GENERADORES PSEUDOALEATORIOS - SIMULACION

Fadua Dora Sicardi **Franco Giangiordano** **Gonzalo Turconi** **Ignacio Curti**

14 de mayo de 2022

ABSTRACT

El presente trabajo de investigación incluye el estudio sobre generadores de números pseudoaleatorios. Indagamos sobre un conjunto de métodos analíticos para reproducir secuencias de números con cierta aleatoriedad, por lo que se denominan pseudoaleatorios, y que tengan periodos suficientemente largos. Evaluamos la calidad de estas secuencias a partir de conceptos probabilísticos y estadísticos.

***K*eywords**

Simulación, Números Pseudoaleatorios, Generadores.

1. Introducción

Los números aleatorios son la base esencial de la simulación. Usualmente, toda la aleatoriedad involucrada en el modelo se obtiene a partir de un generador de números aleatorios que produce una sucesión de valores que supuestamente son realizaciones de una secuencia de variables aleatorias independientes e idénticamente distribuidas U (0, 1) [1]. Posteriormente estos números aleatorios se transforman convenientemente para simular las diferentes distribuciones de probabilidad que se requieran en el modelo. En general, la validez de los métodos de transformación depende fuertemente de la hipótesis de que los valores de partida son realizaciones de variables aleatorias U (0, 1), pero esta suposición realmente no se cumple, puesto que los generadores de números aleatorios son simplemente programas determinísticos que intentan reproducir una sucesión de valores que parezca aleatoria.

2. Marco teórico

Veamos un ejemplo para comenzar a comprender el desarrollo del actual trabajo. Si plantearamos realizar el sorteo del mundial de futbol mediante una computadora, seguramente la audiencia no confiaría en la aleatoriedad del ordenador y se quejaría. Por lo que preferimos un método físico y sencillo de entender, como lo es extraer bolas de un bombo. Debemos tener en cuenta que incluso este tipo de métodos requiere tomar ciertas precauciones, como las pueden ser que todas las bolas deben estar bien mezcladas y deben tener idéntico peso y tamaño. Claramente este procedimiento no es práctico para una simulación computacional que requiere la generación de cientos de miles de números aleatorios. El método más conveniente y más fiable de generar números aleatorios es utilizar algoritmos determinísticos [2] que posean alguna base matemática sólida. Estos algoritmos producen una sucesión de números que se asemeja a la de una sucesión de realizaciones de variables aleatorias U (0, 1). Es por ello que este tipo de números se denominan pseudoaleatorios. Ahora definiremos el algoritmo que los produce:

Nomenclatura

1 Una variable aleatoria tiene un comportamiento uniforme si en intervalosde igual amplitud contenidos en el intervalo [a,b] tienen la misma probabilidad de ocurrir. Se denota con la letra U.

2 Un algoritmo determinista es un algoritmo que, en términos informales, es completamente predictivo si se conocen sus entradas.

Los **números pseudoaleatorios** se generan de manera secuencial con un algoritmo determinístico, formalmente se definen por:

- **Función de inicialización:** recibe un número, la semilla, y pone al generador en su estado inicial.
- **Función de transición:** transforma el estado del generador
- **Función de salidas:** transforma el estado para producir un número fijo de bits (0 o 1).

Una sucesión de bits pseudoaleatorios se obtiene definiendo la semilla y llamando repetidamente la función de transición y la función de salidas.

Un buen generador de números pseudoaleatorios debería tener las siguientes **propiedades:**

- La sucesión de valores que proporcione debería asemejarse a una sucesión de realizaciones independientes de una variable aleatoria U (0, 1).
- Los resultados deben ser reproducibles, en el sentido de que comenzando con las mismas condiciones iniciales debe ser capaz de reproducir la misma sucesión. Esto nos puede permitir depurar fallos del modelo o simular diferentes alternativas del modelo en las mismas condiciones obteniendo una comparación más precisa. Los procedimientos físicos no permiten que los resultados sean reproducibles.
- La sucesión de valores generados debe tener un ciclo no repetitivo tan largo como sea posible.

