

Tarea 5 de Modelos Probabilistas Aplicados

Algoritmos generadores de números pseudo-aleatorios con distribución Uniforme y distribución Normal

5271

6 de octubre de 2020

1. Introducción

En este trabajo se presenta el análisis a varios algoritmos de generación de números pseudo-aleatorios con distribución Uniforme y distribución Normal. Así como el impacto de los parámetros de dichos algoritmos en la calidad de los números pseudo-aleatorios. El análisis será realizado en el programa R versión 4.0.2 [2] en el entorno de desarrollo Rstudio [3]

2. Generador congruencial lineal(Mixto)

Entre los principales generadores de números pseudo-aleatorios que se emplean en la actualidad están los llamados generadores congruenciales lineales, introducidos por Lehmer en 1951. Estos métodos comienza con un valor inicial x_0 (semilla), y los sucesivos valores $x_n, n \geq 0$ se obtienen recursivamente con la ecuación 1:

$$X_{n+1} = (aX_n + c) \bmod m, \quad n \geq 0. \quad (1)$$

Con parámetros:

$$\begin{aligned} m, & \quad \text{el módulo;} & 0 < m. \\ a, & \quad \text{el multiplicador;} & 0 \leq a < m. \\ c, & \quad \text{el incremento;} & 0 \leq c < m. \\ X_0, & \quad \text{la semilla;} & 0 \leq X_0 < m. \end{aligned} \quad (2)$$

Se crea una función con la ecuación 1 en R, como muestra el código 1.

```
1 dist_uniforme = function(n, semilla) {  
2   a = 7  
3   cc = 7  
4   m = 10  
5   datos = numeric()  
6   x = semilla  
7   while (length(datos) < n) {  
8     x = (a * x + cc) %% m  
9     datos = c(datos, x)  
10  }  
11  return(datos / (m - 1))}
```

Tarea5.R

Pero con esto no es suficiente para garantizar la calidad de los números pseudos-aleatorios, una de estas medidas de calidad es el período de los números generados. El período no es más que cada cuantos números generados se vuelve a repetir la secuencia y se representa $\lambda^*(m)$ es decir que si $\lambda^* < m$ el generador no es de buena calidad. Los generadores de buena calidad deben tener un período completo, es decir $\lambda^*(m) = m$. Para garantizar el período completo se tiene de [1]:

Teorema 1 *La secuencia lineal congruencial definida por m, a, c, X_0 tiene período completo sí y solo sí*

- I. c es primo relativo de m .
- II. $b = a - 1$ es múltiplo de $p, \forall p$ primo dividiendo m .
- III. b es múltiplo de cuatro, sí m es un múltiplo de cuatro.

2.1. Selección del módulo

Para la selección del modulo m la siguiente expresión:

$$m = P^e \begin{cases} P \text{ es la base que utiliza} \\ e \text{ es el número de bit} \end{cases} \quad (3)$$

La base mayormente usada es dos y el número de bit es 32. Para la selección del valor para el módulo se creo la función 2.1 en R.

```

1 modu = function (p,e){
2   m = p^e
3   return(m)}

```

Tarea5.R

2.2. Selección del incremento

El incremento o constante aditiva c es un número en el intervalo $0 \leq c < m$ y primo relativo con m , es decir que el Mínimo Común Divisor de $(c, m) = 1$, esto genera una cierta cantidad de posibles valores de c . Para garantizar la elección de un valor de c adecuado se realizo una la función 2.2 en R.

```

1 aditiva= function(m,po){
2   lisc=numeric()
3   for (i in c(1:m)) {
4     if(GCD(m,i)==1){
5       lisc=c(lisc,i)
6     }
7   }
8   c = lisc[po]
9   return(c)
10 }

