Simulación ITAM
24 de febrero de 2018 Jorge de la Vega
2018-I

Tarea 1.

La fecha de entrega es el **16 de febrero de 2018**.

Lecturas

- Robert & Casella Capítulo 2 sección 2.1
- Dagpunar Capítulo 2
- Good random number generators are (not so) easy to find
- Linear Congruential Generator in R

Problemas

1. Probar por inducción matemática que para un generador lineal congruencial (GLC),

$$Z_i \equiv \left[a^i Z_0 + c\frac{a^i - 1}{a-1}\right] \mod m$$

Solución.

Probamos primero para $k = 1$: $Z_1 = [aZ_0 + c\frac{a-1}{a-1}] \mod m = [aZ_0 + c] \mod m$. Esta es la definición de Z_1. Por lo tanto, el resultado es válido para $k = 1$.

Ahora suponemos que el resultado es válido para un índice k. Esta es la hipótesis de inducción.

Queremos probar que el resultado es válido para $k + 1$.

En términos generales, la operación módulo se define como $u \equiv a \mod b$ si existe un entero q tal que $u = bq + a$. Entonces existe q_1 tal que

$$Z_k = mq_1 + \left[a^k Z_0 + c\frac{a^k - 1}{a-1}\right]$$

y también existe q_2 tal que $Z_{k+1} = mq_2 + (aZ_k + c)$. Reemplazando Z_k por la primera ecuación se obtiene que $Z_{k+1} = m(q_2 + aq_1) + a^{k+1}Z_0 + c\frac{a^{k+1}-1}{a-1}$. Por lo tanto, la relación se cumple para los módulos.

La conclusión final es que el resultado es cierto para cualquier k.

□

2. Calcular el periodo de $Z_i \equiv (5Z_{i-1} + 3) \mod 31$.

Solución.

Este es un generador de periodo completo, porque cumple con las condiciones de Hull y Dobell:

```
x <- NULL
x[1] <- 7
for(i in 2:16){x[i] <- (5*x[i-1]+3) %% 16}
x
```

```
 [1]  7  6  1  8 11 10  5 12 15 14  9  0  3  2 13  4
```

□

3. Sin calcular explícitamente ninguna Z_i, determinar cuál de los siguientes GLC's mixtos tienen periodo completo.

   (a) $Z_i \equiv [13Z_{i-1} + 13] \mod 16$.

   (b) $Z_i \equiv [12Z_{i-1} + 13] \mod 16$.

   (c) $Z_i \equiv [25437Z_{i-1} + 35421] \mod 2^{10}$.

   (d) $Z_i \equiv [Z_{i-1} + 12] \mod 13$.

   (e) El glc con parámetros: $a = 2{,}814{,}749{,}767{,}109$, $c = 59{,}482{,}661{,}568{,}307$, $m = 2^{48}$.

Solución.

Lo único que se requiere es verificar las condiciones del teorema de Hull y Dobell.

(a) (a) $c = 13$ y $m = 16$ son primos relativos, (b) los primos divisores de 16 son {2} y 2 divide a $a - 1 = 12$. (c) 4 divide a 16 y 4 divide a 12. Por lo tanto, este generador tiene periodo completo.

(b) (a) $c = 13$ y $m = 16$ son primos relativos, (b) los primos divisores de 16 son {2} y 2 NO divide a $a - 1 = 11$. No tiene periodo completo.

(c) Ver (e) abajo.

```
library(gmp)
```

```
Attaching package: 'gmp'
The following objects are masked from 'package:base':
   %*%, apply, crossprod, matrix, tcrossprod
```

```
a <- as.bigz(25437)
c <- as.bigz(35421)
m <- as.bigz(2^10)
factorize(a)

Big Integer ('bigz') object of length 3:
[1] 3   61  139

factorize(c)

Big Integer ('bigz') object of length 2:
[1] 3     11807

factorize(a-1)

Big Integer ('bigz') object of length 3:
[1] 2    2    6359
```

Entonces, vemos que (a) c y m son primos relativos. (b) 2 es el divisor primo de m, y 2 divide a $a-1$, y (c) m es divisible por 4, que también divide a $a-1$. Por lo tanto, el GLC tiene ciclo completo.

(d) (a) $c = 12$ y $m = 13$ son primos relativos, (b) los primos divisores de 13 son $\{13\}$ y 13 divide a $a - 1 = 0$. La última condición no se requiere ya que 4 no divide a 13. Es de periodo completo.

(e) Para resolver este problema, podemos usar R para hacer aritmética de grandes enteros. El problema con los números es que son muy grandes y en la computadora se desbordan. Un poco de investigación nos lleva al paquete `gmp`, que permite hacer aritmética de precisión múltiple. Por ejemplo, nos permite encontrar la descomposición en primos de grandes enteros. Lo que nos sería de utilidad es decomponer los números en su factorización prima:

```
library(gmp)
a <- as.bigz(2814749767109)
c <- as.bigz(59482661568307)
m <- as.bigz(2^48)
factorize(a)

Big Integer ('bigz') object of length 3:
[1] 7       65111   6175717

factorize(c)

Big Integer ('bigz') object of length 2:
[1] 7850083 7577329

factorize(a-1)

Big Integer ('bigz') object of length 3:
[1] 2           2           703687441777
```

Entonces, vemos que (a) c y m son primos relativos. (b) 2 es el divisor primo de m, y 2 divide a $a-1$, y (c) m es divisible por 4, que también divide a $a-1$. Por lo tanto, el GLC tiene ciclo completo.

