TP1 - SIMULACIÓN DE UNA RULETA

Antonelli, Nicolás
Departamento de Ingeniería en Sistemas
Universidad Tecnológica Nacional - FR Rosario
Rosario, Zeballos 1341
niconelli2@gmail.com

Coauthor's Name
Departamento de Ingeniería en Sistemas
Universidad Tecnológica Nacional - FR Rosario
Rosario, Zeballos 1341
coauthor@example.com

5 de abril de 2020

ABSTRACT

La Simulación. Según R. E. Shannon: "La simulación es el proceso de diseñar un modelo de un sistema real y llevar a término experiencias con él, con la finalidad de comprender el comportamiento del sistema o evaluar nuevas estrategias -dentro de los límites impuestos por un cierto criterio o un conjunto de ellos - para el funcionamiento del sistema"[1].

En este proyecto se ha realizado un análisis numérico y gráfico de los resultados obtenidos al simular una ruleta en Python, desde un punto de vista estadístico y haciendo uso de números pseudoaleatorios.

1. Introducción

Este proyecto es el primer trabajo práctico de la cátedra "Simulación"(UTN FRRO, Departamento ISI). La idea fue que en esta simulación, donde lo que se modeliza es el funcionamiento de una ruleta 'europea'[2], aparte de trabajar con números pseudoaleatorios para representar los números obtenidos al 'girarla' utilizamos listas (arrays) para hacer todo tipo de cálculos estadísticos tales como: la frecuencia relativa de un número en unas N tiradas de ruleta con respecto al total de números, el promedio (media estadística), varianza y desvío de una población conformada por los N números obtenidos, y como estos valores se comportan si vamos creciendo lentamente el N desde 0 hasta un valor dado en un input, como por ejemplo 'N = 500'. Todos estos valores fueron graficados para estudiarlos mejor. A su vez, se utilizaron matríces para poder calcular más de un juego de resultados de Ruleta a la vez, y graficar en simultáneo todo lo obtenido.

2. Marco Teórico

2.1. Definiciones

El valor de la frecuencia de aparición de un número X obtenido al azar en N tiradas de ruleta, es una *Variable Aleatoria Discreta*, por la siguiente definición[3]:

> "Una *variable aleatoria* es una función que asigna a cada elemento del espacio muestral un número real. Una variable aleatoria es *discreta* si su recorrido es un conjunto finito o infinito numerable (susceptible de ser contado)". Lo anterior sucede en este caso con un rango de (0, N), donde N es la cantidad de tiradas, osea la cantidad de veces que hacemos girar la ruleta (N entero positivo).

Y también se tiene definida una población de tamaño N (a la que se calculan parámetros estadísticos) para cada valor entero entre 0 y N: Son los números aleatorios que representan los números obtenidos en N tiradas de la ruleta.

Ejemplo de una población de números aleatorios de tamaño N=10: {24, 1, 10, 0, 0, 6, 12, 36, 21, 35}

2.2. Fórmulas Empleadas

Cálculo de la Frecuencia Relativa de un Número X con respecto a los N números de la tirada. Dónde:

- n es la frecuencia absoluta de un número, es decir, su cantidad de apariciones
- N es el tamaño de la población, es decir los N números obtenidos en la tirada.
- x es el número elejido a ser estudiado, cualquiera entre 0 y 36.

$$f_x = \frac{n_x}{\sum_{i=0}^{N} n_i} = \frac{n_x}{N} \tag{1}$$

Cálculo de la Esperanza Matemática (en este caso es lo mísmo referirse a ella como Media o Promedio) de la Población. Dónde:

- X es un número de la población
- $\Re$ es el recorrido de la población (0, 36)
- $P_x(X)$ es la probabilidad de ocurrencia de X

$$E(X) = \mu_X = \sum_{x \in \Re_x} X P_x(X) \tag{2}$$

Si en vez se procede con una población de Frecuencias de aparición de los números de la ruleta:
Como $P_x(X)$ es lo mismo que hablar de f_x, entonces aplicando la fórmula (1) se puede sacar N (tamaño de la población) como factor común. De esta forma, solo multiplico dentro de la sumatoria, y divido una sola vez en el final, ahorrando tiempo de procesamiento. La fórmula entonces queda como la siguiente:

$$E(X) = \mu_X = \frac{\sum_{x \in \Re_x} X n_x}{N} \tag{3}$$

Cálculo de la Varianza de la Población. Dónde:

