Tarea 2.

La fecha de entrega es el **13 de septiembre de 2018**.

Lecturas

- Robert & Casella Cap.2 sección 2.1.1
- Dagpunar Cap.2
- Good random number generators are (not so) easy to find

Problemas

1. Probar por inducción matemática que para un GLC,

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

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 $k = 1$.
- Hipótesis de inducción: Ahora suponemos que el resultado es válido para un índice k. 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. ¿Qué se puede decir del periodo de $Z_i \equiv 630,360,016 Z_{i-1} \mod 2^{31} - 1$?

Solución.

En este ejemplo, correspondiente a un generador multiplicativo, $c = 0$. $m = 2^{31} - 1$, y $a = 630360016$. Con los teoremas vistos en clase, claramente, al violar la primera condición del teorema de Hull y Dobell. no puede tener periodo completo, ya que $c = 0$.

□

3. Sin calcular ninguna Z_i, determinar cuál de los siguientes GLC's mixtos tienen periodo completo.
   (a) $Z_i \equiv [13 Z_{i-1} + 13] \mod 16$.
   (b) $Z_i \equiv [12 Z_{i-1} + 13] \mod 16$.
   (c) $Z_i \equiv [13 Z_{i-1} + 12] \mod 16$.
   (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) (a) $c = 12$ y $m = 16$ NO son primos relativos. No tiene periodo 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)


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

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 en algunas máquinas, particularmente en las computadoras de 32 bits (no en las de 64 bits).

□

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. Generar 10,000 números con $\mathcal{U}(0,1)$ de Excel. Hacer un breve estudio para probar la calidad de los generadores; aplicar las pruebas de uniformidad e independencia a cada conjunto de datos. Resumir resultados en NO MAS de 2 cuartillas, incluyendo gráficas. De acuerdo a tus resultados, ¿cómo calificarías al generador de Excel?

Solución.

Haré el ejercicio equivalente con datos de `LibreOffice` que es el que uso yo bajo Linux (`LibreOffice` es una versión libre similar a Office de Microsoft). Los datos generados con el comando `ALEATORIO()` están guardados en el archivo `aleatoriosLibreOfficeT2-2018-II.csv` que se encontrará disponible en el apéndice para quien quiera consultarlo.

```
u <- scan("../Tareas/aleatoriosLibreOfficeT2-2018-II.csv")
head(u,20) #muestra los primeros 20 datos


 [1] 0.66240915 0.50384770 0.90719733 0.39324029 0.74814657 0.72356900
 [7] 0.39846114 0.13220532 0.18769793 0.56067698 0.82218120 0.57759397
[13] 0.28664155 0.85749071 0.57103053 0.08885868 0.89685290 0.16818367
[19] 0.30467353 0.60118267
```

```r
length(u)
```

```
[1] 10000
```

El estudio tiene que verificar lo siguiente:

- si los datos se pueden considerar provenientes de una distribución uniforme (con pruebas de bondad de ajuste, qq-plots, histogramas)
- si los datos son independientes (con pruebas como rachas, gaps, función de autocorrelación)

Lo primero que podemos hacer es obtener algunas estadísticas descriptivas de la muestra y hacer un histograma, junto con un qq-plot

