

Generación de números pseudoaleatorios mediante el método de los cuadrados medios

AUTORES/AS

Luis Ríos

Sebastian Zabala

FECHA DE PUBLICACIÓN

4 de noviembre de 2025

Importancia de los números pseudoaleatorios

Los números pseudoaleatorios son fundamentales en diversas áreas de la informática y las matemáticas, incluyendo simulaciones, criptografía, juegos y algoritmos de optimización. A diferencia de los números verdaderamente aleatorios, que se generan a partir de fenómenos físicos impredecibles, los números pseudoaleatorios se generan mediante algoritmos deterministas que producen secuencias de números que imitan las propiedades estadísticas de los números aleatorios.

Métodos comunes para generar números pseudoaleatorios

Existen varios métodos para generar números pseudoaleatorios, entre los cuales destacan:

1. **Generadores Congruenciales Lineales (LCG)**: Este es uno de los métodos más simples y ampliamente utilizados. Utiliza una fórmula matemática para generar una secuencia de números basada en una semilla inicial. La fórmula general es:

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

donde (X) es el número pseudoaleatorio, (a), (c) y (m) son constantes, y (X_0) es la semilla inicial.

2. **Generadores de Mersenne Twister**: Este es un generador de números pseudoaleatorios más avanzado que ofrece una mayor calidad y un período más largo que los LCG. Es ampliamente utilizado en aplicaciones científicas y de simulación.
3. **Generadores basados en hardware**: Algunos sistemas utilizan fuentes de entropía física, como ruido térmico o movimientos del mouse, para generar números aleatorios. Aunque estos no son estrictamente pseudoaleatorios, a menudo se combinan con métodos algorítmicos para mejorar la calidad de los números generados.

Método de los Cuadrados Medios

Uno de los métodos clásicos para generar números pseudoaleatorios es el **método de los cuadrados medios**. Siendo uno de los primeros métodos propuestos históricamente para la generación de números pseudoaleatorios, introducido por John von Neumann en 1946. A pesar de que hoy en día se considera un método con limitaciones importantes en cuanto a la calidad de las secuencias que produce, resulta útil para fines didácticos, pues permite comprender de forma clara cómo un algoritmo determinista puede generar una secuencia de valores a partir de una semilla inicial.

Pasos para aplicar el método de los cuadrados medios

Este método implica los siguientes pasos:

- i. Se escoge un número de cuatro dígitos x_0 (semilla).
- ii. Se eleva al cuadrado (x_0^2) y se toman los cuatro dígitos centrales (x_1).
- iii. Se genera el el número pseudo-aleatorio como

$$u_1 = \frac{x_1}{10^4}$$

- iv. Volver al paso ii y repetir el proceso.

Para obtener los k (número par) dígitos centrales de x_i^2 se puede utilizar que:

$$x_{i+1} = \left\lfloor \left(x_i^2 - \left\lfloor \frac{x_i^2}{10^{(2k-\frac{k}{2})}} \right\rfloor 10^{(2k-\frac{k}{2})} \right) / 10^{\frac{k}{2}} \right\rfloor$$

Implementación del método de los cuadrados medios en R

Ejemplo con semilla 1234 y 100 valores utilizando el paquete `simres`

Para exemplificar el funcionamiento del método de los cuadrados medios, se generarán 100 números pseudoaleatorios utilizando una semilla inicial de 1234 ($x_0 = 1234$) con $k = 4$ dígitos centrales. Todo esto usando la función `simres::rvng()` del paquete `simres`.

```
library(simres)
simres::rvng(100, 1234)
```

```
[1] 0.5227 0.3215 0.3362 0.3030 0.1809 0.2724 0.4201 0.6484 0.0422 0.1780
[11] 0.1684 0.8358 0.8561 0.2907 0.4506 0.3040 0.2416 0.8370 0.0569 0.3237
[21] 0.4781 0.8579 0.5992 0.9040 0.7216 0.0706 0.4984 0.8402 0.5936 0.2360
[31] 0.5696 0.4444 0.7491 0.1150 0.3225 0.4006 0.0480 0.2304 0.3084 0.5110
[41] 0.1121 0.2566 0.5843 0.1406 0.9768 0.4138 0.1230 0.5129 0.3066 0.4003
[51] 0.0240 0.0576 0.3317 0.0024 0.0005 0.0000 0.0000 0.0000 0.0000 0.0000
[61] 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
[71] 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
[81] 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
[91] 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
```

Como se puede observar, llegado un punto de la secuencia, los números generados son todos ceros (0). Relevando así uno de los principales problemas del método de cuadrados medios, **la absorción**. El problema es que si en algún paso se obtiene un número pequeño (pocos dígitos), al llenar con ceros y tomar los dígitos centrales se pueden sacar muchos ceros, llegando así a cero (0). Y si $x_n = 0$, entonces $x_{n+1} = 0$ para siempre.

Advertencia

Advertencia: El método de los cuadrados medios puede degenerar rápidamente al valor 0.

