Trabajo Práctico 1.2 – Simulación

Renzo Aimaretti
renzoceronueve@gmail.com

Facundo Sosa Bianciotto
facundososabianciotto@gmail.com

Vittorio Maragliano
maraglianovittorio@gmail.com

Ignacio Amelio Ortiz
nameliortiz@gmail.com

Nicolás Roberto Escobar
escobar.nicolas.isifrro@gmail.com

Juan Manuel De Elia
juanmadeelia@gmail.com

29 de abril de 2025

Resumen

Este trabajo tiene como objetivo el análisis económico-matemático de estrategias de apuestas en la ruleta, abordado desde la perspectiva del apostador mediante simulaciones computacionales. Se desarrolla un simulador en Python 3.x que reproduce el comportamiento del juego y permite implementar diversas estrategias clásicas como Martingala, D'Alembert , Fibonacci y Paroli. El estudio contempla dos escenarios contrastantes: capital acotado (realista) y capital infinito (idealizado), con especial atención a la frecuencia de bancarrota en el primero. Los resultados obtenidos son representados gráficamente utilizando herramientas como Matplotlib, permitiendo evaluar la efectividad y el riesgo de cada estrategia a través de múltiples ejecuciones. Este enfoque busca desmitificar la posibilidad de obtener beneficios sistemáticos en la ruleta.

1. Introducción

Utilizamos Python para crear una ruleta simulada y probamos varias formas de apostar, entre ellas tres estrategias conocidas: Martingala, D'Alembert y Fibonacci. Además, agregamos una cuarta estrategia que se usa bastante en la práctica pero que no siempre se estudia tanto desde un punto de vista matemático: la estrategia Paroli, también conocida como la .ªnti-martingala", donde en vez de doblar la apuesta al perder, se la dobla cuando se gana, y en la cual se recomienda luego de 3 o 4 jugadas ganadas volver a la apuesta inicial.

Nuestro objetivo no es solo ver si se gana o se pierde, sino entender cómo se comportan estas estrategias cuando se juega con capital limitado (como en la vida real) y cuando se supone un capital infinito (más ideal). También analizamos cuántas veces el jugador termina en bancarrota y qué tan rápido pasa eso dependiendo de la estrategia. A lo largo del trabajo mostramos los resultados mediante gráficos, sacamos conclusiones y reflexionamos sobre si realmente existe una forma ïnteligente"de apostar, o si al final la ruleta sigue siendo un juego de azar, por más cálculos que hagamos.

En este informe, vamos a simular la ruleta con una apuesta de $10, y teniendo en cuenta que el apostador puede realizar los distintos tipos de apuesta:

- Apostar a un número específico (0-36).
- Apostar a un color(rojo o negro)
- Apostar a una docena(primer docena, segunda docena,tercer docena)
- Apostar a una fila(primera fila,segunda fila o tercera fila del tablero)
- Apostar a numero par o numero impar
- Apostar a numero alto o numero bajo(del 1 al 18 o del 19 al 36)

La apuesta que se decida realizar va a ingresarse por argumento en la consola

2. Descripción del programa

2.1. Funciones del programa

El programa está compuesto por varias funciones, destacandose la funcion jugar, la cual dependiendo de la jugada que se decida realizar va a llamar a las distintas funciones para cada juego. A continuación, se describe cada una de ellas en detalle.

2.2. Conceptos teóricos utilizados

Se emplearon las siguientes formulas de estadistica: 1. Frecuencia relativa real:

$$f_{\text{real}} = \frac{n_{\text{real}}}{n_{\text{total}}}$$

donde n_{real} es la frecuencia observada y n_{total} es el total de observaciones.

Se utilizaron las siguientes estrategias de apuestas: 1.Fibonacci : Se utiliza la secuencia de Fibonacci para determinar el monto de la apuesta. La secuencia comienza con 1 y 1, y cada número siguiente es la suma de los dos anteriores. Si se pierde, se avanza un número en la secuencia; si se gana, se retrocede dos números en la secuencia. Se emplea tambien el concepto teorico de la secuencia de Fibonacci, la cual es : 1,1,2,3,5,8,13,21,34,55,89,144.... 2.Martingala : Para la estrategia Martingala se duplica la apuesta después de cada pérdida, y se vuelve a la apuesta inicial después de una victoria. Esta estrategia busca recuperar las pérdidas anteriores con una sola victoria. 3.D'Alembert: La estrategia D'Alembert aumenta la apuesta en una unidad después de una pérdida y la disminuye en una unidad después de una victoria. Esta estrategia busca equilibrar las ganancias y pérdidas a lo largo del tiempo. 4.Paroli : La estrategia Paroli duplica la apuesta luego de cada victoria y vuelve a la apuesta inicial después de una pérdida. Esta estrategia busca maximizar las ganancias durante una racha ganadora.

2.2.1. simular_ruleta

La función `simular_ruleta` es la encargada de generar la tirada de la ruleta, simulando el lanzamiento de la bola y obteniendo un número aleatorio entre 0 y 36. Esta función utiliza la biblioteca `random` de Python para generar números aleatorios:

```
def simular_ruleta():
    return random.randint(0, 36)
```

2.2.2. estrategia

Esta función determina el monto de la próxima apuesta basándose en diferentes algoritmos según la estrategia seleccionada:

```
def estrategia(apuesta_actual, historial, gano, tipo):
    if tipo == 'm':
        return apostar_martingala(apuesta_actual, gano), historial
    elif tipo == 'd':
        return apostar_dalembert(apuesta_actual, gano), historial
    elif tipo == 'f':
        nueva_apuesta, nuevo_historial = apostar_fibonacci(historial, gano)
        return nueva_apuesta, nuevo_historial
    elif tipo == 'p':
        return apostar_paroli(apuesta_actual, gano), historial
    else:
        raise ValueError("Estrategia no reconocida")
```

Recibe como parámetros:

- `apuesta_actual`: Monto de la apuesta actual.

- `historial`: Lista con el historial de apuestas anteriores.
- `gano`: Booleano que indica si se ganó (True) o perdió (False) la última apuesta.
- `tipo`: Tipo de estrategia a aplicar ('m' para Martingala, 'd' para D'Alembert, 'f' para Fibonacci o 'p' para Paroli).

Retorna una tupla con dos elementos:

- El monto de la próxima apuesta calculado según la estrategia elegida.
- El historial de apuestas actualizado (relevante para la estrategia Fibonacci).

2.3. jugar

La función `jugar` tiene como propósito simular una sesión de apuestas en la ruleta, siguiendo una estrategia determinada, durante un número fijo de tiradas. Su diseño permite evaluar el comportamiento del capital a lo largo del tiempo y calcular la frecuencia relativa de aciertos en función de los resultados obtenidos.

```
def jugar(tiradas, numero_elegido, tipo_estrategia, capital_tipo, capital_inicial)
```

Los parámetros de entrada son:

- `tiradas`: cantidad total de jugadas a simular.
- `numero_elegido`: apuesta seleccionada (puede ser un número específico, una docena, una fila, par/impar, rojo/negro, número bajo/alto).
- `tipo_estrategia`: estrategia de progresión aplicada (puede ser Martingala, Fibonacci, D'Alembert o Paroli).
- `capital_tipo`: especifica si el capital es `fijo` o `infinito`. Si es `infinito`, el capital se reinicia cada vez que se agota.
- `capital_inicial`: cantidad de dinero inicial disponible para apostar.