Noten en este problema que si intentan realizar la aritmética de manera directa, el problema genera incongruencias debido al sobreflujo generado por el tamaño de los números.

□

4. Mostrar que el promedio de las U_i's tomadas de un ciclo completo de un GLC de periodo completo es $\frac{1}{2} - \frac{1}{2m}$.

Solución.

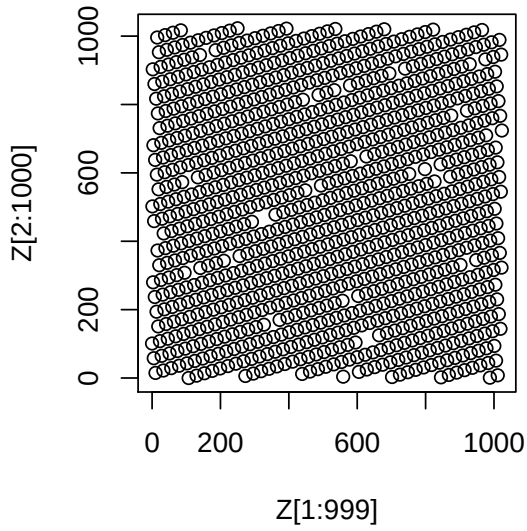
El promedio de las U_i's es $\bar{U} = \frac{1}{m}\sum_{i=1}^{m} U_i = \frac{1}{m}\sum_{i=1}^{m} \frac{Z_i}{m} = \frac{1}{m^2}\sum_{i=1}^{m} Z_i$. Como las Z_i toman todos los posibles valores entre 0 y $m-1$, ya que el periodo es completo, entonces la suma de las variables es equivalente a la suma de los primeros $m-1$ naturales, que es igual a $(m-1)m/2$. Por lo tanto $\bar{U} = \frac{(m-1)m}{2m^2} = \frac{1}{2} - \frac{1}{2m}$.

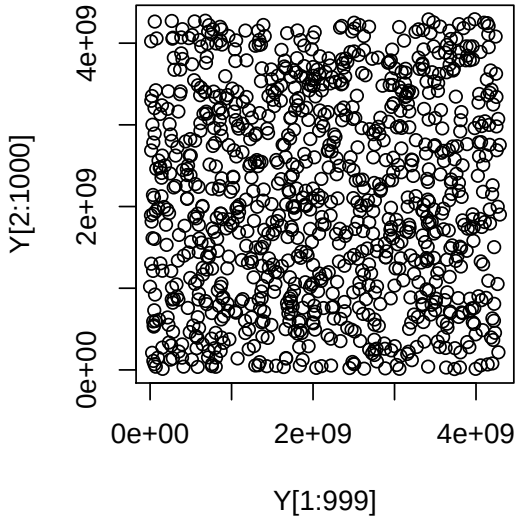
□

5. Dada una sucesión $X_1, X_2, \ldots, X_n$ de $\mathcal{U}(0,1)$ números pseudoaleatorios, podemos hacer una gráfica de dispersión de puntos de (X_i, X_{i+1}) para $i = 1, \ldots, n-1$ para verificar si hay independencia. Hacer esta gráfica para el GLC con parámetros $m = 1{,}024, a = 401, c = 101$ y para el GLC $m = 2^{32}, a = 1{,}664{,}525, c = 1{,}013{,}904{,}223$.

Solución.

```
glc <- function(Z0,a,c,m){
  Z <- (a*Z0 + c) %% m
  return(Z)
}
Z <- 5 #valor inicial arbitrario
for (i in 2:1000) Z[i] <- glc(Z[i-1],a=401,c=101,m=1024)
Y <- 3 #valor inicial arbitrario
for (i in 2:1000) Y[i] <- glc(Y[i-1],a=1664525,c=1013904223,m=2^32)
par(mfrow=c(1,2))
par(pty="s")
plot(Z[1:999],Z[2:1000])
plot(Y[1:999],Y[2:1000])
```





□

6. Probar que la parte fraccional de la suma de uniformes [0,1] $U_1+U_2+\cdots+U_k$ es también uniforme en el intervalo [0,1]

Solución.

Este ejercicio se puede probar por inducción. Para facilitar la notación, definamos $\{x\} = x - \lfloor x \rfloor$ como la parte fraccional de x. La densidad de $\{U_1+U_2\}$ está dada por

$$f_{U_1+U_2}(x) = \begin{cases} x, & 0 \leq x \leq 1 \\ 2-x, & 1 < x \leq 2. \end{cases}$$

La distribución esta dada por la siguiente expresión, de la que ustedes pueden completar los detalles, considerando que U_1+U_2 puede estar entre (0,1) y (1,2)

$$F(x) = \Pr[\{U_1+U_2\} \leq x] = \int_{u=0}^{x} f_{U_1+U_2}(u)\,du + \int_{1}^{1+x} f_{U_1+U_2}(u)\,du = x.$$

□

7. Un generador de Fibonacci obtiene X_{n+1} a partir de X_n y X_{n-1} de la siguiente forma:

$$X_{i+1} \equiv (X_i + X_{i-1}) \mod m$$

donde X_0 y X_1 están especificados.

