Juego de dados

Probabilidad y Estadística Aplicada

Prof: Maglis Mujica

Autores: Juan Nocetti - 5.390.966-5
Franco Barlocco - 5.308.038-6
Ionas Josponis - 5.242.903-0

10 de Mayo de 2024

Índice

Introducción

En este proyecto crearemos un programa en Python sobre un juego de dados entre dos jugadores, posteriormente simularemos partidas cierta cantidad de veces para analizar las distintas estrategias que utilizan los jugadores.
El juego requiere dos dados y se desarrolla entre dos jugadores, Juan y María. Ambos jugadores tiran los dados, y el que obtenga mayor puntaje será el ganador. Los posibles puntajes son {0, 1, 2, 3, 4, 5, 6}.

Resultados	Puntaje
$(4,1)$	1
$(1,4)$	1
$(4,2)$	2
$(2,4)$	2
$(4,3)$	3
$(3,4)$	3
$(4,4)$	4
$(4,5)$	5
$(5,4)$	5
$(6,4)$	6
$(4,6)$	6

Las distintas reglas del juego dependen de las tiradas:

- Puede haber como máximo dos tiradas de ambos dados en simultáneo, y solo se suma cuando un dado tiene valor 4.
- Si en la primera tirada obtienes un dado con el valor 4 puedes mantenerlo, y solo tirar el otro para intentar obtener más puntos.

Cada jugador utiliza una estrategia :

Estrategia de Juan:

- Si obtiene 0 puntos utilizara su segunda tirada de ambos dados.
- Si obtiene 1, 2 o 3 puntos utilizará su segunda tirada en uno de los dados para intentar mejorar su puntaje.
-Si obtiene más de 3 puntos, se plantará en su puntaje y no utilizara su tirada adicional de uno de los dados.

Estrategia de María:
Al tirar después de Juan, María conoce el puntaje de Juan, por lo que intentará ganar. Si en su primera tirada no supera a Juan utiliza la segunda tirada. En caso contrario mantiene su puntaje para así ganar.

Objetivos

1- Crear un programa en Python para simular el juego.
2- Analizar las distintas estrategias de juego que utilizan Juan y María.
3- Desarrollar una estrategia para maximizar las chances de que gane María.
4- Obtener las distintas probabilidades de conseguir los posibles puntajes del juego.
5- Simular el juego 1.000, 10.000 y 100.000 veces y calcular la frecuencia relativa de los ganadores(y/o empates).
6- Analizar si María tiene ventaja sobre Juan al tirar segunda.
.

Marco Teórico

Probabilidad: marco matemático para describir y analizar fenómenos aleatorios, donde se entiende por fenómeno aleatorio a un evento o experimento cuyo resultado no puede ser predecido con certeza.

Espacio muestral: conjunto de resultados posibles de un experimento aleatorio, se le denomina con la letra Ω. En este problema.
Frecuencia relativa: Es la frecuencia absoluta dividida la cantidad de elementos.

Herramientas

- IDE: Es un entorno de desarrollo para poder programar, en nuestra aplicación usamos PyCharm de JetBrains y Visual Studio Code.

- **Python:** Lenguaje de desarrollo.

-GitHub: Es una herramienta para desarrollar software colectivamente.

-Random: Librería de Python para poder generar números aleatorios y realizar operaciones.

-Draw.io: aplicación que permite crear distintos diagramas.

-Drawio Preview: extensión de Visual Studio Code para poder ver los diagramas con extensión .drawio. La puedes obtener aquí https://marketplace.visualstudio.com/items?itemName=purocean.drawio-preview

Desarrollo

En el programa tenemos dos clases, “Utils.py” para realizar todas las funciones necesarias para simular el juego y “Main.py” como clase principal para ejecutar las funciones.

Funciones dentro de la clase Utils

throw_dice() y throw_die(): Estas funciones nos permiten simular la tirada de uno o dos dados utilizando la librería “random”.

```python
import random

def throw_dice():
    results = []
    results.append(random.randint(1, 6))
    results.append(random.randint(1, 6))
    return results

def throw_die():
    return random.randint(1, 6)
```

Figura 1

play(previus_score, show_game): Función la cual contiene la gran mayoría de la lógica para simular el juego de dados. Al ser una función extensa, explicaremos cada parte en profundidad.