Durante la simulación, se ejecutan las siguientes acciones principales:

1. Se inicializa el capital y la apuesta base.
2. Por cada tirada, se simula un resultado de ruleta y se evalúa si se ha ganado la apuesta según la apuesta elegida.
3. En caso de acierto, se incrementa el capital según el tipo de apuesta (pago 1:1, 2:1 o 35:1). En caso contrario, se descuenta el monto apostado.
4. La progresión de la apuesta se ajusta conforme a la estrategia elegida.
5. En caso de que el capital disponible no sea suficiente para realizar la siguiente apuesta y el tipo de capital no sea infinito, se considera que el jugador ha entrado en *banca rota*, lo que implica la finalización inmediata de la simulación de la corrida actual. Para mantener la coherencia estructural con otras corridas que sí completaron la totalidad de las tiradas, el historial de capital se completa artificialmente con el último valor de capital (que en este caso es cero) hasta alcanzar la longitud total de tiradas especificada. Esto permite un análisis gráfico homogéneo y evita desfases en las comparaciones visuales entre corridas.
6. Se almacena el capital luego de cada tirada.

Al finalizar, se construye un historial de capital y se calcula la **frecuencia relativa de éxito acumulada** (`frsa`), que indica la proporción de tiradas en las que el resultado fue favorable respecto a la cantidad de tiradas hasta el momento.
La función retorna:

- `capital_historial`: evolución del capital a lo largo de las tiradas.
- `banca_rota`: valor booleano que indica si el capital se agotó.
- `frsa`: arreglo con la frecuencia relativa acumulada de aciertos por tirada.

Visualización de resultados: funciones de graficación

Para facilitar el análisis visual del comportamiento del capital y la frecuencia relativa de éxito a lo largo de las tiradas, se han desarrollado tres funciones específicas de graficación. A continuación, se detalla su propósito y funcionamiento general:

- `graficar_capital`: Grafica la evolución del capital a lo largo de las tiradas.
- `graficar_frsa`: Grafica la frecuencia relativa de aparición de la opcion elegida.
- `graficar_todas_corridas`: Grafica la evolución del capital para todas

```
    def graficar_capital(capital_historial, titulo, nombre_archivo):
    plt.figure(figsize=(10, 6))
    plt.plot(range(1, len(capital_historial)+1), capital_historial, color='red')
    plt.axhline(y=capital_historial[0], color='blue', linestyle='--', label='Capital Inicial')
    plt.title(titulo)
    plt.xlabel("Tiradas")
    plt.ylabel("Capital")
    plt.grid(True)
    plt.legend()
    plt.savefig(nombre_archivo)
    plt.close()

def graficar_frsa(frsa, titulo, nombre_archivo):
    plt.figure(figsize=(8, 5))
    plt.bar(range(1, len(frsa) + 1), frsa, color='salmon', edgecolor='blue')
    plt.title(titulo)
    plt.xlabel("n (número de tiradas)")
    plt.ylabel("frsa (frecuencia relativa)")
    plt.ylim(0, 1)
    plt.grid(True, linestyle='--', alpha=0.5)
    plt.savefig(nombre_archivo)
    plt.close()

def graficar_todas_corridas(historiales_capital, titulo, nombre_archivo):
    plt.figure(figsize=(10, 6))
    for i, capital_historial in enumerate(historiales_capital):
        plt.plot(range(1, len(capital_historial) + 1), capital_historial,
         label=f"Corrida {i + 1}", alpha=0.7)
    plt.axhline(y=historiales_capital[0][0], color='blue', linestyle='--',
     label='Capital Inicial')
    plt.title(titulo)
    plt.xlabel("Tiradas")
    plt.ylabel("Capital")
    plt.grid(True)
    plt.legend()
    plt.savefig(nombre_archivo)
    plt.close()
```

2.4. Calcular resultado de la tirada

Estas funciones se ocupan de determiar si el numero que salio en la tirada corresponde una fila, una docena, par, impar, rojo o negro. Se llama una funcion u otra dependiendo el tipo de apuesta realizada por el jugador.

```
def get_row(number):
    if number == 0:
        return None
    if number % 3 == 1:
        return "primera"
    elif number % 3 == 2:
        return "segunda"
    else:
        return "tercera"

def get_dozen(number):
    if 1 <= number <= 12:
        return "primera docena"
    elif 13 <= number <= 24:
        return "segunda docena"
    elif 25 <= number <= 36:
        return "tercera docena"
    else:
        return None

def odd_even(number):
    if number == 0:
        return None
    return "par" if number % 2 == 0 else "impar"

def red_black(number):
    if number == 0:
        return None
    if number in [1, 3, 5, 7, 9, 12, 14, 16, 18, 19, 21, 23, 25, 27, 30, 32, 34, 36]:
        return "rojo"
    else:
        return "negro"
```

2.4.1. main

La función principal coordina la ejecución del programa:

```
def main():
    parser = argparse.ArgumentParser()
    parser.add_argument("-c", "--corridas", type=int, default=1, help="Número de corridas")
    parser.add_argument("-n", "--tiradas", type=int, default=1000,
     help="Número de tiradas por corrida")
    parser.add_argument("-e", "--elegido", type=str, default='17', help="Número elegido")
    parser.add_argument("-s", "--estrategia", type=str, required=True,
    help="Estrategia: m (martingala), d (D'Alembert), f (Fibonacci), p (Paroli)")
    parser.add_argument("-a", "--capital", type=str, required=True,
    help="Capital: f (finito), i (infinito)")
    parser.add_argument("-i", "--inicial", type=int, default=1000, help="Capital inicial")
    args = parser.parse_args()

    banca_rotas = 0
```

```
historiales_capital = []

for _ in range(args.corridas):
    capital_historial, banca_rota, frsa = jugar(args.tiradas, args.elegido,
    args.estrategia, args.capital, args.inicial)
    historiales_capital.append(capital_historial)
    if banca_rota or capital_historial[-1] <= 0:
        banca_rotas += 1

graficar_capital(capital_historial, f"Corrida
{args.corridas} - Evolución del Capital", f"capital_corrida_{args.corridas}.png")
graficar_frsa(frsa, f"Corrida
{args.corridas} - Frecuencia Relativa de Aciertos", f"frsa_corrida_{args.corridas}.png")


graficar_todas_corridas(historiales_capital,
"Evolución del Capital - Todas las Corridas", "capital_todas_corridas.png")
print(f"Simulación finalizada.")
print(f"Resultados de la estrategia '{args.estrategia}' con capital '{args.capital}':")
print(f"Capital inicial: {args.inicial}")
print(f"Número elegido: {args.elegido}")
print(f"Número de tiradas por corrida: {args.tiradas}")
print(f"Número de corridas: {args.corridas}")
print(f"Total de bancarrotas: {banca_rotas} de {args.corridas} corridas.")
```

