Simulación ITAM
8 de septiembre de 2020 2020-II

Tarea 1.

La fecha de entrega es el **24 de agosto de 2020**. Enviar **antes de la medianoche** al correo: jorge.delavegagongora@gmail.com (o se convierte en calabaza).

Lecturas

- Robert & Casella Capítulo 2 sección 2.1 y 2.2.
- Dagpunar Capítulo 2
- Good random number generators are (not so) easy to find
- Linear Congruential Generator in R

Problemas

1. Lanzar una moneda honesta 500 veces y hacer una gráfica de:

   i r/n vs n, para $n = 1, 2, \ldots, 500$, donde n es el número de lanzamientos y r es el número de soles para esos n lanzamientos; y

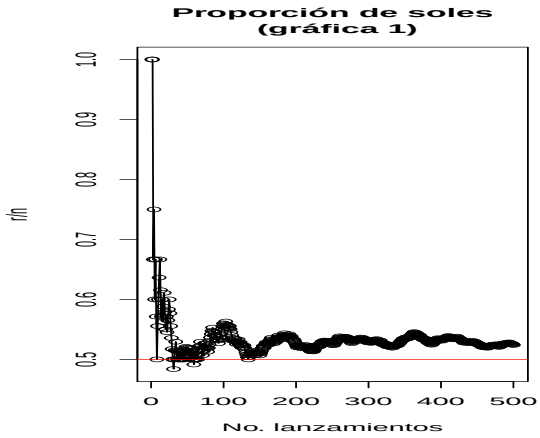
   ii. $(2r - n)$ vs n, la diferencia entre el número de soles y águilas.

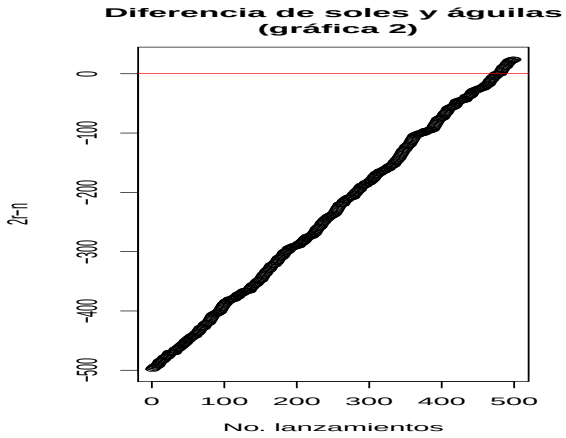
   Comentar sobre el comportamiento de r/n y $(2r - n)$

   Solución.

   Podemos simular el lanzamiento de una moneda generando variables aleatorias Bernoulli con parámetro $p = 0.5$.

```
x <- rbinom(n=500,size=1,prob=0.5)
par(mfrow=c(1,2))
plot(1:500,cumsum(x)/(1:500),type = "o",xlab = "No. lanzamientos", ylab = "r/n",
     main = "Proporción de soles\n (gráfica 1)")
abline(h = 0.5, col = "red")
plot(1:500,(2*cumsum(x)-rep(500,500)),type = "o",xlab = "No. lanzamientos", ylab = "2r-n",
     main = "Diferencia de soles y águilas\n (gráfica 2)")
abline(h=0.5,col="red")
```





Podemos ver que conforme se aumenta el número de lanzamientos, el valor de r/n se aproxima al valor de $p = 0.5$. La diferencia entre águilas y soles se aproxima a 0 conforme aumenta el tamaño de la muestra.

□

2. Dar 5 ejemplos de procesos en los que se puede utilizar simulación.

Solución.

a) Muchos modelos de la vida real son procesos de linea de espera o de teoría de inventarios, estos pueden incluir, por ejemplo: tráfico aéreo, lista de trabajos en una computadora, clientes en gasolineras, distribución de billetes y monedas, llegadas y partidas de trenes en estaciones europeas, etc.

b) Un modelo típico de simulación se da en las finanzas, donde es necesario estimar el precio de instrumentos que dependen de condiciones de azar. como las opciones.

c) En problemas de riesgo de crédito, resulta interesante medir el efecto de las correlaciones entre los acreditados. Incorporar modelos de correlación resulta complicado de modelar matemáticamente, por lo que la simulación es apropiada.

d) Flujos de trabajo para medir tiempos de ejecución de tareas y encontrar rutas críticas en proyectos. En la Universidad Técnica de Eindhoven, en Alemania, Wil van der Aalst se ha dedicado al estudio de la minería de procesos, que se ha aplicado a instituciones como hospitales, banca central, etc.

e) Como ya se ha mencionado en el curso, se puede aplicar la simulación a procesos de optimización e integración de funciones con estructuras complejas.