- X es un número de la población
- $\Re$ es el recorrido de la población (0, 36)
- $P_x(X)$ es la probabilidad de ocurrencia de X
- μ_X es la media de la población, calculada en (3)

$$V(X) = \sigma_X^2 = \sum_{x \in \Re_x} (X - \mu_X)^2 P_x(X) \tag{4}$$

Si en vez se procede una población de Frecuencias de aparición de los números de la ruleta:
Siguiendo el mismo razonamiento utilizado para a partir de la fórumla (2) sacar la fórumla (3), acá también se puede utilizar la fórmula (1) y sacar factor común N y ahorrar tiempo innecesario calculando divisiones demás en el código. La fórmula entonces queda como la siguiente:

$$V(X) = \sigma_X^2 = \frac{\sum_{x \in \Re_x} (X - \mu_X)^2 X n_x}{N} \tag{5}$$

Cálculo del Desvío de la Población. Dónde:

- σ_X es la varianza de la población calculada en (5)

$$D(X) = \sigma_X = \sqrt{\sigma_X^2} \tag{6}$$

3. Estrucura de la Simulación

Para componer este trabajo práctico, se ha utilizado íntegramente el lenguaje **Python**[4], escrito el código en **Visual Studio Code**[5] con su correspondiente *Plugin*. Para llegar al resultado final, se han probado y descartado varias librerías y módulos en el camino, que fueron reemplazadas por otras; tales como: la librería **statistics** que tenía funciones estadísticas fáciles de utilizar, pero ciertos límites en la cantidad de decimales a manejar con *floats*, la librería **random** para números *pseudoaleatorios*, pero se utilizó una mejor, entre otras. A continuación, se presenta lo que sí ha llegado hasta la *versión final*:

3.1. Metodología: Librerías y Módulos Utilizados

Numpy [6] Esta librería tiene de todo. Primero, la utilización de *Arrays y Matrices* en vez de *Listas*, pues estas últimas son objetos, y con los arrays de Numpy se consigue una mayor eficiencia (es recomendable utilizar arrays y matrices de tamaño fijo, pues como no se almacena de forma contigua espacio extra con Numpy, tamaños variables destruyen esta eficiencia mayor pues aumentar el tamaño sería volver a construirlo).

Numpy Tiene un módulo llamado random (Numpy.Random) que se utilizó para conseguir los números pseudoaleatorios de una forma rápida, y si se necesita, se puede generar un Array de N longitud con todos valores aleatorios. Numpy también posee fórmulas estadísticas muy potentes para calcular la *media* (promedio), *varianzas* y *desvíos*.

Por último, también tiene un método llamado *arange* que es como el *for* de siempre, pero con algunos beneficios como que su *Step* puede ser un float, o que se puede generar un array con él.

Pyplot [7] Este módulo de la librería **matplotlib** es ideal para graficar toda la información calculada de una forma simple: unas pocas configuraciones y un array o lista a graficar. Su sintaxis es similar a usar **Matlab**. Los gráficos que genera son muy personalizables y rápidos de generarse. También posee la posibilidad de hacer *Subplots*, es decir más de una gráfica en simultáneo, y plottearlas a todas juntas en una misma ventana.

Time [8] Este módulo de Python fue utilizado para calcular el tiempo de procesamiento intermedio y el tiempo total que tarda el código en ejecutarse correctamente, según el tamaño de tiradas (N) aumenta.

3.2. Método de Resolución Aplicado

Para hacerlo breve, ya que el código de la simulación tiene comentarios paso-a-paso explicando como funciona, la idea es la siguiente:

Primero se pide un número N de tiradas de ruleta, y un número X a estudiar su frecuencia.

Segundo se calculan los valores esperados (Frecuencia Relativa de X, y valores de la población: media, variación y desvío.

Tercero se generan cuatro matríces, una para cada parámetro a estudiar, de tamaño (6xN), que se puede pensar como 6 arrays de tamaño N, uno para cada *Conjunto de Resultados*. **¿Porqué calcular más de un Array de resultados?** Pues porque primero se pide graficar uno solo, pero luego se pide graficar en simultáneo entre 5 y 10 resultados.

Cuarto se llena para cada uno de los 4 parámetro a estudiar, cada uno de los 6 array de resultados, cargando en un for o un arange anidado a otro for o arange: los valores que te da el aplicar la fórmula del parámetro correspondiente a una población de números aleatorios no-fija, ya que arranca en 0 (1era posición del array) y en cada loop va calculando una nueva población con 1 tirada más que la anterior, hasta la última iteración donde calcula los parámetros con respecto a una población N (última población del array).