Esta función:

- Recibe argumentos de línea de comandos para configurar la simulación (número de corridas, tiradas, número elegido, estrategia, tipo de capital y capital inicial).
- Ejecuta la función `jugar` para cada corrida, almacenando los resultados en una lista.
- Genera gráficos de la evolución del capital y la frecuencia relativa de aciertos utilizando las funciones `graficar_capital` y `graficar_frsa`.
- Imprime en la consola un resumen de los resultados obtenidos, incluyendo el número de bancarrotas y el capital final.
- Llama a la función `graficar_todas_corridas` para graficar la evolución del capital
- Muestra un resumen de los resultados obtenidos, incluyendo el número de bancarrotas y el capital final.

2.5. Resultados y Conclusiones

A continuación, se presentan los resultados de las simulaciones realizadas con las distintas estrategias de apuestas: Martingala, D'Alembert, Fibonacci y Paroli. Para cada una se consideraron dos escenarios:

- **Capital finito**: se establece un monto inicial de capital, y se detiene la simulación si se alcanza la bancarrota.
- **Capital infinito**: se asume que el jugador puede continuar apostando indefinidamente sin restricción de fondos.

Para cada combinación de estrategia y escenario, se muestran las siguientes tres gráficas:

- **Evolución del capital** en una corrida individual.

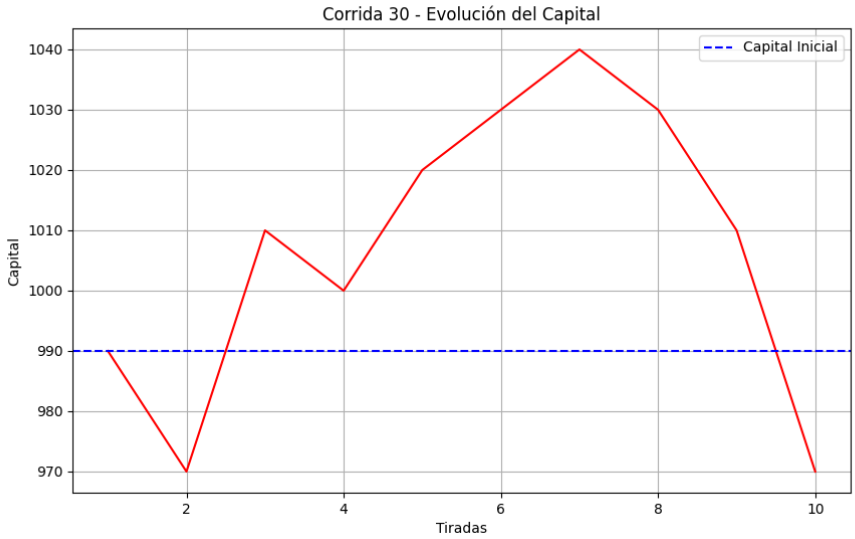

Figura 1: Martingala (finito) - Evolución del capital en una corrida

- **Frecuencia relativa de aciertos (FRSA)** en esa corrida.
- **Evolución del capital en 30 corridas** independientes.

Estrategia Martingala

Capital finito
Capital infinito

Estrategia D'Alembert

Capital finito
Capital infinito

Estrategia Fibonacci

Capital finito
Capital infinito

Estrategia Paroli

Capital finito
Capital infinito

Conclusión general: Las estrategias muestran comportamientos distintos dependiendo del capital disponible. La Martingala, por ejemplo, puede ser rentable en el caso de capital infinito, pero sufre bancarrota