□

3. Una canoa que contiene tres mujeres y tres hombres llega a una isla deshabitada. Discutan la información que requieren para modelar la sociedad de estos individuos y cómo el tamaño de la población crece con el tiempo.

Solución.

Se pueden hacer diferentes supuestos para este problema. Por ejemplo, podemos suponer que sólo un hombre es fértil, que las mujeres sólo pueden tener un determinado número de hijos en su periodo reproductivo, etc. O bien podemos hacer supuestos muy simples y considerar esencialmente árboles de expansión, por ejemplo.

Con el supuesto inicial simple en el que cada hombre mantiene su relación con la misma mujer, y que se puede tener un bebé que vive con probabilidad p cada año y que es hombre o mujer con probabilidad w, y que la edad reproductiva comienza a los 18 años, etc., se podría modelar cómo va creciendo la población y calcular cosas como el tiempo promedio en el que se duplica la población.

□

4. Considerar cómo podrían simular el siguiente modelo de una sala de cirugía que opera bajo citas:

   - Los pacientes se programan para llegar en cada 5 horas.
   - Independientemente de los otros pacientes, cada paciente falla a su cita con probabilidad 0.1
   - Independientemente de los otros pacientes, cada paciente tiene tiempos de llegada con la siguiente distribución:

Tiempo	2 hrs antes	1 hra antes	a tiempo	1 hra tarde	2 hrs tarde
probabilidad	1/10	1/5	2/5	1/5	1/10

   - Los tiempos de consulta tienen la siguiente distribución:

Tiempo en hrs	2	3	4	5	6	7	8	9
probabilidad	1/10	1/10	1/10	1/5	1/5	1/10	1/10	1/10

   - Los pacientes se atienden en el orden en el que llegan.

Solución.

a) Creamos la lista de llegadas. Se genera una posible llegada de un cliente I cada 5 horas como una variable aleatoria **Bernoulli** (0.9) para determinar si el cliente llega o no llega.

b) Sea $X|I = 1$, el tiempo en el que el cliente llega a su cita. Entonces $(X|I = 1) = 5 + u$, donde u tiene la distribución de probabilidad dada para las llegadas.

c) Para cada llegada, hay que generar la duración de la consulta S, con la distribución dada. Ya con la duración y el tiempo de llegada, se puede determinar el tiempo de salida.

