Generadores de números aleatorios con Python

J. M. Puddu[1]

[1] *Facultad de Matemática, Astronomía, Física y Computación, UNC, Argentina*

Received: ... / Accepted: ...

Resumen / Con el fin de estudiar las distintas distribuciones de probabilidad de manera computacional es necesario en principio poder replicar la aleatoriedad que se haya en la naturaleza, esto presenta un desafío ya que implica la generación matemática de números o variables aleatorias. En este trabajo se estudian los distintos generadores de números aleatorios y como poder trabajar con estos para replicar distintos experimentos con ellos.

Keywords / https://github.com/shulypuddu/astrometria/tree/main

1. Introducción

Para poder comprender el análisis y funcionamiento de los generadores de números aleatorios son cruciales entender los siguientes conceptos:

- **Semilla:** valor inicial que se ingresa al generador, a partir de este comienza la sucesión de números no necesariamente forma parte de dicha sucesión.
- **Correlación :** Es una forma de trabajar los valores de la lista entre sí, simplemente se un valor de la sucesión con el siguiente de la misma. Si en vez de ser con su consecutivo se saltea 1 o mas valores de por medio se dice que se realiza un **retardo de i saltos**.
- **k-Momento:** Se define matemáticamente como:

$$E(X^k) = \sum_{i=1}^{N} x_i^k p(x_i) \quad \text{con } k = 1, 2, 3, 4, \ldots$$

Una caracterización "completa" de una dada distribución se logra cuando se estudian todos los momentos hasta orden n de una distribución. Para una distribución uniforme se cumple que

$$E(X^k) = \frac{1}{k+1}$$

Para este trabajo se fueron realizando los distintos generadores y sus aplicaciones en códigos de python, los cuales se pueden encontrar dentro de la carpeta 'Práctico 1' en el repositorio de GitHub que se referencia en este trabajo. Dicha carpeta tiene un archivo con las funciones caseras y otro con las aplicaciones, además de las distintas imágenes utilizadas.

2. Generador lineal congruencial

Este generador es capaz de generar una N cantidad de números en principio aleatorios usando el álgebra de congruencia. A partir de una semilla el generador computa la siguiente ecuación:

$$x_{i+1} = (c + a * x_i) \quad mod(m) \tag{1}$$

Donde c se denomina incremento, a multiplicador y m es el módulo al cual es congruente la ecuación. Esta ecuación nos devuelve números aleatorios entre 0 y $m-1$ por tanto para obtener un generador normalizado debemos dividir cada valor por m.

Este tipo de operaciones termina generando una sucesión de N valores que en principio no tienen una correlación entre ellos, pero en la práctica queda claro que esto es muy dependiente de los valores que se elijan para a, c y m. Si probamos con los siguientes valores ($a = 1$,$c = 57$,$m = 256$,$x_0 = 10$), y generamos distintas cantidades podemos probar que mientras la cantidad de números generados se acerca al valor del módulo la "aleatoriedad" de los números se va perdiendo y al llegar a m números queda claro en la correlación que hay un patrón y que por tanto se pierde la aleatoriedad.

Algo destacable de este generador es que mientras no tomes m como la cantidad de números deseados este generador no produce números repetidos, esto resulta muy útil porque con los números correctos la aleatoriedad de la imagen **??** se puede repetir con volúmenes de números mucho mas grandes. Es entonces que podemos buscar una lista de números 'buenos' con los cuales los patrones son mucho menos notorios y el conjunto formado se puede llamar aleatorio. Según Press & Teukolsky (2007) un set de parámetros útiles son $a = 1664525$,$c = 1013904223$,$m = 2^{32}$, al tomar un módulo tan grande nos asegura que no podremos reconocer un patrón en los valores arrojados por mas que sean, es así tal como se ve en la figura 2.

Una forma de comparar los valores obtenidos es utilizando los k-momentos y comparándolos con los resultados teóricos de los mismos como se ve en la tabla **??**.