A continuación se muestran las operaciones específicas que provocan la caída de la secuencia hacia el cero. Consideraremos el punto de la secuencia en el que comienzan a aparecer valores pequeños, lo cual desencadena la absorción:

$$x_{53} = 3317$$

1. Primer paso

$$x_{53}^2 = 3317^2 = 11002489$$

Tomando los dígitos centrales:

$$11002489 \Rightarrow \boxed{0024}$$

$$x_{54} = 0024 = 24$$

2. Segundo paso

$$x_{54}^2 = 24^2 = 576$$

Tomando los dígitos centrales:

$$00000576 \Rightarrow \boxed{0005}$$

$$x_{55} = 0005 = 5$$

3. Tercer paso

$$x_{55}^2 = 5^2 = 25$$

Tomando los dígitos centrales:

$$00000025 \Rightarrow \boxed{0000}$$

$$x_{56} = 0000 = 0$$

A partir de aquí, la secuencia queda atrapada en un **estado absorbente**:

$$x_{57} = 0 \Rightarrow x_{58} = 0 \Rightarrow x_{59} = 0 \Rightarrow \dots$$

Lo que confirma que, una vez que el método alcanza (0), no es posible recuperar valores distintos de cero, perdiéndose completamente la utilidad del generador.

Ejemplo con semilla 1234 y 100 valores utilizando función propia

A continuación, se presenta una implementación propia del método de los cuadrados medios en R para generar 100 números pseudoaleatorios a partir de una semilla inicial de 1234.

```
cuadrados_medios <- function(n,
                                seed = as.numeric(Sys.time()),
                                k = 4,
                                save_seed = TRUE,
                                warn = TRUE) {
  # n: cantidad de números a generar
  # seed: semilla inicial (entero o numérico). Se truncará y reducirá modulo 10^k.
  # k: número de dígitos centrales (debe ser par)
  # save_seed: si TRUE guarda .rng en globalenv() (semilla final y parámetros)
  # warn: si TRUE muestra advertencias sobre precisión

  if (!is.numeric(n) || n <= 0) stop("n debe ser un entero positivo")
```

```

if ((k %% 2) != 0) stop("k debe ser par")
if (!is.numeric(seed)) seed <- as.numeric(seed)

# normalizar semilla: usar solo sus últimos k dígitos
seed <- floor(seed)
seed <- seed %% (10^k)

# parámetros para extracción por módulo (idénticos al fragmento de simres)
halfk <- k / 2
aux   <- 10^(2*k - halfk) # = 10^(2k - k/2) (entero si k par)
aux2  <- 10^halfk          # = 10^(k/2)

# chequeo de precisión: si z = seed^2 puede exceder la precisión entera segura
# doble precisión en R mantiene enteros exactos hasta ~2^53 (~9e15).
if (warn) {
  max_safe <- 2^53
  # valor máximo de z aproximado: (10^k - 1)^2 ≈ 10^(2k)
  approx_z_max <- (10^k)
  if (10^(2*k) > max_safe) {
    warning(sprintf("Para k = %d la extracción podría perder precisión en z = x^2 (recomenda"))
  }
}

u <- numeric(n)

for (i in seq_len(n)) {
  z <- seed * seed
  # implementación equivalente a:
  # seed <- trunc((z - trunc(z/aux)*aux)/aux2)
  # pero usando operaciones mod y floor:
  # extraer (z mod aux) then dividir por aux2
  seed <- floor((z %% aux) / aux2)
  u[i] <- seed / (10^k)
}

# guardar semilla final y parámetros en el entorno global (similar a simres)
if (isTRUE(save_seed)) {
  assign(".rng",
        list(seed = seed, type = "vm", parameters = list(k = k)),
        envir = globalenv())
}

return(u)
}

```

Generar 100 números pseudoaleatorios con semilla 1234

numeros_pseudoaleatorios <- cuadrados_medios(100, 1234, k = 4)

numeros_pseudoaleatorios

```

[1] 0.5227 0.3215 0.3362 0.3030 0.1809 0.2724 0.4201 0.6484 0.0422 0.1780
[11] 0.1684 0.8358 0.8561 0.2907 0.4506 0.3040 0.2416 0.8370 0.0569 0.3237
[21] 0.4781 0.8579 0.5992 0.9040 0.7216 0.0706 0.4984 0.8402 0.5936 0.2360
[31] 0.5696 0.4444 0.7491 0.1150 0.3225 0.4006 0.0480 0.2304 0.3084 0.5110

```

```
[41] 0.1121 0.2566 0.5843 0.1406 0.9768 0.4138 0.1230 0.5129 0.3066 0.4003
[51] 0.0240 0.0576 0.3317 0.0024 0.0005 0.0000 0.0000 0.0000 0.0000 0.0000
[61] 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
[71] 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
[81] 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
[91] 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
```

Del mismo modo, se puede observar que la secuencia generada por esta función también cae en el estado absorbente de cero (0) después de varios pasos, confirmando nuevamente la limitación del método de los cuadrados medios.

Análisis de la calidad del generador por el método de los cuadrados medios para una secuencia

Para evaluar la calidad del generador de números pseudoaleatorios basado en el método de los cuadrados medios, se pueden utilizar diversas pruebas estadísticas. A continuación, se presentan algunas de las pruebas más comunes:

Propósito del Análisis	Prueba o Gráfico Utilizado
Uniformidad	Prueba de Chi-cuadrado
	Prueba de Kolmogorov-Smirnov
	Histograma
Independencia	Gráfico Secuencial
	Gráfico de Dispersión Retardado
	Correlograma (Función de Autocorrelación)
	Prueba de Ljung-Box

Implementación de pruebas estadísticas en R

Como se pudo observar, el ejemplo anterior con el método de los cuadrados medios mostró una clara limitación al caer en un estado absorbente. Por este motivo, se utilizará otra semilla inicial (x_0) y un valor de k mayor a 4, esto garantizará que por lo menos podamos generar 100 números pseudoaleatorios. Si bien esto no soluciona el problema de la absorción, si que retrasa su aparición. Por último, se implementarán las pruebas estadísticas mencionadas en [R](#) para evaluar la calidad del generador.