Supongan que $m = 5$. Sólo dos ciclos son posibles. Encontrarlos, así como su respectivo periodo.

Solución.

Para el valor $m = 5$ hay $5^2 = 25$ posibles valores iniciales. Obteniendo la secuencia para cada posible valor inicial, podemos construir la siguiente matrix que tiene por columnas cada una de las combinaciones de valores iniciales.

```
fibo <- function(m=5,x0,x1){(x0+x1) %% m}
X <- as.matrix(expand.grid(0:4,0:4)); names(X) <- NULL
M <- NULL
for (j in 1:25){
  x <- as.vector(X[j,])
  for(i in 3:100) x[i] <- fibo(x1=x[i-1],x0=x[i-2])
  M <- cbind(M,x)
}
M
```

```
      x x x x x x x x x x x x x x x x x x x x x x x x x
 [1,] 0 1 2 3 4 0 1 2 3 4 0 1 2 3 4 0 1 2 3 4 0 1 2 3 4
 [2,] 0 0 0 0 0 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4 4 4 4 4
 [3,] 0 1 2 3 4 1 2 3 4 0 2 3 4 0 1 3 4 0 1 2 4 0 1 2 3
 [4,] 0 1 2 3 4 2 3 4 0 1 4 0 1 2 3 1 2 3 4 0 3 4 0 1 2
 [5,] 0 2 4 1 3 3 0 2 4 1 1 3 0 2 4 4 1 3 0 2 2 4 1 3 0
 [6,] 0 3 1 4 2 0 3 1 4 2 0 3 1 4 2 0 3 1 4 2 0 3 1 4 2
 [7,] 0 0 0 0 0 3 3 3 3 3 1 1 1 1 1 4 4 4 4 4 2 2 2 2 2
 [8,] 0 3 1 4 2 3 1 4 2 0 1 4 2 0 3 4 2 0 3 1 2 0 3 1 4
 [9,] 0 3 1 4 2 1 4 2 0 3 2 0 3 1 4 3 1 4 2 0 4 2 0 3 1
[10,] 0 1 2 3 4 4 0 1 2 3 3 4 0 1 2 2 3 4 0 1 1 2 3 4 0
[11,] 0 4 3 2 1 0 4 3 2 1 0 4 3 2 1 0 4 3 2 1 0 4 3 2 1
[12,] 0 0 0 0 0 4 4 4 4 4 3 3 3 3 3 2 2 2 2 2 1 1 1 1 1
[13,] 0 4 3 2 1 4 3 2 1 0 3 2 1 0 4 2 1 0 4 3 1 0 4 3 2
[14,] 0 4 3 2 1 3 2 1 0 4 1 0 4 3 2 4 3 2 1 0 2 1 0 4 3
[15,] 0 3 1 4 2 2 0 3 1 4 4 2 0 3 1 1 4 2 0 3 3 1 4 2 0
[16,] 0 2 4 1 3 0 2 4 1 3 0 2 4 1 3 0 2 4 1 3 0 2 4 1 3
[17,] 0 0 0 0 0 2 2 2 2 2 4 4 4 4 4 1 1 1 1 1 3 3 3 3 3
[18,] 0 2 4 1 3 2 4 1 3 0 4 1 3 0 2 1 3 0 2 4 3 0 2 4 1
[19,] 0 2 4 1 3 4 1 3 0 2 3 0 2 4 1 2 4 1 3 0 1 3 0 2 4
[20,] 0 4 3 2 1 1 0 4 3 2 2 1 0 4 3 3 2 1 0 4 4 3 2 1 0
[21,] 0 1 2 3 4 0 1 2 3 4 0 1 2 3 4 0 1 2 3 4 0 1 2 3 4
[22,] 0 0 0 0 0 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4 4 4 4 4
[23,] 0 1 2 3 4 1 2 3 4 0 2 3 4 0 1 3 4 0 1 2 4 0 1 2 3
[24,] 0 1 2 3 4 2 3 4 0 1 4 0 1 2 3 1 2 3 4 0 3 4 0 1 2
[25,] 0 2 4 1 3 3 0 2 4 1 1 3 0 2 4 4 1 3 0 2 2 4 1 3 0
[26,] 0 3 1 4 2 0 3 1 4 2 0 3 1 4 2 0 3 1 4 2 0 3 1 4 2
[27,] 0 0 0 0 0 3 3 3 3 3 1 1 1 1 1 4 4 4 4 4 2 2 2 2 2
[28,] 0 3 1 4 2 3 1 4 2 0 1 4 2 0 3 4 2 0 3 1 2 0 3 1 4
[29,] 0 3 1 4 2 1 4 2 0 3 2 0 3 1 4 3 1 4 2 0 4 2 0 3 1
[30,] 0 1 2 3 4 4 0 1 2 3 3 4 0 1 2 2 3 4 0 1 1 2 3 4 0
[31,] 0 4 3 2 1 0 4 3 2 1 0 4 3 2 1 0 4 3 2 1 0 4 3 2 1
[32,] 0 0 0 0 0 4 4 4 4 4 3 3 3 3 3 2 2 2 2 2 1 1 1 1 1
[33,] 0 4 3 2 1 4 3 2 1 0 3 2 1 0 4 2 1 0 4 3 1 0 4 3 2
[34,] 0 4 3 2 1 3 2 1 0 4 1 0 4 3 2 4 3 2 1 0 2 1 0 4 3
[35,] 0 3 1 4 2 2 0 3 1 4 4 2 0 3 1 1 4 2 0 3 3 1 4 2 0
[36,] 0 2 4 1 3 0 2 4 1 3 0 2 4 1 3 0 2 4 1 3 0 2 4 1 3
[37,] 0 0 0 0 0 2 2 2 2 2 4 4 4 4 4 1 1 1 1 1 3 3 3 3 3
[38,] 0 2 4 1 3 2 4 1 3 0 4 1 3 0 2 1 3 0 2 4 3 0 2 4 1
[39,] 0 2 4 1 3 4 1 3 0 2 3 0 2 4 1 2 4 1 3 0 1 3 0 2 4
[40,] 0 4 3 2 1 1 0 4 3 2 2 1 0 4 3 3 2 1 0 4 4 3 2 1 0
[41,] 0 1 2 3 4 0 1 2 3 4 0 1 2 3 4 0 1 2 3 4 0 1 2 3 4
[42,] 0 0 0 0 0 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4 4 4 4 4
[43,] 0 1 2 3 4 1 2 3 4 0 2 3 4 0 1 3 4 0 1 2 4 0 1 2 3
```

```
 [44,] 0 1 2 3 4 2 3 4 0 1 4 0 1 2 3 1 2 3 4 0 3 4 0 1 2
 [45,] 0 2 4 1 3 3 0 2 4 1 1 3 0 2 4 4 1 3 0 2 2 4 1 3 0
