# GENERADORES PSEUDOALEATORIOS

**Gianfranco Scaldini**
Cátedra Simulación
UTN - FRRO
Zeballos 1341, S2000
gianscaldinis6@gmail.com

**Luciano Vanelli**
Cátedra Simulación
UTN - FRRO
Zeballos 1341, S2000
lucianovannelli@gmail.com

**Manuel Ángel Bahamonde**
Cátedra Simulación
UTN - FRRO
Zeballos 1341, S2000
manuelbahamonde6@gmail.com

**Ramiro Sammataro**
Cátedra Simulación
UTN - FRRO
Zeballos 1341, S2000
ramasammataro@gmail.com

23 de abril de 2021

**ABSTRACT**

Nuestro trabajo tiene como objetivo analizar cómo se realiza la generación de números pseudoaleatorios abordados por 3 algoritmos diferentes y 4 test diferentes. La programación del trabajo se realizó con el lenguaje Python en su versión 3.7.

## 1. Introducción

Se quiere simular y poder observar la generación de números pseudoaleatorios a través de la utilización de 3 algoritmos. Éstos, serán programados por nosotros y uno será propio del lenguaje Python y de la librería Random. Para el análisis de estos generadores utilizaremos determinados tests para evaluar el comportamiento de los resultados obtenidos.

## 2. Números aleatorios

Un número aleatorio es aquel obtenido al azar, es decir, que todo número tenga la misma probabilidad de ser elegido y que la elección de uno no dependa de la elección del otro. El ejemplo clásico más utilizado para generarlos es el lanzamiento repetitivo de una moneda o dado ideal no trucado. Los números aleatorios permiten a los modelos matemáticos representar la realidad. El desarrollo de las ciencias de la computación se ha edificado, en gran parte, sobre la base de sólidos conceptos de las matemáticas. Los números aleatorios son un buen ejemplo. Estos han sido utilizados tradicionalmente en una gran variedad de aplicaciones (juegos, criptografía, experimentos científicos, etc.), y constituyen el fundamento para el estudio y modelaje de sistemas estocásticos.

La calidad de los números aleatorios es un factor crítico de éxito para la solución de problemas en diferentes áreas; ésta es garantizada gracias a la teoría matemática y a las ventajas que ofrecen los computadores para la implementación de los diferentes métodos propuestos para la generación eficiente de dichos números. Los números pseudoaleatorios son unos números generados por medio de una función (determinista, no aleatoria) y que aparentan ser aleatorios.

Estos números pseudoaleatorios se generan a partir de un valor inicial aplicando iterativamente la función. La sucesión de números pseudoaleatorios es sometida a diversos tests para medir hasta qué punto se asemeja a una sucesión aleatoria . El campo de aplicación de los números pseudoaleatorios es muy amplio, entre ellos se encuentran : simulaciones, muestreos, análisis numéricos, testeo de programas, juegos de azar, toma de decisiones, etc.

Los números aleatorios son útiles para una variedad de propósitos, como encriptar información, determinar las cartas a dar en una mano en un juego de solitario, o simular variables desconocidas de sistemas reales complejos como por ejemplo los mercados de valores, el pronóstico del clima, la programación de vuelos en las aerolíneas o la selección de

muestras representativas de pacientes al probar nuevos medicamentos. Sin embargo, se establece el interrogante de cómo se pueden generar estos números aleatorios en una computadora con naturaleza determinista, después de todo una computadora sigue sus instrucciones a ciegas y, por lo tanto, es completamente predecible.

Existen dos enfoques principales para generar números aleatorios usando una computadora los cuales se investigarán en el presente trabajo: los generadores de números pseudoaleatorios o pseudo-random number generators (PRNG) y los generadores de números aleatorios verdaderos o true random number generators (TRNG). Los enfoques tienen características bastante diferentes y cada uno tiene sus ventajas y sus desventajas.

## 3. Números Pseudoaleatorios

Antes de las computadoras, existieron diferentes métodos para generar secuencias de números aleatorios, tales como los procedimientos físicos(monedas, dados, bolilleros, etc), tablas de 40000 dígitos aleatorios por Tipett en 1927(no resultaron con distribución uniforme),tabla de 100.000 números aleatorios hecha por Kendall en 1939, tabla de 1.000.000 números aleatorios a partir de ruido electrónico hecha por Rand Corporation en 1955, etc.