* El prefijo NR hace referencia a los valores generados con la funcion de generador lineal congruencial con los parámetros obtenidos de Press & Teukolsky (2007).

3. Generador de Fibonacci con Retardo

Este generador parte no de un solo número como semilla sino de una lista inicial de n valores x_n la cual se ex-

✉ *Contact* / julieta.puddu@mi.unc.edu.ar

Cantidad	k=1	k=3	k=7	NRk=1*	NRk=3*	NRk=7*
10	0.401 17	0.187 67	0.991 74	0.348 47	0.132 68	0.124 784
100	0.484 45	0.243 114	0.124 056	0.441 04	0.223 25	0.126 875
1000	0.498 26	0.248 49	0.123 50	0.485 54	0.244 00	0.034 876

tiende usando la siguiente ecuación:

$$x_i = (x_{i-j} + x_{i-k}) \quad mod(m) \tag{2}$$

Donde necesariamente $j < k$ y la lista inicial tiene al menos k elementos y la operación en cada iteración no necesariamente es de suma sino que se puede hacer a partir de cualquier operación binaria.

Este tipo de generadores requiere de otro generador inicial para funcionar, en la función de python se uso el generador lineal congruencial, en la forma que fue definido resultó ser demandante computacionalmente para valores de números no tan grandes. A modo de prueba de su funcionamiento se generaron 10.000 números y se calcularon su valor medio y su varianza obteniendo que:

$$\bar{x} = 0.50084 \qquad \Delta x = 0.08387 \tag{3}$$

Lo cual nos quiere decir que los valores generados se encuentran distribuidos uniformemente entre $[0, 1)$, esto también se puede ver en el gráfico 3

4. Coeficiente de correlación lineal de Pearson

Este coeficiente asume, como lo indica su nombre, que existe una correlación lineal entre los datos. Esta dada por la siguiente ecuación, y por definición mientras menor el número menor la correlación de los valores en la sucesión estudiada:

$$r \equiv \frac{\sum_i (x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_i (x_i - \overline{x})^2}\sqrt{\sum_i (y_i - \overline{y})^2}} \tag{4}$$

Table 1. Coeficientes de correlación de Pearson para diferentes retardos y generadores

Retardo	GLC	Fibonacci
1	-0.101	0.175
2	-0.106	-0.063
3	0.034	-0.106
5	0.027	0.129
7	0.090	0.111
10	-0.008	-0.156

5. Aplicaciones del generador lineal de funciones

La generación de números aleatorios tiene casi como único fin replicar la aleatoriedad en los resultados de los experimentos que se realizan en el mundo real, por tanto es importante poder entender como aplicar nuestros generadores a estos "experimentos virtuales".

5.1. Caminos Aleatorios

Usando el generador lineal congruencial un camino aleatorio donde cada paso se genera aleatoriamente dentro de un intervalo de $[-\sqrt{2}, \sqrt{2}]$, esto se puede iterar las veces deseadas, dando un camino de largo y dirección variable

Como se puede apreciar incluso con la generalidad en ambos casos la distancia al centro crece, esto se debe a que a medida que se iteran distintos pasos la probabilidad de elegir el camino que lleve de nuevo al origen disminuye notablemente.

5.2. Paradoja de Monty Hall

Surge del programa de televisión *Let's Make a Deal* presentado por Monty Hall. En el programa se plantea el siguiente escenario: *Hay 3 puertas: detrás de una hay un auto (premio) y detrás de las otras dos hay cabras. El concursante elige una puerta, luego Monty (que sabe dónde está el auto) abre una de las otras dos puertas que tiene una cabra y le ofrece al concursante la opción de mantenerse con su puerta original o cambiar a la otra puerta cerrada.*

La paradoja surge al preguntarse si conviene o no cambiar la puerta ya elegida. La intuición nos lleva a creer que tenemos un 50% de acertar cambiando la puerta ya que ahora solo hay 2 puertas y detrás de una esta el auto, pero en ese caso no se tiene en cuenta la situación inicial.

