Hoja 6 de problemas y prácticas con R

Estadística Computacional I. Grado en Estadística Departamento de Estadística e Investigación Operativa

Marta Venegas Pardo

Ejercicio 1

1. Responder a los siguientes apartados:

```
library(randtoolbox)
```

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

Tiene muchas funciones relacionadas con el tema.

Generar secuencias de tamaño 500 según los generadores de Mersenne-Twister, Congruencia lineal, "Knuth-TAOCP-2002" y una secuencia determinista.

Generar de congruencia lineal y algunos contrastes

```
library(tseries) # Para contrastes de independencia
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
n<-500
set.seed(1)
(head (xmt<- runif(n) , 10 )) #Mersenne-Twister
```

```
##  [1] 0.26550866 0.37212390 0.57285336 0.90820779 0.20168193 0.89838968
##  [7] 0.94467527 0.66079779 0.62911404 0.06178627
```

```
(head (xcl<- congruRand(n),10)) #Generador de congruencia lineal
```

```
##  [1] 0.6210185 0.4582407 0.6517764 0.4063970 0.3141388 0.7309768 0.5275602
##  [8] 0.7043181 0.4746586 0.5871574
```

```
(head(xta<-knuthTAOCP(n),10)) #Knuth-TAOCP-2002
```

```
##  [1] 0.77559173 0.20262363 0.06309716 0.57894119 0.59421662 0.12908291
##  [7] 0.22554409 0.20561367 0.94868813 0.06787565
```

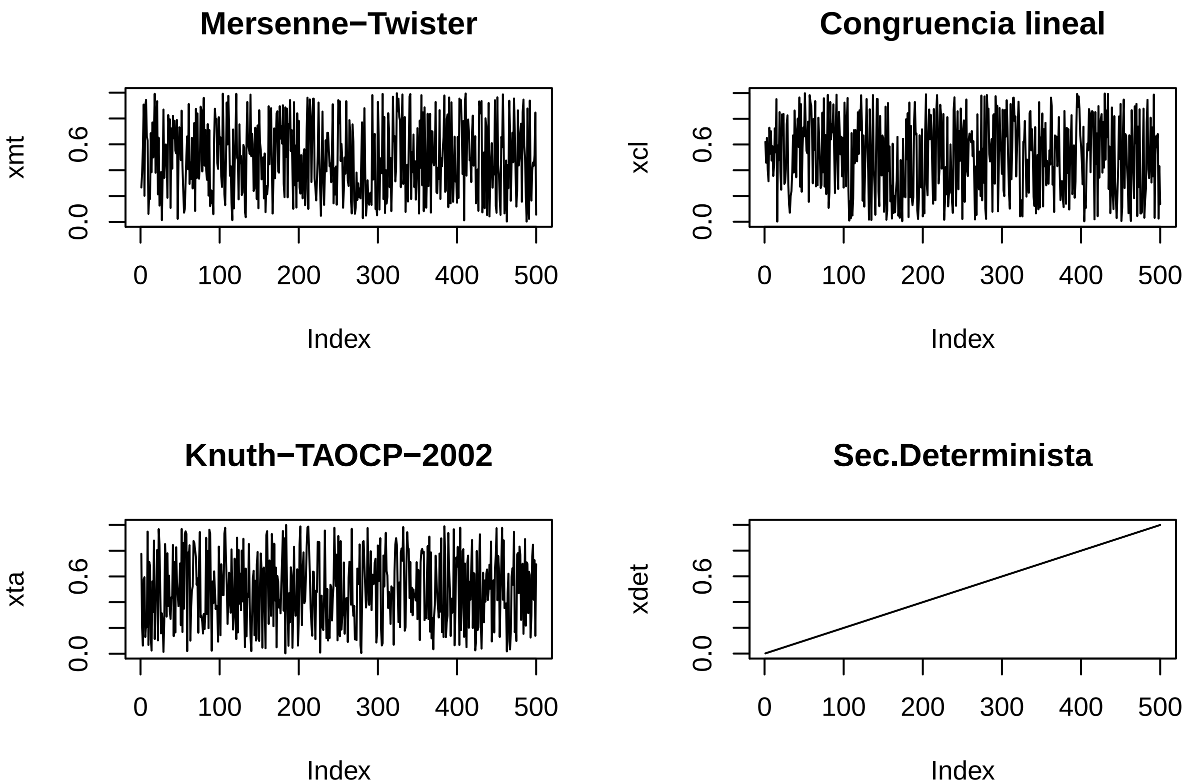
```
(head(xdet<- ((1:n)-0.5)/n,10))
```

```
##  [1] 0.001 0.003 0.005 0.007 0.009 0.011 0.013 0.015 0.017 0.019
```

Dibujar gráficos de líneas e histogramas para las cuatro secuencias.