Por ejemplo, podemos hacer un ejemplo sencillo: suponiendo que tenemos $n = 100$ citas.

```r
n <- 100
X <- D <- numeric(n) #define un vector
(I <- rbinom(n,1,0.9)) #para determinar los clientes que llegaron
```

```
  [1] 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [38] 1 1 1 0 1 1 0 0 1 1 1 1 1 1 1 0 1 0 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1
 [75] 0 1 1 1 1 0 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1
```

```r
k <- length(which(I == 1)) #cuenta qué pacientes llegaron
X[which(I == 1)] <- 5 + sample(-2:2,k,prob=c(0.1,0.2,0.4,0.2,0.1),replace=T)
# duraciones de atención
D[which(I == 1)] <- sample(2:9,k,prob=c(0.1,0.1,0.1,0.2,0.2,0.1,0.1,0.1),replace=T)
W <- cumsum(X) #tiempos de llegadas
(S <- W + D) # tiempos de salida
```

```
  [1]  12  18  19  17  29  35  38  47  49  54  60  64  68  79  79  88  91 100
 [19] 100 108 115 122 123 131 140 138 145 152 157 160 169 173 177 183 184 191
 [37] 193 200 202 211 205 215 221 216 216 225 235 234 239 249 253 258 254 265
 [55] 259 267 275 276 287 286 292 297 298 308 302 309 317 324 326 328 336 337
 [73] 349 347 344 352 360 368 370 364 371 383 383 388 393 391 391 399 412 415
 [91] 416 426 430 438 442 440 450 451 460 463
```

```r
# obtenemos cuántos clientes en espera por cada 5 horas
table(cut(S,breaks=seq(0,500,by=5)))
```

```
    (0,5]     (5,10]    (10,15]    (15,20]    (20,25]    (25,30]    (30,35]    (35,40]
        0          0          1          3          0          1          1          1
   (40,45]    (45,50]    (50,55]    (55,60]    (60,65]    (65,70]    (70,75]    (75,80]
        0          2          1          1          1          1          0          2
   (80,85]    (85,90]    (90,95]   (95,100]  (100,105]  (105,110]  (110,115]  (115,120]
        0          1          1          2          0          1          1          0
 (120,125]  (125,130]  (130,135]  (135,140]  (140,145]  (145,150]  (150,155]  (155,160]
        2          0          1          2          1          0          1          2
 (160,165]  (165,170]  (170,175]  (175,180]  (180,185]  (185,190]  (190,195]  (195,200]
        0          1          1          1          2          0          2          1
 (200,205]  (205,210]  (210,215]  (215,220]  (220,225]  (225,230]  (230,235]  (235,240]
        2          0          2          2          2          0          2          1
 (240,245]  (245,250]  (250,255]  (255,260]  (260,265]  (265,270]  (270,275]  (275,280]
        0          1          2          2          1          1          1          1
 (280,285]  (285,290]  (290,295]  (295,300]  (300,305]  (305,310]  (310,315]  (315,320]
        0          2          1          2          1          2          0          1
 (320,325]  (325,330]  (330,335]  (335,340]  (340,345]  (345,350]  (350,355]  (355,360]
        1          2          0          2          1          2          1          1
 (360,365]  (365,370]  (370,375]  (375,380]  (380,385]  (385,390]  (390,395]  (395,400]
        1          2          1          0          2          1          3          1
 (400,405]  (405,410]  (410,415]  (415,420]  (420,425]  (425,430]  (430,435]  (435,440]
        0          0          2          1          0          2          0          2
 (440,445]  (445,450]  (450,455]  (455,460]  (460,465]  (465,470]  (470,475]  (475,480]
        1          1          1          1          1          0          0          0
 (480,485]  (485,490]  (490,495]  (495,500]
        0          0          0          0
```

□

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

Solución.

Este es un generador que no tiene periodo completo, porque falla la condición 2 del teorema de Hull y Dobell:

- c y m son primos relativos
- $m = 31$ es primo, y la condición dice que si hay un factor primo de m, debe dividir a $a - 1$, que no se cumple,
- Se cumple por excepción.

comprobando empíricamente:

```
x <- NULL
for(j in 1:31){
        x[1] <- j
        for(i in 2:16){x[i] <- (5*x[i-1]+3) %% 31}
        print(x)
}
```

```
 [1]  1  8 12  1  8 12  1  8 12  1  8 12  1  8 12  1
 [1]  2 13  6  2 13  6  2 13  6  2 13  6  2 13  6  2
 [1]  3 18  0  3 18  0  3 18  0  3 18  0  3 18  0  3
 [1]  4 23 25  4 23 25  4 23 25  4 23 25  4 23 25  4
 [1]  5 28 19  5 28 19  5 28 19  5 28 19  5 28 19  5
 [1]  6  2 13  6  2 13  6  2 13  6  2 13  6  2 13  6
 [1] 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7
 [1]  8 12  1  8 12  1  8 12  1  8 12  1  8 12  1  8
 [1]  9 17 26  9 17 26  9 17 26  9 17 26  9 17 26  9
 [1] 10 22 20 10 22 20 10 22 20 10 22 20 10 22 20 10
 [1] 11 27 14 11 27 14 11 27 14 11 27 14 11 27 14 11
 [1] 12  1  8 12  1  8 12  1  8 12  1  8 12  1  8 12
 [1] 13  6  2 13  6  2 13  6  2 13  6  2 13  6  2 13
 [1] 14 11 27 14 11 27 14 11 27 14 11 27 14 11 27 14
 [1] 15 16 21 15 16 21 15 16 21 15 16 21 15 16 21 15
 [1] 16 21 15 16 21 15 16 21 15 16 21 15 16 21 15 16
 [1] 17 26  9 17 26  9 17 26  9 17 26  9 17 26  9 17
 [1] 18  0  3 18  0  3 18  0  3 18  0  3 18  0  3 18
 [1] 19  5 28 19  5 28 19  5 28 19  5 28 19  5 28 19
 [1] 20 10 22 20 10 22 20 10 22 20 10 22 20 10 22 20
 [1] 21 15 16 21 15 16 21 15 16 21 15 16 21 15 16 21
 [1] 22 20 10 22 20 10 22 20 10 22 20 10 22 20 10 22
 [1] 23 25  4 23 25  4 23 25  4 23 25  4 23 25  4 23
 [1] 24 30 29 24 30 29 24 30 29 24 30 29 24 30 29 24
 [1] 25  4 23 25  4 23 25  4 23 25  4 23 25  4 23 25
 [1] 26  9 17 26  9 17 26  9 17 26  9 17 26  9 17 26
 [1] 27 14 11 27 14 11 27 14 11 27 14 11 27 14 11 27
 [1] 28 19  5 28 19  5 28 19  5 28 19  5 28 19  5 28
 [1] 29 24 30 29 24 30 29 24 30 29 24 30 29 24 30 29
 [1] 30 29 24 30 29 24 30 29 24 30 29 24 30 29 24 30
 [1] 31  3 18  0  3 18  0  3 18  0  3 18  0  3 18  0
```