Quinto se toman las posiciones [0] de cada matriz, osea el conjunto de resultados 0: Un array lleno de frecuencias relativas, uno lleno de promedios (medias), otro de varianzas y otro de desvíos. Se plottean los 4 gráficos (stack de 4 subplots) de ese conjunto de resultados en una misma ventana. Luego, se grafican los 6 conjuntos de resultados con un color cada uno, de la misma forma que lo anterior, solo que todos en simultáneo.

4. Gráficas Obtenidas

4.1. Etiquetas de las Gráficas

- **FRE**: Frecuencia Relativa Esperada
- **FRN**: Frecuencia Relativa de un número X con respecto a las N Tiradas
- **VPE**: Valor Promedio Esperado
- **VPN**: Valor Promedio con respecto a las N Tiradas
- **VVE**: Valor de la Varianza Esperado
- **VVN**: Valor de la Varianza con respecto a las N Tiradas
- **VDE**: Valor del Desvío Esperado
- **VDN**: Valor del Desvío con respecto a las N Tiradas

4.2. Población Pequeña

Gráficas obtenidas con el estudio de una población pequeña. Rango de N = (0, 20). Primero gráfica de un solo conjunto resolución, luego de los 6 en simultáneo.

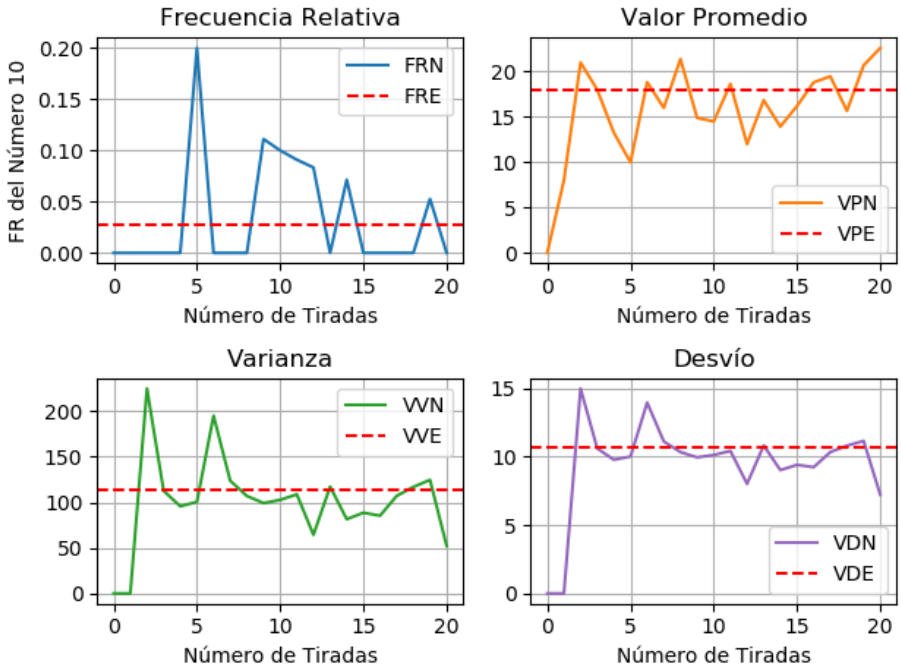

Figura 1: Un conjunto de Resolución. De 0 a 20 Tiradas.