Para una correcta ejecución de la investigación recurrimos a definiciones cruciales como lo son:

El **promedio** μ es una medida de posición. No coincide necesariamente con un valor de la variable. El cálculo del promedio de n observaciones de la variable X (x_i con i = 1, 2,. . . ,n), resulta:

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

La **varianza** σ^2 se define como el promedio de los cuadrados de las diferencias de los datos con promedio:

$$\sigma^2 = \frac{\sum_{i=1}^{N} (x_i - \mu)^2}{N} \tag{2}$$

La **desviación estándar** σ es la raíz cuadrada positiva de la varianza:

$$\sigma = \sqrt{\sigma^2} \tag{3}$$

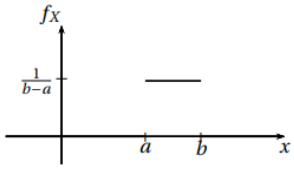
Distribución Normal
Decimos que una variable aleatoria X tiene una distribución normal de parámetros μ y σ donde $\sigma > 0$, y notamos X $\sim \mathcal{N}(\mu, \sigma)$ cuando su densidad de probabilidad es:

$$fx(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-} \frac{(x-\mu)^2}{2\sigma^2} \tag{4}$$

Distribución Uniforme
Decimos que una variable aleatoria X se distribuye uniformemente en el intervalo [a,b] y notamos $X \sim \mathcal{U}(a, b)$ cuando su densidad de probabilidad es:

$$fx(x) = \begin{cases} \frac{1}{b-a} & si \quad x \in [a, b], \\ 0 & en \qquad otro \\ caso \end{cases} \tag{5}$$

Graficamente:



Distribución Chi-Cuadrado
Sean $x_1, x_2, ..., x_n$ variables independientes que siguen una distribución N(0,1)
Sea X una nueva variable definida según:

$$X = x_1^2 + x_2^2 + \ldots + x_n^2 = \sum_{i=1}^{n} x_i^2 \tag{6}$$

en este caso, se dice que X se distribuye como una chi-cuadrado, con n grados de libertad, que representamos como:
$X \rightarrow \chi_n^2$

2.1. Métodos de generación de números pseudoaleatorios

Como ya se comentó, los distintos métodos de simulación requieren disponer de secuencias de números pseudoaleatorios que imiten las propiedades de generaciones independientes de una distribución U (0, 1). En primer lugar nos centraremos en el caso de los generadores congruenciales.

En los **generadores congruenciales lineales** se considera una combinación lineal de los últimos k enteros generados y se calcula su resto al dividir por un entero fijo m. En el método congruencia simple (de orden k= 1), partiendo de una semilla inicial x_0 el algoritmo secuencial es el siguiente:

$$X_{n+1} = (aX_n + c)mod(m) \tag{7}$$

Están determinados por los parámetros:

- Módulo: m > 0
- Multiplicador: $0 \leq$ a < m
- Incremento: c $\leq$ m
- Semilla: $0 \leq X_0 <$ m

donde a (multiplicador), c (incremento) y m (módulo) son enteros positivos fijados de antemano (los parámetros de este generador).

2.1.1. Método de la parte media del cuadrado

El método *middle square*, para generar secuencias de dígitos pseudoaleatorios de 4 dígitos propone iniciar con una semilla de 4 dígitos. Luego se eleva al cuadrado la semilla, produciendo un número de 8 dígitos (si el resultado tiene menos de 8 dígitos se añaden ceros al inicio). Los 4 números del centro serán el siguiente número en la secuencia, y se devuelven como resultado.
Este generador cae rápidamente en ciclos cortos, por ejemplo, si aparece un cero se propagará por siempre por lo tanto no es considerado un método bueno.

2.1.2. Método RAND

Por muchos años (antes de 1995) el generador de la función rand en Matlab fue el generador congruencial:

$$X_{n+1} = (7^5)X_n mod(2^{31} - 1) \tag{8}$$

Una propiedad deseable es que la sucesión parezca una sucesión de observaciones independientes de una Uniforme(0,1).

2.1.3. Método RANDU

RANDU fue generador de números aleatorios ampliamente utilizado en los 60´s y 70´s, se define como:

$$X_{n+1} = (2^{16} + 3)X_n mod(2^{31}) \tag{9}$$

A primera vista las sucesiones se asemejan a una uniforme, sin embargo, cuando se grafican ternas emergen patrones no deseados.

