TP 2.1 - GENERADORES PSEUDOALEATORIOS

Abud Santiago Elias
Legajo 47015
sabudvicco@gmail.com

Buchhamer Ariel
Legajo 46217
arielbuchhamer1@outlook.com

Castellano Marcelo
Legajo 39028
marce.geek22@gmail.com

Dolan Guillermo Patricio
Legajo 46101
guillermo230899@gmail.com

Navarro Franco
Legajo 46387
franconavarro1889@gmail.com

17 de mayo de 2022

ABSTRACT

Los números pseudoaleatorios se generan de manera secuencial con un algoritmo determinístico. Construir un buen algoritmo de números pseudoaleatorios es complicado, por eso hemos hecho un estudio sobre cómo funcionan y de la manera que se comportan.

1. Introducción

Los números aleatorios son muy útiles en distintos ámbitos de aplicación, como pueden ser la simulación, el muestreo, el análisis numérico, la programación de computadoras en general, la toma de decisiones, la estética y la recreación **knuth1997seminumerical**. En nuestro caso, resulta particularmente importante su aplicación al campo de la simulación.

Hace muchísimos años, las personas que necesitaban números aleatorios en sus trabajos científicos tenían que recurrir a fuentes "convencionales" de aleatoriedad (bolilleros, dados, cartas, entre otras) **knuth1997seminumerical**. Más tarde se publicaron tablas, como la publicada en 1927 por H. L. C. Tippett, de 40000 dígitos. Posteriormente surgieron dispositivos que generaban números aleatorios de manera mecánica. La computadora Ferranti Mark I, instalada por primera vez en 1951, tenía una instrucción que obtenía 20 bits aleatorios usando el ruido de una resistencia eléctrica. En 1955 la RAND Corporation publicó una ampliamente usada tabla de un millón de dígitos aleatorios obtenidos con la ayuda de otro dispositivo especial. Una famosa máquina de números aleatorios llamada ERNIE (el primer modelo, construido en 1956 **virtualernie**) fue utilizada por muchos años (desde 1957 hasta 1973 **virtualernie**) para seleccionar los números ganadores de la lotería de bonos *Premium Savings Bonds* británica. Esta última fue luego reemplazada por sucesivas versiones, siendo la quinta la más reciente, de 2019, que funciona con tecnología cuántica, empleando luz en lugar de ruido térmico como las anteriores **virtualernie**. Hoy en día otro generador de números aleatorios reales y de acceso al público en general es el servicio de RANDOM.ORG, que se basa el procesamiento de ruido atmosférico captado mediante un arreglo de receptores de radio **randomorgfaq**.

Con la introducción de las computadoras, se comenzó a buscar maneras eficientes de generar números aleatorios dentro de los programas **knuth1997seminumerical**. Las tablas no eran prácticas porque tenían un tamaño finito, requerían memoria y tiempo para cargarlas, además de que había que mantenerlas.[1] Las máquinas como ERNIE podían servir, pero dificultaban probar el software, debido a la no repetibilidad de sus resultados; por otro lado, esa clase de máquinas tendían a sufrir de malfunciones muy difíciles de detectar **knuth1997seminumerical**.

Fue esta insuficiencia de los métodos mecánicos en los primeros días lo que llevó a un interés en producir números aleatorios empleando las operaciones aritméticas ordinarias de una computadora. John von Neumann sugirió por primera vez este enfoque alrededor de 1946, ideando el método de los cuadrados medios, que luego analizaremos en mayor detalle **knuth1997seminumerical**.

[1] Los avances tecnológicos desde la década de 1990 hasta la actualidad las han hecho nuevamente útiles, debido a la posibilidad de almacenar, transportar y transmitir grandes volúmenes de datos con facilidad **knuth1997seminumerical**.

Estos métodos, que parecen generar números aleatorios, pero no lo hacen realmente, se denominan *pseudoaleatorios* o *cuasialeatorios* **knuth1997seminumerical**. Existen diferentes tipos y familias de generadores números pseudoaleatorios, algunos de ellos son el método de los cuadrados medios y los generadores congruenciales lineales (GCL) (y los generadores congruenciales en general). En nuestro estudio nos enfocaremos en el método de los cuadrados medios y los GCL.

2. Descripción del trabajo

Utilizando en Python 3.7, reproducimos y comparamos algoritmos generadores de números pseudoaleatorios, poniendo a prueba la aleatoriedad de cada método por medio de diferentes pruebas.