Las principales desventajas de todos estos intentos de realizar una secuencia de números aleatorios de manera física son que: no pueden repetirse una misma secuencia, no hay velocidad computacional e incorporar una tabla a la computadora implica gran costo de almacenamiento en relación a la cantidad de números (si se quiere realizar una tabla infinitamente aleatoria). En 1946 Jon Von Neuman sugirió usar las operaciones aritméticas de una computadora para generar secuencias de número pseudoaleatorios, mediante el "método del valor medio". Este generador cae rápidamente en ciclos cortos, por ejemplo, si aparece un cero se propagara por siempre. A inicios de 1950 se exploró el método y se propusieron mejoras, por ejemplo para evitar caer en cero. Metrópolis logró obtener una secuencia de 750,000 números distintos al usar semillas de 38 bits (usaba sistema binario), además la secuencia de Metrópolis mostraba propiedades deseables. No obstante, el método del valor medio no es considerado un método bueno por lo común de los ciclos cortos. A partir de estos primeros métodos de generación de números pseudoaleatorios empezaron a surgir nuevos y mejores métodos para su utilización.

## 4. Aleatoriedad

Cuando se habla de aleatoriedad naturalmente surgen preguntas: qué es, qué significa que algo sea aleatorio o si verdaderamente puede existir algo aleatorio. Esas son preguntas con importancia práctica, ya que la aleatoriedad es sorprendentemente útil en muchos campos de estudio, entre ellos la simulación de eventos, variables y sistemas. Siguiendo ninguna ley, los números aleatorios carecen de previsibilidad. El resultado de todo suceso aleatorio no puede determinarse en ningún caso antes de que este se produzca. Una secuencia aleatoria de eventos, símbolos o pasos a menudo no tiene orden y no sigue un patrón o combinación inteligible. Los eventos aleatorios individuales son, por definición, impredecibles, pero la frecuencia de diferentes resultados en numerosos eventos o ensayos es predecible dado que a menudo siguen una distribución de probabilidad. Por ejemplo, cuando se lanzan dos dados, el resultado de cualquier tirada en particular es impredecible: cada dado puede tomar un valor x  [1; 6] con una probabilidad p = 1/6 pero la suma de 7 ocurrirá dos veces más que 4. En cierto sentido, no existe un número aleatorio por sí solo. No tiene sentido preguntarse si 2 es un número aleatorio. Más bien hablamos de una secuencia de números aleatorios independientes con una distribución específica. De estas observaciones se desprende que esta distribución debe ser uniforme. Esto es lógico, pues si un valor apareciera más veces (por ejemplo la mitad de las tiradas el dado cae en 6) la distribución estaría sesgada y no sería verdaderamente aleatoria (se espera que caiga en 6 la mitad de las veces).

## 5. Generadores reales

Los generadores de números aleatorios reales o TRNG extraen o heredan la aleatoriedad a partir de fenómenos físicos del mundo externo y la introducen en la computadora. Un fenómeno físico realmente bueno para usar es la radioactividad. Los puntos en el tiempo en que se desintegra una fuente radiactiva son completamente impredecibles, y pueden detectarse fácilmente y alimentarse a una computadora. La radiación del ambiente se puede tabular en un periodo corto de tiempo con un contador Geiger y obtiene una secuencia aleatoria de valores. Otro fenómeno físico adecuado es el ruido atmosférico, que es bastante fácil de detectar con una radio normal. Las desventajas de obtener números aleatorios con un método físico es que se necesita un dispositivo externo y no es un proceso rápido. Además los TRNG son no deterministas, lo que significa que una secuencia dada de números no puede reproducirse, propiedad en muchos casos necesaria para testear software o reproducir experimentos.

## 6. Generadores pseudoaleatorios

En la mayoría de los casos no necesitamos producir números verdaderamente aleatorios y en tales casos se puede acudir a los generadores de números pseudoaleatorios o PRNG. Estos son más eficientes, lo que significa que pueden producir muchos números en poco tiempo en comparación con los TRNG. Un generador de números aleatorios consiste en una función que devuelve los valores de una secuencia de números reales ecuación donde cada xi  [0; 1], los cuales se comportan como números aleatorios aunque no lo sean (de ahí el prefijo pseudo). Se aceptan entonces como sustitutos de los números aleatorios para determinadas aplicaciones porque su comportamiento es similar, es decir porque tienen muchas de las propiedades estadísticas más importantes esperadas en los números aleatorios puros: impredecibilidad, uniformidad e independencia.

El algoritmo que genera la secuencia de números se denomina generador se recursivo y puede ser asociado a la relación de recurrencia H de grado k en la Ecuación 1.

$$x_i = H(x_{i-1}, x_{i-2}, ..., x_{i-k}) \tag{1}$$

Todos los valores de la secuencia x1, x2, x3, ... son generados por H a partir de un valor inicial x0 denominado semilla o seed. A partir de esta definición se puede deducir entonces que:

- Los números de la secuencia no son independientes entre sí por el simple hecho de responder a una relación de recurrencia matemática.
- Los PRNG son deterministas (siguen alguna ley), es decir, el generador generará la misma secuencia de números si se introduce la misma semilla al inicio.

El determinismo puede resultar útil si se necesita volver a reproducir la misma secuencia de números en una etapa posterior. La no independencia, por el otro lado, contradice a la naturaleza de la aleatoriedad y no es beneficioso pues puede crear patrones no deseables en la secuencia.