A continuación, se generarán 100 números pseudoaleatorios utilizando el paquete [simres](#), una semilla inicial de 123456 y $k = 6$ dígitos centrales.

```
numeros_generados <- simres::rvng(100, 123456, k=6)
numeros_generados
```

```
[1] 0.241383 0.265752 0.624125 0.532015 0.039960 0.596801 0.171433 0.389273
[9] 0.533468 0.588107 0.869843 0.626844 0.933400 0.235560 0.488513 0.644951
```

```
[17] 0.961792 0.043851 0.922910 0.762868 0.967585 0.220732 0.722615 0.172438
[25] 0.734863 0.023628 0.558282 0.678791 0.757221 0.383642 0.181184 0.827641
[33] 0.989624 0.355661 0.494746 0.773604 0.463148 0.506069 0.105832 0.200412
[41] 0.164969 0.214770 0.126152 0.914327 0.993862 0.761675 0.148805 0.142928
[49] 0.428413 0.537698 0.119139 0.194101 0.675198 0.892339 0.268890 0.301832
[57] 0.102556 0.517733 0.047459 0.252356 0.683550 0.240602 0.889322 0.893619
[65] 0.554917 0.932876 0.257631 0.373732 0.675607 0.444818 0.863053 0.860480
[73] 0.425830 0.331188 0.685491 0.897911 0.244163 0.615570 0.926424 0.261427
[81] 0.344076 0.388293 0.771453 0.139731 0.524752 0.364661 0.977644 0.787790
[89] 0.613084 0.871991 0.368304 0.647836 0.691482 0.147356 0.713790 0.496164
[97] 0.178714 0.938693 0.144548 0.894124
```

Como se puede observar, no se tienen números pseudoaleatorios iguales a cero (0) en esta secuencia generada con una semilla mayor y $k = 6$ dígitos centrales.

Estos mismos resultados se obtienen utilizando una función propia implementada en R:

```
# Generar 100 números pseudoaleatorios con semilla 1234
numeros_pseudoaleatorios2 <- cuadrados_medios(100, 123456, k = 6)
numeros_pseudoaleatorios2
```

```
[1] 0.241383 0.265752 0.624125 0.532015 0.039960 0.596801 0.171433 0.389273
[9] 0.533468 0.588107 0.869843 0.626844 0.933400 0.235560 0.488513 0.644951
[17] 0.961792 0.043851 0.922910 0.762868 0.967585 0.220732 0.722615 0.172438
[25] 0.734863 0.023628 0.558282 0.678791 0.757221 0.383642 0.181184 0.827641
[33] 0.989624 0.355661 0.494746 0.773604 0.463148 0.506069 0.105832 0.200412
[41] 0.164969 0.214770 0.126152 0.914327 0.993862 0.761675 0.148805 0.142928
[49] 0.428413 0.537698 0.119139 0.194101 0.675198 0.892339 0.268890 0.301832
[57] 0.102556 0.517733 0.047459 0.252356 0.683550 0.240602 0.889322 0.893619
[65] 0.554917 0.932876 0.257631 0.373732 0.675607 0.444818 0.863053 0.860480
[73] 0.425830 0.331188 0.685491 0.897911 0.244163 0.615570 0.926424 0.261427
[81] 0.344076 0.388293 0.771453 0.139731 0.524752 0.364661 0.977644 0.787790
[89] 0.613084 0.871991 0.368304 0.647836 0.691482 0.147356 0.713790 0.496164
[97] 0.178714 0.938693 0.144548 0.894124
```

Como se puede observar, la secuencia generada por esta función propia tampoco contiene ceros (0) en los primeros 100 números generados.

A continuación, se aplicarán las pruebas de uniformidad e independencia.

Análisis de Uniformidad

La prueba de uniformidad evalúa si los números generados se distribuyen de manera uniforme en el intervalo $[0, 1)$.

Prueba de Chi-cuadrado

Esta prueba divide el intervalo $[0, 1)$ en subintervalos y compara las frecuencias observadas de números en cada uno con las frecuencias esperadas. La función `simres::freq.test()` es una implementación directa de este contraste para una distribución uniforme.

```
# Aplicamos el test de Chi-cuadrado
simres::freq.test(numeros_generados, nclass = 10)
```

```
Chi-squared test for given probabilities
```

```
data: numeros_generados
X-squared = 8, df = 9, p-value = 0.5341
```

El resultado de la prueba de Chi-cuadrado indica si se puede rechazar la hipótesis nula de que los números generados siguen una distribución uniforme. Un valor p alto sugiere que no hay evidencia suficiente para rechazar la hipótesis nula, indicando una buena uniformidad en la distribución de los números generados.

