
# 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

**04 de Abril de 2024**

# Índice



# Introducción

**Monty Hall**

Monty Hall fue un presentador de programas de televisión canadiense que tenía un concurso televisivo que consiste en un problema matemático el cual se basa en la probabilidad de ganarse un premio dependiendo de ciertos factores, a continuación una cita que describe el problema:

*"En este concurso, el concursante escoge una puerta entre tres, y su premio consiste en lo que se encuentra detrás. Una de ellas oculta un coche, y tras las otras dos hay una cabra. Sin embargo, antes de abrirla, el presentador, que sabe dónde está el premio, abre una de las otras dos puertas y muestra que detrás de ella hay una cabra. Ahora tiene el concursante una última oportunidad de cambiar la puerta escogida* ***¿Debe el concursante mantener su elección original o escoger la otra puerta? ¿Hay alguna diferencia?*** *Estadística para todos"*

# Objetivos

1- Desarrollar un programa en Python que permita simular el programa "Lets make a deal".

2- Simular las posibles estrategias del programa, quedarse con la puerta elegida o cambiar de puerta.

3- Probar el programa 1.000, 10.000 y 100.000 veces con cada estrategia.

4- Analizar y comparar estas estrategias con los resultados de la simulación

.

# 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 denota con la letra Ω. En este problema

**Frecuencia absoluta:** La frecuencia absoluta de un valor X es la cantidad de veces que el valor se encuentra en el conjunto.

**Frecuencia relativa:** Es la frecuencia absoluta dividida la cantidad de elementos.

**Porcentaje**: Para calcular el porcentaje de un valor hay que hacer una regla de tres, dividir nuestro valor por la cantidad de elementos y lo multiplicamos entre 100.

# 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.

# Desarrollo

Para implementar el programa, creamos dos clases utils y main. Dentro de estas clases hay distintos métodos que hacen posible simular el programa televisivo.

Funciones dentro de la clase utils:

***crearPuertas():*** Esta función genera aleatoriamente una lista de tres elementos, ya que se genera un número del 1 al 3, el cual corresponde a la posición donde se ubicará el número 1, en las posiciones restantes se ingresarán números con valor 0.
Los ceros corresponden a las cabras, el uno al auto y las posiciones de la lista a las puertas.

```
def crearPuertas():
```

```
    puertas = []
    num = random.randint(0,2)
    for i in range(3):
        if (i == num):
            puertas.append(1)
        else:
            puertas.append(0)
    return puertas
```

***def montyHall():***

Este método simula el juego y permite al usuario decidir si elegir quedarse con la puerta que eligió o cambiar a la otra. Utiliza el metodo crearPuertas(), simula la puerta elegida por el usuario y luego crea una lista con las puertas que tienen valor 0 (esto significa que tiene detras una cabra).
Básicamente se crea una lista vacía en la cuál mediante un bucle “for” se chequea si en la puerta hay una cabra o no y si no es la puerta elegida por el usuario. En el caso de que el usuario acierte al auto, en la lista habrán dos elementos y posteriormente se muestra de forma aleatoria una de las puertas que posee la cabra.
La otra opción es que el usuario elija una cabra y en la lista sin premio quede una cabra, la cuál será revelada su posición en la lista. Luego se pregunta al usuario si se queda en su puerta o cambia a la otra y finalmente se retorna si detrás de la puerta seleccionada hay una cabra o un auto.

```
def montyHall():
    puertas = crearPuertas()
    seleccion = random.randint(0, 2)

    print("Su selección es:", seleccion)

    puertas_sin_auto = []

    for i in range(3):
        if i != seleccion and puertas[i] == 0:
            puertas_sin_auto.append(i)

    if len(puertas_sin_auto) == 2:
        # la funcion random.choice() elige un elemento aleatorio de una
lista
        puerta_abierta = random.choice(puertas_sin_auto)
    else:
        puerta_abierta = puertas_sin_auto[0]
```

```
    print("La puerta", puerta_abierta, "no tiene el premio")

    print("Desea cambiar de puerta? Responda si o no")
    eleccion = input().lower()

    if "si" in eleccion:
        for i in range(3):
            if i != seleccion and i != puerta_abierta:
                seleccion = i
                break

    if puertas[seleccion] == 1:
        print("¡Felicidades, ha ganado!")
        return True
    else:
        print("Lo siento, ha perdido")
        return False
```