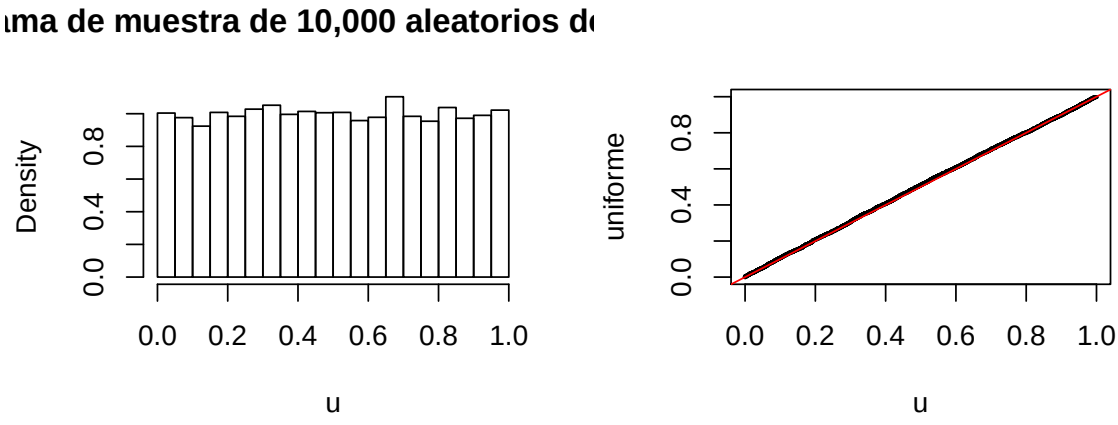
```r
summary(u)
```

```
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max. 
0.0000975 0.2547939 0.5002788 0.5010814 0.7486289 0.9999213 
```

```r
var(u)
```

```
[1] 0.08282431
```

```r
par(mfrow = c(1,2))
hist(u,prob = T, breaks = 20, main = "Histograma de muestra de 10,000 aleatorios de LibreOffice")
qqplot(u,runif(10000),ylab="uniforme",pch=16,cex=0.3)
abline(a=0,b=1,col="red")  #agrega una linea identidad para comparar
```




```r
ks.test(u,"punif")
```

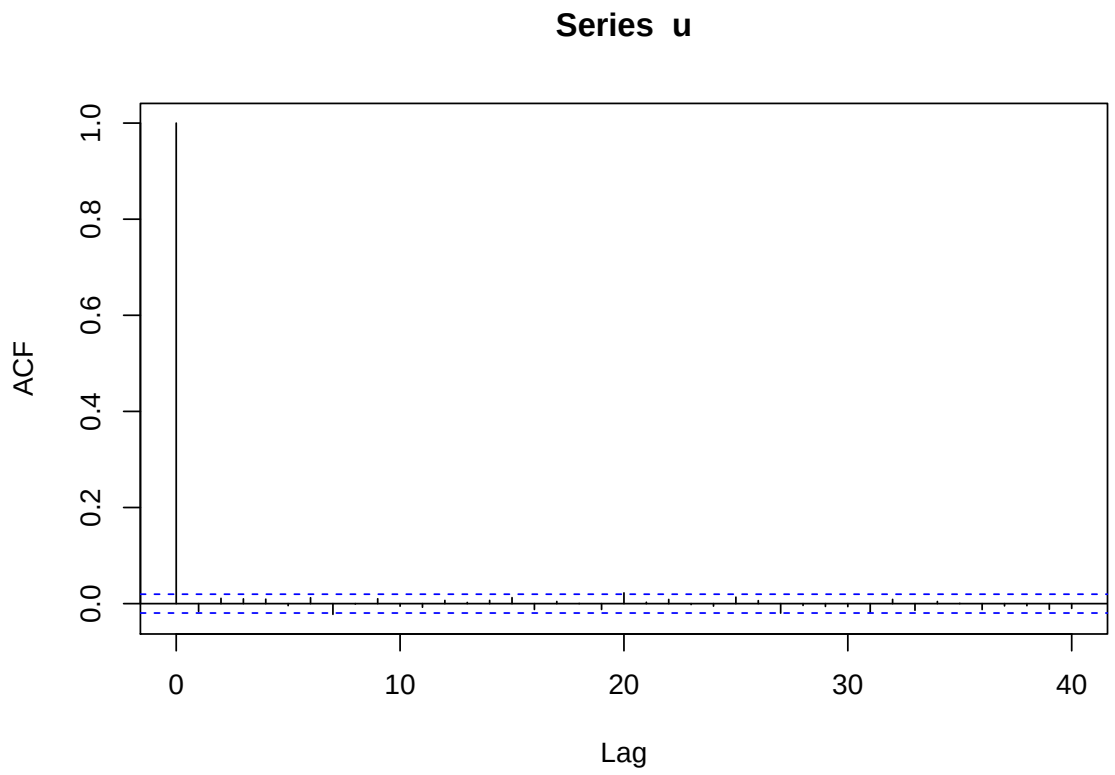
```
One-sample Kolmogorov-Smirnov test

data:  u
D = 0.0064599, p-value = 0.7982
alternative hypothesis: two-sided
```

Podemos ver que la media y la mediana casi coinciden en el valor de la media y mediana de una distribución uniforme en (0,1), que es 0.5, y que la varianza es muy cercana también a 1/12. El histograma y el qq-plot muestran que los datos sí se comportan de manera uniforme muy aproximadamente.