Prueba de Kolmogorov-Smirnov

La prueba de Kolmogorov-Smirnov compara la distribución empírica de los números generados con la distribución teórica uniforme. La función `ks.test()` en R se utiliza para realizar esta prueba.

```
# Aplicamos el test de Kolmogorov-Smirnov
ks.test(numeros_generados, "punif", 0, 1)
```

```
Asymptotic one-sample Kolmogorov-Smirnov test
```

```
data: numeros_generados
D = 0.062556, p-value = 0.8287
alternative hypothesis: two-sided
```

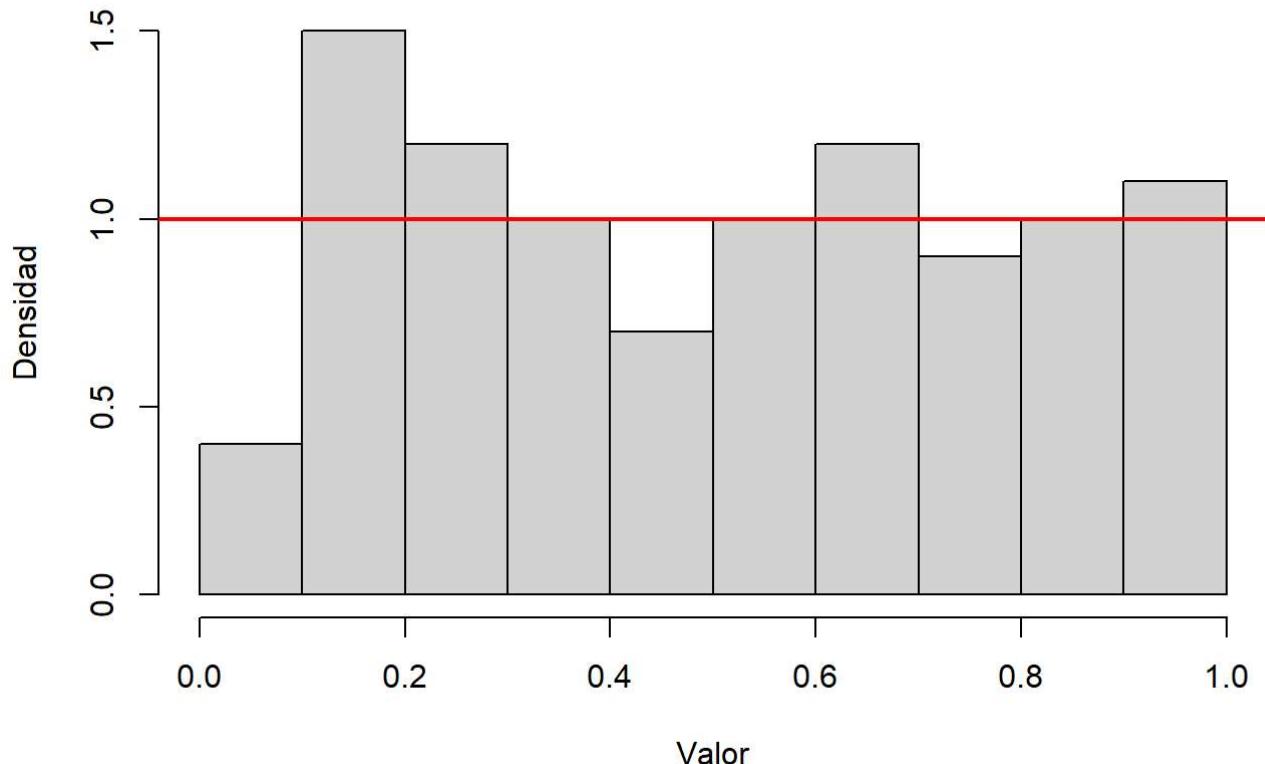
El resultado de la prueba de Kolmogorov-Smirnov proporciona un valor p que indica si se puede rechazar la hipótesis nula de que los números generados siguen una distribución uniforme. Un valor p alto sugiere que no hay evidencia suficiente para rechazar la hipótesis nula, indicando una buena uniformidad en la distribución de los números generados.

Análisis Gráfico: Histograma

Un histograma es una representación gráfica que muestra la distribución de los números generados. Si los números son uniformemente distribuidos, el histograma debería mostrar barras de alturas similares en todo el rango.

```
# Histograma de la secuencia generada
hist(numeros_generados, freq = FALSE, main = "Histograma de Números Pseudoaleatorios",
      xlab = "Valor", ylab = "Densidad", breaks = 10)
abline(h = 1, col = "red", lwd = 2) # Línea de la densidad uniforme teórica
```

Histograma de Números Pseudoaleatorios



El histograma muestra la distribución de los números pseudoaleatorios generados. La línea roja representa la densidad uniforme teórica. Si las barras del histograma están cerca de esta línea, indica que los números generados están distribuidos uniformemente.