Los algoritmos generadores de números pseudoaleatorios utilizados fueron:

1. Método de los cuadrados medios, en dos variantes:
   a) 4 dígitos
   b) 10 dígitos (aproximadamente 34 bits como máximo, que da un rango similar a los GCL estudiados)
2. Generador congruencial lineal (GCL), en sus formas:
   a) Parámetros de glibc ($m = 2^{31}, a = 1103515245, c = 12345$)
   b) Parámetros arbitrarios ($m = 2^{31}, a = 1000, c = 151$)
3. Generador del módulo `random` de Python (generador Mersenne Twister (MT19937)

Y las pruebas realizadas, fueron:

1. Test de χ^2
2. Test de póker
3. Test de rachas
4. Test de frecuencia (monobit)

3. Marco teórico

Existen diferentes tipos y familias de generadores de números pseudoaleatorios, algunos de ellos son el método de los cuadrados medios y los generadores congruenciales lineales (y los generadores congruenciales en general).

A grandes rasgos, un generador de números pseudoaleatorios se basa en aplicar una serie de operaciones aritméticas a un estado inicial (la *semilla*) para obtener el siguiente, y así sucesivamente.

Período de un generador Debido a que el estado de un generador es finito, finalmente el mismo se repetirá en algún punto y habrá un ciclo que se repite infinitamente. La cantidad de iteraciones necesaria para que esto ocurra es lo que se denomina el *período del generador*. Según el tipo de generador, sus parámetros y tal vez la semilla empleada puede ser que el mismo sea el máximo soportado por el tamaño de su estado, o que sea menor.[2] Normalmente es deseable que este sea suficientemente grande, idealmente el máximo posible. Sin embargo, vale la pena notar que un período grande no determina que un generador es bueno, debemos verificar que los números que se generan se comportan como si fueran aleatorios.

Recordemos que lo que nos interesa para trabajar con un buen generador de números aleatorios es que la distribución de los números obtenidos tiene que ser uniforme, no deben de haber correlaciones entre los términos de la secuencia, el período debe ser lo más largo posible, y el algoritmo debe ser de ejecución rápida **webfisica2004generacion**.

El problema es saber qué generador de números es mejor, ya que la razón es que si su generador de números aleatorios es bueno, es igualmente probable que aparezca cada posible secuencia de valores **randomorganalysis**. Esto significa que un buen generador de números aleatorios también producirá secuencias que parecen no aleatorias para el ojo humano y que también fallan en cualquier prueba estadística a la que podamos exponerlo.

Es imposible probar definitivamente la aleatoriedad **randomorganalysis**. Una forma de aproximar esto es tomar muchas secuencias de números aleatorios de un generador dado y someterlos a una batería de pruebas estadísticas. A medida

[2]Lo que necesariamente es igual al tamaño de los valores enteros que genera, por ejemplo el generador Mersenne Twister (MT19937), en su versión original, produce enteros de 32 bits, con un estado interno de 624 palabras de 32 bits (19968 bits en total) y tiene un período de $2^{19937} - 1$ **matsumoto1998mersenne**.

que las secuencias pasan más pruebas, aumenta la confianza en la aleatoriedad de los números y también la confianza en el generador. Sin embargo, debido a que esperamos que algunas secuencias no parezcan aleatorias, debemos esperar que algunas de las secuencias fallen al menos en algunas de las pruebas. Sin embargo, si muchas secuencias fallan en las pruebas, deberíamos sospechar.

Hay varias formas de examinar un generador de números aleatorios, las distintas pruebas estadisticas son: análisis visual simple, análisis estadístico de Charmaine Kenny y análisis estadístico de Louise Foley **randomorganalysis**.

3.1. Tipos de generadores

3.1.1. Método de los cuadrados medios

Como se mencionó anteriormente, este método fue desarrollado por Jonh von Neumann, quien sugirió usar las operaciones aritméticas de una computadora para generar secuencias de números pseudoaleatorios. Con este procedimiento se pueden generar números pseudoaleatorios de 4 dígitos de la siguiente forma **ortiz2018numeros**:

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 al inicio).
3. 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.

A inicios de 1950s se exploró el método y se propusieron mejoras, por ejemplo para evitar caer en cero **ortiz2018numeros**. Metrópolis logró obtener una secuencia de 750,000 números distintos al usar semillas de 38 bits (usaba el sistema binario), además la secuencia de Metrópolis mostraba propiedades deseables. No obstante, el método del valor medio no es considerado un buen método por lo común de los ciclos cortos.