En la figura 2, definimos la función “*play*” la cual tiene dos parámetros:

-*previus_score*: Se usa para que el programa sepa qué jugador está jugando. Si este parámetro es ‘None’, significa que es el turno de Juan(ya que juega primero sin saber el puntaje de María). Si este contiene un valor entero, significa que está jugando María ya que conoce el puntaje final del jugador

anterior. En la figura 3 se puede observar como dependiendo de este parámetro el programa imprime un nombre u otro(en caso que se desee imprimir).

-*show_game*: Es un valor booleano que si es verdadero muestra el paso a paso del juego y el puntaje final. Sino, solamente retorna el puntaje.
Es útil por si realizamos muchas ejecuciones y no deseamos ver el transcurso del juego, sino que solamente deseamos ver el puntaje final de cada jugador.
Se optó por agregar este parámetro para no tener que repetir código, y lograr tener la funcionalidad de imprimir el proceso del juego.
En la figura 3 y 4 se puede observar este comportamiento.

También inicializamos las variables locales "*die1*" y "*die2*" para asignar los resultados de cada dado en el primer lanzamiento

```
def play(previus_score, show_game):
    score = 0
    first_throw_result = throw_dice()
    die1 = first_throw_result[0]
    die2 = first_throw_result[1]
```

Figura 2

```
if show_game:
        # If previus_score is None, Juan is playing, otherwise
María is playing
        if previus_score == None:
            print("Juan turns\n")
        else:
            print("María turns\n")
```

Figura 3

```
if show_game:
        print("First throw: ")
        print("First die value: ", die1)
        print("Second die value: ", die2, "\n")
```

Figura 4

Determinar si el juego termina o si se necesitan más lanzamientos

La figura 5 muestra el caso de que ambos dados tengan un valor de 4. Se desarrolló así ya que Juan se conforma con un valor mayor que 3. En el caso de María, si el puntaje de Juan es mayor al de ella en su primera tirada, volverá a tirar un dado para ver si puede superarlo. El caso de empate está explicado en la conclusión número 3. Finalmente si el puntaje de María(en su primera tirada) supera al de Juan, no vuelve a tirar.

```
    # The player got a 4 in both dice, the final score is 4
    if(die1 == 4 and die2 == 4):
        if previus_score!=None and previus_score > 4:
            # María must throw again because she will lose
            if show_game: print("Throw again one die.")
            die1 = throw_die()
            if show_game: print("First die new value: ", die1)
        score = die1
        if show_game: print("Final score: ", score, "\n")
        return score
```

Figura 5

En la figura 6 representa el caso el cual uno de los dados es 4 y el otro es mayor que 4.
Para Juan, el puntaje final es el valor del dado que no es 4.
Para María, si el puntaje de Juan es menor o igual al de ella en su primera tirada, no vuelve a tirar su segundo dado. Si el puntaje de Juan es mayor al de María, ella vuelve a tirar un dado.

```
       else:
        # The player got a 4 in one of the dice, the final
score is the other die value
        if (die1 == 4 and die2 >= 4) or (die2 == 4 and die1 >=
4 ):
            # This if represents the case when Juan is playing
or Juan's result is
            # lower or the same than María first throw.
            if(previus_score == None or previus_score <= 4):
                if die1 == 4:
                    score = die2
                else:
                    score = die1
            else:
```

```
                # María will lose if she does not throw again
the die distinct to 4
                if die1 == 4:
                    if show_game: print("Throw again second
die.")
                    die2 = throw_die()
                    if show_game: print("Second die new value:
", die1)
                    score = die2
                else:
                    if show_game: print("Throw again first
die.")
                    die1 = throw_die()
                    if show_game: print("First die new value:
", die1)
                    score = die1
            if show_game: print("Final score: ", score, "\n")
            return score
```

Figura 6

Caso donde uno de los dados es 4 y el otro dado es menor o igual a 3

Si el primer dado(‘die1’) es 4 y el otro(‘die2’) es menor o igual a 3 entonces:

- Si “*previus_score*” es None, lanzamos el segundo dado nuevamente ya que es la estrategia de Juan. Si el nuevo valor de ‘die2’ es diferente, se actualiza el score.
- Si es el turno de María, lanza nuevamente el ‘die2’ si el puntaje de Juan es mayor o igual a el actual valor del ‘die2’.