 [46,] 0 3 1 4 2 0 3 1 4 2 0 3 1 4 2 0 3 1 4 2 0 3 1 4 2
 [47,] 0 0 0 0 0 3 3 3 3 3 1 1 1 1 1 4 4 4 4 4 2 2 2 2 2
 [48,] 0 3 1 4 2 3 1 4 2 0 1 4 2 0 3 4 2 0 3 1 2 0 3 1 4
 [49,] 0 3 1 4 2 1 4 2 0 3 2 0 3 1 4 3 1 4 2 0 4 2 0 3 1
 [50,] 0 1 2 3 4 4 0 1 2 3 3 4 0 1 2 2 3 4 0 1 1 2 3 4 0
 [51,] 0 4 3 2 1 0 4 3 2 1 0 4 3 2 1 0 4 3 2 1 0 4 3 2 1
 [52,] 0 0 0 0 0 0 4 4 4 4 4 3 3 3 3 3 2 2 2 2 2 1 1 1 1
 [53,] 0 4 3 2 1 4 3 2 1 0 3 2 1 0 4 2 1 0 4 3 1 0 4 3 2
 [54,] 0 4 3 2 1 3 2 1 0 4 1 0 4 3 2 4 3 2 1 0 2 1 0 4 3
 [55,] 0 3 1 4 2 2 0 3 1 4 4 2 0 3 1 1 4 2 0 3 3 1 4 2 0
 [56,] 0 2 4 1 3 0 2 4 1 3 0 2 4 1 3 0 2 4 1 3 0 2 4 1 3
 [57,] 0 0 0 0 0 2 2 2 2 2 4 4 4 4 4 1 1 1 1 1 3 3 3 3 3
 [58,] 0 2 4 1 3 2 4 1 3 0 4 1 3 0 2 1 3 0 2 4 3 0 2 4 1
 [59,] 0 2 4 1 3 4 1 3 0 2 3 0 2 4 1 2 4 1 3 0 1 3 0 2 4
 [60,] 0 4 3 2 1 1 0 4 3 2 2 1 0 4 3 3 2 1 0 4 4 3 2 1 0
 [61,] 0 1 2 3 4 0 1 2 3 4 0 1 2 3 4 0 1 2 3 4 0 1 2 3 4
 [62,] 0 0 0 0 0 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4 4 4 4 4
 [63,] 0 1 2 3 4 1 2 3 4 0 2 3 4 0 1 3 4 0 1 2 4 0 1 2 3
 [64,] 0 1 2 3 4 2 3 4 0 1 4 0 1 2 3 1 2 3 4 0 3 4 0 1 2
 [65,] 0 2 4 1 3 3 0 2 4 1 1 3 0 2 4 4 1 3 0 2 2 4 1 3 0
 [66,] 0 3 1 4 2 0 3 1 4 2 0 3 1 4 2 0 3 1 4 2 0 3 1 4 2
 [67,] 0 0 0 0 0 3 3 3 3 3 1 1 1 1 1 4 4 4 4 4 2 2 2 2 2
 [68,] 0 3 1 4 2 3 1 4 2 0 1 4 2 0 3 4 2 0 3 1 2 0 3 1 4
 [69,] 0 3 1 4 2 1 4 2 0 3 2 0 3 1 4 3 1 4 2 0 4 2 0 3 1
 [70,] 0 1 2 3 4 4 0 1 2 3 3 4 0 1 2 2 3 4 0 1 1 2 3 4 0
 [71,] 0 4 3 2 1 0 4 3 2 1 0 4 3 2 1 0 4 3 2 1 0 4 3 2 1
 [72,] 0 0 0 0 0 4 4 4 4 4 3 3 3 3 3 2 2 2 2 2 1 1 1 1 1
 [73,] 0 4 3 2 1 4 3 2 1 0 3 2 1 0 4 2 1 0 4 3 1 0 4 3 2
 [74,] 0 4 3 2 1 3 2 1 0 4 1 0 4 3 2 4 3 2 1 0 2 1 0 4 3
 [75,] 0 3 1 4 2 2 0 3 1 4 4 2 0 3 1 1 4 2 0 3 3 1 4 2 0
 [76,] 0 2 4 1 3 0 2 4 1 3 0 2 4 1 3 0 2 4 1 3 0 2 4 1 3
 [77,] 0 0 0 0 0 2 2 2 2 2 4 4 4 4 4 1 1 1 1 1 3 3 3 3 3
 [78,] 0 2 4 1 3 2 4 1 3 0 4 1 3 0 2 1 3 0 2 4 3 0 2 4 1
 [79,] 0 2 4 1 3 4 1 3 0 2 3 0 2 4 1 2 4 1 3 0 1 3 0 2 4
 [80,] 0 4 3 2 1 1 0 4 3 2 2 1 0 4 3 3 2 1 0 4 4 3 2 1 0
 [81,] 0 1 2 3 4 0 1 2 3 4 0 1 2 3 4 0 1 2 3 4 0 1 2 3 4
 [82,] 0 0 0 0 0 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4 4 4 4 4
 [83,] 0 1 2 3 4 1 2 3 4 0 2 3 4 0 1 3 4 0 1 2 4 0 1 2 3
 [84,] 0 1 2 3 4 2 3 4 0 1 4 0 1 2 3 1 2 3 4 0 3 4 0 1 2
 [85,] 0 2 4 1 3 3 0 2 4 1 1 3 0 2 4 4 1 3 0 2 2 4 1 3 0
 [86,] 0 3 1 4 2 0 3 1 4 2 0 3 1 4 2 0 3 1 4 2 0 3 1 4 2
 [87,] 0 0 0 0 0 3 3 3 3 3 1 1 1 1 1 4 4 4 4 4 2 2 2 2 2
 [88,] 0 3 1 4 2 3 1 4 2 0 1 4 2 0 3 4 2 0 3 1 2 0 3 1 4
 [89,] 0 3 1 4 2 1 4 2 0 3 2 0 3 1 4 3 1 4 2 0 4 2 0 3 1
 [90,] 0 1 2 3 4 4 0 1 2 3 3 4 0 1 2 2 3 4 0 1 1 2 3 4 0
 [91,] 0 4 3 2 1 0 4 3 2 1 0 4 3 2 1 0 4 3 2 1 0 4 3 2 1
 [92,] 0 0 0 0 0 4 4 4 4 4 3 3 3 3 3 2 2 2 2 2 1 1 1 1 1
 [93,] 0 4 3 2 1 4 3 2 1 0 3 2 1 0 4 2 1 0 4 3 1 0 4 3 2
 [94,] 0 4 3 2 1 3 2 1 0 4 1 0 4 3 2 4 3 2 1 0 2 1 0 4 3
 [95,] 0 3 1 4 2 2 0 3 1 4 4 2 0 3 1 1 4 2 0 3 3 1 4 2 0
 [96,] 0 2 4 1 3 0 2 4 1 3 0 2 4 1 3 0 2 4 1 3 0 2 4 1 3
 [97,] 0 0 0 0 0 2 2 2 2 2 4 4 4 4 4 1 1 1 1 1 3 3 3 3 3
 [98,] 0 2 4 1 3 2 4 1 3 0 4 1 3 0 2 1 3 0 2 4 3 0 2 4 1
 [99,] 0 2 4 1 3 4 1 3 0 2 3 0 2 4 1 2 4 1 3 0 1 3 0 2 4
[100,] 0 4 3 2 1 1 0 4 3 2 2 1 0 4 3 3 2 1 0 4 4 3 2 1 0
```