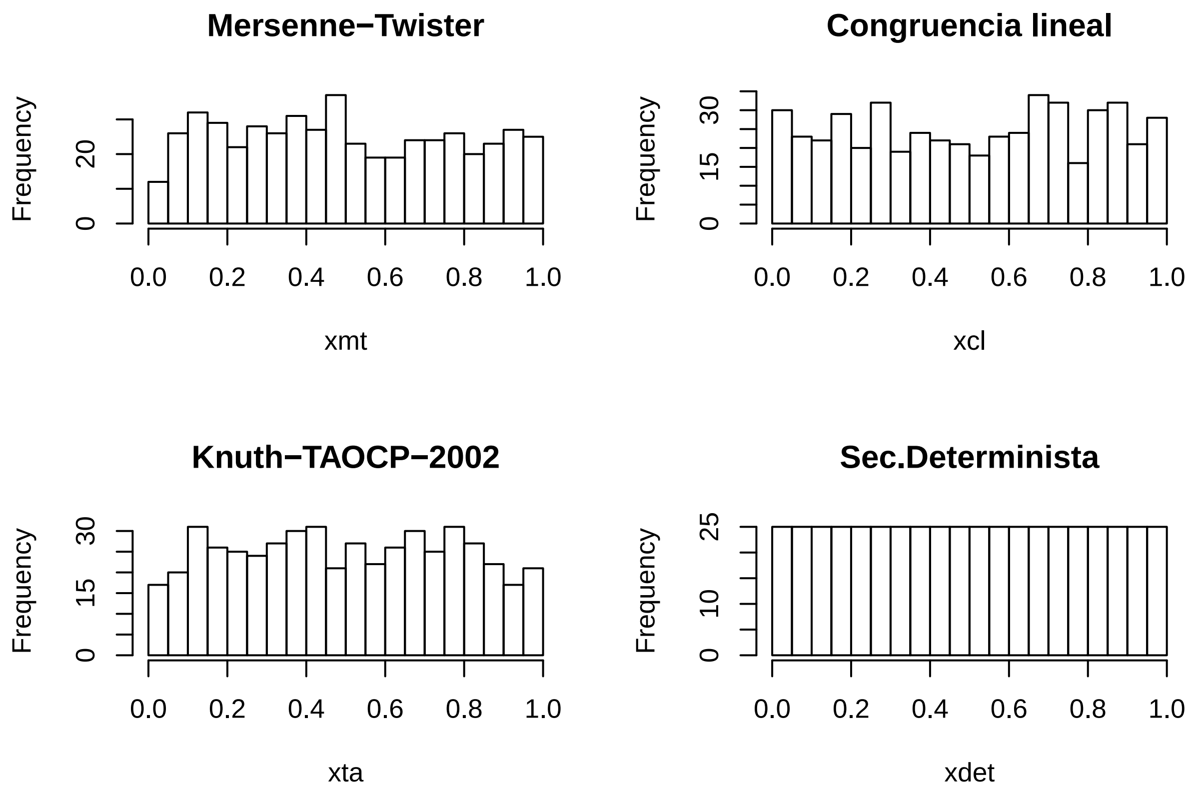
Gráficos de líneas con plot

```
par(mfrow=c(2,2))
plot(xmt,type="l",main="Mersenne-Twister")
plot(xcl,type="l",main="Congruencia lineal")
plot(xta,type="l",main="Knuth-TAOCP-2002")
plot(xdet,type="l",main="Sec.Determinista")
```



Histogramas con hist

```
par(mfrow=c(2,2))
hist(xmt,br=20,main="Mersenne-Twister")
hist(xcl,br=20,main="Congruencia lineal")
hist(xta,br=20,main="Knuth-TAOCP-2002")
hist(xdet,br=20,main="Sec.Determinista")
```




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

Aplicar contrastes de aleatoriedad sobre las cuatro secuencias.

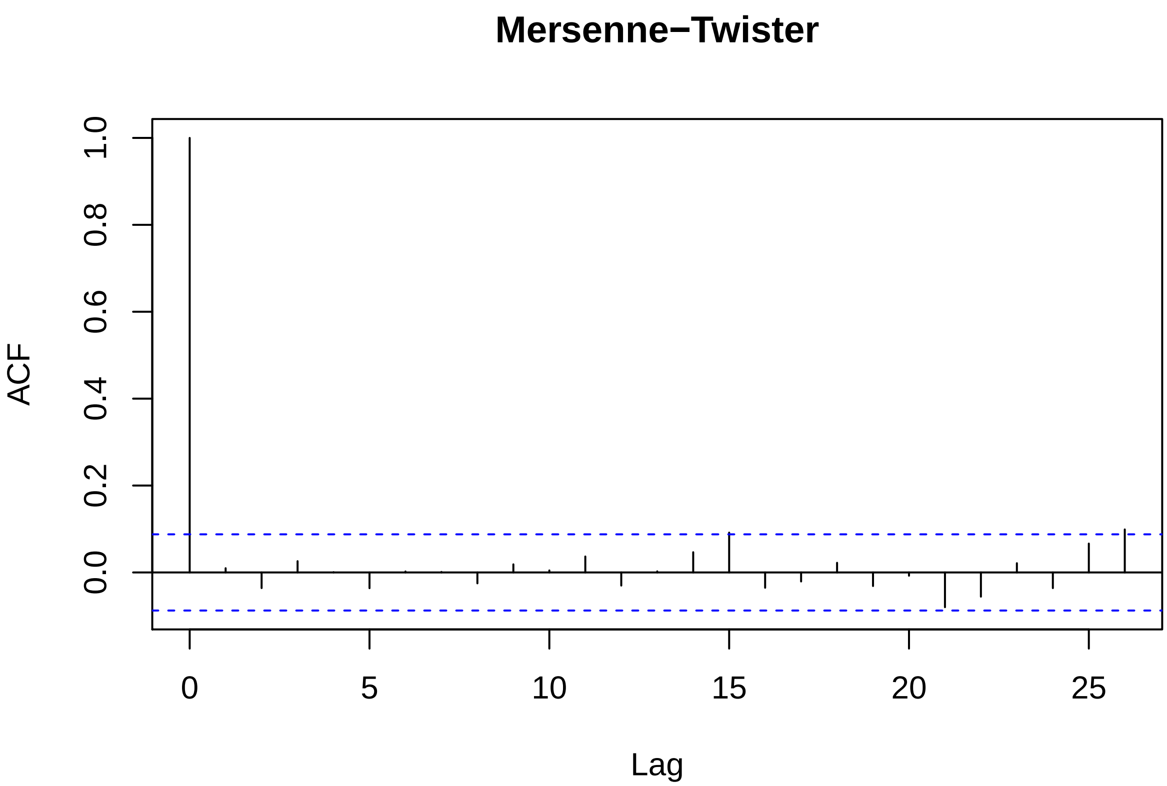
```
contrastes_aleato<-function(x,titulo) {
print(titulo)
ks.test(x, "punif")
gap.test(x) #Por defecto, l=0, u=0.5
order.test(x,d=5) #d puede ser 2,3,4,5, pero n debe ser múltiplo de d freq.test(x, 1:4)#Por defecto, se
serial.test(x,d=5) #n debe ser múltiplo de d (t=2)
poker.test(x)
print(runs.test(factor(x>0.5)))
acf(x,main=titulo)
}
```