Se muestra el resultado final y la función termina retornando el puntaje “*score*”.

```
# The player got a 4 in one of the dice and a number lower
than 3 in the other, can throw the other die again
if(die1==4 or die2==4):
    if die1 == 4 and die2 <= 3:
        if(previus_score == None):
                # Juan strategy, he must throw again
                if show_game: print("Throw again the second
die.")
                die2 = throw_die()
```

```
                if show_game: print("Second die new value: ",
die2)
                score = die2
                if show_game: print("Final score: ", score,
"\n")
                return score
        else:
                # María already knows Juan's result
            if(previus_score >= die2):
                # María must throw again because she will lose
                if show_game: print("Throw again the second
die.")
                    die2 = throw_die()
                    if show_game: print("Second die new value:
", die2)
                        score = die2
                    if show_game: print("Final score: ",
score, "\n")
                        return score
```

Figura 7

En la figura 8 realizamos el mismo procedimiento para cuando los valores de los dados están invertidos. ‘die2’ igual a 4 y ‘die2’ menor o igual a 3.

```
if die2 == 4 and die1 <= 3:
     if(previus_score == None):
          # Juan strategy, he must throw again
          if show_game: print("Throw again the first die.")
          die1 = throw_die()
          if show_game: print("First die new value: ", die1)
          score = die1
          if show_game: print("Final score: ", score, "\n")
          return score
     else:
         # María already knows Juan's result
         if(previus_score > die1):
          # María must throw again because she will lose
```

```
                if show_game: print("Throw again the first
die.")
            die2 = throw_die()
            if show_game: print("First die new value: ",
die2)
            if (previus_score == die2):
            # María  throw again because she is more
probably to win with a new value
                if show_game: print("Throw again the second
die.")
                die2 = throw_die()
                if show_game: print("Second die new value:
", die2)
                score = die1
                if show_game: print("Final score: ", score,
"\n")
                return score
```

Figura 8

Caso el cual ninguno de los dados salga el valor 4, se debe realizar otro lanzamiento.

En la figura 9, se tira nuevamente ambos dados. Si uno de los dados tiene valor 4, retorna el valor del otro dado.

```
if die1 != 4 and die2 != 4:
            if show_game: print("You must throw again \n")
            second_throw_results = throw_dice()
            die1 = second_throw_results[0]
            die2 = second_throw_results[1]
            if show_game:
                print("Second throw: ")
                print("First die value: ", die1)
                print("Second die value: ", die2, "\n")
            if die1 == 4:
                score = die2
            if die2 == 4:
                score = die1
```

Figura 9

Por último en la figura 10, se retorna el score final

```
if show_game: print("Final score: ", score, "\n")
    return score
```

Figura 10

def winner_and_results(show_game): Esta función simula las tiradas de Juan y María, comparara sus puntajes y retorna el ganador(o si hubo empate). Se aprecia en la figura 11.

```
def winner_and_results(show_game):
    juan_results = play(None, show_game)
    maria_results = play(juan_results, show_game)
    if juan_results > maria_results:
        winner = "Juan wins"
    elif juan_results < maria_results:
        winner = "María wins"
    else:
        winner = "Draw"

    return winner, juan_results, maria_results
```

Figura 11

def print_results(): Método que imprime por consola los resultados de la partida de Juan y María. Figura 12.

```
def print_results():
    print("Dices game\n")
    winner, results_juan, results_maria =
winner_and_results(True)
    print("Final results of the game between Juan and María
after throwing the dices:")
    print("\tJuan: ", results_juan)
    print("\tMaría: ", results_maria)
    print(winner)
```

Figura 12

def simulate_games(n, show_game): Esta función permite simular partidas entre los jugadores, muestra los resultados y la frecuencia relativa.

Parámetros de la función:

- “*n*”: Número de juegos a simular.
- “*show_game*”: Como ya mencionamos, es un booleano que indica si se deben mostrar detalles del juego en la consola.

Inicialización de contadores:

- Se inicializan tres contadores “*juan_wins”*, “*maria_wins*” y “*draws*” con valor 0.

Simulación de juegos:

- Se usa un bucle for para simular los n juegos.
- En cada iteración, se llama a “*winner_and_results*(*show_game*)” para obtener al ganador.

Dependiendo del resultado:

- Si winner es "Juan wins", se incrementa “*juan_wins”*.
- Si winner es "María wins", se incrementa “*maria_wins”*.
- Si es un empate ("Draw"), se incrementa “*draws”*.