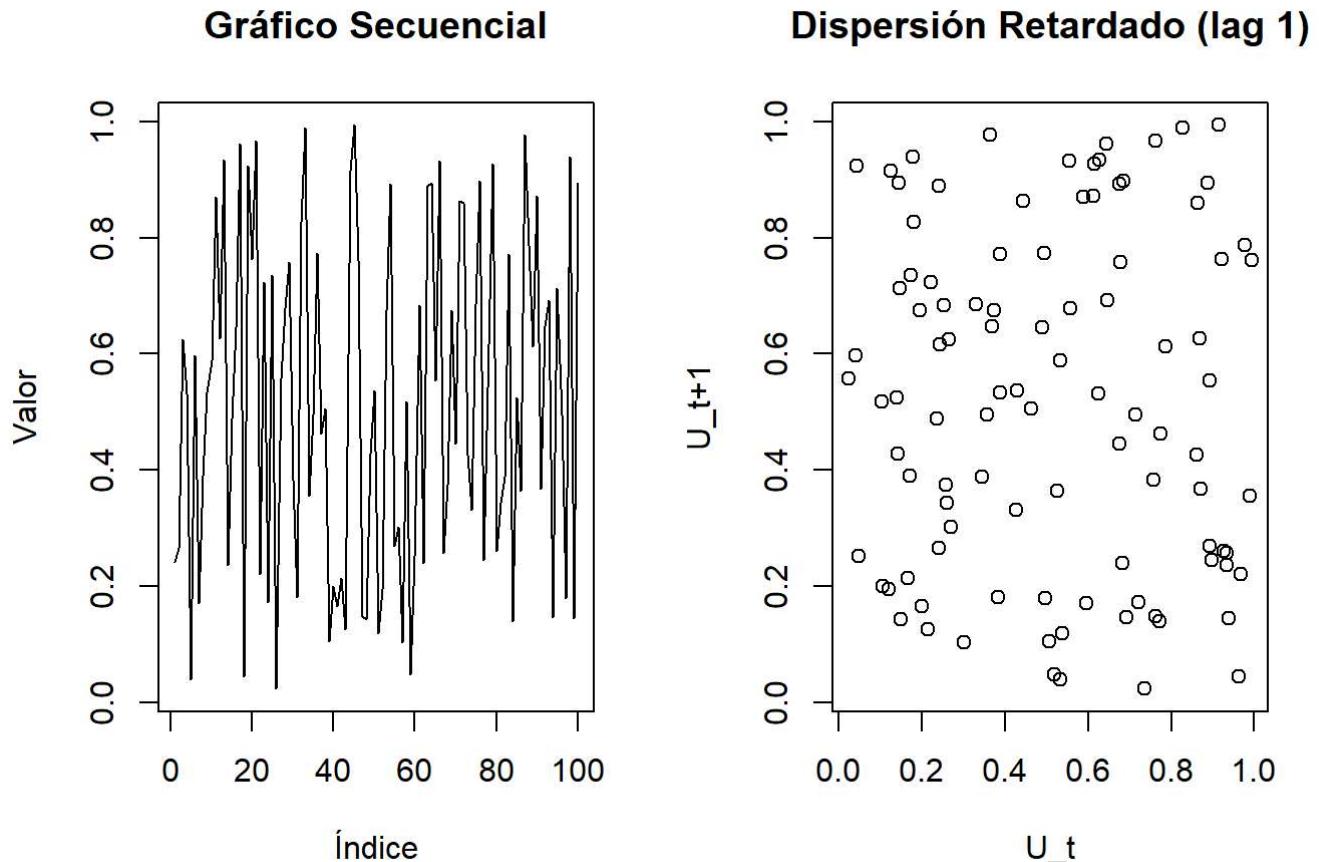
Análisis de Independencia

Estas pruebas verifican si los números en la secuencia son independientes entre sí, un requisito fundamental para la aleatoriedad.

Gráfico Secuencial y de Dispersión Retardado

Estos gráficos son herramientas visuales para detectar patrones. El gráfico secuencial muestra los valores en el orden en que fueron generados, mientras que el de dispersión retardado compara cada valor con el siguiente

```
# Gráfico secuencial y de dispersión retardado
par(mfrow = c(1, 2))
plot(as.ts(numeros_generados), main = "Gráfico Secuencial", xlab = "Índice", ylab = "Valor")
plot(numeros_generados[-100], numeros_generados[-1],
     main = "Dispersión Retardado (lag 1)", xlab = "U_t", ylab = "U_{t+1}")
```



```
par(mfrow = c(1, 1))
```

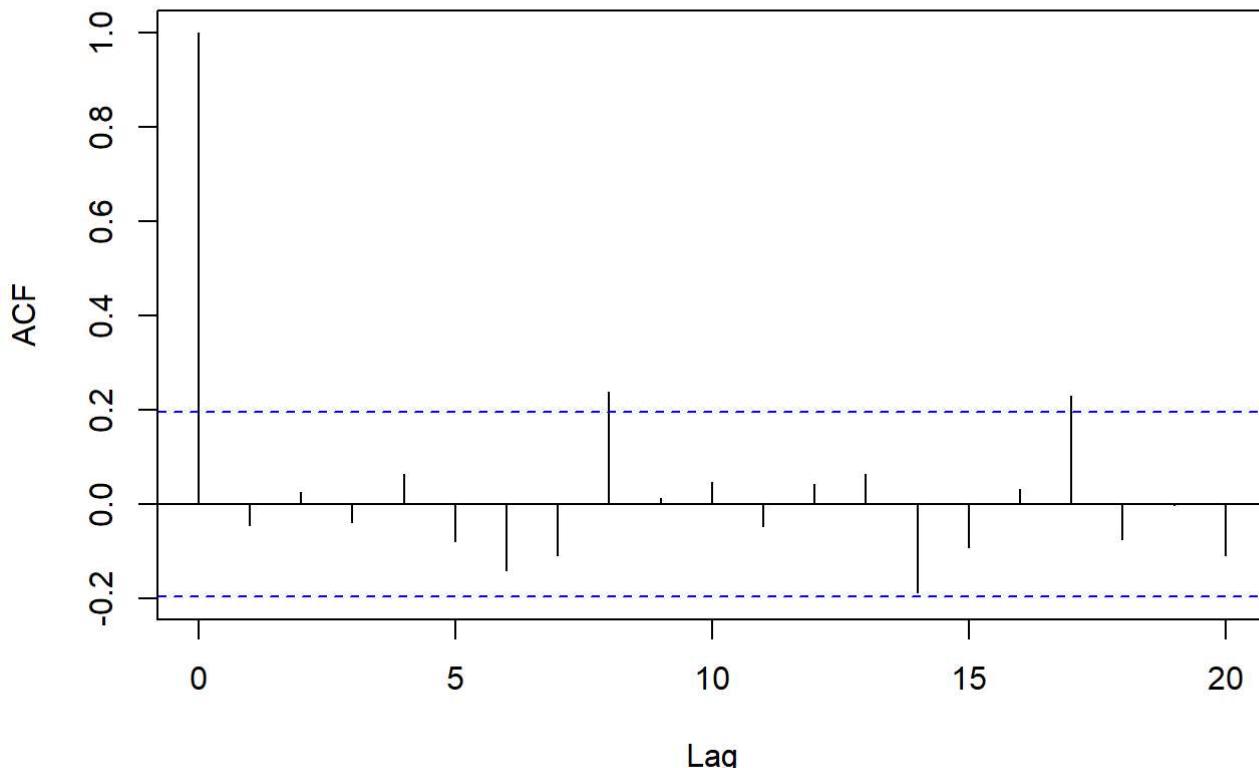
El gráfico secuencial muestra la evolución de los números generados a lo largo del tiempo. Si no hay patrones evidentes, sugiere independencia. El gráfico de dispersión retardado (lag 1) compara cada número con el siguiente. Si los puntos están dispersos sin formar patrones claros, indica independencia entre los números consecutivos.