```

Tarea5.R

2.3. Selección del multiplicador

Para la elección acertada del multiplicador a se tiene las siguientes expresiones:

$$a = 1 + MCM(P_1, P_2, P_3, \dots, P_{k-1}, P_k, 4) * t, \quad t \in (Z^+ \cup \{0\}) \text{ si cuatro divide a } m.$$

$$a = 1 + MCM(P_1, P_2, P_3, \dots, P_{k-1}, P_k) * t, \quad t \in (Z^+ \cup \{0\}) \text{ si cuatro no divide a } m.$$

Teniendo en cuenta estas expresiones y con el apoyo en la función *LCM* de R que calcula el Mínimo Común Múltiplo (MCM), se crea la función 2.3 en R para la correcta selección del parámetro a .

```
1 multiplicador = function(m,t){
2   a = 0
3   s = numeric()
4   d = numeric()
5   if ((m %%4 )!=0){
6     s = primeFactors(m)
7     d = s[!duplicated(s)]
8     if(length(d) == 1){
9       a = 1 + d * t
10
11     } else {
12       a = 1 + LCM(d) * t
13     } else {
14       s = primeFactors(m)
15       d = s[!duplicated(s)]
16       d = c(d,4)
17       a = 1 + LCM(d) * t
18     }
19   return(a)}

```

Tarea5.R

2.4. Selección de la semilla

Para la selección de la semilla X_0 solo se debe tener en cuenta que debe encontrarse en el rango $0 \leq X_0 < m$.

2.5. Generador congruencial lineal(Mixto) de período completo

Con las funciones 2.1, 2.2, 2.3, se modifica la función 2, dando lugar a la función 2.5 que garantiza el período completo y la posibilidad de reproducir la secuencia si fuera necesario, dando como resultados números pseudo-aleatorios de calidad. Esto se comprueba en la figura 1 de la página 5 y en los resultados de la prueba estadística de uniformidad e independencia Chi-cuadrado con un valor del estadístico $X^2 = 11,424$ y el valor $p = 0,9088$, no se rechaza la hipótesis nula. Por tanto, los números son independientes y uniformes.

```
1 uniforme_comp = function(n,p,e,t,ps,po) {
2   m = modu(p,e)
3   s = semilla(m,ps)
4   a = multiplicador(m,t)
5   c = aditiva(m,po)
6   datos = numeric()
7   x = s

```

```

8   while (length(datos) < n) {
9       x = (a * x + c) %%m
10      datos = c(datos, x)
11  }
12  return(datos / (m - 1))
13 }

```

Tarea5.R

3. Transformada de Box-Muller

la Transformada de Box-Muller es método para generar pares de independiente, estándar, normalmente distribuido. Este método fue llevado a un programa de R como se muestra en el código 3. A partir de este código se realizó una experimentación donde se variaron los parámetros z_0 , z_1 y se utilizó en uno de los casos el generador *uniformeGLC* propuesto.

```

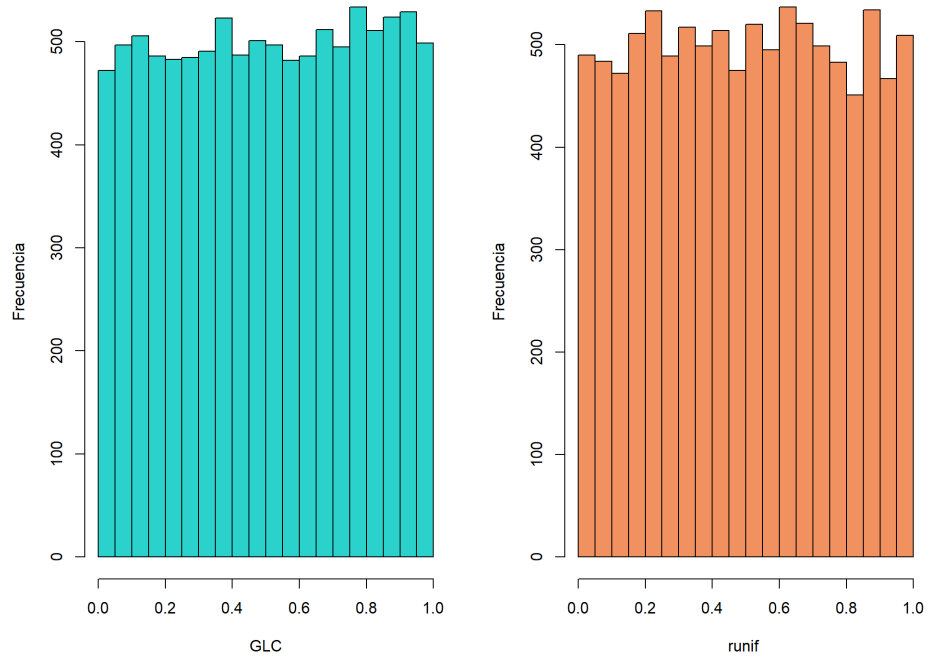
1  gaussian = function (mu, sigma) {
2    u = runif(2)
3    z0 = sqrt(-2*log(u[1])) * cos(2*pi*u[2])
4    z1 = sqrt(-2*log(u[1])) * sin(2*pi*u[2])
5    datos = c(z0, z1)
6    return (sigma * datos + mu)