Impresión de resultados finales:

- Al final del bucle, se imprime el número total de juegos jugados.
- Se muestran cuántas veces ganó Juan, cuántas veces ganó María, y cuántos empates hubo.
- Se calculan y muestran las frecuencias relativas de las victorias para cada jugador y empates.

```
def simulate_games(n, show_game):
    juan_wins = 0
    maria_wins = 0
    draws = 0
    for i in range(n):
        winner, _, _ = winner_and_results(show_game)
        if winner == "Juan wins":
            juan_wins += 1
        elif winner == "María wins":
            maria_wins += 1
        else:
            draws += 1
    print("Results after played ", n, " games:")
    print("\tJuan has won ", juan_wins, " times")
    print("\tMaría has won ", maria_wins, " times")
    print("\tThere were ", draws, " draws\n")
```

```
    print("\tJuan relative frequence:", float(juan_wins/n))
    print("\tMaría relative frequence:", float(maria_wins/n))
    print("\tDraws relative frequence:", float(draws/n), "\n")
```
Figura 13

Clase “Main”
Esta clase es donde ejecutamos el programa, simplemente importamos la clase “utils” para tener todas las funciones y llamamos a los métodos necesarios con los parámetros que deseemos.

```
import  utils

# For play one time
utils.print_results()

# To play 1000, 10000, 100000 times
utils.simulate_games(1000, False)
utils.simulate_games(10000, False)
utils.simulate_games(100000, False)
```
Figura 14

Conclusiones

1- Logramos implementar un programa en Python que simule el juego entre Juan y María.

2- En el programa analizamos las estrategias de los jugadores. La estrategia de Juan consiste en conformarse con puntajes mayores a 3 ya que sabe que es un puntaje alto y tiene pocas chances de perder.
La estrategia de María consiste en usar la probabilidad ante los casos de empate para tratar de superar a Juan o no.

3- La técnica desarrollada para que María sea más eficiente en su juego(pierda menos) consiste en los casos de empate. Si en su primera tirada empata con Juan, se analiza las probabilidades de ganar, volver a empatar y perder en su segunda tirada. Si las chances de ganar son mayores a las de perder, María vuelve a tirar.

Si en la primera tirada de María, ambos jugadores empatan con un puntaje menor o igual a 3, María tira nuevamente el dado que no es 4.Esto se debe a que tiene una mayor probabilidad de ganar que perder.

Por ejemplo, en el caso que ambos empaten con un puntaje de 3, la probabilidad de ganar es de $\frac{3}{6}$, de empatar es de $\frac{1}{6}$ y de perder es de $\frac{2}{6}$.
Justificamos con el valor de empate igual a 3 ya que en los casos de empate con valores 1 y 2, la probabilidad de ganar es aún mayor.

Si en la primera tirada de María, ambos jugadores empatan con un puntaje mayor a 3, María no tira nuevamente. Esto debido a que la probabilidad de perder es mayor a la probabilidad de ganar.
Supongamos que ambos jugadores empatan con un puntaje de 4 La probabilidad de perder es de $\frac{3}{6}$, de empatar es de $\frac{1}{6}$ y de ganar es de $\frac{2}{6}$.
Mencionamos este ejemplo ya que en los casos de empatar con los valores 5 y 6, la probabilidad de perder aumenta.

4- Realizamos un diagrama con la herramienta “Draw.io” que nos permitió obtener todas las probabilidades posibles de conseguir los distintos puntajes del juego(diagrama cargado en el repositorio Github)

5- Desarrollamos una función que permite repetir el juego 1.000, 10.000 y 100.000 veces, y calcular la frecuencia relativa de cada simulación.

6- Antes de desarrollar el juego nos parecía que María tenía una amplia ventaja en el juego al conocer el puntaje de Juan. Sin embargo, los resultados de la simulación muestran como Juan tiene ventaja en el juego.

Pasos a futuro

-Generar casos de pruebas unitarias de cada función para validar un buen comportamiento del juego.
-Implementar una interfaz gráfica que nos permita simular el juego con elementos visuales.
- Agregar funcionalidad para que cada persona nueva que juegue genere sus propias estrategias.
-Simular el juego pero con la diferencia de que Juan y María tengan 3 oportunidades de tirar los dados y analizar las opciones.

Bibliografía

Universo Formulas. (s.f.). Frecuencia relativa. Recuperado de https://www.universoformulas.com/estadistica/descriptiva/frecuencia-relativa/