Por ejemplo algo que haría que una secuencia de números parezca menos aleatoria sería que el mismo patrón de números se repita con un periodo chico o que haya una clara monotonía de crecimiento o decrecimiento en los números de la secuencia. Se puede buscar entonces un comportamiento similar a la independencia (pseudo-independencia) haciendo que los elementos de la secuencia tengan mínima correlación.

## 7. Método de la parte media del cuadrado

Un método simple de PRNG para generar secuencias de dígitos pseudoaleatorios de 4 dígitos es el método de la parte media del cuadrado o middle square de Jon Von Neuman (1946).

1. Se inicia con una semilla de 4 dígitos.
2. La semilla se eleva al cuadrado, produciendo un número de 8 dígitos. Si el resultado tiene menos de 8 dígitos se añaden ceros no significativos al inicio.
3. Los 4 números del centro serán el siguiente número en la secuencia, y se devuelven como resultado.

En la práctica el método de la parte media del cuadrado no es uno muy bueno. Los números generados son de 4 dígitos y este generador cae rápidamente en ciclos cortos de números en el rango [0;9999] dependiendo de la semilla, o si aparece un cero. En la Figura 2 se muestra una simulación del método en la cual se puede observar claramente como empieza a ciclar los números generados una vez que un número se repite. La función cuadrática no es muy buena para cubrir el rango de números posibles a generar. Cómo se extraen los dígitos del medio, se encuentra que solo el 60 % de los diferentes valores realmente se alcanzan efectivamente independientemente de la semilla.

## 8. Periodicidad

Como el método de la parte media del cuadrado, todos los PRNG son periódicos, lo que significa que todas las secuencias de números pseudoaleatorios eventualmente se repiten. Las secuencias son cíclicas y la cantidad de términos en las secuencias antes de que se repita se denomina periodo. Si bien la periodicidad casi nunca es una característica deseable, los PRNG modernos tienen un periodo que es tan largo que puede ignorarse para la mayoría de los propósitos prácticos. La solución óptima sería hacer el periodo del ciclo lo suficientemente grande (ajustando los parámetros del generador) para que la secuencia se repita más tarde que temprano. En teoría, si los números de la secuencia xi N i=1

son generados sobre un conjunto cíclico Zp es imposible ir más de p términos de la secuencia sin repetir algún número por el principio del palomar. Es decir, m huecos pueden albergar como mucho m objetos si cada uno de los objetos está en un hueco distinto, así que el hecho de añadir otro objeto fuerza a volver a utilizar alguno de los huecos.

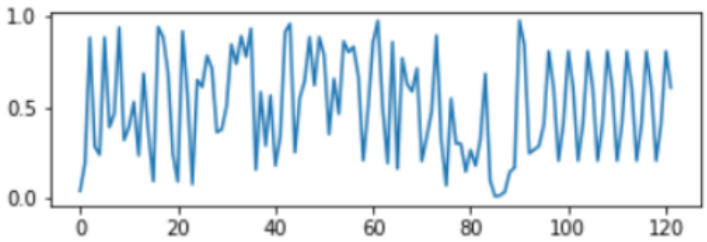



Figura 1: secuencia de números generados por el método de la parte media del cuadrado.

## 9. Comparativa

Las características de los distintos generadores hacen que los PRNG sean adecuados para aplicaciones donde se requieren muchos números y donde es útil que la misma secuencia se pueda reproducir fácilmente. Ejemplos populares de tales aplicaciones son las aplicaciones de simulación y modelado. Los PRNG no son adecuados para aplicaciones en las que es importante que los números sean realmente impredecibles,como el cifrado de datos y los juegos de azar y en tales casos sería recomendable utilizar algún TRNG.

| | **TRNG** | **PRNG** |
|---|---|---|
| **Mecanismo** | Físico | Matemático |
| **Eficiencia** | Pobre | Alta |
| **Independencia** | Si | No |
| **Determinismo** | No determinista | Determinista |
| **Periodicidad** | Aperiódico | Periódico |
| **Distribución** | Uniforme | Uniforme |

Figura 2: Comparación de características de los diferentes tipos de generadores de números aleatorios.

## 10. Generador lineal congruencial

Un generador lineal congruencial (GLC) es un algoritmo que permite obtener una secuencia de números pseudoaleatorios calculados con una función lineal definida a trozos discontinua. Es uno de los métodos más antiguos y conocidos para la generación de números pseudoaleatorios. La teoría que sustenta el proceso es relativamente fácil de entender, el algoritmo en si es de fácil implementación y su ejecución es rápida, especialmente cuando el hardware del ordenador puede soportar aritmética modular al truncar el bit de almacenamiento correspondiente. En nuestro presente trabajo utilizaremos 3 generador GCL: El generador Rand (generador de la función rand en Matlab) , Randu(fue un generador de números aleatorios ampliamente utilizado en los 60s y 70s), y un GCL genérico. En nuestro caso, elegimos una buena semilla para que se cumpla la condición de aleatoriedad debido a que si la semilla es demasiado pequeña, esta no se cumple.