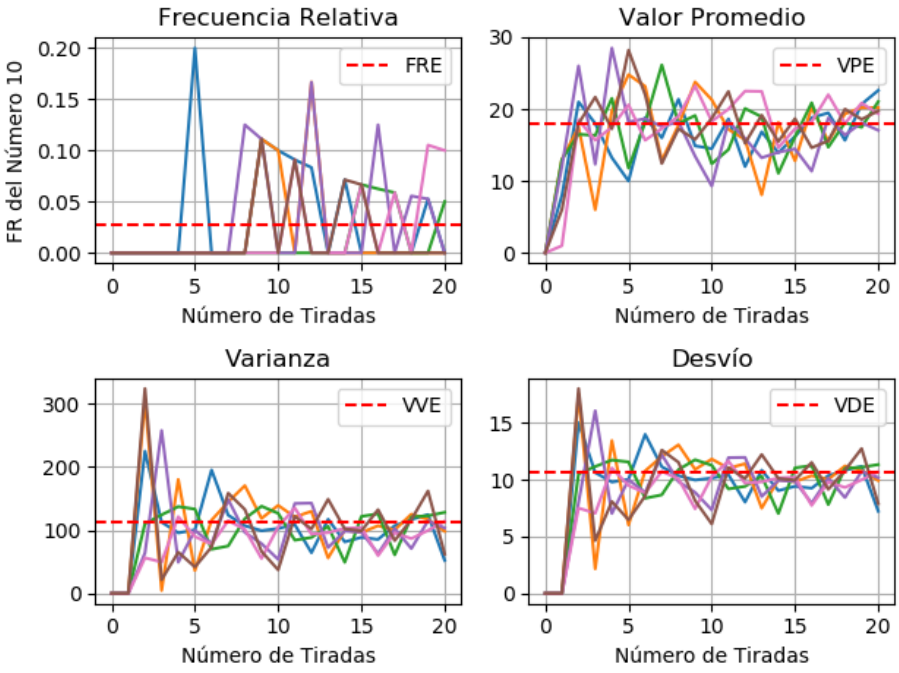

Figura 2: Los 6 conjuntos de Resolución en simultáneo. De 0 a 20 Tiradas.

4.3. Población Mediana

Gráficas obtenidas con el estudio de una población mediana. Rango de N = (0, 120). Primero gráfica de un solo conjunto resolución, luego de los 6 en simultáneo.

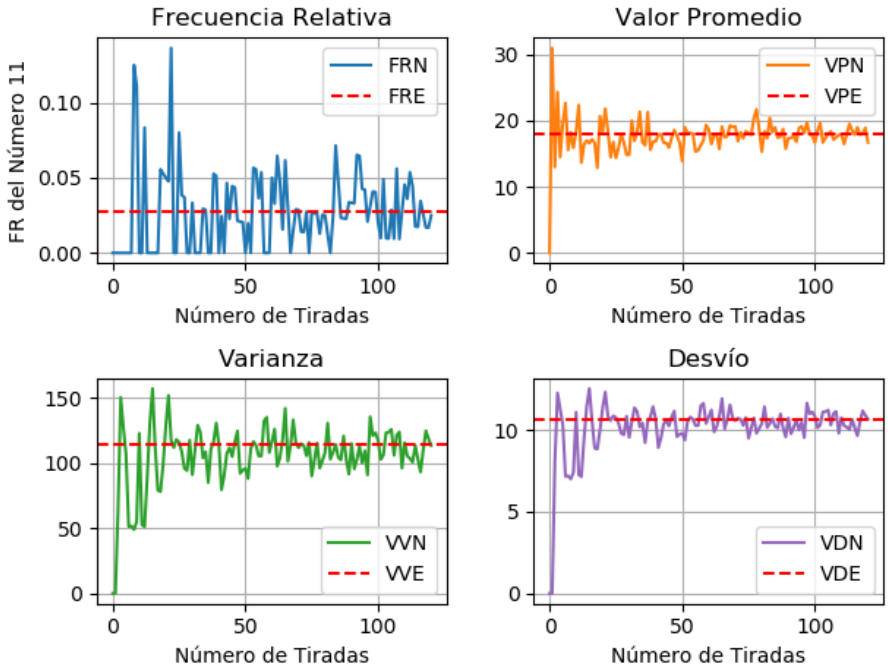

Figura 3: Un conjunto de Resolución. De 0 a 120 Tiradas.