Mersenne-Twister

```
contrastes_aleato(xmt,"Mersenne-Twister")
```

```
## [1] "Mersenne-Twister"
##
##          Gap test
##
## chisq stat = 16, df = 9, p-value = 0.064
##
##       (sample size : 500)
##
## length   observed freq        theoretical freq
```

```
## 1             48              62
## 2             37              31
## 3             16              16
## 4             11              7.8
## 5             5           3.9
## 6             2           2
## 7             0           0.98
## 8             0           0.49
## 9             1           0.24
## 10            1           0.12
##
##           Order test
##
## chisq stat = 121, df = 119, p-value = 0.44
##
##       (sample size : 500)
##
##  observed number  0 1 0 0 2 1 1 1 0 1 3 2 2 0 0 2 2 3 0 1 1 0 0 0 0 3 0 0 1 2 1 1 0 0 1 2 0 0 0 1 1
##  expected number  0.83
##
##           Serial test
##
## chisq stat = 33, df = 24, p-value = 0.1
##
##       (sample size : 500)
##
##  observed number  9 7 7 5 9 12 8 11 12 16 9 11 17 5 14 19 12 7 9 7 13 10 8 8 5
##  expected number  10
##
##           Poker test
##
## chisq stat = 3.3, df = 4, p-value = 0.5
##
##       (sample size : 500)
##
##  observed number  0 6 56 35 3
##  expected number  0.16 9.6 48 38 3.8
##
##  Runs Test
##
## data:  factor(x > 0.5)
## Standard Normal = -0.8, p-value = 0.4
## alternative hypothesis: two.sided
```



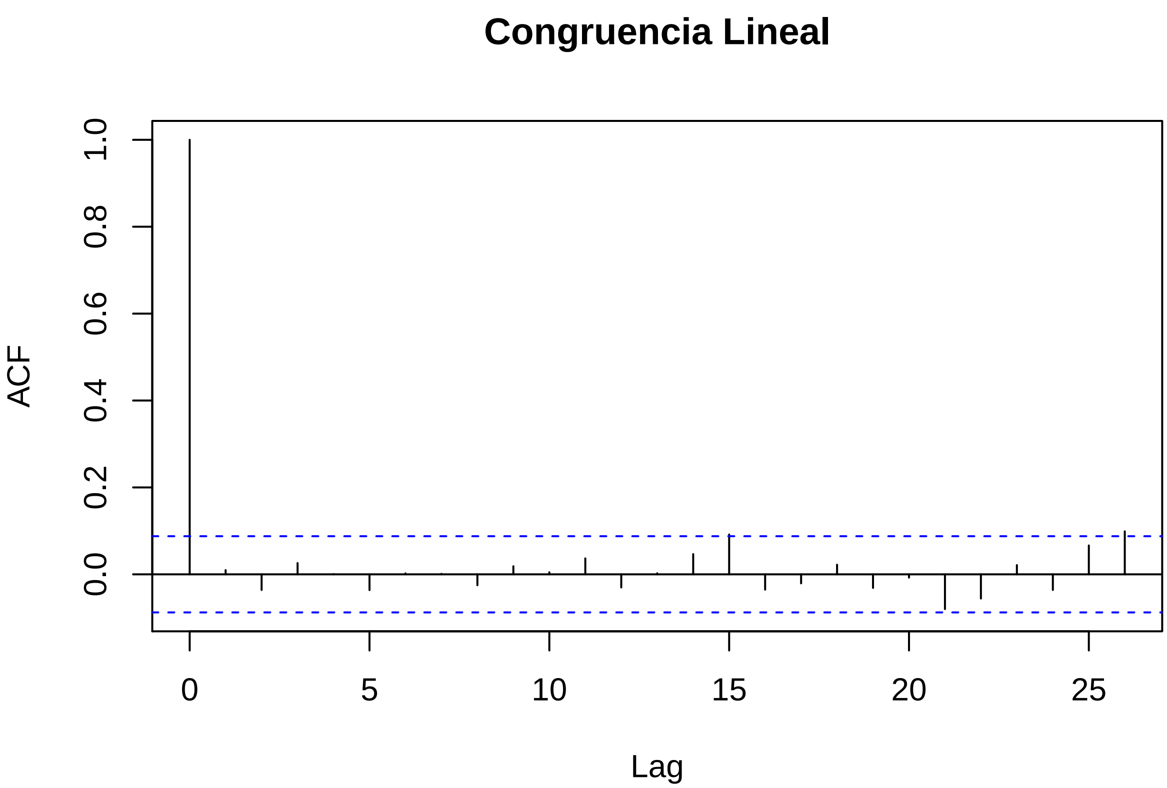
No se sale mucho de las líneas, para entender las autocorreaciones.

Congruencia lineal

```
contrastes_aleato(xmt,"Congruencia Lineal")
```

```
## [1] "Congruencia Lineal"
##
##           Gap test
##
## chisq stat = 16, df = 9, p-value = 0.064
##
##       (sample size : 500)
##
## length   observed freq       theoretical freq
## 1             48              62
## 2             37              31
## 3             16              16
## 4             11              7.8
## 5             5           3.9
## 6             2           2
## 7             0           0.98
## 8             0           0.49
## 9             1           0.24
## 10            1           0.12
##
##           Order test
##
## chisq stat = 121, df = 119, p-value = 0.44
```

```
##
##       (sample size : 500)
##
##  observed number  0 1 0 0 2 1 1 1 0 1 3 2 2 0 0 2 2 3 0 1 1 0 0 0 0 3 0 0 1 2 1 1 0 0 1 2 0 0 0 1 1
##  expected number  0.83
##
##           Serial test
##
## chisq stat = 33, df = 24, p-value = 0.1
##
##       (sample size : 500)
##
##  observed number  9 7 7 5 9 12 8 11 12 16 9 11 17 5 14 19 12 7 9 7 13 10 8 8 5
##  expected number  10
##
##           Poker test
##
## chisq stat = 3.3, df = 4, p-value = 0.5
##
##       (sample size : 500)
##
##  observed number  0 6 56 35 3
##  expected number  0.16 9.6 48 38 3.8
##
##  Runs Test
##
## data:  factor(x > 0.5)
## Standard Normal = -0.8, p-value = 0.4
## alternative hypothesis: two.sided
```