```

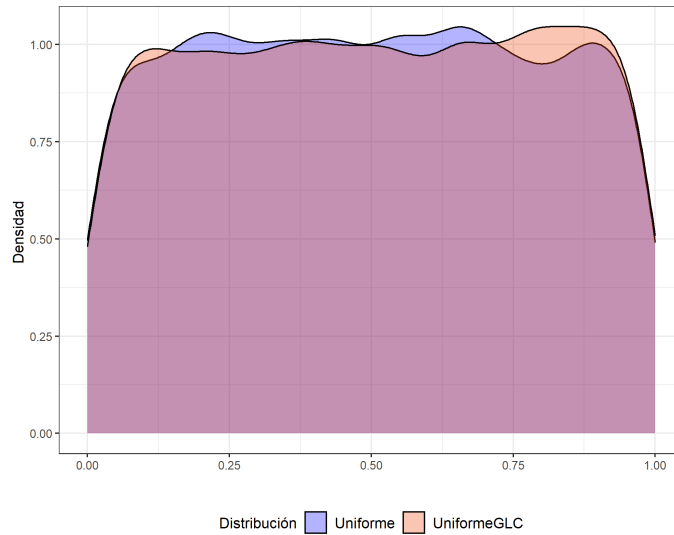
Tarea5.R

En la imagen 2 de la página 6 se muestran los histogramas de las diferentes variantes analizadas, como se puede observar la variante la sdos variantes representan una distribución normal, por la prueba de Shapiro–Wilk con un valor p de 0.98 para la a) y 0.87 para b).

El código general se encuentra disponible en el repositorio. <https://github.com/Albertomnoa/Tareas>