## 11. Test utilizados

Tests Estadísticos: Los tests estadísticos (o contrastes) son el instrumento para validar o rechazar las hipótesis de modelación probabilistas. Ellos tratan de distinguir lo que es plausible de lo que es muy poco verosímil, en el marco de un modelo dado. Los test de Chi-cuadrado y de Poker que se mencionan a continuación se utilizarán para poder verificar la uniformidad de los números aleatorios generados. Esto nos va a permitir saber si los números generados son una muestra de la distribución uniforme en el intervalo [0,1]. Los demás test de corridas no demostrarán la uniformidad de la distribución sino que permitirán visualizar la aleatoriedad de los números generados

### 11.1. Test de Bondad de Chi-Cuadrado

La prueba chi-cuadrado es una de las más conocidas y utilizadas para analizar variables nominales o cualitativas, es decir, para determinar la existencia o no de independencia entre dos variables. Que dos variables sean independientes significa que no tienen relación, y que por lo tanto una no depende de la otra, ni viceversa. Dividiremos en 10 intervalos a los números y los agruparemos para realizar este test((0;0.1), ...,(0.9;1)]

### 11.2. Test de Poker

Prueba grupos de números juntos como una mano de poker y compara cada mano con la mano esperada usando la prueba Chi-cuadrada. Esta prueba examina en forma individual los dígitos del número pseudoaleatorio generado. La forma como esta prueba se realiza es tomando 5 dígitos a la vez y clasificándolos como : Par, dos pares, tercia, póker, quintilla full y todos diferentes. Las probabilidades para cada una de las manos del póker diferentes se muestran a continuación: Todos diferentes = 0.3024. Un par = 0.504. Dos pares = 0.108. Tercia = 0.072. Full = 0.009. Quintilla = 0.0001.

A modo de simplificación, nosotros optamos por tomar los primeros 3 dígitos decimales del número generado y no los primeros 5 como en el caso del test de póker original(Los dígitos pueden ser todos diferentes, dos iguales o tres iguales).

### 11.3. Test de Corridas

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. Si tenemos una secuencia de números de tal manera que a cada uno de los números siga otro mayor la secuencia dada será ascendente (arriba). Si cada número va seguido por otro menor, la secuencia será descendente (abajo).

### 11.4. Test de Corridas de Arriba y De Abajo de la Media para Números Uniformemente Distribuidos.

Es generalmente la prueba principal usada para verificar la dependencia. Esta prueba detecta si un patrón inaceptable estadísticamente que se incrementa o decrece existe entre números adyacentes en un flujo de números. El procedimiento de esta prueba consiste en determinar una secuencia de + y -, de acuerdo con una comparación entre los símbolos del conjunto ri y 0.5. Denotaremos con +, a aquel número que se encuentre por arriba de 0.5.Por lo contrario, si el número se encuentra debajo de 0.5, lo denotaremos con -. Después se determina el número de corridas observadas Co y los valores de n,n0 y n1 (n es el total de números, n0 cantidad de símbolos -", y n1 cantidad de símbolos ").

## 12. Algoritmos de los tests utilizados

En estas sección presentaremos los algoritmos utilizados por cada test, para que se pueda entender de una forma más clara cómo es el procedimiento de los mismos.

### 12.1. Test de bondad Chi-cuadrado:

Procedimiento:

1. Partir el soporte de F en c celdas que son exhaustivas y mutuamente excluyentes.
2. Contar el número de observaciones en cada celda Oi.
3. Calcular el valor esperado en cada celda bajo F: ei=npi.
4. Calcular la estadística de prueba.
5. Verificar que el valor obtenido sea menor al valor obtenido en tabla con un porcentaje confianza determinado y acorde a los grados de libertad.

### 12.2. Test de Poker:

Procedimiento:

1. Determinar la categoría de cada número del conjunto ri.

2. Contabilizar los números ri de la misma categoría o clase para obtener la frecuencia observada (Oi).
3. Calcular el estadístico de la prueba.
4. Finalmente comparar el estadístico.

### 12.3. Test Corridas:

Procedimiento:

1. Primeramente le asignaremos un signo a cada número de la secuencia ya sea + ó - , eso dependerá de lo siguiente.
2. Si a un número le sigue otro mayor, se le asigna +. Esto es si Xi <Xi +1 el signo asignado será (+). Siendo Xi unnúmero de la muestra o secuencia de números.
3. Si el número siguiente es menor, se le da un signo -. Esto es si Xi >Xi +1 el signo asignado será (-).
4. Se continuará con la comparación de los números y la asignación de su signo correspondiente hasta N-1. Es decir hasta el penúltimo número de la secuencia, ya que al último número le sigue un evento nulo(no es posible compararlo con otro número).