3.1.2. Generadores congruenciales lineales

Los generadores congruenciales lineales (GCL) fueron introducidos en 1949 por D.H. Lehmer **ortiz2018numeros**, son muy populares y utilizados incluso hoy en día, en casos donde no se requieran mejores generadores (por ejemplo, no se pueden usar en aplicaciones criptográficas). Tienen la forma:

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

Donde a es el multiplicador, m el módulo, c el incremento y X la semilla.

Los GCLs continúan siendo utilizados en muchas aplicaciones porque con una elección cuidadosa de los parámetros (la elección de los parámetros determina la calidad del generador) pueden pasar muchas pruebas de aleatoriedad, son rápidos y requieren poca memoria **ortiz2018numeros**.[3]

Según los parámetros de este generador, existen distintas variantes, cada una con sus propiedades particulares. Algunas de estas variantes son: la sugerida como ejemplo por ANSI C/ISO 9899:1990 (y 1999, 2011, 2017) ($m = 2^{31}, a = 1103515245, c = 12345$)[4], el implementado por la biblioteca glibc (usado por GCC) que emplea los mismos parámetros que el de ANSI C (sin truncar los bits) **enwiki2022linear**. Otra variante es el infame**knuth1997seminumerical** generador RANDU de Fortran ($m = 2^{31}, a = 65539, c = 0$), que ya no se utiliza.

3.2. Pruebas estadísticas

Las pruebas estadísticas se llevan a cabo mediante la parte de la inferencia estadística conocida como *prueba o test de hipótesis*.

Una **hipótesis estadística**, o simplemente *hipótesis*, es una afirmación acerca del valor de un único parámetro (característica de la población o de una distribución de probabilidad), de los valores de varios parámetros, o de la forma de toda una distribución de probabilidad **devore2015probability**. Cuando se lleva a cabo una prueba de hipótesis, se ponen en consideración dos hipótesis contradictorias. El objetivo es decidir, basándose en la información muestral, cuál de las dos es correcta.

[3] Quizás esto no sea tan relevante hoy en día, pero sí que lo es especialmente en dispositivos muy limitados (como pueden ser los microcontroladores).

[4] Aunque en este caso solamente devuelve los 15 bits más significativos de los valores generados

En la prueba de hipótesis estadísticas, el problema se formula de manera que una de las afirmaciones se vea inicialmente favorecida. Esta afirmación no se rechaza en favor de la alternativa salvo que la evidencia la contradiga y provea un fuerte respaldo por la alternativa **devore2015probability**.

Hipótesis nula y alternativa La **hipótesis nula**, denominada H_0, es la afirmación que inicialmente se asume como verdadera **devore2015probability**. La **hipótesis alternativa**, denominada H_1, es la afirmación contraria a H_0. La hipótesis nula será rechazada en favor de la alternativa solo si la evidencia muestral sugiere que H_0 es falsa. Si la muestra no la contradice significativamente, se puede seguir creyendo en su verosimiltud. Las dos conclusiones posibles son entonces *rechazar* H_0 o *no rechazar* H_0.

Una **prueba de hipótesis** es un método para decidir si la hipótesis nula debería ser rechazada, usando datos de una muestra **devore2015probability**. Ésta debería rechazarse solo si los datos de la muestra sugieren fuertemente que la hipótesis nula no es cierta. En ausencia de tal evidencia, H_0 no debería ser rechazada, puesto que aun es lo bastante plausible.

H_0 se establece generalmente como una afirmación de igualdad. Si θ denota al parámetro de interés, la hipótesis nula tendrá la forma $H_0 : \theta = \theta_0$, donde θ_0 es un número conocido como el *valor nulo* del parámetro (el valor de θ afirmado por la hipótesis nula).

La alternativa a la hipótesis nula $H_0 : \theta = \theta_0$ tendrá una de las formas siguientes:

1. $H_1 : \theta > \theta_0$ la hipótesis nula implícita es $\theta \leq \theta_0$,
2. $H_1 : \theta < \theta_0$ la hipótesis nula implícita es $\theta \geq \theta_0$) o
3. $H_1 : \theta \neq \theta_0$

Estadístico de prueba Un estadístico de prueba es una función sobre los datos de la muestra que se utiliza como base para decidir si se rechaza H_0 **devore2015probability**. El estadístico seleccionado debe distinguir efectivamente las dos hipótesis. Es decir, los valores del estadístico usualmente observados cuando H_0 es verdadera deberían ser muy diferentes de los típicamente observados cuando no lo es.