Prueba de Autocorrelación (Correlograma y Test de Ljung-Box)

El correlograma (página 36) muestra las autocorrelaciones para diferentes retardos (lags). Para una secuencia independiente, estas autocorrelaciones deberían estar dentro de los límites de confianza (líneas azules). El test de Ljung-Box es la prueba formal asociada.

```
# Correlograma
acf(numeros_generados, main = "Función de Autocorrelación")
```

Función de Autocorrelación



El corregograma muestra las autocorrelaciones de la secuencia de números generados para diferentes retardos (lags). Si las barras están dentro de los límites de confianza (líneas azules), sugiere que no hay autocorrelación significativa, indicando independencia entre los números generados.

```
# Test de Ljung-Box
Box.test(numeros_generados, lag = 10, type = "Ljung-Box")
```

Box-Ljung test

```
data: numeros_generados
X-squared = 11.542, df = 10, p-value = 0.3168
```

El resultado del test de Ljung-Box proporciona un valor p que indica si se puede rechazar la hipótesis nula de independencia. Un valor p alto sugiere que no hay evidencia suficiente para rechazar la hipótesis nula, indicando independencia entre los números generados.

Comportamiento Estadístico General y Limitaciones del Período

Aunque esta secuencia particular de 100 números pasa las pruebas de uniformidad e independencia, es crucial recordar la principal debilidad del método de los cuadrados medios: su corto período y su tendencia a degenerar. Como se demostró en el documento original, con una semilla como 1234 y $k=4$, la secuencia degenera rápidamente a cero. La elección de una semilla y un valor de k más grandes (123456 y $k=6$) ha permitido generar una secuencia corta que parece estadísticamente aceptable. Sin embargo, para aplicaciones serias que requieren secuencias largas (miles o millones de números), este generador no es fiable, ya que eventualmente entrará en un ciclo corto o degenerará.

Conclusión

Para la secuencia corta de 100 valores analizada, el generador de cuadrados medios con la semilla y k especificados produce resultados que son consistentes con una muestra uniforme e independiente. Pasa satisfactoriamente las pruebas de Chi-cuadrado, Kolmogorov-Smirnov, Ljung-Box y los análisis gráficos. No obstante, este resultado no valida al generador para un uso general, debido a sus conocidas y graves deficiencias estructurales, como el corto período y la degeneración, que lo hacen inadecuado para la mayoría de las aplicaciones de simulación, que lo hacen inadecuado para la mayoría de las aplicaciones de simulación.

Análisis de la Calidad del generador por el método de los cuadrados medios sobre Múltiples Secuencias

Para obtener una evaluación más robusta de la calidad del generador de números pseudoaleatorios basado en el método de los cuadrados medios, se pueden generar múltiples secuencias independientes y aplicar las pruebas estadísticas a cada una. Esto permite observar la consistencia del generador a través de diferentes ejecuciones.

Por estos motivos se aplicará la siguiente metodología para evaluar la calidad del generador sobre múltiples secuencias:

Paso	Descripción	Detalles Clave
1	Generación de Secuencias	Se generarán 100 secuencias de 100 números cada una, utilizando el método de los cuadrados medios (k=6) con 100 semillas aleatorias distintas.
2	Aplicación de Pruebas	A cada una de las 100 secuencias se le aplicarán individualmente las pruebas de Chi-cuadrado, Kolmogorov-Smirnov (para uniformidad) y Ljung-Box (para independencia).
3	Ánalisis Agregado	Se recopilarán los p-valores de todas las pruebas para calcular la proporción de rechazos (fallos) y se analiza su distribución para evaluar la consistencia del generador.