Podemos ver que uno de los ciclos es el que sólo tiene el 0, que tiene periodo 0, y el otro ciclo que se repite es el de la secuencia siguiente, con periodo 22:

1 0 1 1 2 3 0 3 3 1 4 0 4 4 3 2 0 2 2 4 1 0

□

8. Genera $10{,}000$ números con una semilla de $Z_0 = 1$ usando el generador $Z_n = 7^5 Z_{n-1} \mod (2^{31}-1)$ Clasifica los números en 10 celdas de igual tamaño y prueben por uniformidad usando la prueba χ^2 con un nivel de confianza del 90 %. Aplicar también la prueba de rachas.

Solución.

Usamos la misma función que en un problema previo (problema 5), con los parámetros dados.

```
options(scipen=10) #para poder ver los números sin notación científica
Z <- 1 #valor inicial dado.
for (i in 2:10000) Z[i] <- glc(Z[i-1],a=7^5,c=0,m=2^31-1)
Z <- Z/(2^31-1)
head(Z)
```

```
[1] 0.0000000004656613 0.0000078263692594 0.1315377881431662
[4] 0.7556053221950332 0.4586501319234493 0.5327672374121692
```

Ahora, usamos la función que definí en clase (o cualquier otra que les sirva)

```
prueba.chisq.uniforme <- function(x,k=ceiling(length(x)/5)){
        n <- length(x)
        part <- seq(0,1,length=k+1) #partición
        z <- hist(x,breaks = part, plot = F)$counts
        ch <- (k/n)*sum((z-n/k)^2) #estadística chi
        pval <- pchisq(ch,k-1,lower.tail = F)
        return(list(part=part,freqs = z, estadística = ch, pval = pval))
}
prueba.chisq.uniforme(Z,k=10)
```

```
$part
 [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0

$freqs
 [1]  994 1007  998  958 1000 1049  989  963 1026 1016

$estadística
[1] 6.676

$pval
[1] 0.6708111
```