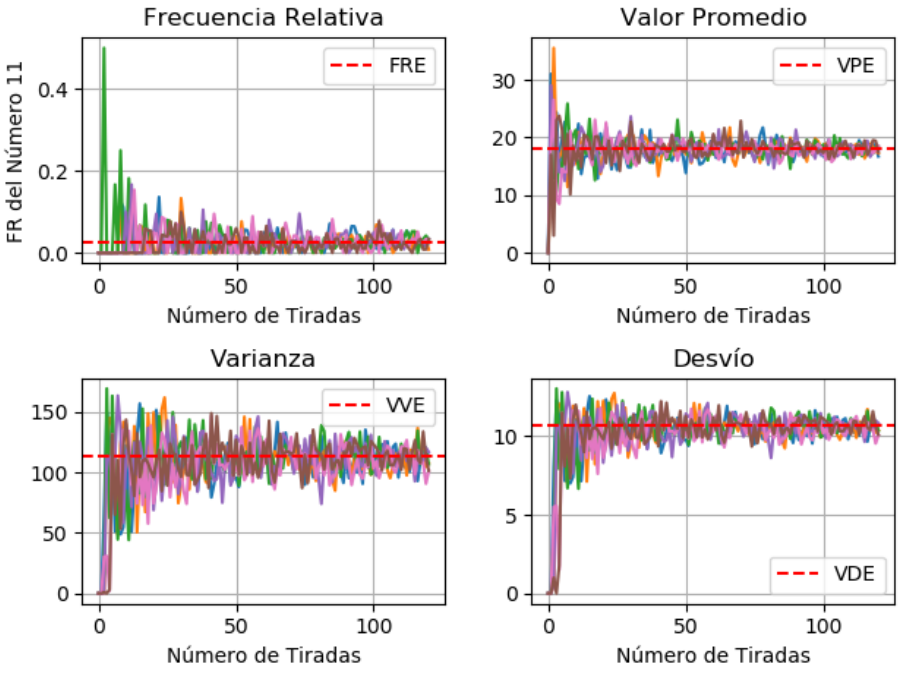

Figura 4: Los 6 conjuntos de Resolución en simultáneo. De 0 a 120 Tiradas.

4.4. Población Grande

Gráficas obtenidas con el estudio de una población grande. Rango de N = (0, 1000). Primero gráfica de un solo conjunto resolución, luego de los 6 en simultáneo.

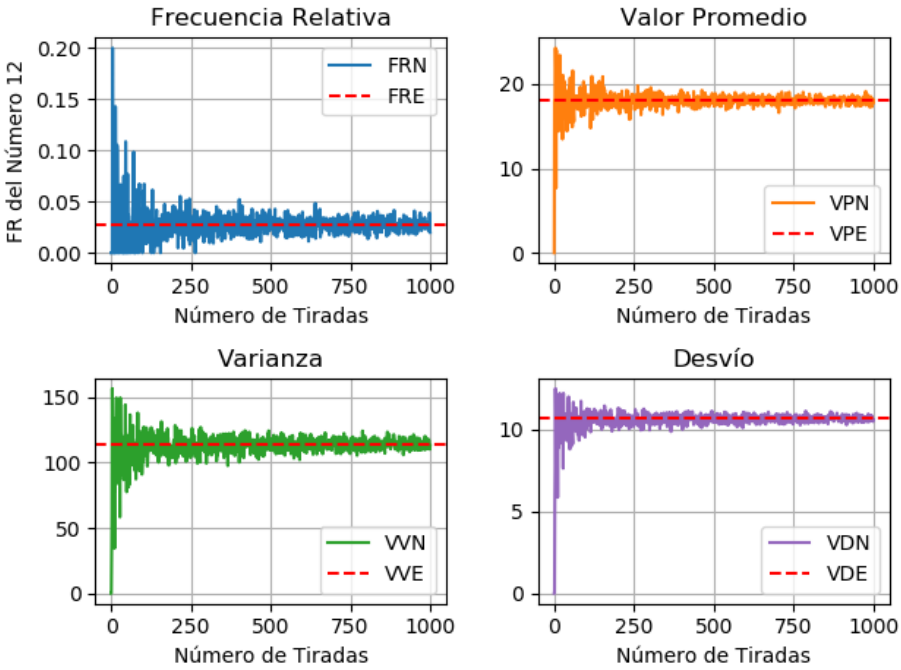

Figura 5: Un conjunto de Resolución. De 0 a 1000 Tiradas.