2.2. Tests para determinar el comportamiento de los generadores

Es imposible probar de manera absoluta si una secuencia de números y el generador que la creó es verdaderamente aleatoria. La manera de pragmática de probar la aleatoriedad es tomar varias secuencias de cierto generador y someterlas a una serie de tests estadísticos. A medida que las secuencias aprueban los tests, aumenta la confianza en la aleatoriedad y por lo tanto aumenta la confianza en el generador. Sin embargo, deberíamos esperar que algunas secuencias no parezcan aleatorias y, por lo tanto, esperamos que algunas de ellas no superen algunas pruebas. Por otro lado, si muchas secuencias no superan la prueba deberíamos comenzar a dudar de su aleatoriedad.
En estadística y análisis preliminar de datos, las pruebas de aleatoriedad (o tests de aleatoriedad) son pruebas estadísticas usadas para decidir si una determinada muestra o conjuntos de datos responde a un patrón o puede considerarse aleatoria. El ojo humano no es muy bueno discriminando aleatoriedad y las escalas de las gráficas, es por ello que resulta conveniente hacer pruebas estadísticas para evaluar la calidad de los generadores de números pseudoaleatorios. Hay dos tipos de pruebas:

- **Empíricas:** evalúan estadísticas de sucesiones de números.
- **Teóricas:** se establecen las características de las sucesiones usando métodos de teoría de números con base en la regla de recurrencia que generó la sucesión.

2.2.1. Test visual

Una manera de evaluar un generador de números es crear una visualización de los números que produce. Las personas son muy buenas a la hora de detectar patrones, y las visualizaciones nos permiten usar nuestra visión y nuestra mente para este propósito. Mientras no consideremos este tipo de enfoque como un análisis exhaustivo o formal, es una buena y rápida manera de obtener una impresión aproximada del rendimiento de un determinado generador.

2.2.2. Arriba y abajo test

El método llamado prueba de corridas por arriba y abajo de la media consiste en lo siguiente:

• Denotaremos con un número (1) a aquel número que se encuentre por debajo de la media.
• Denotaremos con un número (0) a aquel número que se encuentre por arriba de la media.

Este procedimiento consiste en determinar una secuencia de unos y ceros de acuerdo a la comparación de cada número que cumpla con la condición de ser mayor o igual a 0.5 (en el caso de los ceros) o ser menor a 0.5 (en el caso de los unos). Luego se determina el número de corridas y los valores de n1 y n2.

Valores que se emplean:
c_o = es el número de corridas en la secuencia, en otras palabras, es la cantidad de unos o ceros consecutivos.
n_0 = cantidad de ceros en la secuencia.
n_1 = cantidad de unos en la secuencia.
n = suma entre n_0 y n_1.

Luego se calcula el valor esperado, la varianza del número de corridas, y elestadístico Z_0 con las siguientes ecuaciones:

- Valor esperado:

$$\mu c_o = \frac{2n_0n_1}{n} + \frac{1}{2} \tag{1}$$

- Variancia del número de corridas:

$$\sigma^2 c_o = \frac{2n_0n_1(2n_0n_1 - n)}{n^2(n-1)} \tag{2}$$

- El estadístico:

$$Z_0 = \frac{c_o - \mu c_o}{\sigma c_o} \tag{3}$$

Para saber si el estadístico está fuera del intervalo se emplea la siguiente fórmula:

$$-Z_{\frac{\alpha}{2}} \leq Z_0 \leq Z_{\frac{\alpha}{2}} \tag{4}$$

Si la condición anterior se cumple, entonces se concluye que los números evaluados son independientes, de lo contrario se rechaza al conjunto.

Las pruebas de independencia consisten en demostrar que los números generados son estadísticamente independientes entre si, esto es que no dependen uno de otro. Una prueba de Corridas es un método que nos ayuda a evaluar el carácter de aleatoriedad de una secuencia de números estadísticamente independientes y números uniformemente distribuidos. Es decir dado una serie de números determinar si son o no aleatorios. Este método es uno de los mas sencillos ya que solo implica el diferenciar cuales números están arriba o debajo de la media, pero su sencillez no implica que su importancia sea menor.