### 12.4. Test Corridas de Arriba y Abajo:

Procedimiento:

1. Primeramente le asignaremos un signo a cada número de la secuencia ya sea + ó - , eso dependerá de lo siguiente.
2. Denotaremos con un signo - a aquel número que se encuentre por debajo de la media. Denotaremos con un signo + a aquel número que se encuentre por arriba de la media.
3. Se continuará con la comparación de los números y la asignación de su signo correspondiente hasta N-1. Es decir hasta el penúltimo número de la secuencia, ya que al último número le sigue un evento nulo(no es posible compararlo con otro número).

## 13. Metodología

Utilizando el lenguaje de programación python desarrollamos un programa con el propósito de generar números pseudoaleatorios a través de 3 algoritmos distintos. Utilizaremos las librerías random,numpy y math para estas tareas. En el presente trabajo, realizamos 4 tests diferentes con el objetivo de poder corroborar con mayor precisión si los números generados son realmente pseudoaleatorios. En base a diferentes generadores, entre los cuales se encuentran el generador de cuadrado medio, GCL randu y el propio de python (Mersenne Twister) ,implementamos los test que fueron: Test de bondad de Chi Cuadrado, test de poker, test de corridas y el de corridas de arriba y abajo de la media para números uniformemente distribuidos. Realizamos una gran cantidad de simulaciones para cada algoritmo de generación de números aleatorios para poder observar los resultados de los tests que nos indicarán si efectivamente los resultados son pseudoaleatorios. A partir de dichas simulaciones sacamos las respectivas conclusiones.

## 14. Exposición de resultados y análisis de los mismos

### 14.1. Tabla comparativa de resultados

En la siguiente tabla expresaremos los resultados obtenidos por medio de los cuatro test, a partir de los tres generadores. A través de las simulaciones realizadas pudimos observar que los resultados arrojados por los test dieron los valores esperados para que los generadores resulten generadores de números pseudoaleatorios(todos los test resultaron correctamente) salvo en el caso del generador del cuadrado medio en algunos test.

| Generadores | Test Chi-Cuadrado | Pasó |
|---|---|---|
| Randu | Valor de Chi-cuadrado obtenido(11.559) <Valor de la tabla(16.92) | Verdadero |
| Mersenne Twister | Valor de Chi-cuadrado obtenido(7.150) <Valor de la tabla(16.92) | Verdadero |
| Cuadrado Medio | Valor de Chi-cuadrado obtenido(134660.22 ) <Valor de la tabla(16.92) | Falso |

Al realizar el test de Chi-cuadrado con los numeros generados, se puede observar que el valor del estadistico Chicuadrado obtenido para cada generador es menor al valor que se puede obtener con la tabla con un 95 9 grados de libertad, excepto para el generador cuadrado medio que se observa un valor excesivamente grande. Al realizar el test de Poker

| Generadores | Test de Poker | Pasó |
|---|---|---|
| Randu | Valor de Chi-cuadrado obtenido(4.7959) <Valor de la tabla(5.99) | Verdadero |
| Mersenne Twister | Valor de Chi-cuadrado obtenido(0.5079) <Valor de la tabla(5.99) | Verdadero |
| Cuadrado Medio | Valor de Chi-cuadrado obtenido(42629.857) <Valor de la tabla(5.99) | Falso |

con los números generados, se puede observar que el valor del estadístico Chi-cuadrado obtenido para cada generador es menor al valor que se puede obtener con la tabla con un 95 de libertad, excepto para el generador cuadrado medio que se observa un valor excesivamente grande.

| Generadores | Test de Corridas | Pasó |
|---|---|---|
| Randu | Valor de Zo obtenido(-0.3679) <Zmax(1.96) | Verdadero |
| Mersenne Twister | Valor de Zo obtenido(0.7746) <Zmax(1.96) | Verdadero |
| Cuadrado Medio | Valor de Zo obtenido(-1.9335)<Zmax(1.96) | Verdadero |

En este Test lo que se pretende es agrupar las secuencias de mismo signo para poder obtener los valores que permitirán realizar la prueba.Todos los generadores lograron pasar este test