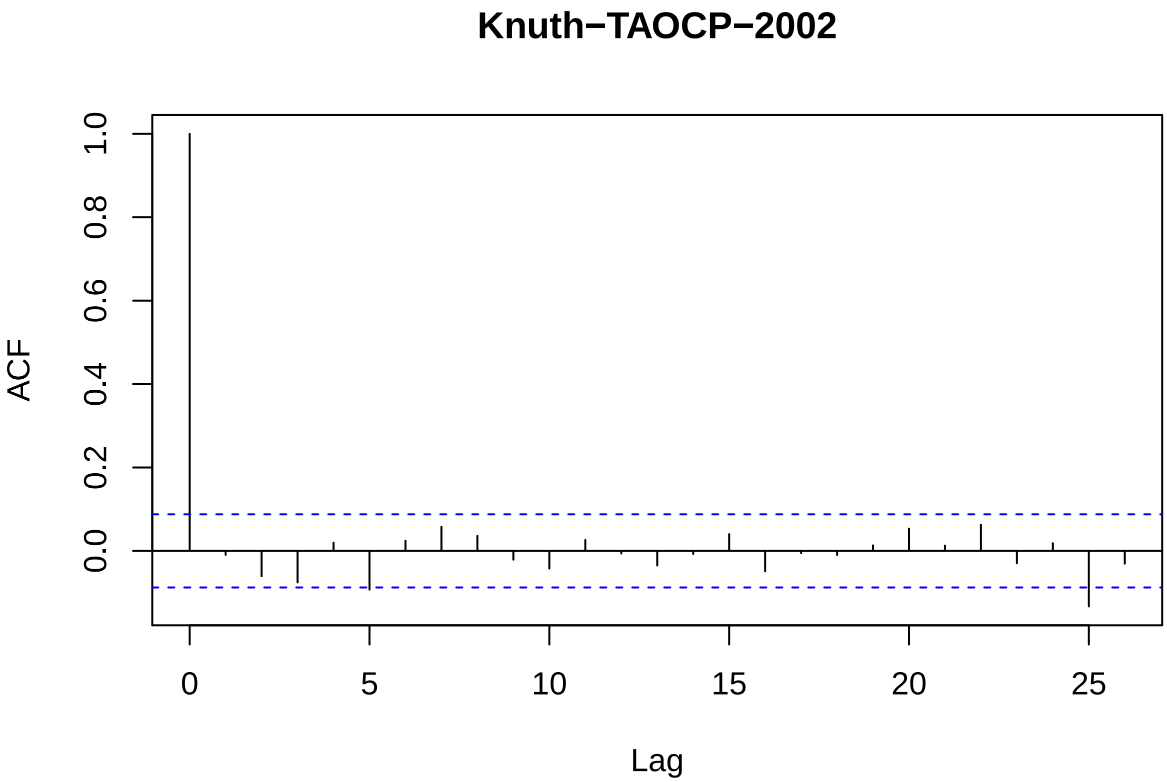
Los p-valores no están tan al límite como alguno de los anteriores.

Knuth-TAOCP-2002

```
contrastes_aleato(xta,"Knuth-TAOCP-2002")
```

```
## [1] "Knuth-TAOCP-2002"
##
##           Gap test
##
## chisq stat = 2.9, df = 9, p-value = 0.97
##
##       (sample size : 500)
##
## length   observed freq       theoretical freq
## 1             61              62
## 2             29              31
## 3             20              16
## 4             7           7.8
## 5             4           3.9
## 6             3           2
## 7             1           0.98
## 8             0           0.49
## 9             0           0.24
## 10            0           0.12
##
##           Order test
##
## chisq stat = 123, df = 119, p-value = 0.38
##
##       (sample size : 500)
##
##  observed number  1 0 0 0 1 0 2 1 0 2 0 2 1 1 1 1 1 2 2 0 0 0 1 0 0 0 1 1 0 1 0 2 2 0 0 0 1 1 3 1 0 0
##  expected number  0.83
##
##           Serial test
##
## chisq stat = 28, df = 24, p-value = 0.26
##
##       (sample size : 500)
##
##  observed number  4 9 9 17 8 14 12 11 11 8 12 15 6 14 7 9 5 12 11 6 8 9 9 16 8
##  expected number  10
##
##           Poker test
##
## chisq stat = 2.3, df = 4, p-value = 0.68
##
##       (sample size : 500)
##
##  observed number  0 7 48 43 2
##  expected number  0.16 9.6 48 38 3.8
##
##  Runs Test
```

```
##
## data:  factor(x > 0.5)
## Standard Normal = 0.001, p-value = 1
## alternative hypothesis: two.sided
```