La prueba de Kolmogorov-Smirnov nos dice que no podemos rechazar la hipótesis nula ya que el p-value resultante es muy alto.

Para verificar independencia, podemos considerar la función de autocorrelación, la prueba de rachas (disponible como vimos en `randtoolbox`) y la prueba de gaps (disponible en `randtests` (también podemos usar las construidas por nosotros).

```
acf(u)
```




```
library(randtoolbox)
```

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

```
gap.test(u)
```

```

 Gap test

chisq stat = 13, df = 13, p-value = 0.42

 (sample size : 10000)
```

```
length observed freq theoretical freq
1   1255   1250
2   629   625
3   325   312
4   150   156
5   77   78
6   42   39
7   19   20
8   13   9.8
9   1   4.9
10   0   2.4
11   1   1.2
12   0   0.61
13   0   0.31
14   1   0.15
```

```
library(randtests)
```

```
Attaching package: 'randtests'
The following object is masked from 'package:randtoolbox':
    permut
```

```
runs.test(u)
```

```
Runs Test

data:  u
statistic = 0.5, runs = 5000, n1 = 5000, n2 = 5000, n = 10000,
p-value = 0.6
alternative hypothesis: nonrandomness
```

Podemos ver que el correlograma muestra que todas las autocorrelaciones de orden superior son no significativas. En el caso de la prueba de gaps y la prueba de rachas ambas tampoco son significativas.

En conclusión, parece que el generador de números aleatorios de la hoja de cálculo de `LibreOffice` pasa las diversas pruebas de aleatoriedad. Averiguando un poco más, el generador que usa `Calc LibreOffice` es el de Mersenne-Twister 19937.

□

6. Probar que la parte fraccional de la suma de uniformes en [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 0 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 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 0 2 2 2 2 2 4 4 4 4 4 1 1 1 1 1 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 0 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 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 0 4 4 4 4 4 3 3 3 3 3 2 2 2 2 2 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 0 2 2 2 2 2 4 4 4 4 4 1 1 1 1 1 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 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 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
 [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 0 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 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 0 2 2 2 2 2 4 4 4 4 4 1 1 1 1 1 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 $Z_n = 7^5 Z_{n-1}$ mód $(2^{31} - 1)$ 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 función `glc`.

```
glc <- function(Z0,a,c,m){
        Z <- (a*Z0 + c) %% m
        return(Z)
}
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] 0.00000000047 0.00000782637 0.13153778814 0.75560532220 0.45865013192
[6] 0.53276723741
```

Ahora, usamos la función `prueba.chisq.uniforme` 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] 6.7

$pval
[1] 0.67
```

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] -1.7
```

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

```
[1] 0.042
```

En este caso, el p-value es 0.04, que de acuerdo al criterio dado, es significativo, por lo que hay evidencia marginal para rechazar la hipótesis de independencia, al 90 % de confianza.