Valor p El *valor p* o *p-value* es la probabilidad, calculada asumiendo que la hipótesis nula es verdadera, de obtener un valor del estadístico de prueba al menos tan contradictorio con H_0 como el calculado con los datos disponibles de la muestra **devore2015probability**. En un análisis de prueba de hipótesis se arriba a una conclusión seleccionando un número α, denominado el nivel de significancia de la prueba, que es razonablemente cercano a 0. Entonces H_0 será rechazada en favor de H_1 si valor p $\leq \alpha$, mientras que H_0 no será rechazada (todavía considerada verosímil) si valor p $< \alpha$. Los niveles de significancia más usados en la práctica son (en orden) $\alpha = 0{,}05; 0{,}01; 0{,}001;$ y $0{,}10$.

La distribución del valor p Cuando se repite un experimento varias veces o se obtienen varias muestras de una misma población, al realizar pruebas de hipótesis, se obtiene un valor p para cada repetición del experimento. Como el valor p está basado en el análisis de variables aleatorias, es también una variable aleatoria, que para estadísticos de prueba continuos se distribuye de manera uniforme sobre el intervalo [0, 1] cuando la hipótesis nula es verdadera, sin importar el tamaño de la muestra **hung1997behavior**. En contraste, la distribución del valor p bajo la hipótesis alternativa es una función del tamaño de la muestra y el valor verdadero o el rango de valores verdaderos del parámetro bajo prueba. Para el caso de la prueba χ^2 de Pearson, aunque la distribución del estadístico es discreta, el valor p se calcula basado en su distribución χ^2 asintótica **wang2019p**. Por lo tanto, el valor p se distribuye de manera prácticamente uniforme cuando el tamaño de la muestra es lo suficientemente grande.

3.2.1. Prueba χ^2 de Pearson

La prueba χ^2 de Pearson se considera una prueba no paramétrica que mide la discrepancia entre una distribución observada y otra teórica (bondad de ajuste), indicando en qué medida las diferencias existentes entre ambas, de haberlas, se deben al azar en el contraste de hipótesis **eswiki2022pearson**.

Pone a prueba una hipótesis nula que establece que la distribución de frecuencia de ciertos eventos obervados en una muestra es consistente con una distribución teórica en particular. Los eventos considerados deben ser mutuamente excluyentes y tener una probabilidad total igual a 1. Un caso común para esto es donde cada uno de los eventos cubren un resultado de una variable categórica **enwiki2022pearson**.

La prueba χ^2 de Pearson se usa para realizar distintos tipos de contrastes, como bondad de ajuste, homogeneidad, e independencia **enwiki2022pearson**. Para nuestro estudio, como los valores generados son números enteros y deberían estar distribuidos de manera uniforme, llevamos a cabo pruebas de bondad de ajuste a la distribución uniforme discreta.

Una prueba de bondad de ajuste determina si una distribución de frecuencia observada difiere de una distribución teórica dada.

Planteamos el test de hipótesis siguiente, la hipótesis nula (H_0) de que los datos siguen la distribución esperada y la alternativa (H_1), de que los datos no siguen la misma:

$$H_0 : \text{no hay diferencia entre las distribuciones}$$
$$H_1 : \text{hay diferencia entre las distribuciones}$$

El procedimiento de cálculo de la prueba de χ^2 para bondad de ajuste incluye los pasos siguientes **enwiki2022pearson**:

1. Calcular el estádistico χ^2, que se asemeja a una suma normalizada de los desvíos al cuadrado entre las frecuencias observadas y las teóricas (ecuación 2).
2. Determinar los grados de libertad, **gl**, del estadístico: para nuestro caso, de bondad de ajuste, gl = $n - m$, donde n es el número de valores distintos de la distribución, y m es el número de parámetros ajustados para hacer que la distribución se ajuste mejor a las observaciones: el número de valores reducidos por el número de parámetros ajustados en la distribución.
3. Seleccionar el nivel deseado de confianza (nivel de significancia o el nivel alfa correspondiente) para el resultado de la prueba. Usualmente y para nuestro caso seleccionamos un alfa de 0.05, lo que corresponde a un 95 % de confianza.
4. Comparar χ^2 con una distribución χ^2 con gl grados de libertad para obtener el valor p correspondiente y emplear y el nivel de confianza seleccionado (de un solo lado, puesto que la prueba es solamente en una dirección, esto es, ¿es el valor del estadístico de prueba mayor que valor el crítico? o ¿es el valor p menor o igual al alfa?), lo que en muchos casos da una buena aproximación de la distribución χ^2.
5. Rechazar o mantener la hipótesis nula de que la distribución de frecuencias observadas es la misma que la teórica empleando el valor p correspondiente. Si el valor p es menor o igual que el nivel alfa seleccionado, se rechaza la hipótesis nula (H_0) y se acepta la alternativa (H_1), con el nivel de confianza seleccionado. Si en cambio el valor p supera dicho umbral, no se puede llegar a una conclusión clara, y se mantiene la hipótesis nula (no la podemos rechazar), lo que no significa necesariamente que la misma sea aceptada.