Determinista

```
contrastes_aleato(xdet,"Determinista")
```

```
## [1] "Determinista"
##
##           Gap test
##
## chisq stat = 1.4e+73, df = 249, p-value = 0
##
##       (sample size : 500)
##
## length   observed freq       theoretical freq
## 1            0           62
## 2            0           31
## 3            0           16
## 4            0           7.8
## 5            0           3.9
## 6            0           2
## 7            0           0.98
## 8            0           0.49
## 9            0           0.24
## 10           0           0.12
## 11           0           0.061
```

```
## 12              0            0.031
## 13              0            0.015
## 14              0            0.0076
## 15              0            0.0038
## 16              0            0.0019
## 17              0            0.00095
## 18              0            0.00048
## 19              0            0.00024
## 20              0            0.00012
## 21              0            6e-05
## 22              0            3e-05
## 23              0            1.5e-05
## 24              0            7.5e-06
## 25              0            3.7e-06
## 26              0            1.9e-06
## 27              0            9.3e-07
## 28              0            4.7e-07
## 29              0            2.3e-07
## 30              0            1.2e-07
## 31              0            5.8e-08
## 32              0            2.9e-08
## 33              0            1.5e-08
## 34              0            7.3e-09
## 35              0            3.6e-09
## 36              0            1.8e-09
## 37              0            9.1e-10
## 38              0            4.5e-10
## 39              0            2.3e-10
## 40              0            1.1e-10
## 41              0            5.7e-11
## 42              0            2.8e-11
## 43              0            1.4e-11
## 44              0            7.1e-12
## 45              0            3.6e-12
## 46              0            1.8e-12
## 47              0            8.9e-13
## 48              0            4.4e-13
## 49              0            2.2e-13
## 50              0            1.1e-13
## 51              0            5.6e-14
## 52              0            2.8e-14
## 53              0            1.4e-14
## 54              0            6.9e-15
## 55              0            3.5e-15
## 56              0            1.7e-15
## 57              0            8.7e-16
## 58              0            4.3e-16
## 59              0            2.2e-16
## 60              0            1.1e-16
## 61              0            5.4e-17
## 62              0            2.7e-17
## 63              0            1.4e-17
## 64              0            6.8e-18
## 65              0            3.4e-18
```