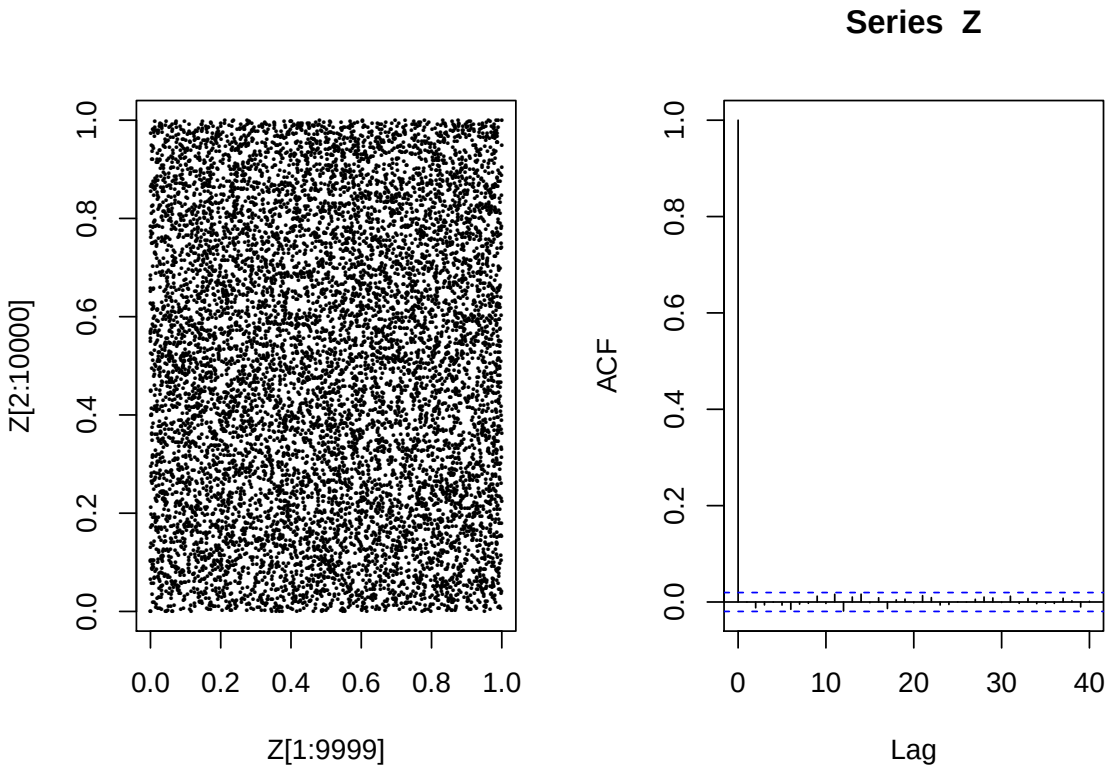
□

9. Aplicar a los datos del ejercicio las pruebas de correlación, gaps y poker.

Solución.

Se aplicarán las pruebas que les dí en clase. Primero haremos dos gráficas, la correspondiente a un valor de u_i y su rezago u_{i-1}, y la función de autocorrelación

```
par(mfrow=c(1,2))
plot(Z[1:9999],Z[2:10000],pch=16,cex=0.3) #hacemos los puntos muy chicos para evitar la saturación visual
acf(Z)
```



La gráfica de autocorrelación y la de rezagos no muestran comportamientos fuera de la uniformidad y de correlación en las observaciones.

Ahora consideremos las pruebas de gap y de póker que vimos en clase:

```
source("../scripts/pruebas.r")
prueba.poker(Z)


[[1]]
     Esperado Observado
[1,]     5040      5039
[2,]     4320      4287
[3,]      270       313
[4,]      360       343
[5,]       10        18

$Estadistica
[1] 14

$pval
[1] 0.0064


gap.test(Z)
```

```
 Gap test

chisq stat = 7.6, df = 13, p-value = 0.87

 (sample size : 10000)

length observed freq theoretical freq
1   1207   1250
2   642   625
3   313   312
4   161   156
5   70   78
6   34   39
7   22   20
8   12   9.8
9   5   4.9
10   1   2.4
11   1   1.2
12   0   0.61
13   1   0.31
14   0   0.15
```

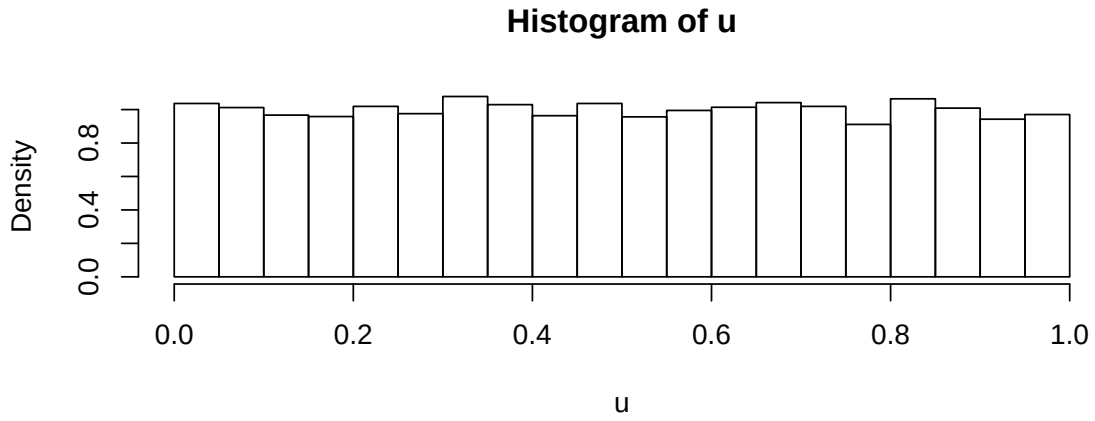
En esta muestra de valores, la muestra pasa la prueba de gaps, pero no pasa la prueba de póker. Recuerden que esto no implica que la muestra esté mal, ocasinalmente un conjunto de datos puede no pasar una prueba La prueba de póker es sencilla y sólo está tomando en consideración cuatro dígitos, probablemente debería de tomar más. En general la muestra se ve razonablemente uniforme.