Entonces, con un alto p-value, no tenemos evidencia para rechazar la hipótesis de uniformidad.
Para la prueba de rachas:

```
nrachas <- function(x){
        n <- length(x)
        signo <- x[-1] - x[-n]
        s <- ifelse(signo<0,-1,1)
        R <- 1 + sum(s[-1] != s[-(n-1)]) #cuenta los cambios de signo
        return(R)
}
nr <- nrachas(Z)
z <- (nr-(2*length(Z)-1)/3)/sqrt((16*nr-29)/90);z
```

```
[1] -1.672862
```

```
pnorm(z,.05) #para prueba de dos lados al nivel 90%
```

```
[1] 0.04245673
```

En este caso, el p-value es 0.0424567, que de acuerdo al criterio dado, es significativo, por lo que hay evidencia marginal para rechazar la hipótesis de independencia, al 90 % de confianza.

□

9. El método del cuadrado medio de John von Neumann es el siguiente: comenzando con $Z_0 \in \{0, 1, \ldots, 99\}$, definir Z_n para $n \in \mathbb{N}$ a ser los dos dígitos de enmedio del número de 4 dígitos Z_{n-1}^2. Si Z_{n-1}^2 no tiene 4 dígitos, se le pegan a la izquierda con ceros. Por ejemplo, si $Z_0 = 64$, tenemos que $Z_0^2 = 4096$ y entonces $Z_1 = 09 = 9$. En el siguiente paso, encontramos que $Z_1^2 = 81 = 0081$, así que $Z_2 = 08 = 8$.

   - Escriban una función que calcule Z_n a partir de Z_{n-1}.
   - La salida del cuadrado medio tiene bucles. Por ejemplo, una vez que $Z_N = 0$, tendremos que $Z_n = 0$ para toda $n \geq N$. Escriban un programa que encuentre todos los ciclos del método del cuadrado medio y lístenlos.
   - Comenten sobre la calidad del método como generador de números aleatorios.

Solución.

La función que se solicita es, en el caso de `R`, la siguiente, que es una ligera modificación de la función que vimos en clase (la vista en clase, se terminaba cuando el número era muy corto, y no agregaba ceros a la izquierda). Esta función no da sólo el siguiente valor, sino que calcula toda la secuencia:

```
cm <- function(x){
  #a partir de la semilla x, se genera una sucesión de valores,
  #tomando los valores de enmedio de la serie
  u <- x/10^nchar(as.character(x))
  z <- x
  repeat{
    #verifica que el número tenga suficientes dígitos
    z <- z^2
    n <- nchar(z)
    if (n < 4) z <- paste(as.character(rep(4-n,0)),as.character(z),sep="")
    #partiendo de un tamaño de 7 dígitos, escogemos los dígitos centrales y vamos recorriendo
    z <- as.numeric(as.character(substr(z,floor(n/2),floor(n/2)+1)))
    u <- append(u,z/10000)
    if ((u[length(u)] == 0 ) | length(unique(u)) < length(u)) break #si el último número es 0 o ya se repiten termina
  }
  return(u)
}
cm(64) #ejemplo.
```

```
 [1] 0.6400 0.0009 0.0081 0.0056 0.0013 0.0016 0.0025 0.0062 0.0084 0.0005
[11] 0.0025
```

El siguiente inciso pide calcular las secuencias que se generan con cada número inicial. Podemos construir una lista con los ciclos de cada semilla.

```
Ciclo <- list(NULL)
for (i in 0:99) Ciclo[[i+1]] <- cm(i)
Ciclo[1:10] #ejemplos
```

```
[[1]]
[1] 0 0

[[2]]
[1] 0.1000 0.0001 0.0001
```

```
[[3]]
[1] 0.2000 0.0004 0.0016 0.0025 0.0062 0.0084 0.0005 0.0025

[[4]]
 [1] 0.3000 0.0009 0.0081 0.0056 0.0013 0.0016 0.0025 0.0062 0.0084 0.0005
[11] 0.0025

[[5]]
[1] 0.4000 0.0016 0.0025 0.0062 0.0084 0.0005 0.0025

[[6]]
[1] 0.5000 0.0025 0.0062 0.0084 0.0005 0.0025

[[7]]
[1] 0.6000 0.0036 0.0029 0.0084 0.0005 0.0025 0.0062 0.0084

[[8]]
[1] 0.7000 0.0049 0.0040 0.0060 0.0060

[[9]]
 [1] 0.8000 0.0064 0.0009 0.0081 0.0056 0.0013 0.0016 0.0025 0.0062 0.0084
[11] 0.0005 0.0025

[[10]]
 [1] 0.9000 0.0081 0.0056 0.0013 0.0016 0.0025 0.0062 0.0084 0.0005 0.0025
```

```
unlist(lapply(Ciclo,length))
```

```
  [1]  2  3  8 11  7  6  8  5 12 10  3 11 10  7  9  7  6 15 10  8  4 14  6
 [24]  6  4  6  7 12 14  6  4 16  9 13  8  7  7  8 14  6  3  7 16  6 13  9
 [47] 12  5  5  4  3  3  5  5 16  9  8  4  8  6  3 12  6 16 11  7  8  6  6
 [70] 16  4  8 11 10  6  6 15 14 13  4  4  9 12  8  6  7  7  9  7 14  3 15
 [93] 13 12  9  9 15  4  3  5
```

Ya vimos que este generador no es bueno, ya que genera ciclos muy cortos, los ciclos más largos en este ejemplo son de longitud 16.