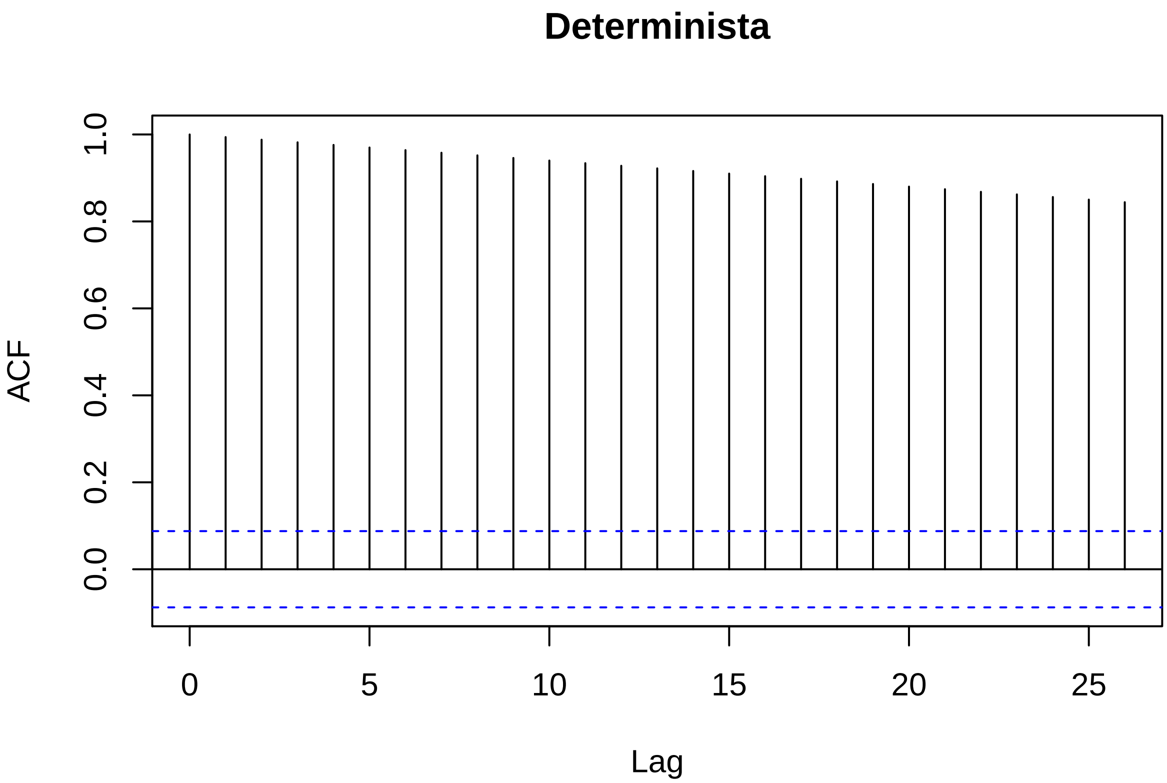
```
## 66              0             1.7e-18
## 67              0             8.5e-19
## 68              0             4.2e-19
## 69              0             2.1e-19
## 70              0             1.1e-19
## 71              0             5.3e-20
## 72              0             2.6e-20
## 73              0             1.3e-20
## 74              0             6.6e-21
## 75              0             3.3e-21
## 76              0             1.7e-21
## 77              0             8.3e-22
## 78              0             4.1e-22
## 79              0             2.1e-22
## 80              0             1e-22
## 81              0             5.2e-23
## 82              0             2.6e-23
## 83              0             1.3e-23
## 84              0             6.5e-24
## 85              0             3.2e-24
## 86              0             1.6e-24
## 87              0             8.1e-25
## 88              0             4e-25
## 89              0             2e-25
## 90              0             1e-25
## 91              0             5e-26
## 92              0             2.5e-26
## 93              0             1.3e-26
## 94              0             6.3e-27
## 95              0             3.2e-27
## 96              0             1.6e-27
## 97              0             7.9e-28
## 98              0             3.9e-28
## 99              0             2e-28
## 100             0             9.9e-29
## 101             0             4.9e-29
## 102             0             2.5e-29
## 103             0             1.2e-29
## 104             0             6.2e-30
## 105             0             3.1e-30
## 106             0             1.5e-30
## 107             0             7.7e-31
## 108             0             3.9e-31
## 109             0             1.9e-31
## 110             0             9.6e-32
## 111             0             4.8e-32
## 112             0             2.4e-32
## 113             0             1.2e-32
## 114             0             6e-33
## 115             0             3e-33
## 116             0             1.5e-33
## 117             0             7.5e-34
## 118             0             3.8e-34
## 119             0             1.9e-34
```

```
## 120             0            9.4e-35
## 121             0            4.7e-35
## 122             0            2.4e-35
## 123             0            1.2e-35
## 124             0            5.9e-36
## 125             0            2.9e-36
## 126             0            1.5e-36
## 127             0            7.3e-37
## 128             0            3.7e-37
## 129             0            1.8e-37
## 130             0            9.2e-38
## 131             0            4.6e-38
## 132             0            2.3e-38
## 133             0            1.1e-38
## 134             0            5.7e-39
## 135             0            2.9e-39
## 136             0            1.4e-39
## 137             0            7.2e-40
## 138             0            3.6e-40
## 139             0            1.8e-40
## 140             0            9e-41
## 141             0            4.5e-41
## 142             0            2.2e-41
## 143             0            1.1e-41
## 144             0            5.6e-42
## 145             0            2.8e-42
## 146             0            1.4e-42
## 147             0            7e-43
## 148             0            3.5e-43
## 149             0            1.8e-43
## 150             0            8.8e-44
## 151             0            4.4e-44
## 152             0            2.2e-44
## 153             0            1.1e-44
## 154             0            5.5e-45
## 155             0            2.7e-45
## 156             0            1.4e-45
## 157             0            6.8e-46
## 158             0            3.4e-46
## 159             0            1.7e-46
## 160             0            8.6e-47
## 161             0            4.3e-47
## 162             0            2.1e-47
## 163             0            1.1e-47
## 164             0            5.3e-48
## 165             0            2.7e-48
## 166             0            1.3e-48
## 167             0            6.7e-49
## 168             0            3.3e-49
## 169             0            1.7e-49
## 170             0            8.4e-50
## 171             0            4.2e-50
## 172             0            2.1e-50
## 173             0            1e-50
```

```
## 174            0            5.2e-51
## 175            0            2.6e-51
## 176            0            1.3e-51
## 177            0            6.5e-52
## 178            0            3.3e-52
## 179            0            1.6e-52
## 180            0            8.2e-53
## 181            0            4.1e-53
## 182            0            2e-53
## 183            0            1e-53
## 184            0            5.1e-54
## 185            0            2.5e-54
## 186            0            1.3e-54
## 187            0            6.4e-55
## 188            0            3.2e-55
## 189            0            1.6e-55
## 190            0            8e-56
## 191            0            4e-56
## 192            0            2e-56
## 193            0            1e-56
## 194            0            5e-57
## 195            0            2.5e-57
## 196            0            1.2e-57
## 197            0            6.2e-58
## 198            0            3.1e-58
## 199            0            1.6e-58
## 200            0            7.8e-59
## 201            0            3.9e-59
## 202            0            1.9e-59
## 203            0            9.7e-60
## 204            0            4.9e-60
## 205            0            2.4e-60
## 206            0            1.2e-60
## 207            0            6.1e-61
## 208            0            3e-61
## 209            0            1.5e-61
## 210            0            7.6e-62
## 211            0            3.8e-62
## 212            0            1.9e-62
## 213            0            9.5e-63
## 214            0            4.7e-63
## 215            0            2.4e-63
## 216            0            1.2e-63
## 217            0            5.9e-64
## 218            0            3e-64
## 219            0            1.5e-64
## 220            0            7.4e-65
## 221            0            3.7e-65
## 222            0            1.9e-65
## 223            0            9.3e-66
## 224            0            4.6e-66
## 225            0            2.3e-66
## 226            0            1.2e-66
## 227            0            5.8e-67
```

```
## 228           0           2.9e-67
## 229           0           1.4e-67
## 230           0           7.2e-68
## 231           0           3.6e-68
## 232           0           1.8e-68
## 233           0           9.1e-69
## 234           0           4.5e-69
## 235           0           2.3e-69
## 236           0           1.1e-69
## 237           0           5.7e-70
## 238           0           2.8e-70
## 239           0           1.4e-70
## 240           0           7.1e-71
## 241           0           3.5e-71
## 242           0           1.8e-71
## 243           0           8.8e-72
## 244           0           4.4e-72
## 245           0           2.2e-72
## 246           0           1.1e-72
## 247           0           5.5e-73
## 248           0           2.8e-73
## 249           0           1.4e-73
## 250           1           6.9e-74
##
##           Order test
##
## chisq stat = 11900, df = 119, p-value = 0
##
##       (sample size : 500)
##
##  observed number  100 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  expected number  0.83
##
##           Serial test
##
## chisq stat = 1000, df = 24, p-value = 8.9e-196
##
##       (sample size : 500)
##
##  observed number  0 0 50 50 0 0 0 0 50 50 0 0 0 0 50 0 0 0 0 0 0 0 0 0 0
##  expected number  10
##
##           Poker test
##
## chisq stat = 2504, df = 4, p-value = 0
##
##       (sample size : 500)
##
##  observed number  0 0 0 0 100
##  expected number  0.16 9.6 48 38 3.8
##
##  Runs Test
##
## data:  factor(x > 0.5)
```

```
## Standard Normal = -22, p-value <2e-16
## alternative hypothesis: two.sided
```



Test con generadores (no secuencias)

Le pasamos la función generadora y hará un estudio.

```
coll.test(runif,2^7,2^10)
```

```
##
##            Collision test
##
## chisq stat = 0.008, df = 1, p-value = 0.93
##
## Poisson approximation (sample number : 1000 / sample size : 128 / cell number : 1e+06)
##
##  collision        observed        expected
##  number           count       count
##       0           992             992
##       1           8           7.8
```

p-valor alto, no rechazo.

```
#m=2^10 secuencias de tamaño n=2^7
##por defecto tdim=2
coll.test(congruRand,2^7,2^10)
```

```
##
##            Collision test
##
## chisq stat = 0.4, df = 1, p-value = 0.53
```

```
##
## Poisson approximation (sample number : 1000 / sample size : 128 / cell number : 1e+06)
##
##  collision        observed        expected
##  number           count       count
##      0            994             992
##      1            6           7.8
```

```
coll.test(knuthTAOCP,2^7,2^10)
```

```
##
##          Collision test
##
## chisq stat = 0.66, df = 1, p-value = 0.42
##
## Poisson approximation (sample number : 1000 / sample size : 128 / cell number : 1e+06)
##
##  collision        observed        expected
##  number           count       count
##      0            990             992
##      1            10              7.8
```

Ejercicio 2

Comparar empíricamente los rendimientos de los siguientes generadores de números aleatorios en (0,1), Mersenne-Twister, Congruencia Lineal, K2nuth-TAOCP-2002,2 además de un generador de números determinista, según los resultados del test de Kolmogorov-Smirnov y el test de huecos.