□

10. Generar 1,500 números del generador RANDU. Hacer una prueba de Kolmogorov-Smirnov al 95 % de confianza.

Solución.

```
seed <- 100   #establece la semilla para la muestra RANDU
RANDU <- function() {seed <-(3 * seed+1) %% (64); seed/(64)} #generador RANDU
for(i in 1:1500) u <- c(u,RANDU()) #genera muestra de 1500 uniformes RANDU
head(u)


[1] 0.66 0.50 0.91 0.39 0.75 0.72


hist(u, prob=T, breaks=30)  #Un histograma para ver el comportamiento de los datos
```



```
#Hacemos la prueba
ks.test(u,"punif")


Warning in ks.test(u, "punif"): ties should not be present for the Kolmogorov-Smirnov test



One-sample Kolmogorov-Smirnov test

data:  u
D = 0.008, p-value = 0.4
alternative hypothesis: two-sided
```

Vemos que los datos en una dimensión se ven razonablemente bien, y que los datos pasan la prueba de uniformidad, aun cuando sabemos que el generador no es del todo bueno.

□

11. La página The number e to one million digits (https://apod.nasa.gov/htmltest/gifcity/e.1mil) contiene el primer millón de dígitos de e (pueden usar cualquier otra página). 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 (esta está hecha para π):