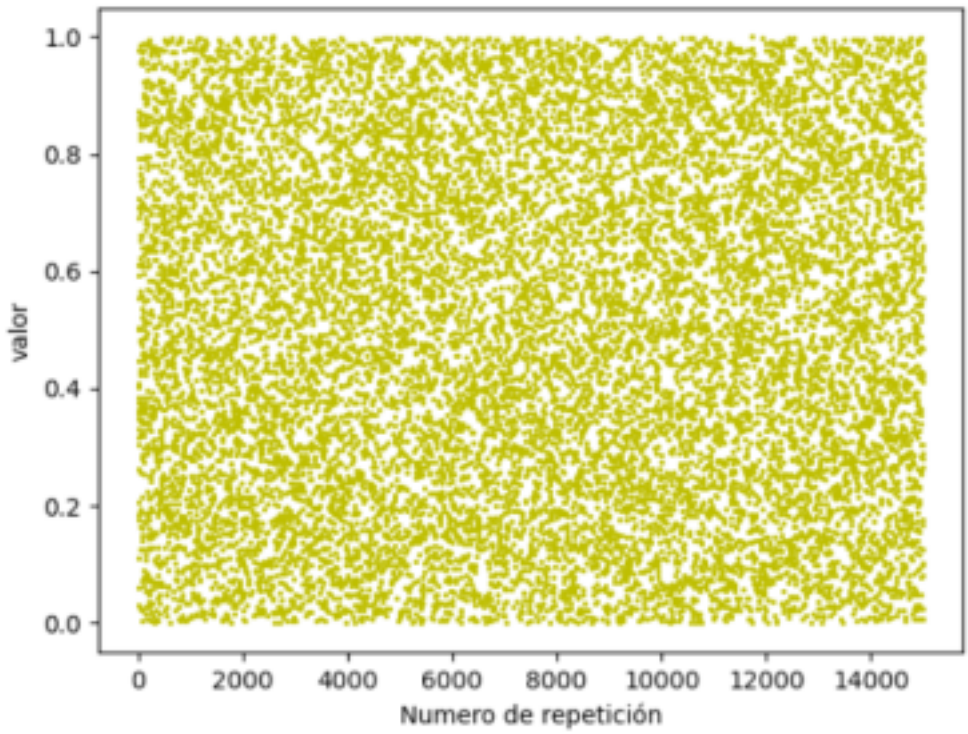
| Generadores | Test de corridas arriba y abajo | Pasó |
|---|---|---|
| Randu | Valor de Zo obtenido(-0.6726) <Zmax(1.96) | Verdadero |
| Mersenne Twister | Valor de Zo obtenido(1.16135) <Zmax(1.96) | Verdadero |
| Cuadrado Medio | Valor de Zo obtenido(-1.2099) <Zmax(1.96) | Verdadero |

Este método es uno de los mas sencillos ya que solo implica el diferenciar cuales números están arriba o abajo de la media, pero su sencillez no implica que su importancia sea menor. Este test todos los generadores lo lograron pasar.

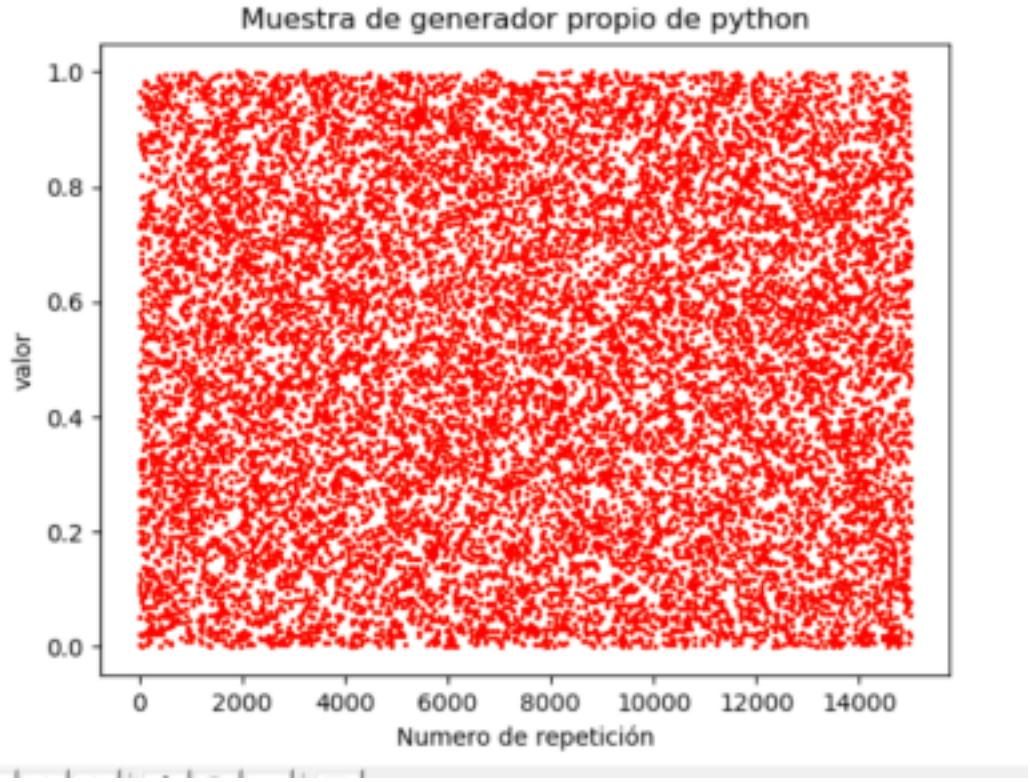
| Generadores | Tipo de generador | Test Chi-Cuadrado | Test de Poker | Test de Corridas | Test corridas Arriba y Abajo |
|---|---|---|---|---|---|
| Randu | Valor de Zo obtenido(-0.6726) <Zmax(1.96) | OK | OK | OK | OK |
| Mersenne Twister | Valor de Zo obtenido(1.16135) <Zmax(1.96) | OK | OK | OK | OK |
| Cuadrado Medio | Valor de Zo obtenido(-1.2099) <Zmax(1.96) | FALSO | FALSO | OK | OK |

