TP 1.1 - SIMULACIÓN DE UNA RULETA

Bruno A. Dolce
UTN - FRRo
Zeballos 1341, S2000
brunodolce96@gmail.com

Juan Pablo Castelli
UTN - FRRo
Zeballos 1341, S2000
juancastelli58@gmail.com

April 23, 2020

1 Introducción

Nos referimos a aleatoriedad a la cualidad de un suceso de no poder ser previsto y evitado. Los juegos de azar son excelentes ejemplos para simular eventos de naturaleza meramente aleatoria. Simulando los mismos es posible comprobar si una simulación concreta dada se acerca al carácter mostrado por el juego real, es decir, si podemos considerar la tendencia a la aleatoriedad de los distintos lanzamientos simulados.

Una ruleta francesa posee 37 casillas, con números posibles a salir del 0 al 36, de igual probabilidad. Debido a esto, los valores que puede asumir una jugada de lanzamiento tienen una probabilidad de ocurrencia de $\frac{1}{37}$ (esto es, aproximadamente, una probabilidad de 0,027).

2 Objetivos

El objetivo de éste trabajo es evaluar la calidad de la aleatoriedad de las funciones de generación de números aleatorios en el lenguaje de codificación Python. Teniendo la frecuencia relativa, esperanza, desvío y varianza esperados y calculando los mismos parámetros en cada iteración de la ruleta para luego compararlas visualmente en graficas y concluir si el generador estudiado tiende a la aleatoriedad.

3 Metodología de Trabajo

Se realizaron una cantidad n de llamadas a la función random.randint para simular los número obtenidos en las n tiradas de la ruleta. De estos resultados se fueron calculando cada vez, en conjunto con las tiradas anteriores, la frecuencia relativa del número elegido y, la esperanza, el desvío y la varianza del conjunto total de resultados. Se realizaron, en primer instancia, gráficas para una primer ejecución del programa, así visualizar el comportamiento de un caso en particular y posteriormente gráficas en conjunto para varias iteraciones de éste, y para poder comparar la conducta de nuestra ruleta en varias oportunidades.

3.1 Fórmulas

Frecuencia relativa Se realizó utilizando la siguiente fórmula escrita en código Python, para calcular las ocurrencias del número elegido (frecuencia absoluta) con respecto a la cantidad de tiradas:

$$\frac{x_i}{n} \tag{1}$$

Cálculo de la esperanza.

Se utilizó el método numpy.average que metemáticamente se puede formular de la siguiente manera.

$$\frac{\sum_{i=1}^{n} x_i}{n} \tag{2}$$

Cálculo del desvío estándar. Para obtenerlo se usó el método numpy.std, que expresado con notación metemática la fórmula sería la siguiente

$$\sqrt{\frac{\sum_{i=1}^{n}(x_i - \overline{x})^2}{n-1}} \tag{3}$$

Cálculo de la varianza. En cuanto a la varianza se ejecutó la función numpy.var que escrita en notación matemática es como sigue

$$\frac{\sum_{i=1}^{n}(x_i - \overline{x})^2}{n-1} \tag{4}$$

3.2 Herramientas utilizadas

Lenguaje de programación. Se utilizó el lenguaje de programación **Python** en su versión 3 para el desarrollo de la simulación. Se utilizaron además módulos de la biblioteca estándar de Python, asi como bibliotecas externas para añadir más funcionalidad.

Bibliotecas y módulos.

Para generar los números aleatorios se empleó el módulo nativo de Python **random** y su método:

random.randint que obtiene un número entero pseudo-aleatorio que se encuentra entre el número mínimo y máximo pasados como parámetros. En nuestro caso el 0 y el 36.

Se utilizó la biblioteca ***NumPy v1.18*** para calcular los parámetros estadísticos anteriormente comentados.

numpy.average que emplea los siguiente métodos interiormente,

$$sum(a * weights)/sum(weights) \tag{5}$$

Siendo *a* cada elemento de la lista pasada como parámetro y *weights* el peso asignado a cada elemento de dicha lista, en éste caso, al ser null éste parámetro se considera como 1 para todos los elementos, siendo en definitiva la suma de todos los elementos de la lista sobre la cantidad de elementos de ésta.

numpy.std que internamente aplica las siguientes funciones al arreglo usado como parámetro.

$$sqrt(mean(abs(x - x.mean()) * *2)) \tag{6}$$

Siendo *x* cada elemento de la lista pasada como parámetro y *x.mean()* la media ariemética de la misma lista.

numpy.var que realiza las siguientes operaciones al parámetro.

$$mean(abs(x - x.mean()) * *2) \tag{7}$$

Funcionando *x* y *x.mean()* igual que en numpy.std, ya que aplica los mismo métodos a excepción de la radicación.

Para graficar los arreglos que almacenan los parámetros estadísticos anteriores se utilizó la biblioteca ***Matplotlib 3.1.1***, en particular el método *pyplot* perteneciente a dicha biblioteca.