Utilizar 1000 muestras de tamaño 100 y analizar de forma numérica y gráfica los resultados.

```
library(randtoolbox)
M<-1000
n<-100
pvaloresKS<- matrix(NA,M,4)
pvaloresGap<- matrix(NA,M,4)
colnames(pvaloresKS)<- c("Mersenne-Twister","Congruencia Lineal", "Knuth-TAOCP-2002","Determinista")
colnames(pvaloresGap)<- c("Mersenne-Twister","Congruencia Lineal", "Knuth-TAOCP-2002","Determinista")

for (i in 1:M) {
if (i%%50==0) {
cat("Muestra ",i,"de ",M,"\n")
}
xmt<- runif(n) #Mersenne-Twister
xcl<- congruRand(n) #Generador de congruencia lineal
xta<-knuthTAOCP(n) #Knuth-TAOCP-2002
xdet<- ((1:n)-0.5)/n #Secuencia determinista
pvaloresKS[i,]<- c(ks.test(xmt, "punif")$p.value,
                   ks.test(xcl, "punif")$p.value,
                   ks.test(xta, "punif")$p.value,
                   ks.test(xdet, "punif")$p.value)
pvaloresGap[i,]<- c(gap.test(xmt,echo=FALSE)$p.value,
                    gap.test(xcl,echo=FALSE)$p.value,
                    gap.test(xta,echo=FALSE)$p.value,
                    gap.test(xdet,echo=FALSE)$p.value)
}
```

```
## Muestra  50 de  1000
## Muestra  100 de  1000
## Muestra  150 de  1000
## Muestra  200 de  1000
## Muestra  250 de  1000
## Muestra  300 de  1000
## Muestra  350 de  1000
## Muestra  400 de  1000
## Muestra  450 de  1000
## Muestra  500 de  1000
## Muestra  550 de  1000
## Muestra  600 de  1000
## Muestra  650 de  1000
## Muestra  700 de  1000
## Muestra  750 de  1000
## Muestra  800 de  1000
## Muestra  850 de  1000
## Muestra  900 de  1000
## Muestra  950 de  1000
## Muestra  1000 de  1000
```

```
head(pvaloresKS,10)
```

```
##       Mersenne-Twister Congruencia Lineal Knuth-TAOCP-2002 Determinista
##  [1,]            0.442               0.47             0.76            1
##  [2,]            0.840               0.90             0.78            1
##  [3,]            0.519               0.21             0.99            1
##  [4,]            0.715               0.48             0.88            1
##  [5,]            0.466               0.12             0.38            1
##  [6,]            0.035               0.28             0.74            1
##  [7,]            0.933               0.34             0.94            1
##  [8,]            0.530               0.69             0.57            1
##  [9,]            0.954               0.23             0.41            1
## [10,]            0.976               0.75             0.22            1
```

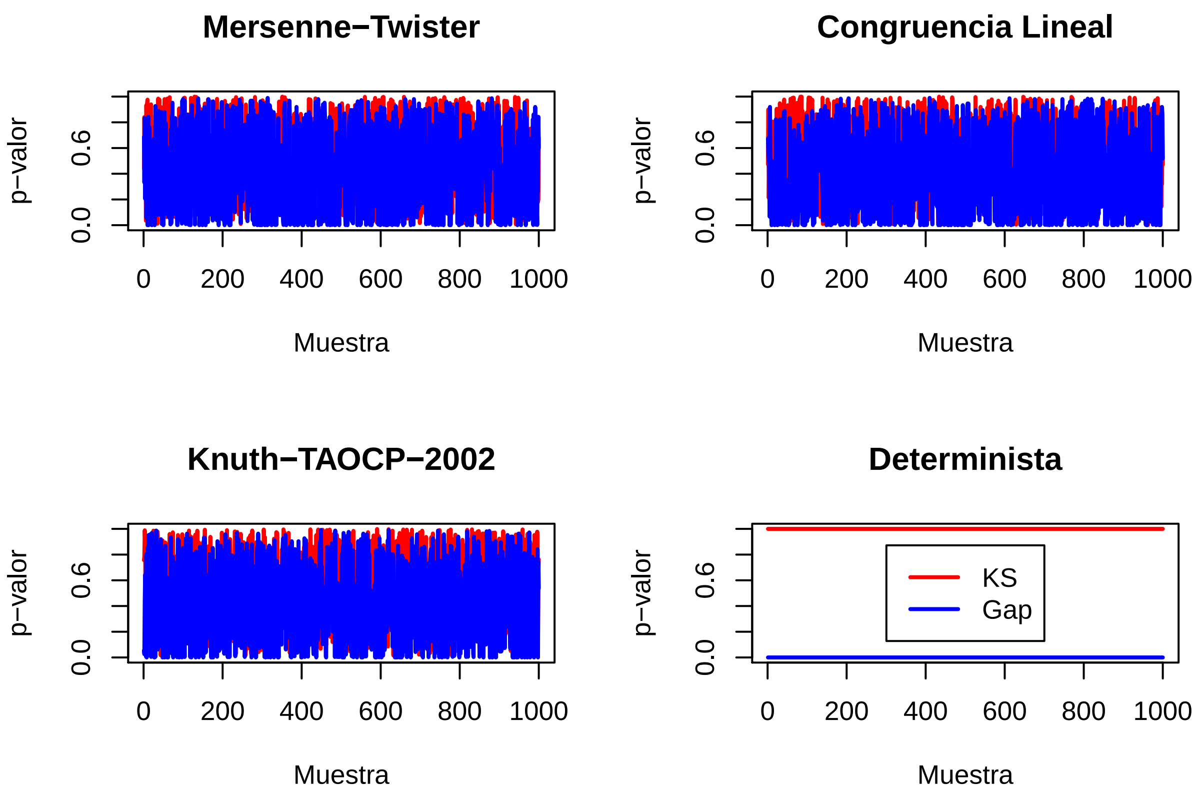
```
head(pvaloresGap,10)
```

```
##       Mersenne-Twister Congruencia Lineal Knuth-TAOCP-2002 Determinista
##  [1,]          6.9e-01            6.8e-01          5.4e-02            0
##  [2,]          3.3e-01            5.3e-01          2.6e-02            0
##  [3,]          8.3e-01            5.3e-01          4.4e-01            0
##  [4,]          2.0e-01            4.8e-01          6.4e-01            0
##  [5,]          7.4e-01            6.7e-02          1.5e-01            0
##  [6,]          6.0e-01            9.2e-01          7.3e-03            0
##  [7,]          5.8e-02            5.8e-01          8.0e-01            0
##  [8,]          3.6e-01            3.0e-01          1.7e-13            0
##  [9,]          5.0e-01            8.0e-02          4.7e-01            0
## [10,]          4.1e-06            1.8e-06          8.5e-01            0
```

```
#Ejercicio: dibujar en una sola pantalla los M p-valores KS y Gap para los #cuatro generadores
```

```
 par(mfrow=c(2,2))
for (i in 1:4)
{
plot(pvaloresKS[,i],type="l",
     main=colnames(pvaloresKS)[i],
```

```
     xlab="Muestra",ylab="p-valor",
     col="red",lwd=2,ylim=c(0,1))
lines(pvaloresGap[,i], col="blue",lwd=2)
}
legend("center",col=c("red","blue"),lty=1,
       lwd=2,legend=c("KS","Gap"))
```