***def MontyHallCambiaPuerta():*** función que simula el juego utilizando la estrategia del cambio de puerta.

```
def MontyHallCambiaPuerta():
    puertas = crearPuertas()
    seleccion = random.randint(0, 2)

    puertas_sin_auto = []

    for i in range(3):
        if i != seleccion and puertas[i] == 0:
            puertas_sin_auto.append(i)

    if len(puertas_sin_auto) == 2:
        # la funcion random.chocie() elige un elemento aleatorio de una
lista
        puerta_abierta = random.choice(puertas_sin_auto)
    else:
        puerta_abierta = puertas_sin_auto[0]

    for i in range(3):
            if i != seleccion and i != puerta_abierta:
                seleccion = i
                break
```

```
    if puertas[seleccion] == 1:
        return True
    else:
        return False
```

***def MontyHallNoCambiaPuerta():*** método que simula el juego utilizando la estrategia de no cambiar de puerta.

```
def MontyHallNoCambiaPuerta():
    puertas = crearPuertas()
    seleccion = random.randint(0, 2)

    if puertas[seleccion] == 1:
        return True
    else:
        return False
```

**Funciones dentro de la clase main:**

***simularJuegoCambiandoDePuerta(veces):*** método para simular el juego usando la estrategia del cambio de puerta las veces que consideres necesarias. El parámetro veces le indica a la función cuántas veces se tiene que simular el juego.

```
def simularJuegoCambiandoDePuerta(veces):
    cantidad_ganadas = 0
    cantidad_perdidas = 0
    for i in range(veces):
        if utils.MontyHallCambiaPuerta() == True:
            cantidad_ganadas += 1
        else:
            cantidad_perdidas += 1


    print("\nIteracion ", limite)
    print("Cantidad de veces que ganó:", cantidad_ganadas, ",
Frecuencia relativa = ",cantidad_ganadas/limite, ", Porcentaje de
ganar: ",int(cantidad_ganadas/limite*100),"%")
```

```
    print("Cantidad de veces que perdió:", cantidad_perdidas, ",
Frecuencia relativa = ",cantidad_perdidas/limite, ", Porcentaje de
perder: ",int(cantidad_perdidas/limite*100),"%")
```

***simularJuegoSinCambiarDePuerta(veces):*** método para simular tantas veces el juego usando la estrategia de quedarse con la primera puerta elegida. El parámetro veces le indica a la función cuántas veces se tiene que simular el juego.

```
def simularJuegoSinCambiarDePuerta(veces):
    cantidad_ganadas = 0
    cantidad_perdidas = 0
    for i in range(veces):
        if utils.MontyHallNoCambiaPuerta() == True:
            cantidad_ganadas += 1
        else:
            cantidad_perdidas += 1


    print("\nIteracion de", veces)
    print("Cantidad de veces que ganó:", cantidad_ganadas, ",
Frecuencia relativa = ",cantidad_ganadas/veces, ", Porcentaje de ganar:
",int(cantidad_ganadas/veces*100),"%")
    print("Cantidad de veces que perdió:", cantidad_perdidas, ",
Frecuencia relativa = ",cantidad_perdidas/veces, ", Porcentaje de
perder: ",int(cantidad_perdidas/veces*100),"%")
```

Testeo de probabilidades para 1.000, 10.000 y 100.000 con las distintas probabilidades:

```
print("Simulación de Monty Hall cambiando de puerta \n")
simularJuegoCambiandoDePuerta(1000)
simularJuegoCambiandoDePuerta(10000)
simularJuegoCambiandoDePuerta(100000)
print("-----------------------------------------------------------------
---------------------------------")
print("\nSimulación de Monty Hall sin cambiar de puerta \n")
simularJuegoSinCambiarDePuerta(1000)
simularJuegoSinCambiarDePuerta(10000)
simularJuegoSinCambiarDePuerta(100000)
```