□

10. Generar 15 números usando la semilla $Z_0 = 1$ del siguiente generador: $Z_n = (5Z_{n-1} + 1) \mod 16$. Hacer una prueba de Kolmogorov-Smirnov al 95 % de confianza.

Solución.

El ejercicio es mecánico, con todo lo que ya hicimos previamente

```
seed <- 1
GLC <- function() {
seed «- (seed * 5 + 1) %% (16)
seed/(16)
}
u <- NULL
for(i in 1:15) u[i] <- GLC()
u
```

```
 [1] 0.3750 0.9375 0.7500 0.8125 0.1250 0.6875 0.5000 0.5625 0.8750 0.4375
[11] 0.2500 0.3125 0.6250 0.1875 0.0000
```

```
ks.test(sort(u,decreasing=F),"punif")
```

```
One-sample Kolmogorov-Smirnov test

data:  sort(u, decreasing = F)
D = 0.066667, p-value = 1
alternative hypothesis: two-sided
```

No tenemos evidencia para rechazar la hipótesis de uniformidad.

□

11. La página PI DAY (http://www.piday.org/million/) contiene el primer millón de dígitos de π. Considerando estos dígitos:

    - Realizar un histograma y verificar la hipótesis de que los dígitos corresponden a una distribución uniforme discreta.
    - Verificar independencia de los dígitos, considerando las pruebas de gaps, de poker y de rachas.

      Una idea de ver los datos está en la siguiente imagen:

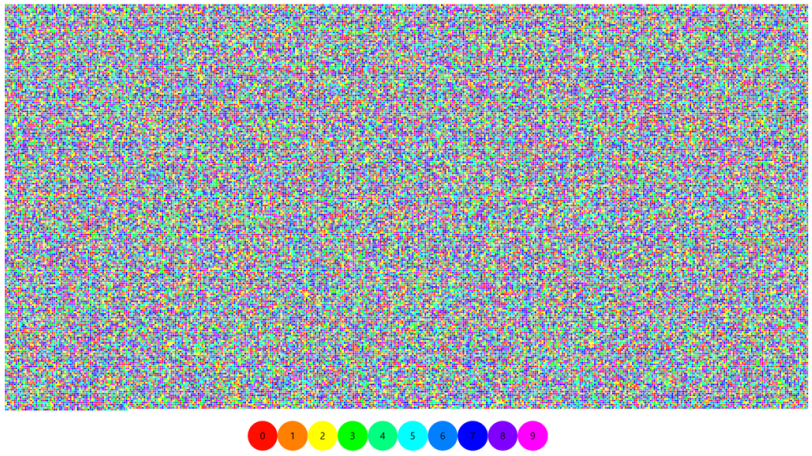


Figura 1: Cómo se ven los primeros 100,000 dígitos de π.

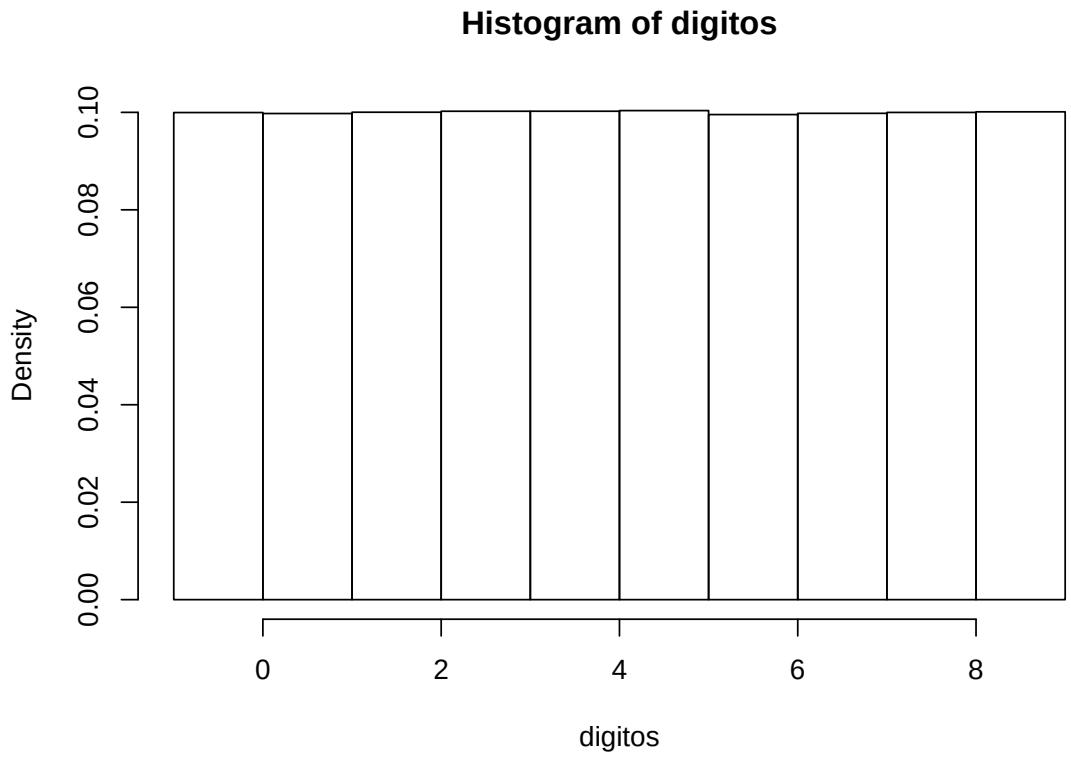
Solución.