2.2.3. Run test

Codificaremos valores por encima de la mediana como positivos y valores por debajo de la mediana como negativos. Una ejecución se define como una serie de valores positivos (o negativos) consecutivos. La prueba de ejecuciones se define como:

Ho = la secuencia se produjo de manera aleatoria.
Ha = la secuencia no se produjo de manera aleatoria

Estadística de la prueba = La estadística de la prueba es:

$$Z = \frac{R - \bar{R}}{s_R} \tag{1}$$

donde R es el número observado de carreras, $\bar{R}$, es el número esperado de ejecuciones, y s_R es la desviación estándar del número de carreras. Los valores de $\bar{R}$ y s_R se calculan de la siguiente manera:

$$\bar{R} = \frac{2n_1n_2}{n_1 + n_2} + 1 \tag{2}$$

$$s_R^2 = \frac{2n_1n_2(2n_1n_2 - n_1 - n_2)}{(n_1 + n_2)^2(n_1 + n_2 - 1)} \tag{3}$$

con n_1 y n_2 denotando el número de valores positivos y negativos en la serie.

La prueba de ejecuciones rechaza la hipótesis nula si:

$$|Z| > Z_{1-\frac{\alpha}{2}} \tag{4}$$

Para una prueba de muestra grande (donde n_1> 10 y n_2 > 10), la estadística de la prueba se compara con una tabla normal estándar. Es decir, en el nivel de significancia del 5 %, una estadística de prueba con un valor absoluto superior a 1,96 indica no aleatoriedad. Para una prueba de ejecuciones de muestra pequeña, hay tablas para determinar valores críticos que dependen de los valores de n_1 y n_2.

2.2.4. Chi Square test

Una prueba de chi-cuadrado se usa en estadística para probar la independencia de dos eventos. Dados los datos de dos variables, podemos obtener el conteo observado O y el conteo esperado E. Chi-Square mide cómo el conteo esperado E

y el conteo observado O se desvían entre sí.

$$X_c^2 = \sum \frac{(O_i - E_i)^2}{E_i} \tag{1}$$

Donde:
c = grados de libertad
O = valores observados
E = valores esperados

Consideremos un escenario en el que necesitamos determinar la relación entre la característica de categoría independiente (predictor) y la característica de categoría dependiente (respuesta). En la selección de características, nuestro objetivo es seleccionar las características que dependen en gran medida de la respuesta. Cuando dos características son independientes, el conteo observado está cerca del conteo esperado, por lo que tendremos un valor de Chi-Cuadrado más pequeño. Un valor tan alto de Chi-Cuadrado indica que la hipótesis de independencia es incorrecta. En palabras simples, cuanto mayor sea el valor de Chi-Cuadrado, la función depende más de la respuesta y se puede seleccionar para el entrenamiento del modelo.

2.2.5. Poker test

El *Poker test* consiste en tomar una serie de números decimales, truncar los primeros *n* decimales y evaluar la cantidad de veces que aparecen cada combinacion de números decimales. Por ejemplo, para *n = 3* se cuentan la cantidad de veces que un número tiene tres decimales iguales, luego las veces que aparecen dos decimales iguales y uno distinto y, finalmente, la cantidad de veces que aparecen tres decimales distintos. Para otro valores de *n* se aplica el mismo razonamiento.
Una vez encontrada la frecuencia relativa de cada combinación en la población se procede a realizar:

$$X^2 = \sum \frac{(f_o - f_t)^2}{f_t} \tag{1}$$

donde f_o refiere a la frecuencia relativa de cada combinación y f_t refiere al valor esperado de dicha frecuencia.

Una vez obtenido el valor de X^2 se elige un grado de confianza para y los grados de libertad para encontrar un valor de chi-cuadrado. Si el valor de X^2 es menor al de chi-cuadrado se aprueba el test.
La cantidad apropiada de grados de libertad es uno menos que el número de intervalos de clase, es decir, la cantidad de combinaciones distintas de decimales.