En un principio hay 3 puertas, por tanto no importa cual elijas hay una probabilidad de 1/3 de que detrás este el auto. Por tanto una vez que Monty revela una de las puertas donde hay una cabra le quedan 2 escenarios al participante:

- **Si eligió inicialmente el auto (Prob**= 1/3**):** Monty revela 1 cabra y por tanto todavia hay otra, por lo que si cambia de puerta pierde.
- **Si eligió inicialmente una cabra (Prob**= 2/3**):** Monty reveló la otra cabra obligatoriamente y por tanto si cambia la puerta gana.

Por tanto hay mas probabilidades de ganar si se elige cambiar la puerta. El error de la paradoja se encuentra en pensar que al abrir una puerta las otras dos se vuelven equiprobables de tener el auto y además no se tiene en cuenta que la puerta que abre Monty necesariamente tiene un cabra y que esto es información importante al análisis de probabilidades.

5.3. Clasificación de galaxias

De las observaciones se ha llegado a la conclusión de que las galaxias se clasifican según su morfología en *elípticas, espirales, enanas* y *irregulares*. Para una dada galaxia

se ha contabilizado la probabilidad de que entren en cada una de las morfologías existentes: 40% elípticas, 30% espirales, 20% enanas y 10% irregulares.

A partir de esta información se puede generar una cantidad N de galaxias que respeten estas probabilidades, obteniendo la siguiente distribución:

5.4. Jugar a los dados

Un dado cuyo lados tienen el peso distribuido equitativamente da una distribución de probabilidad constante para todas sus caras.

Ahora si se tiene 2 dados tanto el espacio muestral como la variable aleatoria cambian, ahora X

Ahora para repetir el experimento con python se definen 2 funciones: "dados" y "dado doble", ambas funcionan con las mismas funciones, solo que una trabaja con la variable aleatoria de nuestro experimento (ambos dados) y para la otra se generan las tiradas de los dados por separado y luego se suman entre sí. Para cada caso se observan los siguientes resultados 7 y 8 respectivamente.

Se nota claramente que no se obtiene el mismo resultado, esto se explica ya que las funciones no se estan ajustando a la misma distribución de probabilidad, ya que al sumar esta cambia.

References

Press W., Teukolsky S., 2007, *Numerical Recipes with Source Code CD-ROM 3rd Edition: The Art of Scientific Computing*, Cambridge University Press