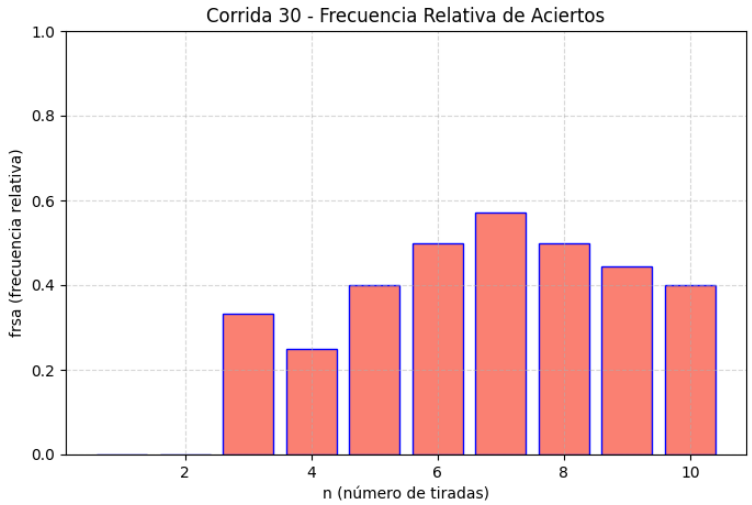

Figura 2: Martingala (finito) - Frecuencia relativa de aciertos

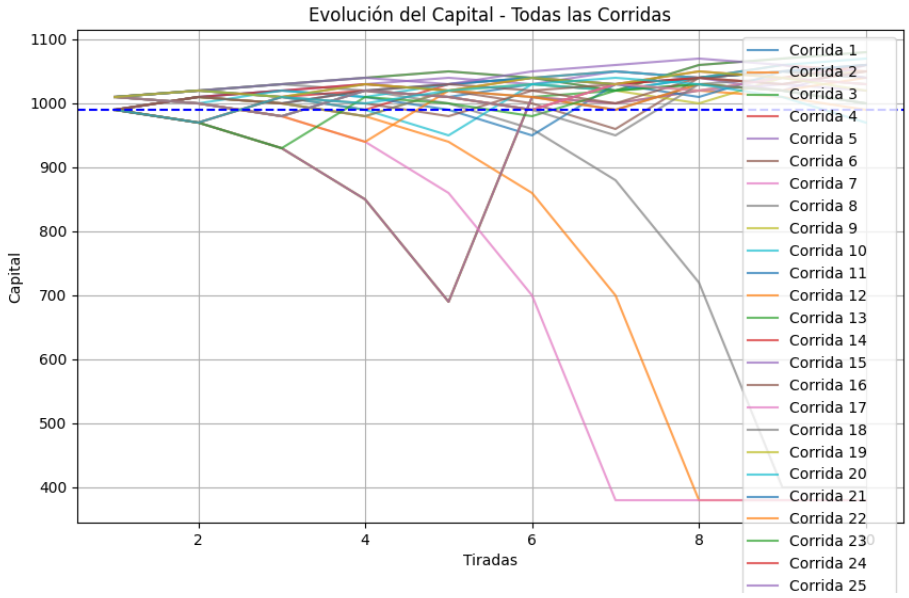

Figura 3: Martingala (finito) - Capital en 30 corridas

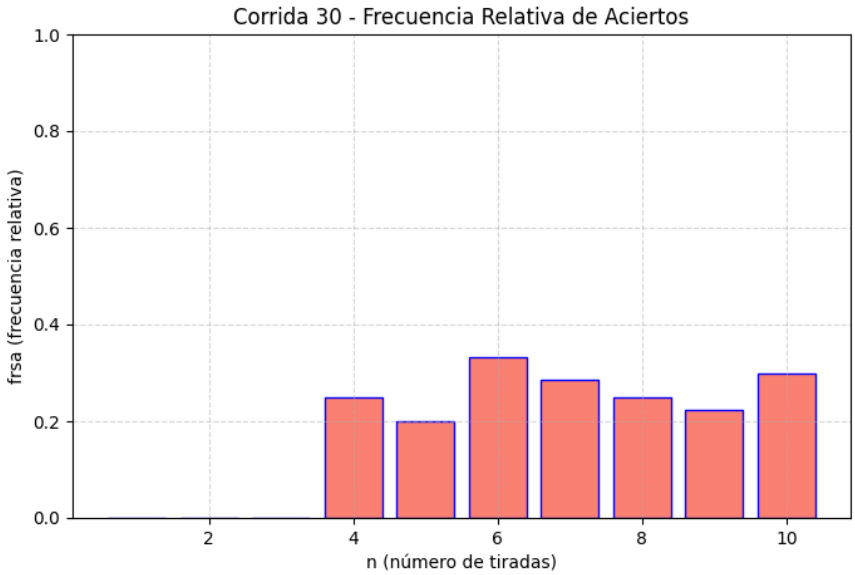

Figura 4: Martingala (infinito) - Evolución del capital en una corrida

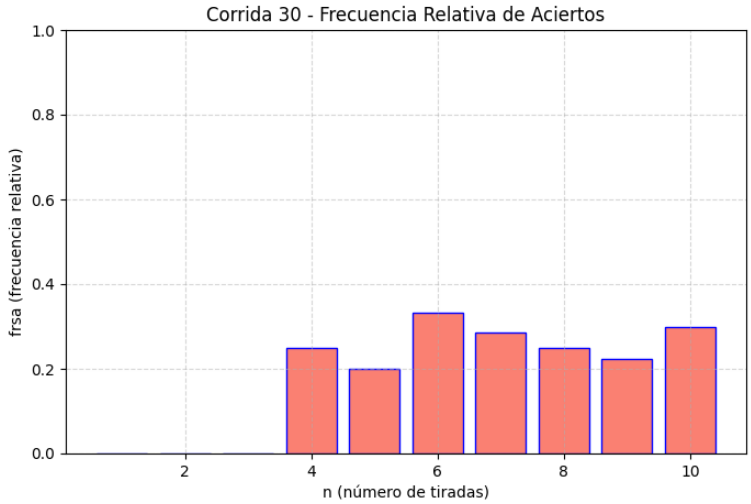

Figura 5: Martingala (infinito) - Frecuencia relativa de aciertos

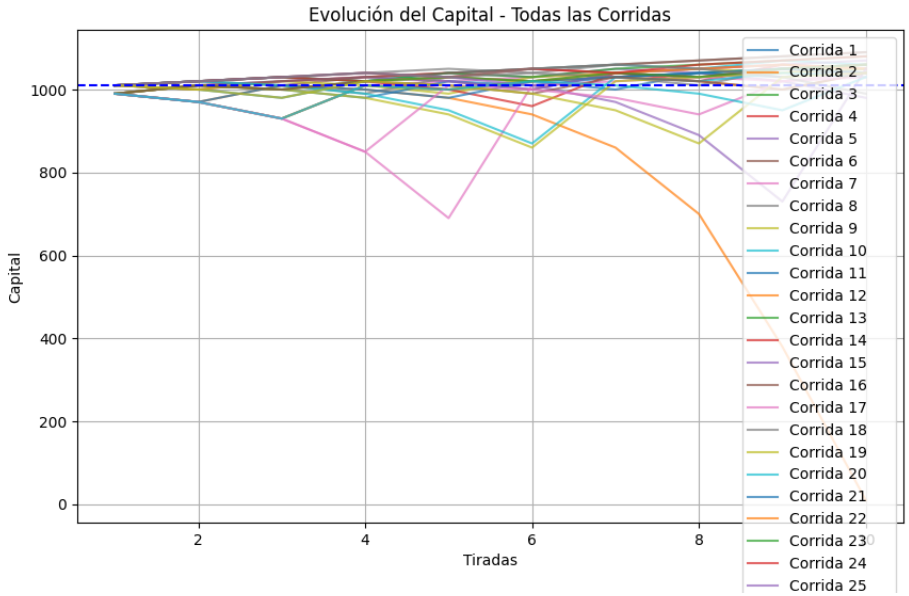

Figura 6: Martingala (infinito) - Capital en 30 corridas