### 14.2. Gráficos de dispersión

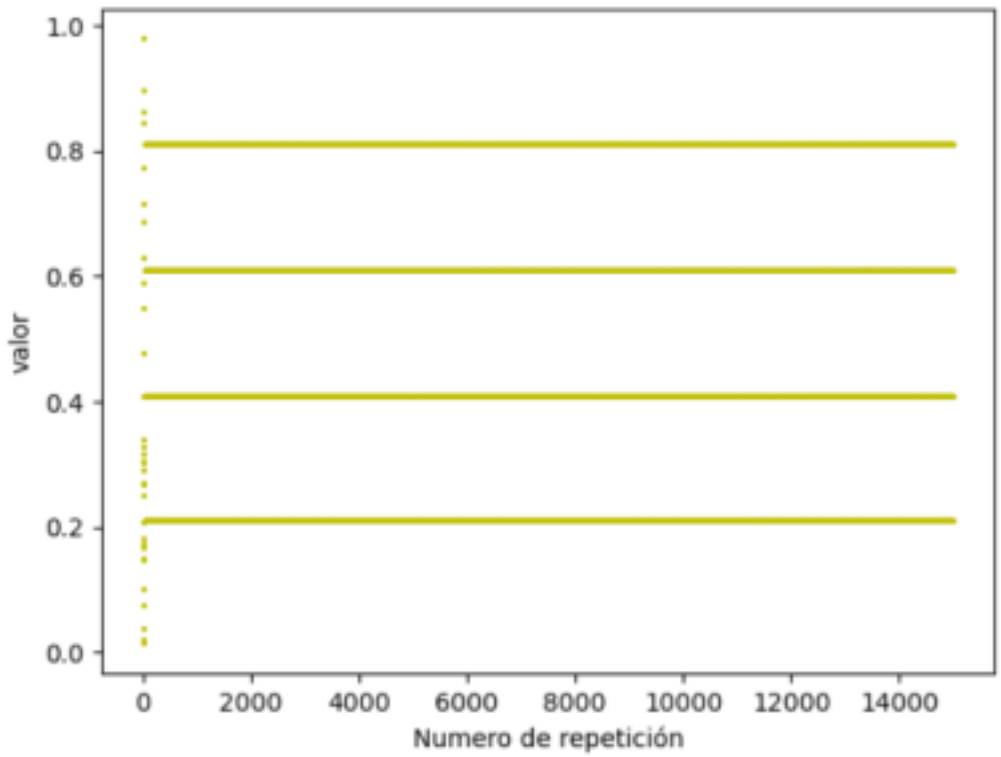
En los siguientes gráficos se representarán las dispersiones de los valores obtenidos por cada uno de los generadores. La primer imagen representa el generador Randu, la segunda al generador de python y la tercera al de cuadradro medio.




En esta gráfica, podemos observar la dispersión que tienen los números generados por el generador Randu. En el eje de las abscisas se encuentran los números de repeticiones del generado y en el eje de las ordenadas, el número generado. Se observa que existe realmente una dispersión.



En esta gráfica, podemos observar la dispersión que tienen los números generados por el generador Mersenne twister. En el eje de las abscisas se encuentran los números de repeticiones del generado y en el eje de las ordenadas, el número generado. Se observa que existe realmente una dispersión.




En esta gráfica, podemos observar la dispersión que tienen los números generados por el generador Cuadrado medio. En el eje de las abscisas se encuentran los números de repeticiones del generado y en el eje de las ordenadas, el número generado. Se observa que no existe una dispersión y los mismos números se repiten de manera constante.