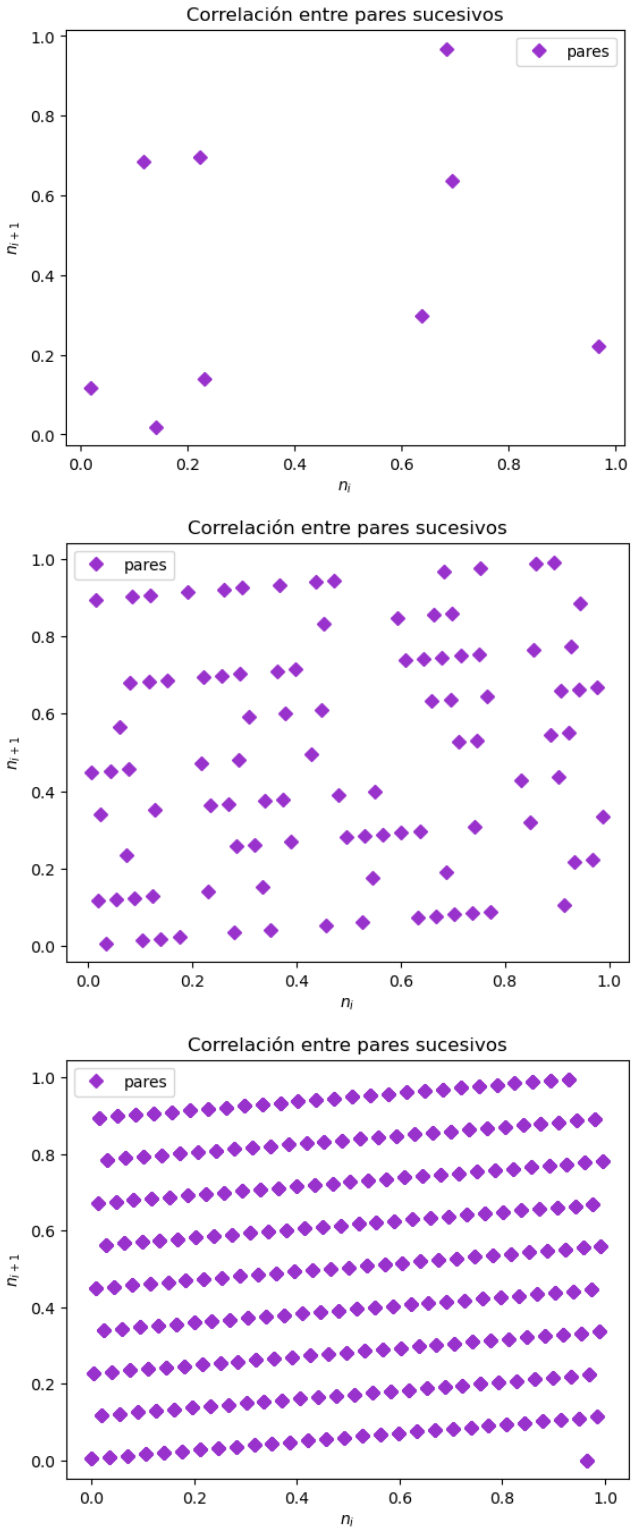

Fig. 1. Generador lineal congruencial con parámetros de prueba, graficando 10, 100 y 1000 valores.

Suma X	**Resultados posibles**	**Nº de resultados**	**Probabilidad** $P(X = x)$
2	(1,1)	1	1/36
3	(1,2), (2,1)	2	1/18
4	(1,3), (2,2), (3,1)	3	1/12
5	(1,4), (2,3), (3,2), (4,1)	4	1/9
6	(1,5), (2,4), (3,3), (4,2), (5,1)	5	5/36
7	(1,6), (2,5), (3,4), (4,3), (5,2), (6,1)	6	1/6
8	(2,6), (3,5), (4,4), (5,3), (6,2)	5	5/36
9	(3,6), (4,5), (5,4), (6,3)	4	1/9
10	(4,6), (5,5), (6,4)	3	1/12
11	(5,6), (6,5)	2	1/18
12	(6,6)	1	1/36

Table 2. Distribución de probabilidad de la suma de dos dados justos

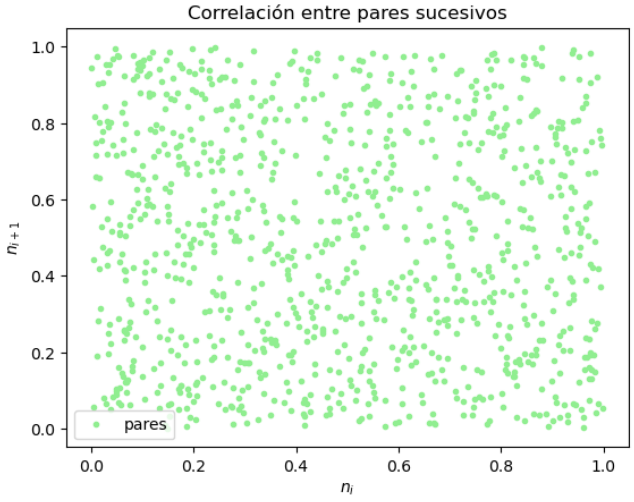

Fig. 2. 1000 números generados con GLC con parámetros recomendados por Press & Teukolsky (2007)