3. Exposición de los resultados y análisis de los mismos

A continuación se detallan los valores de los intervalos de factibilidad utilizando una confianza de 95 % y con 2 grados de libertad:

Intervalos de factibilidad			
Run Test	Chi Square test	Arriba y Abajo test	Poker test
-1,96 < $Z_0 < 1,96$	$Z_0 < 5,99$	-1,96 < $Z_0 < 1,96$	$Z_0 < 5,99$

3.1. Análisis visual

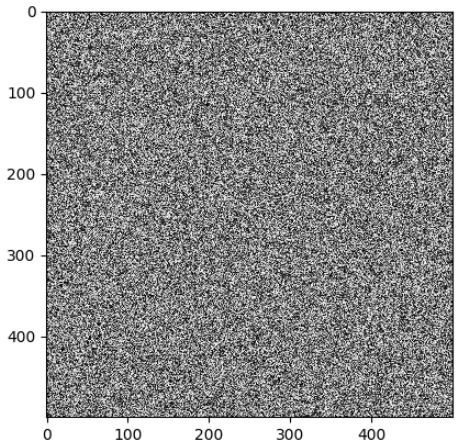


Figura 1: *Mapa de bits del generador Randu*

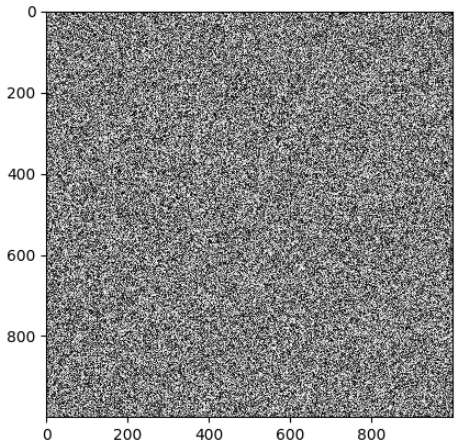


Figura 3: *Mapa de bits del generador Python*

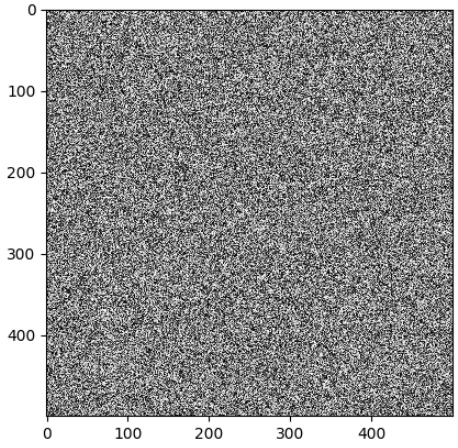


Figura 2: *Mapa de bits del generador Rand*

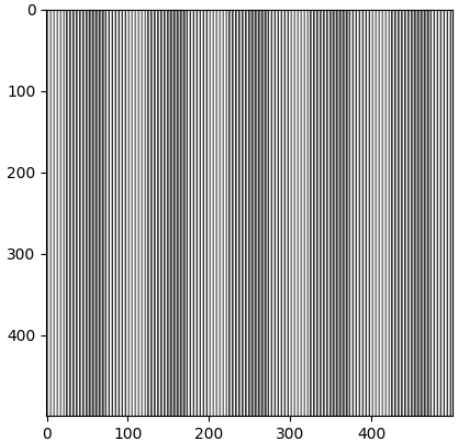


Figura 4: *Mapa de bits del generador Mid Square*

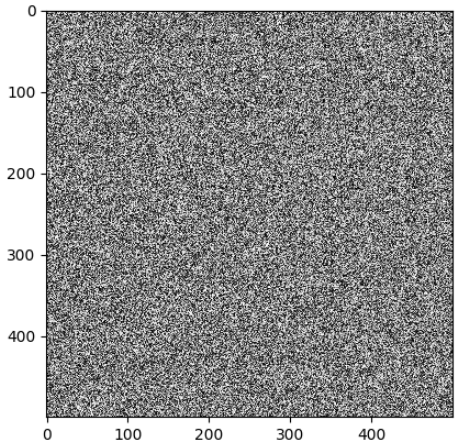


Figura 5: *Mapa de bits del generador Random.org*

A continuación se detalla la tabla correspondiente con los resultados obtenidos al finalizar la simulación