Implementación en R

Primero, preparamos el entorno y definimos los parámetros para la simulación.

```
# Parámetros de la simulación
num_secuencias <- 100
n_por_secuencia <- 100
k <- 6
alpha <- 0.05

# Generar 100 semillas aleatorias de 6 dígitos para asegurar consistencia con k
set.seed(123) # Para la reproducibilidad de las semillas
semillas <- sample(100000:999999, num_secuencias)

# Vectores para almacenar los p-valores de cada prueba
p_valores_chisq <- numeric(num_secuencias)
p_valores_ks <- numeric(num_secuencias)
p_valores_ljungbox <- numeric(num_secuencias)
```

Ahora, iteramos sobre las semillas, generamos cada secuencia y aplicamos las pruebas.

```
# Bucle para generar secuencias y aplicar pruebas
for (i in 1:num_secuencias) {
  # Generar la secuencia
  seq_actual <- simres::rvng(n_por_secuencia, seed = semillas[i], k = k)

  # 1. Prueba de Chi-cuadrado para uniformidad
  test_chisq <- suppressWarnings(
    simres::freq.test(seq_actual, nclass = 10)
  )
  p_valores_chisq[i] <- test_chisq$p.value

  # 2. Prueba de Kolmogorov-Smirnov para uniformidad
  # Se suprimen las advertencias sobre empates, que son esperadas.
  test_ks <- suppressWarnings(ks.test(seq_actual, "punif"))
  p_valores_ks[i] <- test_ks$p.value

  # 3. Prueba de Ljung-Box para independencia
  test_ljungbox <- Box.test(seq_actual, lag = 10, type = "Ljung-Box")
  p_valores_ljungbox[i] <- test_ljungbox$p.value
}
```

Finalmente, analizamos los resultados agregados de las pruebas.

Resumen Comparativo de Resultados

Con los p-valores de las 100 simulaciones, podemos calcular la proporción de veces que cada prueba rechazó la hipótesis nula (es decir, falló en demostrar calidad estadística) a un nivel de significancia de 0.05.

```
# Calcular la proporción de rechazos
rechazos_chisq <- sum(p_valores_chisq < alpha) / num_secuencias
rechazos_ks <- sum(p_valores_ks < alpha) / num_secuencias
rechazos_ljungbox <- sum(p_valores_ljungbox < alpha) / num_secuencias

# Crear tabla de resumen
```

```

resumen <- data.frame(
  Prueba = c("Chi-cuadrado (Uniformidad)", "Kolmogorov-Smirnov (Uniformidad)", "Ljung-Box (Independencia)"),
  Proporcion_Rechazos = c(rechazos_chisq, rechazos_ks, rechazos_ljungbox),
  Proporcion_Aceptaciones = c(1 - rechazos_chisq, 1 - rechazos_ks, 1 - rechazos_ljungbox)
)

knitr::kable(resumen,
  caption = "Proporción de secuencias que pasan o fallan las pruebas de calidad (alpha = 0.05)",
  col.names = c("Prueba Estadística", "Proporción de Fallos (Rechazo H0)", "Proporción de Aceptaciones (No Rechazo H0)")
)

```

Proporción de secuencias que pasan o fallan las pruebas de calidad (alpha = 0.05).

Prueba Estadística	Proporción de Fallos (Rechazo H ₀)	Proporción de Aceptaciones (No Rechazo H ₀)
Chi-cuadrado (Uniformidad)	0.14	0.86
Kolmogorov-Smirnov (Uniformidad)	0.13	0.87
Ljung-Box (Independencia)	0.16	0.84

El resumen muestra la proporción de secuencias que pasaron o fallaron cada una de las pruebas estadísticas. Idealmente, para un buen generador de números pseudoaleatorios, se esperaría que la proporción de rechazos sea cercana al nivel de significancia (0.05 no en este caso), indicando que las secuencias generadas no son consistentes con las propiedades esperadas de uniformidad e independencia.

Esto último debido a que los resultados muestran una clara deficiencia en el generador. Si el generador produjera secuencias verdaderamente aleatorias, esperaríamos que la proporción de fallos (rechazos de la hipótesis nula) fuera aproximadamente del 5% (alpha = 0.05). Sin embargo, los resultados obtenidos son:

- 14% para la prueba de Chi-cuadrado.
- 13% para la prueba de Kolmogorov-Smirnov.
- 16% para la prueba de Ljung-Box.

Estas proporciones son casi tres veces mayores que el nivel de significancia esperado. Esto indica que el generador falla en producir secuencias que aparenten ser uniformes e independientes con mucha más frecuencia de la debida, revelando una inconsistencia sistemática en su comportamiento.