En este ejercicio el tema es obtener los dígitos de π en un archivo.

```
digitos <- scan(file="https://www.angio.net/pi/digits/pi1000000.txt",
                what="character")
digitos <- as.numeric(unlist(strsplit(digitos,""))[-(1:2)])
table(digitos)
```

```
digitos
     0      1      2      3      4      5      6      7      8      9
 99959  99758 100026 100229 100230 100359  99548  99800  99985 100106
```

```
hist(digitos,breaks=-1:9, probability=T)
```




```
o <- table(digitos)
e <- rep(100000,10)
chi2 <- sum((o-e)^2/e)
pchisq(q=chi2,df=9,lower.tail=F)
```

```
[1] 0.7878669
```

Las pruebas de independencia a continuación.

Prueba de rachas:

```
library(randtests)
runs.test(digitos)
```

```
Runs Test

data:  digitos
statistic = 1.4872, runs = 445040, n1 = 499800, n2 = 399970, n =
899770, p-value = 0.137
alternative hypothesis: nonrandomness
```

Prueba de gaps:

```
library(randtoolbox)
```

```
Loading required package: rngWELL
This is randtoolbox. For overview, type 'help("randtoolbox")'.

Attaching package: 'randtoolbox'
```

```
The following object is masked from 'package:randtests':

    permut


gap.test(digitos)



 Gap test

chisq stat = 123580, df = 20, p-value = 0

 (sample size : 1000000)

length observed freq theoretical freq
1   81050   125000
2   8100   62500
3   781   31250
4   84   15625
5   6   7812
6   0   3906
7   0   1953
8   0   977
9   0   488
10   0   244
11   0   122
12   0   61
13   0   31
14   0   15
15   0   7.6
16   0   3.8
17   0   1.9
18   0   0.95
19   0   0.48
20   0   0.24
21   0   0.12
```

La prueba de gaps no la pasa, pero sí la de rachas.

□

12. Si dos dados están cargados de tal manera que en un dado, el valor 1 aparecerá exactamente el doble de veces que los otros valores, y el otro dado está igualmente cargado hacia el 6, calculen la probabilidad p_s de que un total exactamente igual a s aparecerá en la suma de los dos dados, para $2 \leq s \leq 12$.

Solución.

Para que el dado con el valor 1 cargado tenga el doble de probabilidad que el resto de las caras, se debe cumplir que $P(X = 1) = 2k$ donde $k = P(X = j)$ para $j \neq 1$. Entonces $2k + 5k = 1$ por lo que $k = 1/7$. Lo mismo pasará para el otro dado, pero con la cara 6 cargada.

Entonces podemos construir la tabla siguiente:

$S = i$	2	3	4	5	6	7	8	9	10	11	12
$P(S = i)$	$\frac{2}{49}$	$\frac{3}{49}$	$\frac{4}{49}$	$\frac{5}{49}$	$\frac{6}{49}$	$\frac{9}{49}$	$\frac{6}{49}$	$\frac{5}{49}$	$\frac{4}{49}$	$\frac{3}{49}$	$\frac{2}{49}$

□

13. Algunos dados que fueron cargados como se describe en el problema anterior se lanzaron 144 veces, y se observaron los siguientes valores:

Valor de $s =$	2	3	4	5	6	7	8	9	10	11	12
número observado Y_s	2	6	10	16	18	32	20	13	16	9	2

Apliquen la prueba χ^2 a estos valores, usando las probabilidades teóricas para dos dados 'honestos'? ¿La prueba detecta que los dados no son honestos? Si no, explicar porqué no.

Solución.

La distribución esperada está dada, así que la prueba es directa:

```
o <- c(2,6,10,16,18,32,20,13,16,9,2)
e <- 144*c(1,2,3,4,5,6,5,4,3,2,1)/36
chi2 <- sum((o-e)^2/e)
pchisq(q=chi2,df=11,lower.tail=F)


[1] 0.74
```

La conclusión de la prueba es que no se tiene suficiente evidencia estadística para rechazar la hipótesis nula de que las muestra proviene de esos dados, esto es, la prueba no detecta que los datos no son honestos.

Esto se debe a dos razones: 1) Las probabilidades no son muy diferentes entre sí, la suma de los dados tiende a aminorar las diferencias, y 2) la muestra es relativamente pequeña para detectar las diferencias.

Por ejemplo, si simulo 1000 lanzamientos en lugar de 144:

```
x <- sample(2:12,size=1000,replace=T,prob=c(2,3,4,5,6,9,6,5,4,3,2)/49)
o <- table(x)
e <- 1000*c(1,2,3,4,5,6,5,4,3,2,1)/36
chi2 <- sum((o-e)^2/e)
pchisq(q=chi2,df=11,lower.tail=F)


[1] 0.00031
```

□