # Cassino ganha o valor da aposta perdida
                else:
                    saldoJogador[i][partidaAtual] = 0

            partidaAtual += 1
```

```
# Função principal para rodar a simulação
def rodarSimulacao():
    global partidaAtual
    while partidaAtual < QUANTIDADE_MAXIMA_PARTIDAS:
        atualizarSaldoJogadores()

    print(f'Lucro do Cassino: {lucroCassino} reais")
    print(f'Quantidade de partidas: {partidaAtual}")
    print(f'Quantidade de vitórias: {quantidadeVitorias}")
    print(f'Quantidade de derrotas: {quantidadeDerrotas}")

if __name__ == "__main__":
    rodarSimulacao()
```

APÊNDICE E – ESTRATÉGIA I (A MAIS POPULAR DE *BLACKJACK*).

Mão do jogador	Mão do crupiê									
	2	3	4	5	6	7	8	9	10	A
5	x	x	x	x	x	x	x	x	x	x
6	x	x	x	x	x	x	x	x	x	x
7	x	x	x	x	x	x	x	x	x	x
8	x	x	x	x	x	x	x	x	x	x
9	x	D	D	D	D	x	x	x	x	x
10	D	D	D	D	D	D	D	D	x	x
11	D	D	D	D	D	D	D	D	D	x
12	x	x				x	x	x	x	x
13						x	x	x	x	x
14						x	x	x	x	x
15						x	x	x	x	x
16						x	x	x	x	x
17										
18										
19										
20										

21										
A-2	x	x	x	D	D	x	x	x	x	x
A-3	x	x	x	D	D	x	x	x	x	x
A-4	x	x	D	D	D	x	x	x	x	x
A-5	x	x	D	D	D	x	x	x	x	x
A-6	x	D	D	D	D	x	x	x	x	x
A-7		D	D	D	D			x	x	x
A-8										
A-9										
A-10										
A-A	S	S	S	S	S	S	S	S	S	S
2-2	S	S	S	S	S	S	x	x	x	x
3-3	S	S	S	S	S	S	x	x	x	x
4-4	x	x	x	S	S	x	x	x	x	x
5-5	D	D	D	D	D	D	D	D	x	x
6-6	S	S	S	S	S	S	x	x	x	x
7-7	S	S	S	S	S	S	x	x	x	x
8-8	S	S	S	S	S	S	S	S	S	S
9-9	S	S	S	S	S		S	S		
10-10										

X = Pedir.

D = Dobrar.

S = Dividir.

Ficar onde não há ação indicada.

APÊNDICE F – ESTRATÉGIA DE *BLACKJACK* II.

Mão do jogador	Mão do crupiê									
	2	3	4	5	6	7	8	9	10	A
5	x	x	x	x	x	x	x	x	x	x
6	x	x	x	x	x	x	x	x	x	x
7	x	x	x	x	x	x	x	x	x	x

8	x	x	x	x	x	x	x	x	x	x
9	x	D	D	D	D	x	x	x	x	x
10	D	D	D	D	D	D	D	D	x	x
11	D	D	D	D	D	D	D	D	D	x
12						x	x	x	x	x
13						x	x	x	x	x
14						x	x	x	x	x
15						x	x	x	x	x
16										
17										
18										
19										
20										
21										
A-2	x	x	x	D	D	x	x	x	x	x
A-3	x	x	x	D	D	x	x	x	x	x
A-4	x	x	D	D	D	x	x	x	x	x
A-5	x	x	D	D	D	x	x	x	x	x
A-6		D	D	D	D					
A-7		D	D	D	D					
A-8										
A-9										
A-10										
A-A	S	S	S	S	S	S	S	S	S	S
2-2	S	S	S	S	S	S	x	x	x	x
3-3	S	S	S	S	S	S	x	x	x	x
4-4	x	x	x	S	S	x	x	x	x	x
5-5	D	D	D	D	D	D	D	D	x	x
6-6	S	S	S	S	S	S	x	x	x	x
7-7	S	S	S	S	S	S	x	x	x	x
8-8	S	S	S	S	S	S	S	S	S	S
9-9	S	S	S	S	S		S	S		
10-10										

X = Pedir.

D = Dobrar.

S = Dividir.

Ficar onde não há ação indicada.

APÊNDICE G – ESTRATÉGIA DE *BLACKJACK* III.

Mão do jogador	Mão do crupiê									
	2	3	4	5	6	7	8	9	10	A
5	x	x	x	x	x	x	x	x	x	x
6	x	x	x	x	x	x	x	x	x	x
7	x	x	x	x	x	x	x	x	x	x
8	x	x	x	x	x	x	x	x	x	x
9	x	D	D	D	D	x	x	x	x	x
10	D	D	D	D	D	D	D	D	x	x
11	D	D	D	D	D	D	D	D	D	x
12	x	x	x	x	x	x	x	x	x	x
13	x	x				x	x	x	x	x
14						x	x	x	x	x
15						x	x	x	x	x
16						x	x	x	x	x
17						x	x	x	x	x
18										
19										
20										
21										
A-2	x	x	x	D	D	x	x	x	x	x
A-3	x	x	x	D	D	x	x	x	x	x
A-4	x	x	D	D	D	x	x	x	x	x
A-5	x	x	D	D	D	x	x	x	x	x
A-6	x	D	D	D	D	x	x	x	x	x
A-7	x	D	D	D	D	x	x	x	x	x

A-8		D	D	D	D			x	x	x
A-9										
A-10										
A-A	S	S	S	S	S	S	S	S	S	S
2-2	S	S	S	S	S	S	x	x	x	x
3-3	S	S	S	S	S	S	x	x	x	x
4-4	x	x	x	S	S	x	x	x	x	x
5-5	D	D	D	D	D	D	D	D	x	x
6-6	S	S	S	S	S	S	x	x	x	x
7-7	S	S	S	S	S	S	x	x	x	x
8-8	S	S	S	S	S	S	S	S	S	S
9-9	S	S	S	S	S	S	S	S	S	S
10-10	S	S	S	S	S	S	S	S	S	S

X = Pedir.

D = Dobrar.

S = Dividir.

Ficar onde não há ação indicada.