Distribución de los P-valores

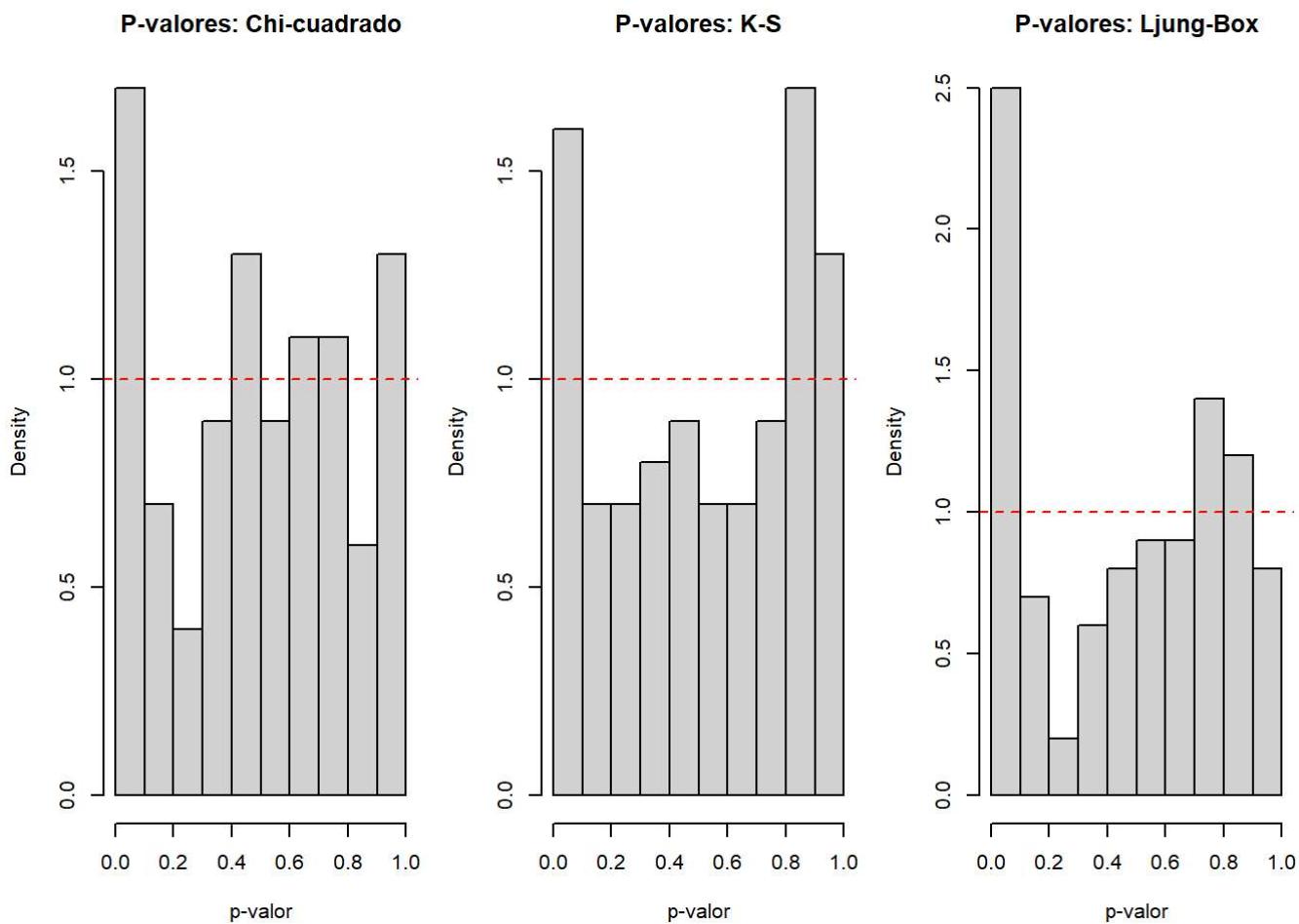
Además de la proporción de rechazos, es útil visualizar la distribución de los p-valores obtenidos en las pruebas para evaluar si se comportan como se esperaría bajo la hipótesis nula.

```

# Gráficos de los p-valores
par(mfrow = c(1, 3), mar = c(4, 4, 3, 1))
hist(p_valores_chisq, main = "P-valores: Chi-cuadrado", xlab = "p-valor", freq = FALSE, breaks = 10)
abline(h = 1, col = "red", lty = 2)
hist(p_valores_ks, main = "P-valores: K-S", xlab = "p-valor", freq = FALSE, breaks = 10)
abline(h = 1, col = "red", lty = 2)

```

```
hist(p_valores_ljungbox, main = "P-valores: Ljung-Box", xlab = "p-valor", freq = FALSE, breaks
abline(h = 1, col = "red", lty = 2)
```



```
par(mfrow = c(1, 1))
```

Los histogramas de los p-valores se desvían notablemente de la distribución uniforme esperada (representada por la línea roja discontinua), lo que confirma las deficiencias del generador:

- P-valores de Chi-cuadrado: Se observa una fuerte acumulación de p-valores en el extremo izquierdo del histograma (intervalo [0.0, 0.1]). Esto indica que un número desproporcionadamente alto de secuencias fallaron la prueba de uniformidad, lo cual es consistente con la tasa de rechazo del 14% observada en la tabla.
- P-valores de K-S: Este gráfico muestra una distribución en forma de "U", con picos tanto en valores muy bajos como muy altos. Esta es una señal clásica de un generador defectuoso. No solo produce secuencias que se desvían significativamente de la uniformidad (p-valores bajos), sino que también genera secuencias que son "demasiado perfectas" o sospechosamente uniformes (p-valores altos), algo que tampoco es esperable en un proceso verdaderamente aleatorio.
- P-valores de Ljung-Box: Al igual que en la prueba de Chi-cuadrado, se observa una clara tendencia a generar p-valores bajos, lo que llevó a una tasa de rechazo del 16% y evidencia problemas sistemáticos con la independencia de los números generados.

Conclusión Final sobre Estabilidad y Confiabilidad

El análisis sobre 100 secuencias independientes demuestra de manera concluyente que el método de los cuadrados medios no es un generador de números pseudoaleatorios confiable ni estable, ni siquiera para secuencias cortas con semillas aparentemente bien elegidas.

Mientras que el análisis de una sola secuencia puede, por azar, no revelar problemas, la evaluación repetida muestra un patrón claro de fallos estadísticos. Las tasas de rechazo para las pruebas de uniformidad (14% y 13%) e independencia (16%) son casi tres veces superiores al nivel de significancia esperado del 5%. Esta discrepancia, visualmente confirmada por la distribución no uniforme de los p-valores, indica que el método no genera de forma consistente secuencias que imiten adecuadamente las propiedades de una muestra aleatoria.

En definitiva, el método de los cuadrados medios es estadísticamente deficiente. Su valor es puramente histórico y didáctico. Para cualquier aplicación práctica, es imperativo utilizar generadores modernos y robustos, como el Mersenne-Twister, que ha superado rigurosas baterías de pruebas y es el predeterminado en R, que ha superado rigurosas baterías de pruebas y es el predeterminado en R.