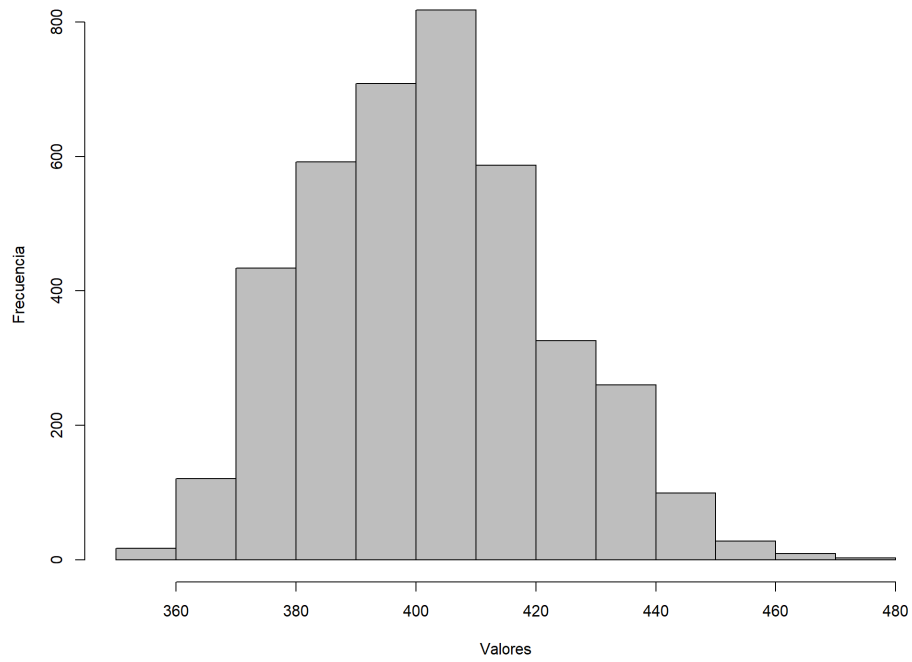


(a) Histogramas de frecuencia de los números pseudo-aleatorios creados por la función *uniformeGCL* presentada y *runif* de R.

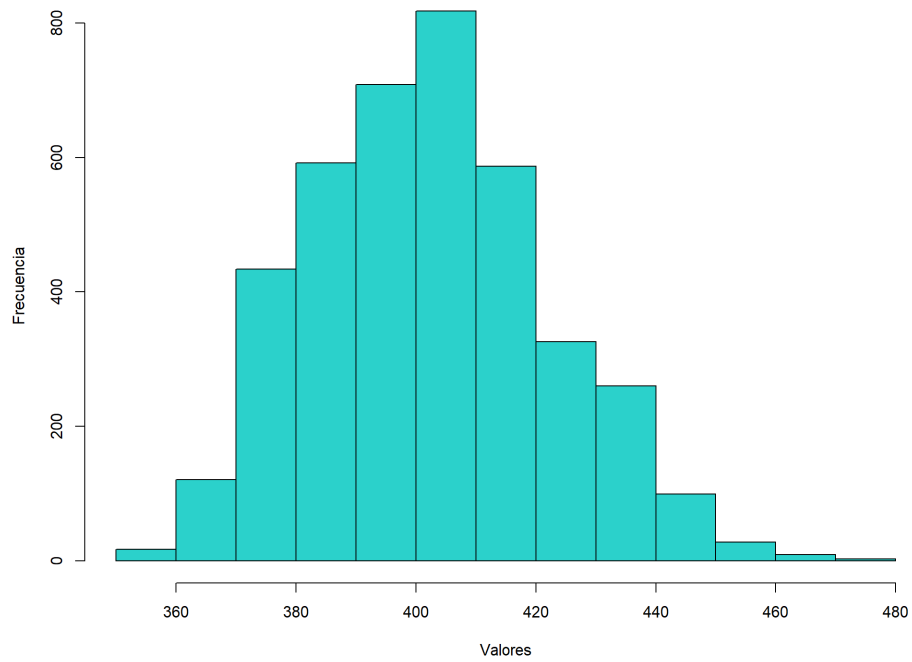


(b) Diagrama de densidad de ambas poblaciones generadas decreciente

Figura 1: Comparación de ambos métodos de generación de números pseudo-aleatorios



(a) Histograma de frecuencia de la distribución normal creada por *runif*



(b) Histograma de frecuencia de la distribución normal creada por *UniformeGLC*

Figura 2: Comparación de ambos métodos de generación de números pseudo-aleatorios

Referencias

- [1] Donald E. Knuth. *The Art of Computer Programming, Volume 2 (3rd Ed.): Seminumerical Algorithms*. Addison-Wesley Longman Publishing Co., Inc., USA, 1997.
- [2] R Core Team. R: Un lenguaje y un entorno para la informática estadística, 2020.
- [3] RStudio Team. Rstudio: Entorno de desarrollo integrado para R, 2020.