Prueba de bondad de ajuste - distribución discreta uniforme En este caso se dividen N observaciones entre n valores **enwiki2022pearson**. Una aplicación simple es probar la hipótesis de que, en la población general, los distintos valores se producirán con la misma frecuencia. La frecuencia teórica absoluta para cualquier valor (bajo la hipótesis nula de una distribucíón discreta uniforme) se calcula como

$$E_i = \frac{N}{n},$$

y la reducción en los grados de libertad es $p = 1$, dado que las frecuencias observadas O_i deben cumplir con la restricción de sumar N. Esto es, los grados de libertad resultan ser gl = $n - 1$

El valor del estadístico de prueba es

$$\chi^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i} = N \sum_{i=1}^{n} \frac{(O_i/N - p_i)^2}{p_i} \tag{2}$$

donde

- χ^2: estadístico acumulativo de prueba de Pearson, que se aproxima asintóticamente a una distribución χ^2
- O_i: número de observaciones de tipo i
- $E_i = Np_i$: la cantidad esperada (teórica) de observaciones de tipo i, afirmada por la hipótesis nula de que la fracción de observaciones de tipo i en la población es p_i

Finalmente, el estadístico χ^2 puede entonces emplearse para calcular un valor p comparando el valor del estadístico con una distribución χ^2 **enwiki2022pearson**. Esto es

$$\text{valor p} = 1 - P(X \leq \chi^2) \tag{3}$$

Como ya se ha mencionado, si el valor p resulta menor o igual que el nivel de significancia $\alpha = 0{,}05$, rechazamos la hipótesis nula y concluimos con un 95 % de confianza que los valores generados difieren de manera estadísticamente significativa de una distribución discreta uniforme y por lo tanto no son aleatorios, en caso contrario, concluimos que no se puede rechazar la hipótesis nula y por lo tanto los valores pueden ser aleatorios.

Implementación En nuestro trabajo empleamos la función `scipy.stats.chisquare`, del paquete SciPy **scipy2021chisquare**, que se encarga tanto de calcular el estadístico como el valor p correspondiente y ya tiene en cuenta por defecto la reducción en los grados de libertad $p = 1$.

3.2.2. Prueba de Póker

Esta prueba examina en forma individual los dígitos del número pseudoaleatorio generado. La forma como esta prueba se realiza es tomando 5 dígitos a al vez y clasificándolos como : Par, dos pares, tercia, póker, full, quintilla, y todos diferentes.

Las probabilidades para cada una de las manos del póker diferentes se muestran enseguida:

1. Todos diferentes = 0.3024
2. Un par = 0.504
3. Dos pares = 0.108
4. Tercia = 0.072
5. Póker = 0.009
6. Full = 0.0045
7. Quintilla = 0.0001

Con las probabilidades anteriores y con el número de números pseudoaleatorios generados, se puede calcular la frecuencia esperada de cada posible resultado, la cual al compararse con la frecuencia observada, produce el estadístico:

$X_0^7 = \sum_{i=1}^{7} \frac{(FO_i - FE_i)^2}{FE_i}$

Si $X_0^2 < X_{\alpha,6}^2$, entonces los números pasan la prueba, y no podemos rechazar la hipótesis de que los números fueron generados aleatoriamente e independientemente.

Implementación En este trabajo utilizamos 5 números de cada muestra y los 7 casos descriptos más arriba.

Usamos una significación del 5 %, y revisamos las cantidades obtenidas para acumular o ignorar las que no sean suficientes.