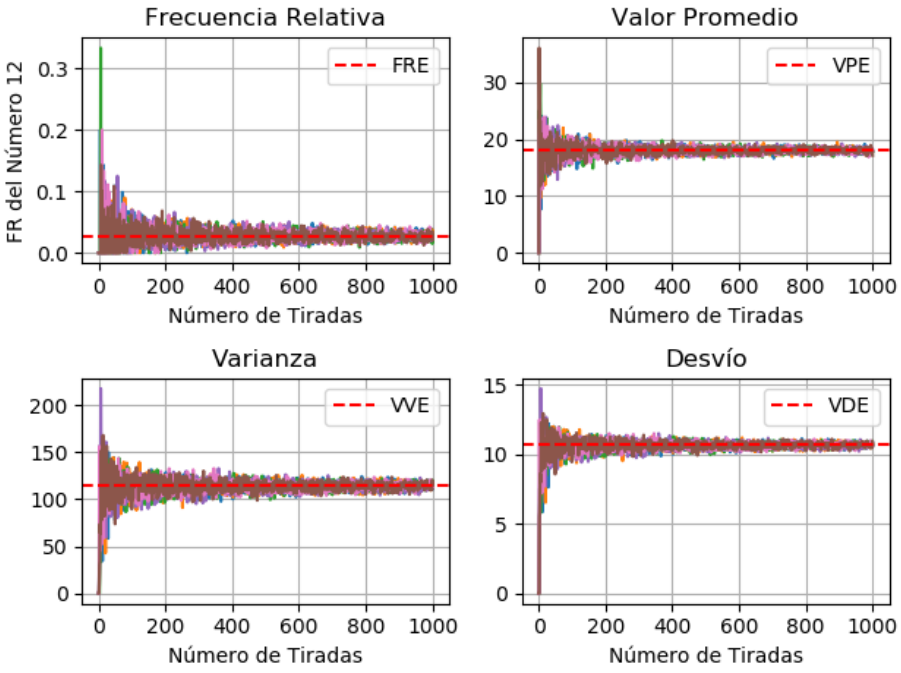

Figura 6: Los 6 conjuntos de Resolución en simultáneo. De 0 a 1000 Tiradas.

5. Conclusiones

Luego de analizar el comportamiento de los conjuntos de resultados en las gráficas, se aprecia que en un N pequeño los valores obtenidos están bastante alejados de los valores esperados. Sin embargo, aumentando N, estos valores van çonvergiendo"lentamente a un rango de valores muy próximo a los valroes esperados. La forma gráfica del acotamiento de estos valores conforme aumenta N recuerda a el movimiento armónico amortiguado.

Podemos concluir entonces que si N tiende a ∞, todos los valores obtenidos serán aproximadamente iguales a los valores esperados.

Otra observación interesante es que si en las gráficas donde aparecen en simultáneo varios conjuntos de resultados tomamos los valores de cada una y (sin realizar cálculos, a simple vista) podemos ver que los promedios de los mismos, aún a pesar de lo errático q son en poblaciones pequeñas, son bastante más próximos a los valores esperados.

No es menor resaltar que también el tiempo de procesamiento juega un papel importante, y la optimización del código es vital para que funcionamiento del mismo sea lo más rápido posible, ya que la complejidad del mismo es aproximadamente O(N!), es decir, N factorial. A continuación mencionaré valores de tiempo aproximados. Se ha probado en una PC (Procesador: AMD Ryzen 5 de 1era Generación) que con un N=1.000 se tarda 0.5s en calcular y graficar 6 conjuntos de resultados, con N=10.000 se tarda 10.0s en la misma tarea, y ya con un N=100.000 se tardan 620s (mas de 10 minutos!). Sin embargo, cuando se comenzó con el código, en los primeros intentos tardaba lo que tarda N=100.000 en calcular y graficar solo con N=10.000, por lo que la mejora es notable.

Podría mejorarse aún más si se redujera la complejidad de O(N!) a O(N) haciendo un estudio con un enfoque muestral en vez de poblacional: tener una sola población de tamaño N y calculando con muestras cuyo tamaño irían en el rango (0, N) de esa misma población (en la última iteración sería la población completa), en vez de generar una población nueva de números aleatorios por cada entero entre 0 y N.

Referencias

[1] Wikipedia ES. Definición de Simulación.
https://es.wikipedia.org/wiki/Simulación

[2] Wikipedia ES. Definición de Ruleta.
https://es.wikipedia.org/wiki/Ruleta

[3] Raúl Katz - Pablo Sabatinelli (2018). Probabilidad y Estadística: Variables Aleatorias Discretas y algunas Distribuciones de Probabilidad

[4] Python Doc. Python 3.8.2 | Documentación Oficial.
https://docs.python.org/3/

[5] Microsoft (VSC). Visual Studio Code Official Webpage
https://code.visualstudio.com/

[6] Numpy Doc. Numpy 1.18 | Documentación Oficial.
https://numpy.org/doc/1.18/

[7] Pyplot Doc. Pyplot 3.1.1 | Documentación Oficial.
https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.html

[8] Time Doc. Time Module (Python) | Documentación Oficial.
https://docs.python.org/3/library/time.html