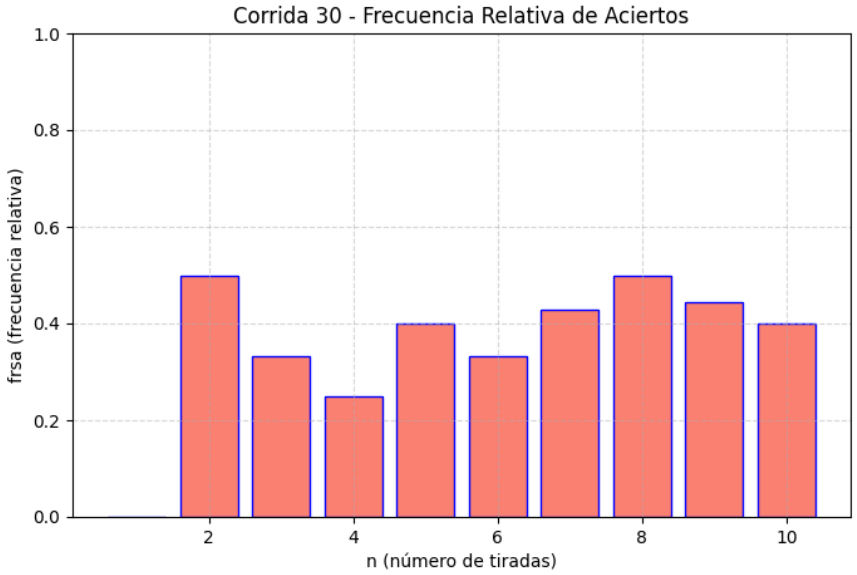

Figura 7: D'Alembert (finito) - Evolución del capital en una corrida

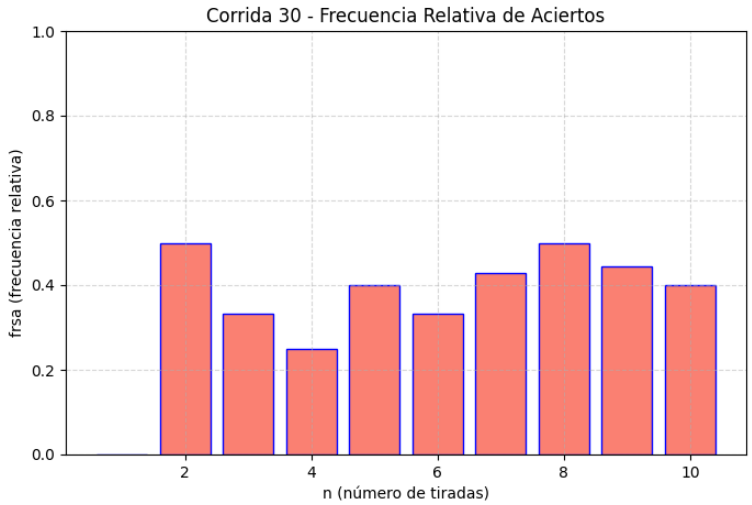

Figura 8: D'Alembert (finito) - Frecuencia relativa de aciertos

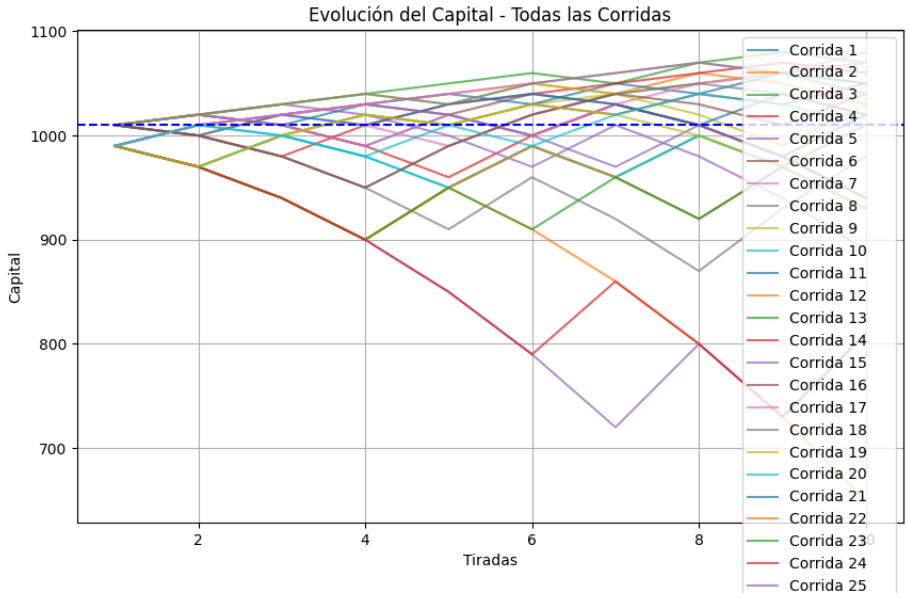

Figura 9: D'Alembert (finito) - Capital en 30 corridas

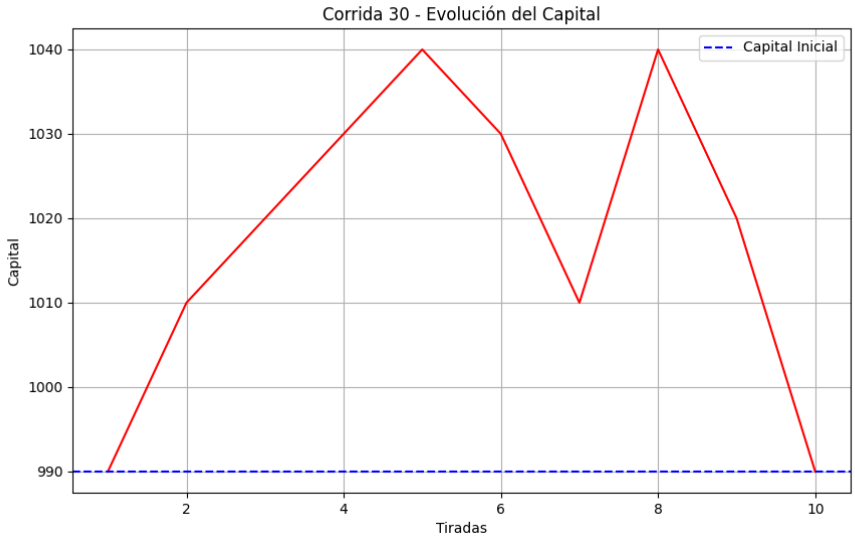

Figura 10: D'Alembert (infinito) - Evolución del capital en una corrida

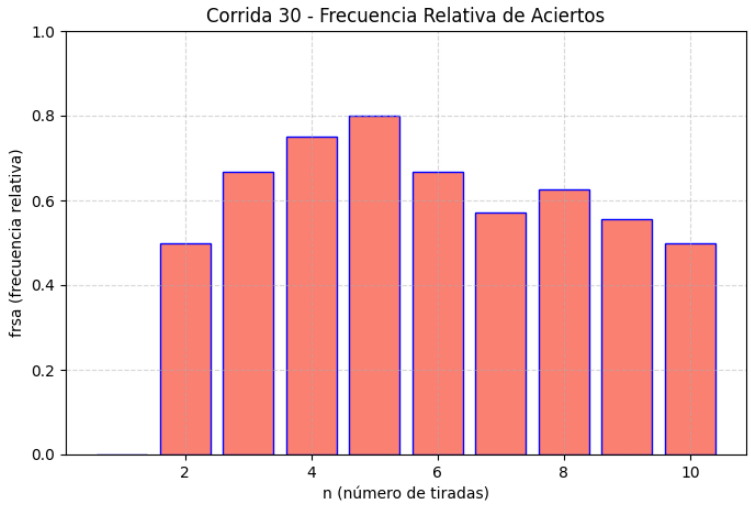

Figura 11: D'Alembert (infinito) - Frecuencia relativa de aciertos