Parámetros utilizados. Para realizar el experimento se utilizaron los siguientes parámetros y valores:

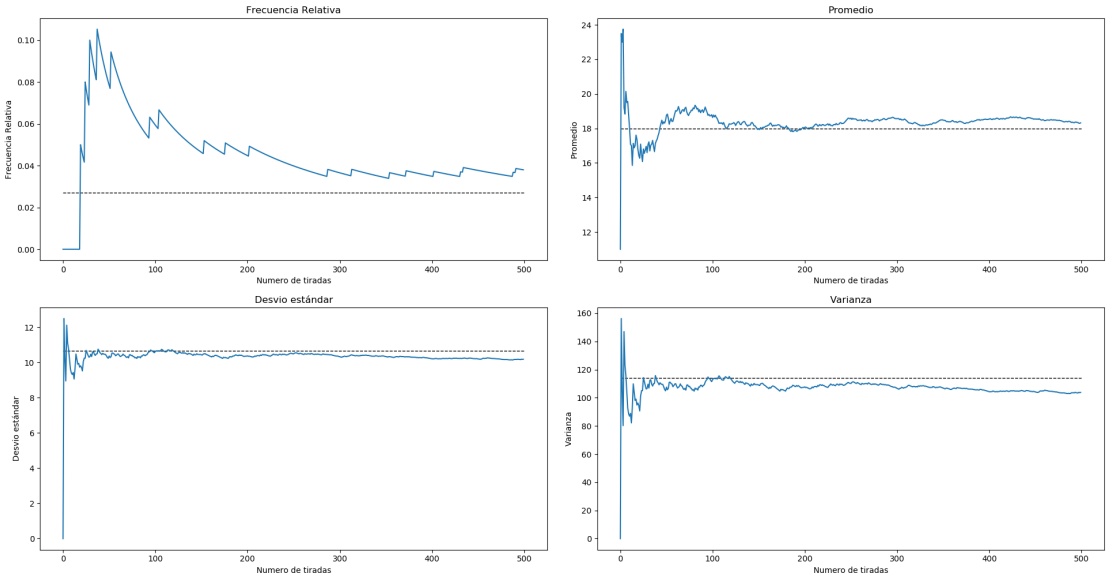
Cantidad de tiros: 500

Cantidad de simulaciones: 1 (para el primer conjunto de gráficas) y 5 (para el segundo conjunto de gráficas)

4 Resultados Obtenidos

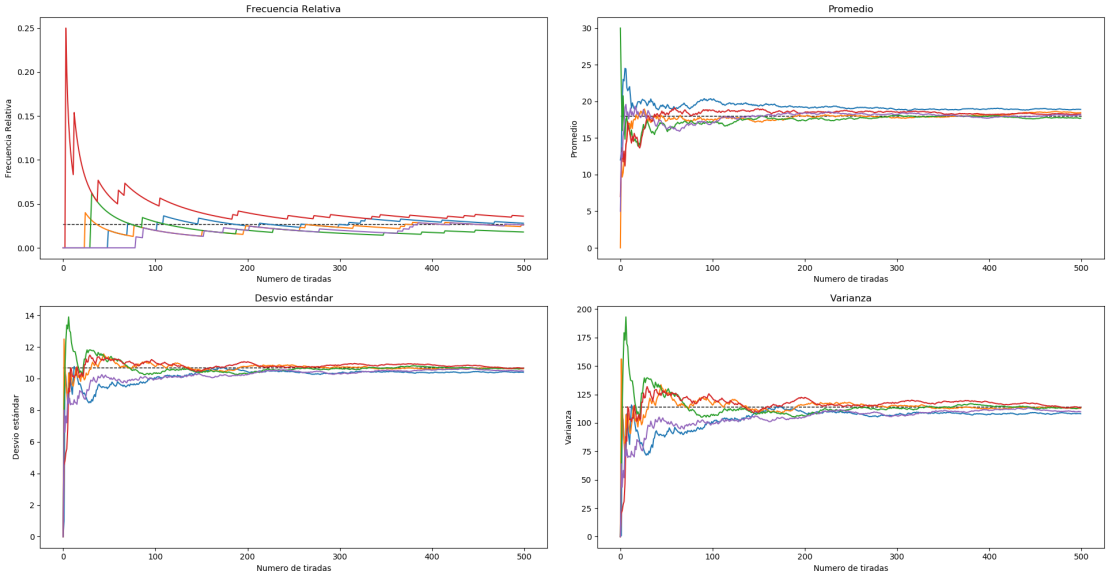
4.1 Resultados para una corrida

Se registran las gráficas obtenidas para una sola corrida de la simulación.



4.2 Resultados para varias corridas

Se registran las gráficas obtenidas para cinco corridas de la simulación, superponiendo en cada gráfica los resultados obtenidos en cada iteración.



5 Conclusiones

Luego de haber realizado una cantidad estadísticamente significativa de repeticiones del experimento, se puede proceder a analizar las gráficas que conjuntan las n iteraciones. Se observa que en las primeras "tiradas" es cuando las gráficas difieran drásticamente una de la otra, debido a la poca información (y disparidad en ella) que se recopila hasta entonces, es luego de, aproximadamente, 200 ejecuciones de la ruleta dónde podemos advertir que todos los parámetros estadísticos estudiados comienzan a tender al valor esperado. Se concluye de ésta forma que el generador de números

aleatorios utilizado para nuestro ensayo efectivamente tiende a la aleatoriedad y a la distribución que emplea, uniforme en éste caso.

References

[1] NumPy v1.18 Manual, numpy.average In *https://numpy.org/doc/stable/reference/generated/numpy.average.html#numpy.average.*

[2] NumPy v1.18 Manual numpy.std In *https://numpy.org/doc/stable/reference/generated/numpy.std.html#numpy.std*

[3] NumPy v1.18 Manual numpy.var In *https://numpy.org/doc/stable/reference/generated/numpy.var.html#numpy.var*.

[4] Matplotlib.pyplot 3.1.1 version Manual In *https://matplotlib.org/3.1.1/api/pyplot*$_{s}ummary.html$.

[5] Python Manual v3.8.2 random.randint In *https://docs.python.org/3/library/random.html*.