## 15. Fórmulas

### 15.1. Fórmulas empleados para los generadores

Generador Randu:

$$x_{n+1} = (2^{16} + 3)x_n mod(2^{31}) \tag{2}$$

Donde:

- m = 2 elevado a 31
- a = 2 elevado a 16
- c = 3
- n = 0, 1, 2, 3, ...
- X0= cualquier número entre 0 y 2 elevado a (31  1)

Generador GCL Genérico:

$$X_{n+1} = (ax_n + c)mod(m) \tag{3}$$

Donde:

- m es el "módulo", m >0
- a es el multiplicador, 0 a <m
- c es el incremento, c m
- X0 es la semilla, 0 X0 <m

### 15.2. Fórmulas empleadas para los test

#### 15.2.1. Test de Bondad de Chi-Cuadrado:

$$x^{(2)} = \sum_{c}^{i=1} \frac{(o_i - e_i)^2}{e_i} \tag{4}$$

Donde:

- e es el valor esperado
- o es el valor observado en cada intervalo

#### 15.2.2. Test de Póker:

$$x^3 = \sum_{3}^{i=1} \frac{(FO_i - FE_i)^{(2)}}{FE_i} \tag{5}$$

Donde:

- FE es el valor esperado
- FO es la cantidad observada en cada intervalo

#### 15.2.3. Test de Corridas:

Media:

$$\mu_\alpha = \frac{2N - 1}{3} \tag{6}$$

Desviación:

$$(\sigma_\alpha)^2) = \frac{16N - 29}{90} \tag{7}$$

Fórmula Z:

$$Z = \frac{\alpha_1 - \mu_\alpha}{\sigma_\alpha} \tag{8}$$

Donde:

- N es la cantidad de números generados.
- mu es la media.
- sigma es el desvío.
- a1 es la cantidad de sucesiones.

#### 15.2.4. Test de Corridas de Arriba y De Abajo de la Media para Números uniformemente distribuidos:

Media:

$$\mu_b = \frac{2n_1n_2(2n_1n_2 - N)}{N^2(N-1)} \tag{9}$$

Desviación:

$$\sigma_b = \frac{2n_1n_2(2n_1n_2 - N)}{N^2(N-1)} \tag{10}$$

Fórmula Z:

$$Z = \frac{b - \mu_b}{\sigma_b} \tag{11}$$

Donde:

- N es la cantidad de números generados.
- mu es la media.
- sigma es el desvío.
- n1 es la cantidad de números por encima de la media.
- n2 es la cantidad de números por debajo de la media.
- b es la cantidad de sucesiones.

## 16. Conclusiones

Al llevar a cabo un experimento de simulación es necesario asegurar desde un inicio que las distintas variables aleatorias generadas sean uniformes e independientes para no generar efectos secundarios o arrastrar errores y terminar llevando a cabo experimentos sesgados o erróneos. Luego de realizar múltiples simulaciones y al visualizar lo observado, llegamos a la conclusión de que los números generados por el generador GCL randu y Mersenne twister (de la librería random de python) se distribuyen de manera pseudoaleatoria, mientras que la distribución generada por el método de los cuadrados medios no es pseudoaleatoria.

Esto no implica que los números generados sean efectivamente aleatorios , sino que para los tests que nosotros realizamos, se corrobora la aleatoriedad. Sin embargo, no es imposible que si se realizaran más pruebas con distintos test, ellas podrían abordar resultados fallidos.

El generador que implica mayor ineficiencia al momento de generar números aleatorios es el realizado a partir del método de los cuadrados medios, debido a los números que se repiten de manera constante, y los bucles que este genera, tal como un bucle infinito de ceros. Sin embargo ,su creación para dicha época fue más que aceptada, debido a la novedad del algoritmo que planteaba este, permitiendo dar paso a los otros algoritmos que surgieron en la posteridad. Este generador logró pasar los test de corridas debido a que estos contemplan la uniformidad de los números generados, y al repetirse una misma secuencia de números cada cierta cantidad de veces, es indudable que pasaría la prueba. A pesar de que 3 de los generadores que planteamos hayan pasado las 4 pruebas no nos indican que específicamente sean los mejores generadores que se puedan utilizar para fines más prácticos. Sin embargo, podemos afirmar que dichos generadores pueden servir para los fines expuestos en nuestro trabajo, que efectivamente es corroborar la aleatoriedad de los números generados.

## Referencias

[1] Modelos y simulación: http://www2.famaf.unc.edu.ar/~jgimenez/Modelos_y_Simulacion/2013/clase5.pdf

[2] Método de los cuadrados medios: https://es.wikibooks.org/wiki/M%C3%A9todo_de_los_cuadrados_medios_para_la_generaci%C3%B3n_de_n%C3%BAmeros_pseudoaleatorios

[3] Generador lineal congruencial: https://es.wikipedia.org/wiki/Generador_lineal_congruencial

[4] Generador de números pseudoaleatorios: https://es.wikipedia.org/wiki/Generador_de_n%C3%BAmeros_pseudoaleatorios

[5] Número aleatorio: https://es.wikipedia.org/wiki/N%C3%BAmero_aleatorio

[6] Número pseudo-aleatorio: https://es.wikipedia.org/wiki/N%C3%BAmero_pseudoaleatorio

[7] Prueba chi-cuadrado: https://psicologiaymente.com/miscelanea/prueba-chi-cuadrado

[8] Métodos para probar números aleatorios: http://hemaruce.angelfire.com/Unidad_III.pdf