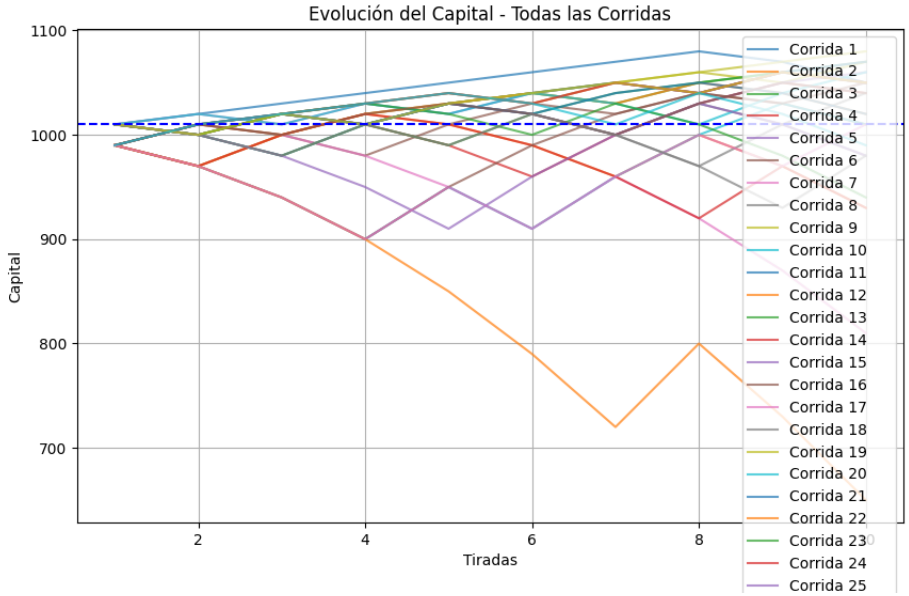

Figura 12: D'Alembert (infinito) - Capital en 30 corridas

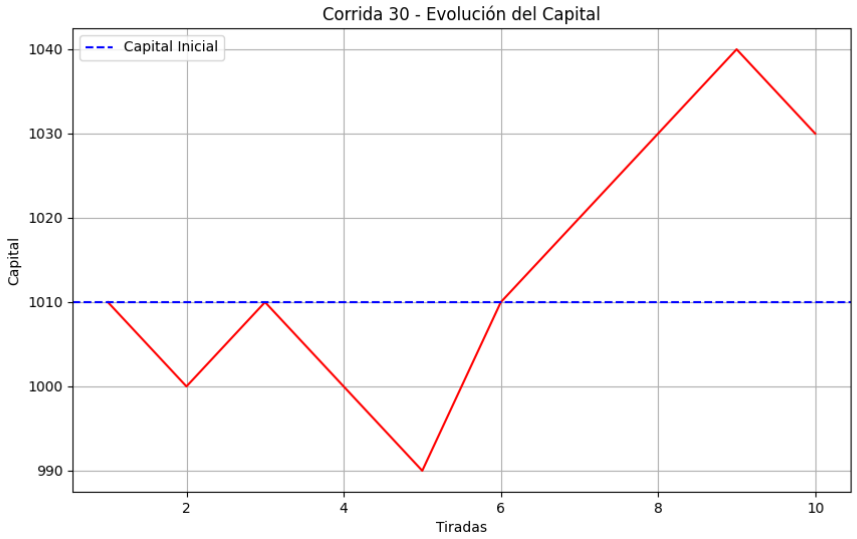

Figura 13: Fibonacci (finito) - Evolución del capital en una corrida

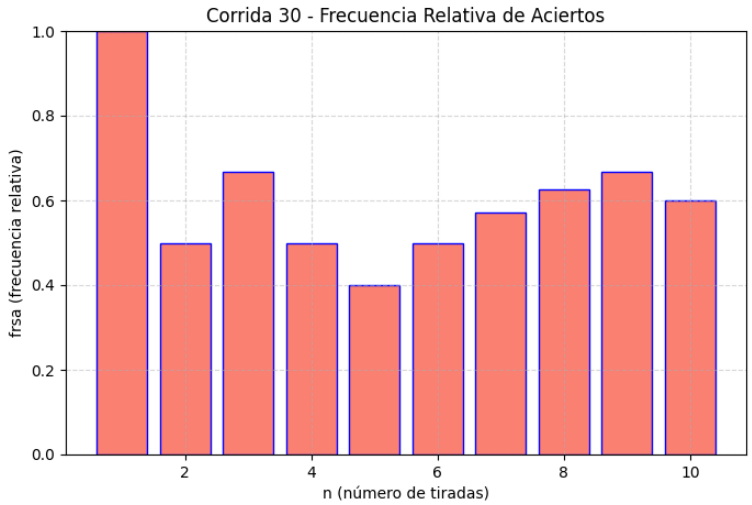

Figura 14: Fibonacci (finito) - Frecuencia relativa de aciertos

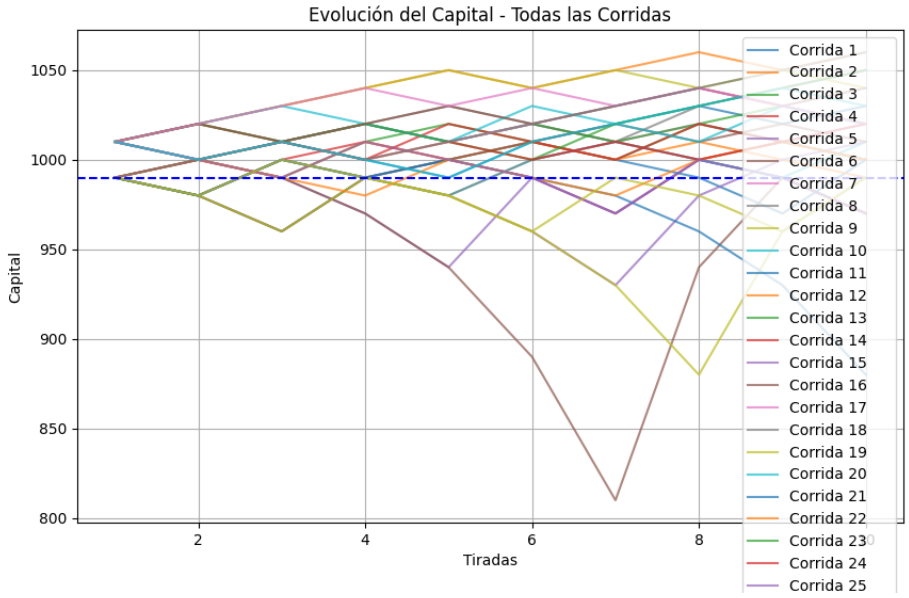

Figura 15: Fibonacci (finito) - Capital en 30 corridas

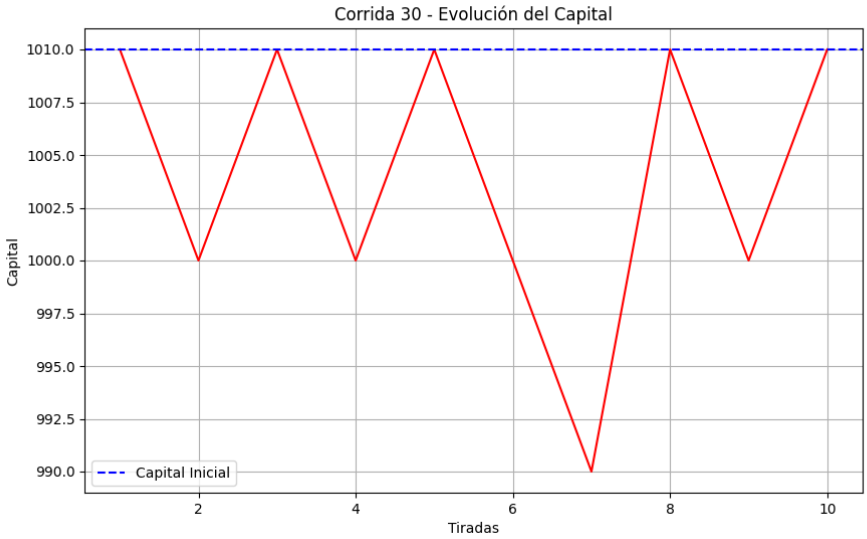

Figura 16: Fibonacci (infinito) - Evolución del capital en una corrida

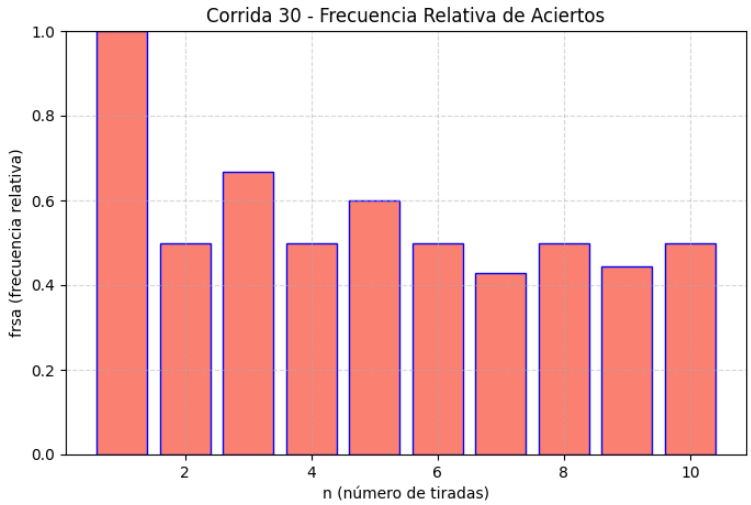