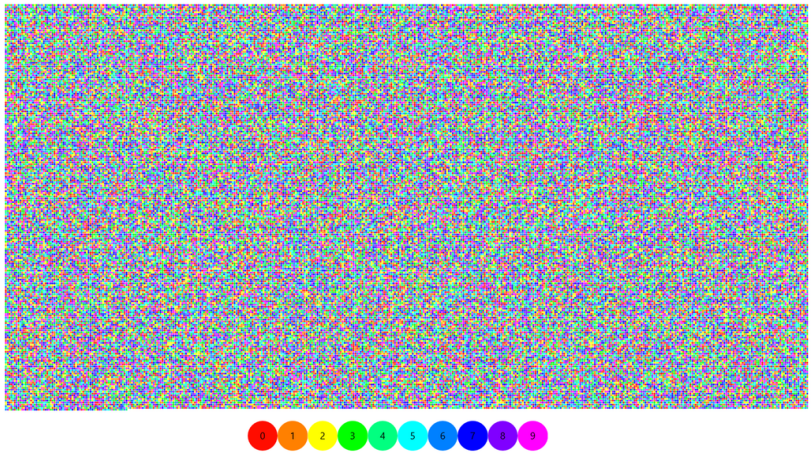


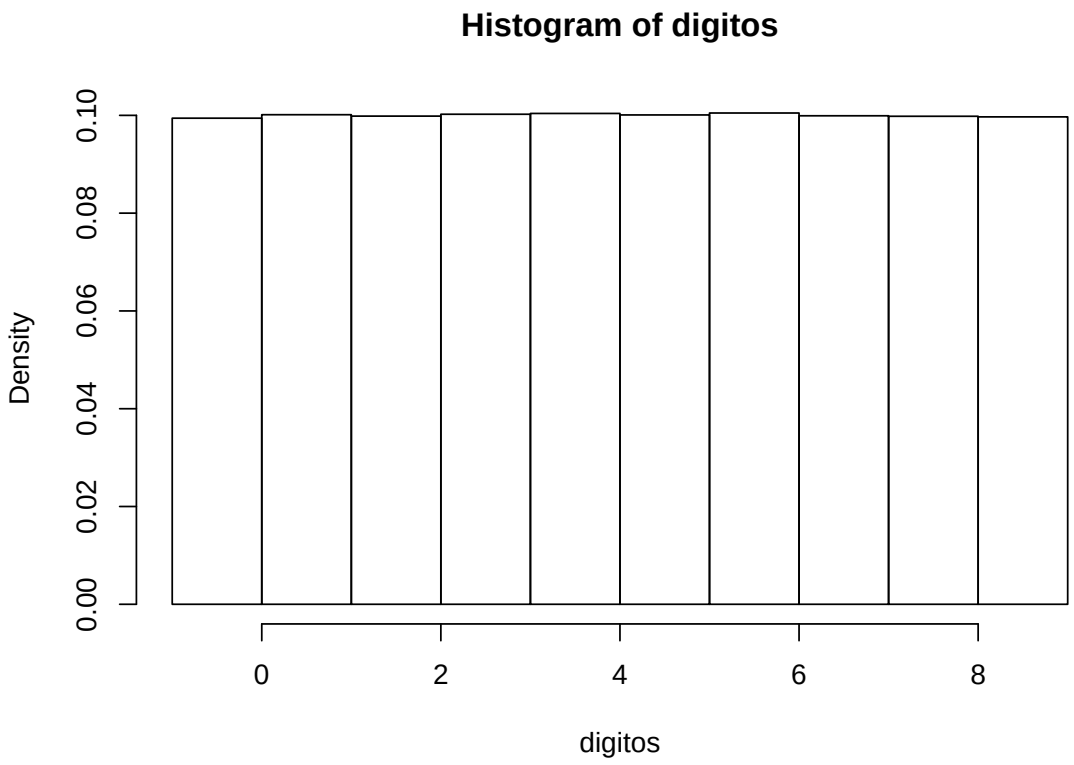
Figura 1: Cómo se ven los primeros 100,000 dígitos de π.

Solución.

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

```
#Para el número e tenemos que leer el archivo y convertir los decimales a dígitos

datos <- scan("https://apod.nasa.gov/htmltest/gifcity/e.1mil",skip=31,what="character")
datos <- datos[1:12502]  #eliminamos los últimos renglones que no tienen dígitos
```

```
#convertimos cada linea a dígitos

digitos <- as.numeric(unlist(strsplit(datos,""))[-(1:2)])


Warning: NAs introducidos por coerción


digitos <- digitos[!is.na(digitos)]  #quita los espacios en bloanco que había en las líneas
table(digitos)


digitos
     0      1      2      3      4      5      6      7      8      9
 99425 100136  99847 100231 100390 100089 100481  99913  99816  99692


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] 0.39
```

Podemos ver que los dígitos están distribuídos de manera uniforme. Las pruebas de independencia a continuación. Sería interesante ver otro tipo de pruebas para la distribución de los dígitos por pares, por tercias, etc.

Prueba de rachas:

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

```
Runs Test

data:  digitos
statistic = 1, runs = 400000, n1 = 500000, n2 = 400000, n =
900000, p-value = 0.2
alternative hypothesis: nonrandomness
```

Prueba de gaps:

```
library(randtoolbox)
gap.test(digitos)
```

```
 Gap test

chisq stat = 124002, df = 20, p-value = 0

 (sample size : 1000020)

length observed freq theoretical freq
1   80698   125002
2   7981   62501
3   815   31251
4   64   15625
5   8   7813
6   4   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 (curiosamente, lo mismo pasa con π).

□

12. Escriban un programa que utilice el método de la transformación inversa para generar números de la densidad siguiente: $f(x) = \frac{1}{x^2} I(x \geq 1)$. Para probar su programa, hagan un histograma de 10,000 números junto con la densidad f. Verificar la hipótesis de que la muestra sigue la distribución teórica dada y hacer un $qq-$plot e interpretar.

Solución.

La función de distribución correspondiente a esta densidad es $F(x) = \int_1^x u^{-2}\, du = -u^{-1}|_1^x = 1 - 1/x$. Invirtiendo la función obtenemos $x = \frac{1}{1-u}$.