Resultados					
Generador	Run Test	Chi Square test	Arriba y Abajo test	Poker test	Visual test
Mid Square	3.200007483566187	3.365377246589806	$Z \leq 0{,}002000016000188$	13.8405481062408	NO
Rand	-0.097615732846377	1.669123398022814	$Z \leq -0{,}09562462744531$	8.09219567038458	OK
Randu	-1.331949015208048	1.667251286584182	$Z \leq -1{,}34195557364742$	8.09257804100064	OK
Random.org	-0.716263946279912	1.675056706524938	$Z \leq -0{,}71827630731985$	8.09218397524057	OK
Python	-0.325921208202092	1.667598276985541	$Z \leq -0{,}31992994964926$	8.09219082018241	OK

Cuadro 1: Tabla con los resultados de los tests - 25000 números pseudoaleatorios

Referencias a los resultados obtenidos en la tabla:

- Pasaron la prueba
- No pasaron la prueba

4. Conclusión

En el presente trabajo, llevamos a cabo una serie de pruebas estadísticas para determinar la efectividad de algunos generadores pseudoaleatearios, el problema es que matemáticamente no existe la prueba perfecta, por lo tanto, podríamos encontrarnos con secuencias que pasaran todas estas pruebas sin ser realmente aleatorias.

Es necesario un análisis matemático riguroso para determinar el grado de aleatoriedad de un generador pseudoaleatorio y resulta de extrema importancia debido a que tiene aplicaciones en sistemas de cifrado, protección de aplicaciones web, etc.

Los resultados que obtuvimos en base a los tests realizados, se encuentran dentro de los parámetros esperados debido a que ningún generador logró superar la totalidad de pruebas que se llevaron a cabo, quedando en evidencia, la falta de aletoriedad de las secuencias generadas por los mismos, sin embargo algunos generadores presentaron un mejor desempeño que otros más primitivos como el Middle Squared Method.

En base a las simulaciones realizadas, podemos decir que:

- El método de la parte media del cuadrado (Middle Squared Method) realmente no es confiable y no se puede utilizar en un estudio de estadística computacional o Computer Science, así como para aplicaciones específicas en seguridad informática entre otras cosas. Las secuencias de números pseudoaleatorios generadas por este método presentan patrones evidentes de repetición, claramente observables mediante el test visual, siendo el único generador que no logró superar dicha prueba.

- En cuanto al comportamiento de RAND vs RANDU, el RANDU suele superar algunas pruebas con una performance mucho mejor que la del metodo RAND, sin embargo se requerirían más tipos de pruebas diferentes para concluir esto con mayor determinación.
- Con respecto al generador propio de Python, lo hemos utilizado sin realizarle modificaciones y los resultados obtenidos no son tan prometedores como esperábamos.

A partir de la información que recopilamos, concluímos que los generadores utilizados, podrían ser empleados en diferentes campos como el modelado por computadora, estadística, o incluso en criptografía, así como para su aplicación en sistemas que exigen estos números para seguridad informática entre otras aplicaciones, puesto que no tenemos pruebas contundentes que nos hagan dejar de recomendarlos.

Referencias

[1] Python
https://python-para-impacientes.blogspot.com/2014/08/graficos-en-ipython.html
[2] Probabilidad y Estadistica
https://relopezbriega.github.io/blog/2015/06/27/probabilidad-y-estadistica-con-python/
[3] Python tests
https://www.itl.nist.gov/div898/handbook/eda/section3/eda35d.htm
[4] Random.org
https://www.random.org/analysis/
[5] Knuth hablando sobre RANDU. Art of Computer Programming, Volume 2: Seminumerical Algorithms - Donald E. Knuth – 4 de noviembre 1997 - ISBN 0-201-03822-6 (Versión original 1981)
[6] Test Arriba y Abajo
https://es.wikipedia.org/wiki/Top-down$_{y}bottom - up$
[7] Metodo chi-cuadrado
https://psicologiaymente.com/miscelanea/prueba-chi-cuadrado
[8] Números pseudoaleatorios
https://tereom.github.io/est-computacional-2018/numeros-pseudoaleatorios.html