El periodo es 3.

□

6. 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

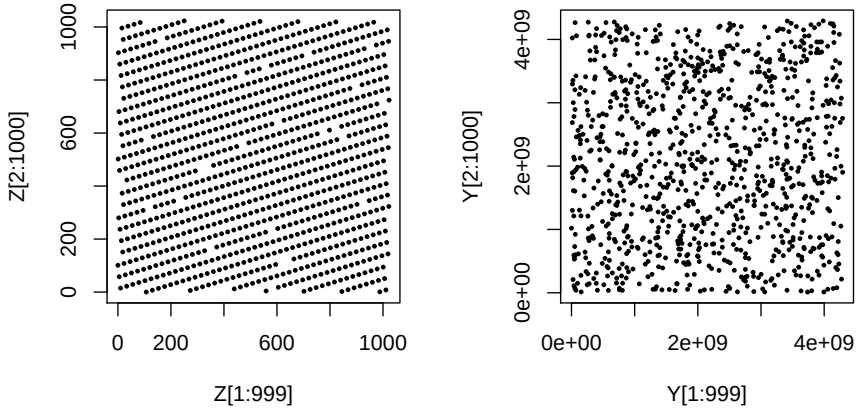
$\bar{U} = \frac{(m-1)m}{2m^2} = \frac{1}{2} - \frac{1}{2m}$.

□

7. 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.

```r
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],pch=16,cex=0.5)
plot(Y[1:999],Y[2:1000],pch=16,cex=0.5)
```



□

8. 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)

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

□

9. Un generador de Fibonacci obtiene el valor 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 0 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 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 4 4 4 4 4 3 3 3 3 3 2 2 2 2 2 1 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 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 0 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 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 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 0 4 4 4 4 4 3 3 3 3 3 2 2 2 2 2 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 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

□

10. 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.
- Hacer un diagrama como el mostrado en clase.

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] 1e-01 1e-04 1e-04

[[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  6  4
 [26]  6  7 12 14  6  4 16  9 13  8  7  7  8 14  6  3  7 16  6 13  9 12  5  5  4
 [51]  3  3  5  5 16  9  8  4  8  6  3 12  6 16 11  7  8  6  6 16  4  8 11 10  6
 [76]  6 15 14 13  4  4  9 12  8  6  7  7  9  7 14  3 15 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.