```
x <- 1/(1-runif(10000))
x[1:100] #muestra
```

```
  [1]    1.1    1.4    1.3    5.5    2.6   55.0    2.0    1.7    1.0    1.1
 [11]    3.8   12.8    1.3 2032.6    1.2    9.5    1.5    1.0    3.7   12.1
 [21]    1.1    2.3    2.2    2.6    4.7    1.1    1.0    3.8    4.0    1.1
 [31]    1.1    2.6    4.5    1.5    5.7    1.1    1.4    1.7    3.0    1.3
 [41]    1.3    3.2    1.5    1.4    1.2    1.5    3.5    6.8    1.1    4.3
```

```
 [51]    1.2    4.6    5.7    1.3    1.0    4.2    1.3   16.0    9.7   35.5
 [61]   12.2   14.3    1.4    7.3    1.2   46.5    2.0    2.4    1.1    1.9
 [71]    1.1   18.4    2.2    1.9    3.4   63.6    8.7    3.9    5.1    1.1
 [81]    8.8    1.5    5.0    2.1 2513.4    1.5    7.3    2.6    5.3    2.5
 [91]    1.4    3.6    1.8   11.2    1.4    1.8    1.1    1.7    7.5    3.4
```

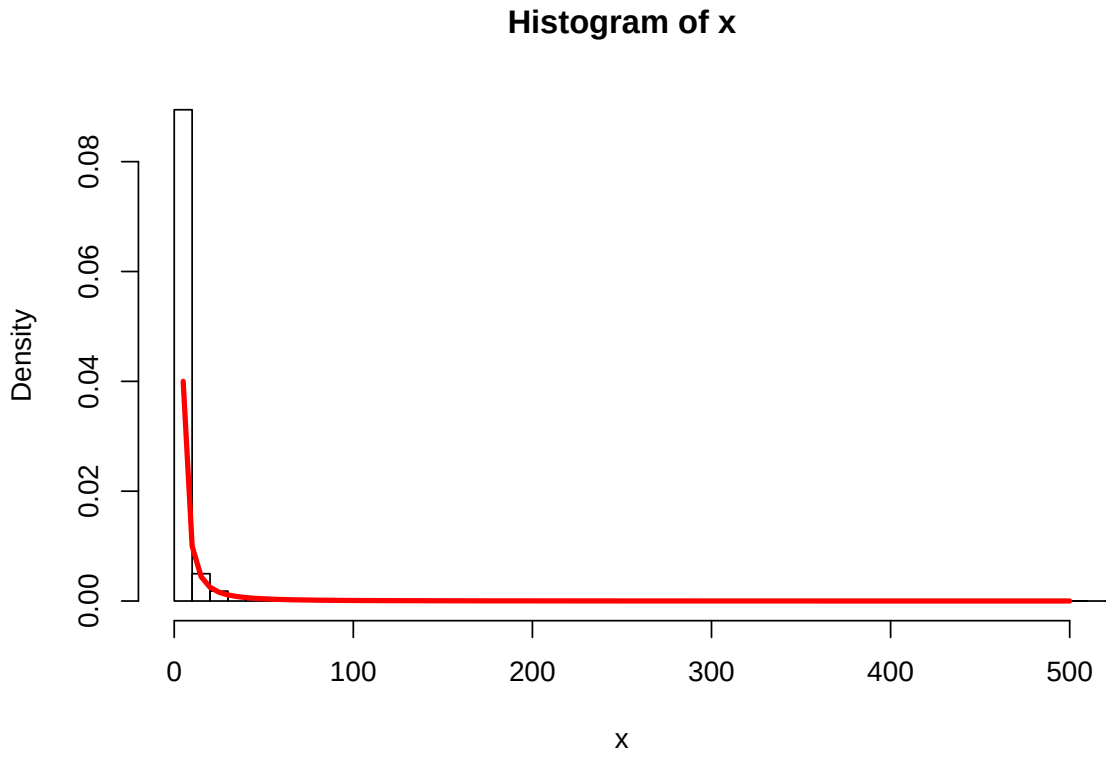
```
summary(x)   #vemos qué posibles valores toma
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
      1       1       2      10       4    7819
```

```
var(x)       # en las estadísticas sumarias, no se incluye la varianza.
```

```
[1] 11683
```

```
hist(x,xlim=c(0,500),breaks=700,prob=T)
curve(1/x^2*ifelse(x>1,1,0),from=0,to=500,col="red",add=T,lwd=3)
```



Esta variable aleatoria tiene valores outliers como en el caso de la distribución Cauchy.

□