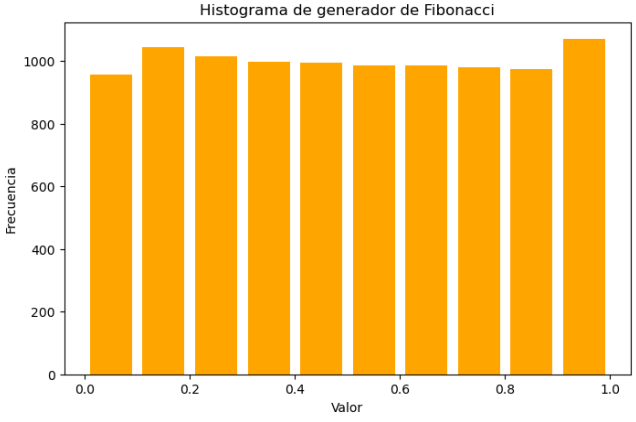

Fig. 3. Frecuencia de los valores generados

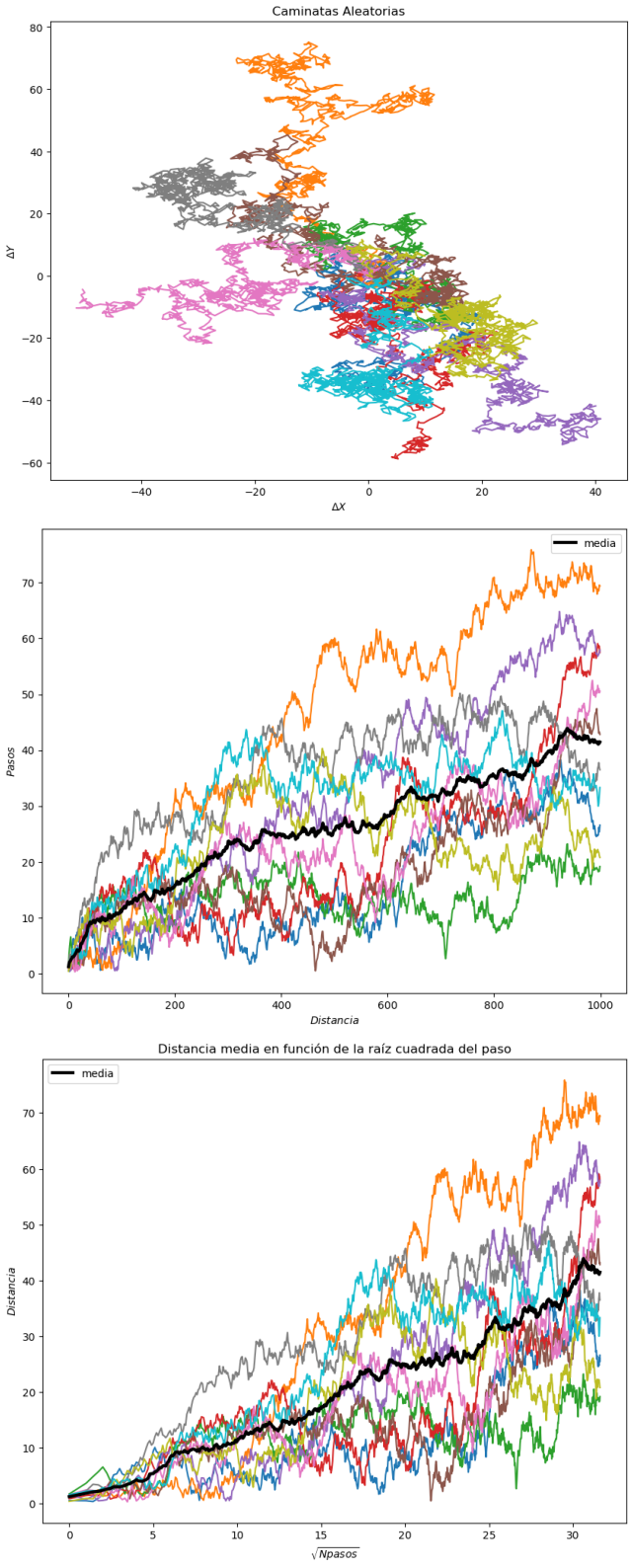

Fig. 4. Se generarón 10 caminos aleatorios de 1000 pasos cada uno.