□

11. La siguiente página 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 Figura 11:

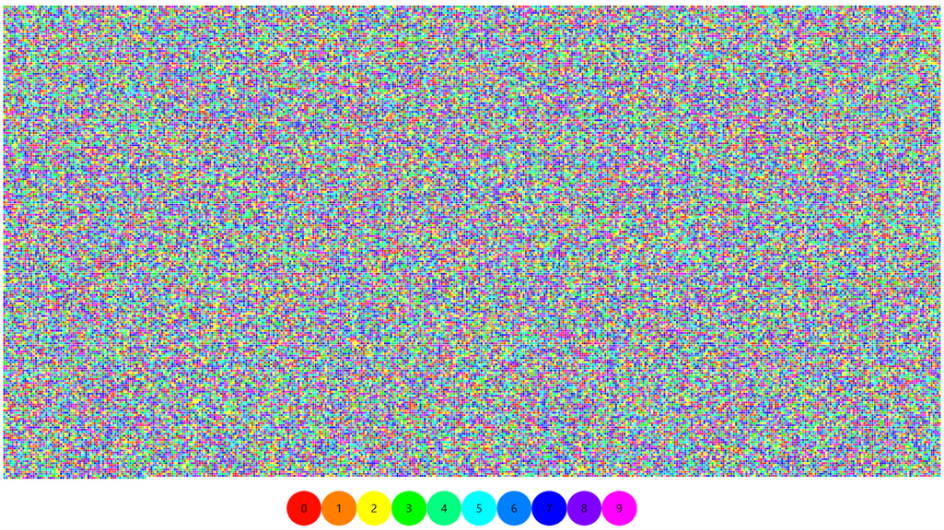


Figura 1: Cómo se ven los primeros 100,000 dígitos de π.

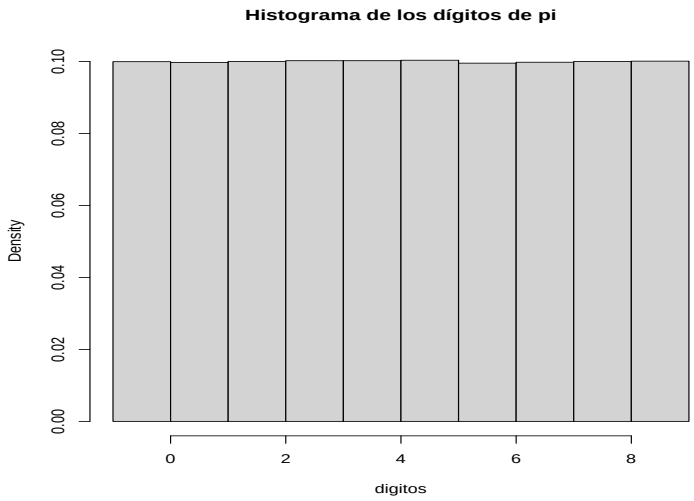
Solución.

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

```r
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
```

```r
hist(digitos, breaks = -1:9, probability = T,
     main = "Histograma de los dígitos de pi")
```




```r
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:

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

```
Runs Test

data:  digitos
statistic = 1.4872, runs = 445045, n1 = 499798, n2 = 399972, n =
899770, p-value = 0.137
alternative hypothesis: nonrandomness
```

Prueba de gaps:

```r
library(randtoolbox)
```

```
Loading required package: rngWELL
This is randtoolbox. For an 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 cumplir con la condición, sea p la probabilidad de cualquiera de las caras del 2 al 6. Entonces $2p + 5p = 1$ y por lo tanto $p = 1/7$. Entonces en el primer dado el 1 aparecerá con probabilidad $2/7$ y el resto de los números con probabilidad $1/7$, y en el otro dado, la cara con el 6 es el que aparecerá con probabilidad $2/7$.

Para la suma, tenemos como ejemplo los siguientes:

$s = 2$: sólo se da con el par (1,1) con probabilidad $\frac{2}{49}$.

$s = 3$: Se puede dar con los pares (1,2) ($\frac{2}{49}$) o (2,1) ($\frac{1}{49}$). Entonces su probabilidad es $\frac{3}{49}$

$s = 4$: Se puede dar con (1,3) ($\frac{2}{49}$), (2,2) ($\frac{1}{49}$), (3,1) ($\frac{1}{49}$). En total, $\frac{4}{49}$

Y así sucesivamente, para finalmente obtener:

$s = 2$	$s = 3$	$s = 4$	$s = 5$	$s = 6$	$s = 7$	$s = 8$	$s = 9$	$s = 10$	$s = 11$	$s = 12$
$\frac{2}{49}$	$\frac{3}{49}$	$\frac{4}{49}$	$\frac{5}{49}$	$\frac{6}{49}$	$\frac{9}{49}$	$\frac{6}{49}$	$\frac{5}{49}$	$\frac{4}{49}$	$\frac{3}{49}$	$\frac{2}{49}$

□