Figura 17: Fibonacci (infinito) - Frecuencia relativa de aciertos

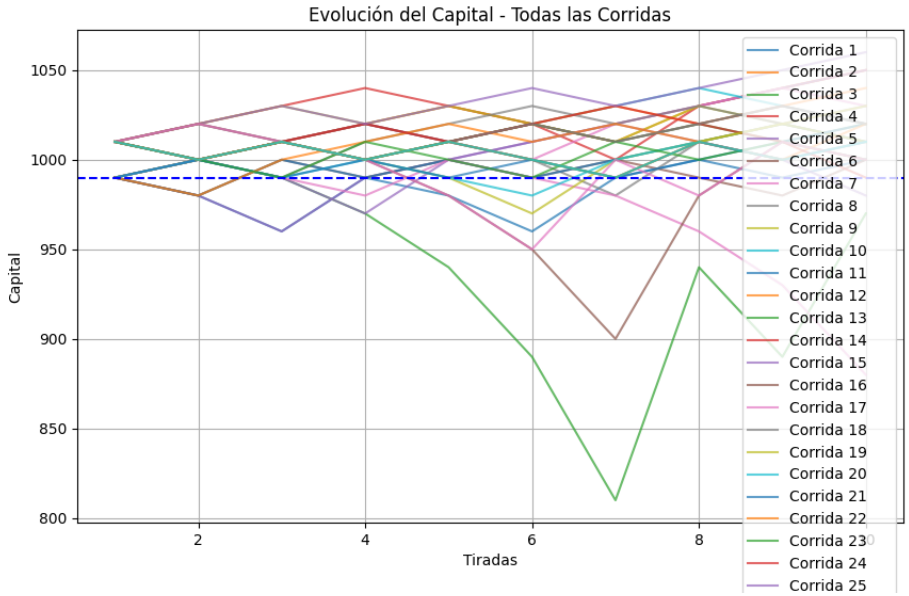

Figura 18: Fibonacci (infinito) - Capital en 30 corridas

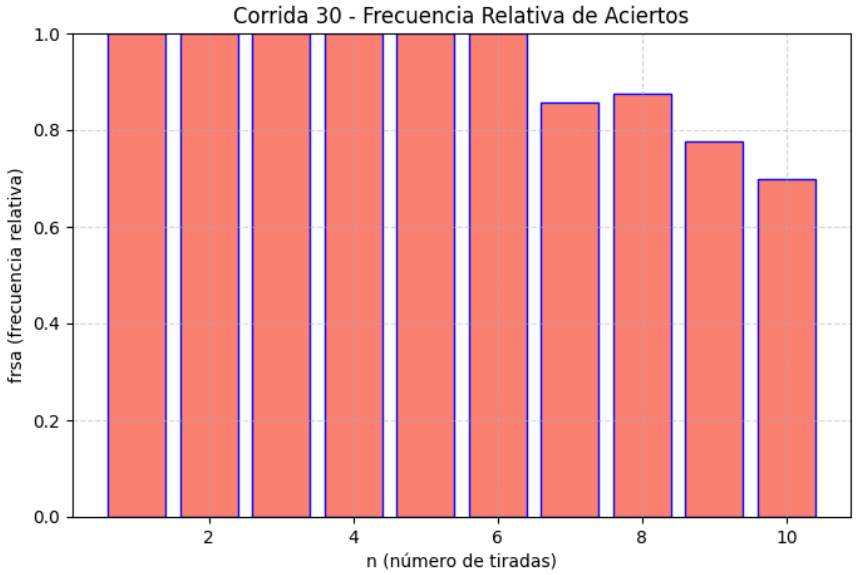

Figura 19: Paroli (finito) - Evolución del capital en una corrida

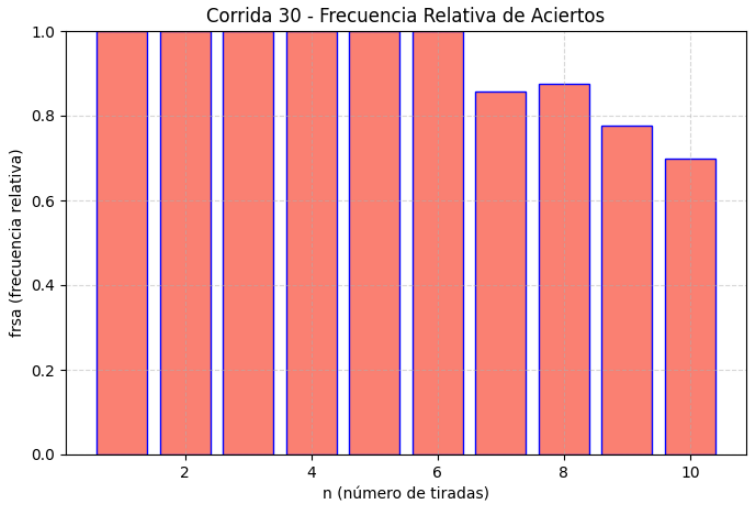

Figura 20: Paroli (finito) - Frecuencia relativa de aciertos

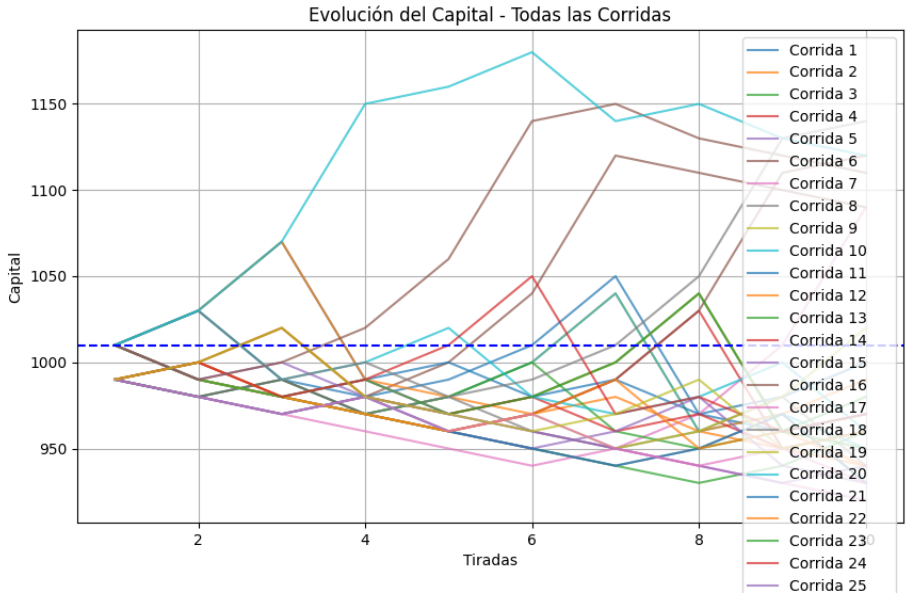

Figura 21: Paroli (finito) - Capital en 30 corridas

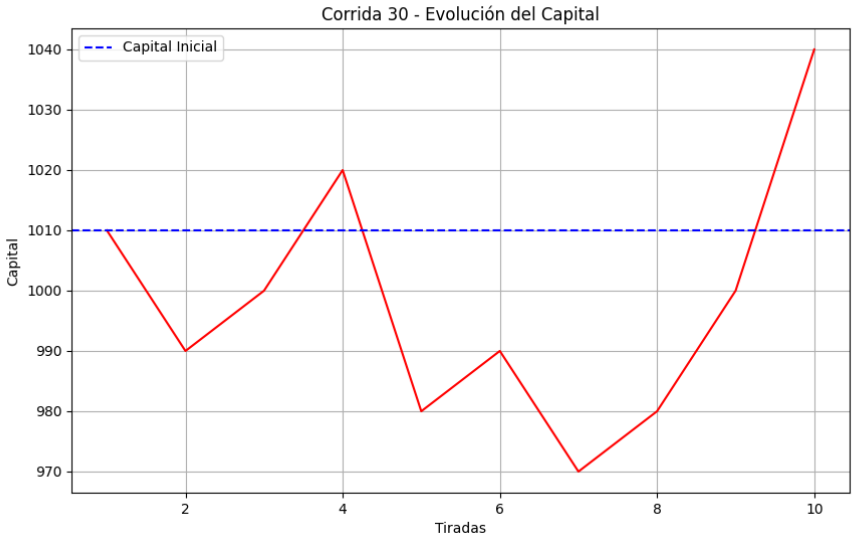