**Resultados**:

Cambiando de puerta

```
Simulación de Monty Hall cambiando de puerta


Iteracion  1000
Cantidad de veces que ganó: 658 , Frecuencia relativa =  0.658 , Porcentaje de ganar:  65 %
Cantidad de veces que perdió: 342 , Frecuencia relativa =  0.342 , Porcentaje de perder:  34 %

Iteracion  10000
Cantidad de veces que ganó: 6753 , Frecuencia relativa =  0.6753 , Porcentaje de ganar:  67 %
Cantidad de veces que perdió: 3247 , Frecuencia relativa =  0.3247 , Porcentaje de perder:  32 %

Iteracion  100000
Cantidad de veces que ganó: 66575 , Frecuencia relativa =  0.66575 , Porcentaje de ganar:  66 %
Cantidad de veces que perdió: 33425 , Frecuencia relativa =  0.33425 , Porcentaje de perder:  33 %
```

No cambiando de puerta

```
Simulación de Monty Hall sin cambiar de puerta


Iteracion de 1000
Cantidad de veces que ganó: 344 , Frecuencia relativa =  0.344 , Porcentaje de ganar:  34 %
Cantidad de veces que perdió: 656 , Frecuencia relativa =  0.656 , Porcentaje de perder:  65 %

Iteracion de 10000
Cantidad de veces que ganó: 3359 , Frecuencia relativa =  0.3359 , Porcentaje de ganar:  33 %
Cantidad de veces que perdió: 6641 , Frecuencia relativa =  0.6641 , Porcentaje de perder:  66 %

Iteracion de 100000
Cantidad de veces que ganó: 33431 , Frecuencia relativa =  0.33431 , Porcentaje de ganar:  33 %
Cantidad de veces que perdió: 66569 , Frecuencia relativa =  0.66569 , Porcentaje de perder:  66 %
```

# Conclusiones

1. Se implementó correctamente un programa el cual simula el programa televisivo "Lets make a deal".
2. Existe una funcionalidad que le permite al usuario cambiar de puerta si así lo desea.
3. Se probó cada estrategia 1.000, 10.000 y 100.000 veces para poder realizar un análisis.
4. Al probar cada estrategia dichas cantidades de veces, concluimos que si bien al principio pensábamos que cambiar de puerta no afecta las probabilidades de ganar, a medida que realizamos pruebas nos dimos cuenta que en realidad sí afecta.
   Si cambiamos de puerta, tenemos 2 de cada 3 chances de ganar el auto, es decir ⅔.
   En el caso de no cambiar de puerta, tenemos 1 de cada 3 chances de ganar el auto, ⅓ de posibilidades.
   Los resultados de cada prueba demuestran que es más probable ganar el premio si cambias de puerta.

# Pasos a futuro

Como pasos a futuro nos gustaría implementar una interfaz gráfica en la cual el usuario pueda jugar al famoso programa de Monty Hall.
También nos gustaría darle libertad a los participantes dentro del juego, que puedan elegir diferentes combinaciones de puertas y premios.
Por último, estaría bueno realizar un análisis psicológico a través de encuestas para saber cual es la estrategia que más eligen las personas sin conocer el problema de Monty Hall.

# Bibliografía


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

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

Uncomo. (s.f.). ¿Cómo sacar un porcentaje de una cifra?. Mundo Deportivo. Recuperado de **https://www.mundodeportivo.com/uncomo/educacion/articulo/como-sacar-un-porcentaje-de-una-cifra-3733.html**

Estadística Para Todos. (s.f.). Monty Hall: El problema del presentador de televisión. Recuperado de **https://estadisticaparatodos.es/taller/montyhall/montyhall.html**