3.2.3. Prueba de rachas

El contraste de rachas permite verificar la hipótesis nula de que la muestra es aleatoria, es decir, si las sucesivas observaciones son independientes. Este contraste se basa en el número de rachas que presenta una muestra. Una racha se define como una secuencia de valores muestrales con una característica común precedida y seguida por valores que no presentan esa característica. Así, se considera una racha la secuencia de k valores consecutivos superiores o iguales a la media muestral (o a la mediana o a la moda, o a cualquier otro valor de corte) siempre que estén precedidos y seguidos por valores inferiores a la media muestral (o a la mediana o a la moda, o a cualquier otro valor de corte).

El número total de rachas en una muestra proporciona un indicio de si hay o no aleatoriedad en la muestra. Un número reducido de rachas (el caso extremo es 2) es indicio de que las observaciones no se han extraído de forma aleatoria, los elementos de la primera racha proceden de una población con una determinada característica (valores mayores o menores al punto de corte) mientras que los de la segunda proceden de otra población. De forma idéntica un número excesivo de rachas puede ser también indicio de no aleatoriedad de la muestra.

Si la muestra es suficientemente grande y la hipótesis de aleatoriedad es cierta, la distribución muestral del número de rachas, R, puede aproximarse mediante una distribución normal de parámetros μ_R y σ_R:

$\mu_R = \frac{2n_1 n_2}{n} + 1$

$\sigma_R = \sqrt{\frac{2n_1 n_2 (2n_1 n_2 - n)}{n^2 (n-1)}}$

donde n1 es el número de elementos de una clase, n2 es el número de elementos de la otra clase y n es el número total de observaciones.

Tras esto, si estandarizamos el valor:

$Z = \frac{R - \mu_R}{\sigma_R}$

podemos usar la función de distribución acumulada para obtener el valor p.

Implementación La función que implementa el test de rachas realiza exactamente este algoritmo, utilizando la librería st incluída en Python para calcular la mediana de la tirada de números aleatorios y el módulo stat de scipy para evaluar el valor z con la función de distribución normal.

3.2.4. Prueba monobit

Esta prueba se concentra en la proporción de zeros y unos de una secuencia entera. El propósito es encontrar si el número de unos y zeros en una secuencia es aproximadamente el mismo, como se esperaría de una verdadera secuencia aleatoria.

Para ellos calculamos la siguiente prueba estadística:

$S_{obs} = \frac{|n_1 - n_0|}{\sqrt{n}}$

Donde n_1 es la cantidad de 1s, n_0 es la cantidad de 0s, y n es la cantidad total de bits.

Luego calculamos el valor p con la función error complementario:

$p = 1 - \frac{2}{\sqrt{\pi}} \int_0^{S_{obs}/\sqrt{2}} e^{-t^2}\, dt$

Implementación Para la implementación de esta prueba, tras transformar los números generados en bits y contar las cantidades, utilizamos varias funciones del módulo Math de Python para poder conseguir el valor p.

4. Análisis de resultados

Salida del programa:

Generadores pseudoaleatorios	rachas	chi²	poker	monobit
GCL ANSI C	QUIZÁS	QUIZÁS	QUIZÁS	QUIZÁS
GCL m=2³² a=10³ c=151	NO	QUIZÁS	NO	NO
MT19937 de Python	QUIZÁS	QUIZÁS	QUIZÁS	QUIZÁS
Cuadrados Medios (4 dígitos)	QUIZÁS	NO	NO	NO
Cuadrados Medios (10 dígitos)	QUIZÁS	QUIZÁS	QUIZÁS	NO

Donde "NO" significa que los datos presentan características que no poseen los generadore ideales de números aleatorios, y "QUIZÁS" significa que estas características no se encuentran (esto no es suficiente para declarar la aleatoriedad del método).

5. Conclusiones

Con las gráficas obtenidas y las pruebas realizadas hemos descubierto que tanto el generador congruencial lineal (parámetros glibc) y el generador Mersenne Twister (MT19937) de Python parecen ser generadores bastante buenos, mientras que el GCL con valores arbitrarios y el generador por el método de los cuadrados medios no resultan buenos generadores. En cuanto al método de los cuadrados medios, es interesante mencionar que, trabajando con 10 dígitos, las gráficas obtenidas pueden mostrar una aparente aleatoriedad, sin embargo es durante las pruebas estadísticas donde pueden notarse sus puntos débiles. Como conclusión final, es claro que es necesario realizar más pruebas, tal vez con distintos niveles de significancia, para poder extraer conclusiones más exactas sobre la aleatoriedad de los generadores.