Figura 22: Paroli (infinito) - Evolución del capital en una corrida

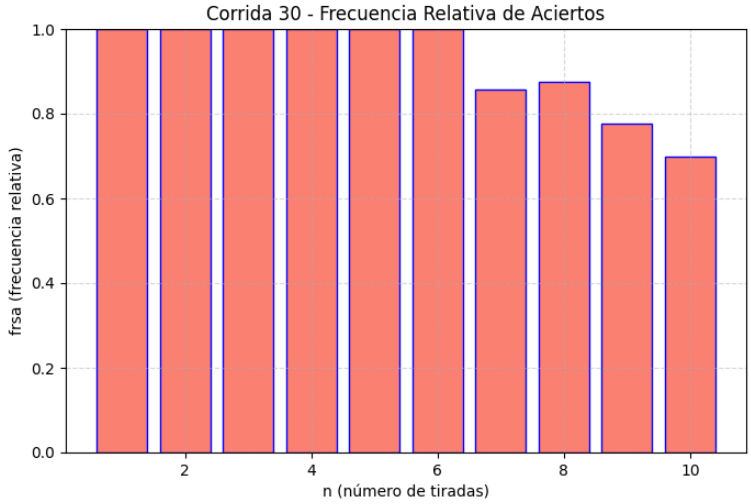

Figura 23: Paroli (infinito) - Frecuencia relativa de aciertos

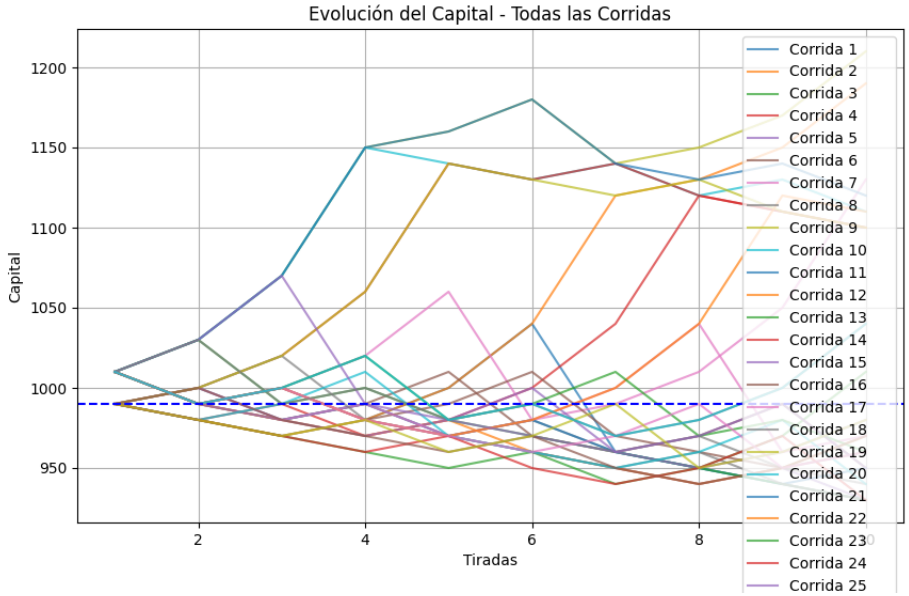

Figura 24: Paroli (infinito) - Capital en 30 corridas

rápidamente con capital finito. Estrategias como Paroli y D'Alembert presentan una progresión más conservadora, lo que reduce el riesgo, pero también limita las ganancias. En todos los casos, se confirma que ninguna estrategia asegura una ganancia sostenida en el largo plazo.