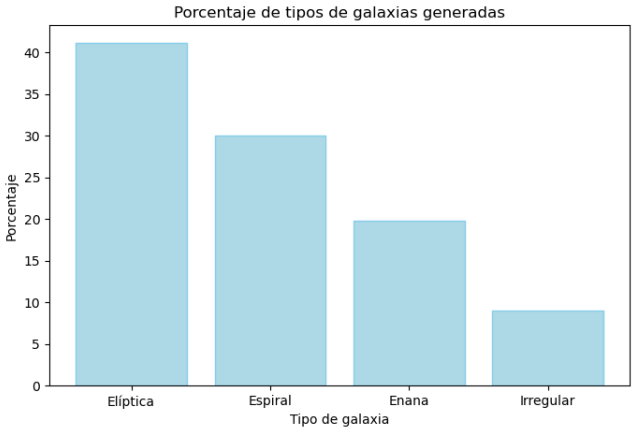

Fig. 5. Frecuencia de las galaxias según su morfología.

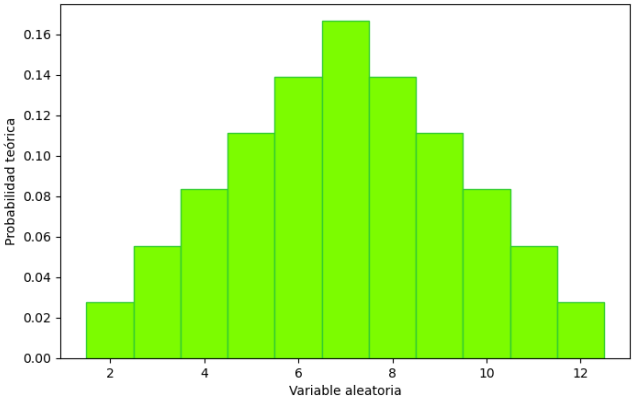

Fig. 6. Función distribución de probabilidad para la suma de dos dados.

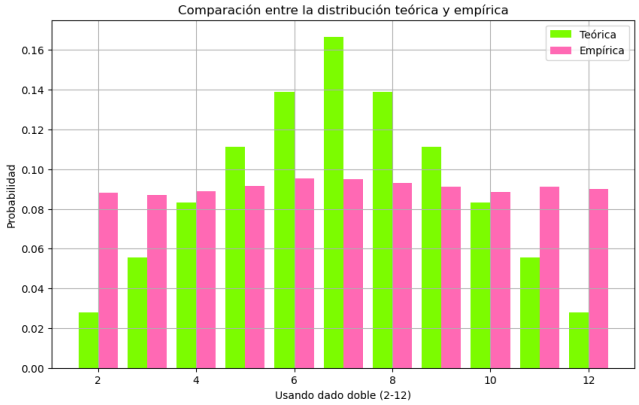

Fig. 7. Comparación de la distribución teórica con la distribución obtenida al tirar un dado doble

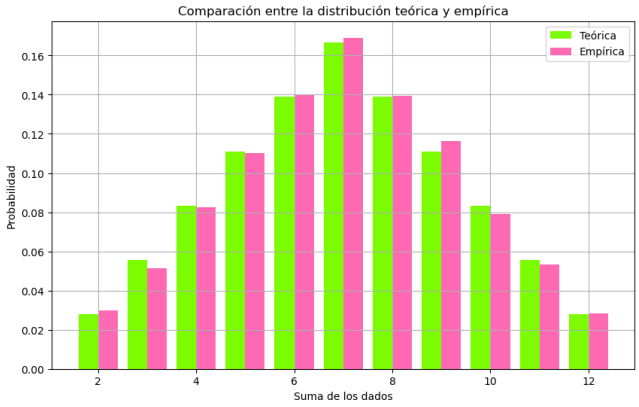

Fig. 8. Comparación de la distribución teórica con la distribución obtenida al sumar las tiradas de dos dados