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

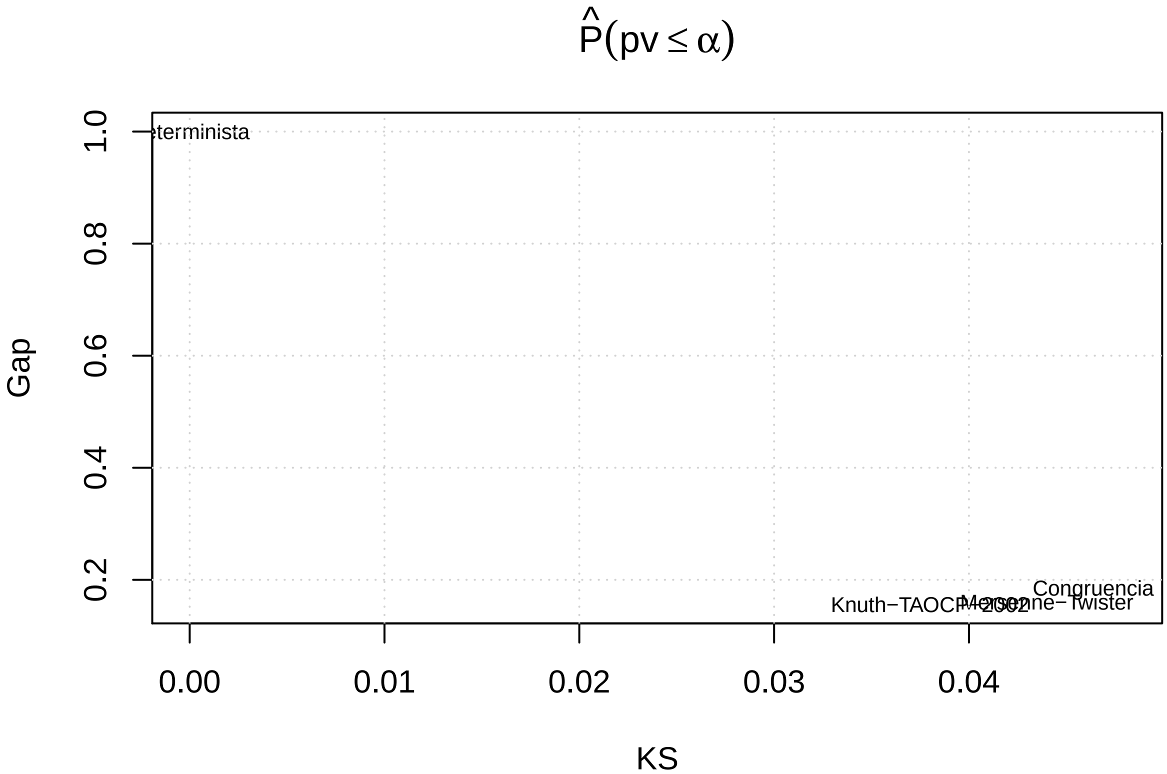
Estimar P[p<alfa]

```
alfa<- 0.05
rendim<- matrix(NA,4,2)
colnames(rendim)<- c("KS","Gap")
rownames(rendim)<- c("Mersenne-Twister","Congruencia Lineal", "Knuth-TAOCP-2002","Determinista")
for (i in 1:4)
rendim[i,]<-c(mean(pvaloresKS[,i]<=alfa),
              mean(pvaloresGap[,i]<=alfa))
#Ejercicio: completar
rendim
```

```
##                       KS  Gap
## Mersenne-Twister   0.044 0.16
## Congruencia Lineal 0.048 0.18
## Knuth-TAOCP-2002   0.038 0.16
## Determinista       0.000 1.00
```

```
plot(rendim,type="n",
     main=expression(hat(P)(pv<=alpha)))
```

```r
text(rendim,label=rownames(rendim),cex=0.7)
grid()
```



Cuántas veces es mayor un pv que el otro

```r
for (i in 1:4)
  cat(colnames(pvaloresKS)[i],mean(pvaloresKS[,i]>pvaloresGap[,i]),"\n")
```

```
## Mersenne-Twister 0.59
## Congruencia Lineal 0.62
## Knuth-TAOCP-2002 0.61
## Determinista 1
```

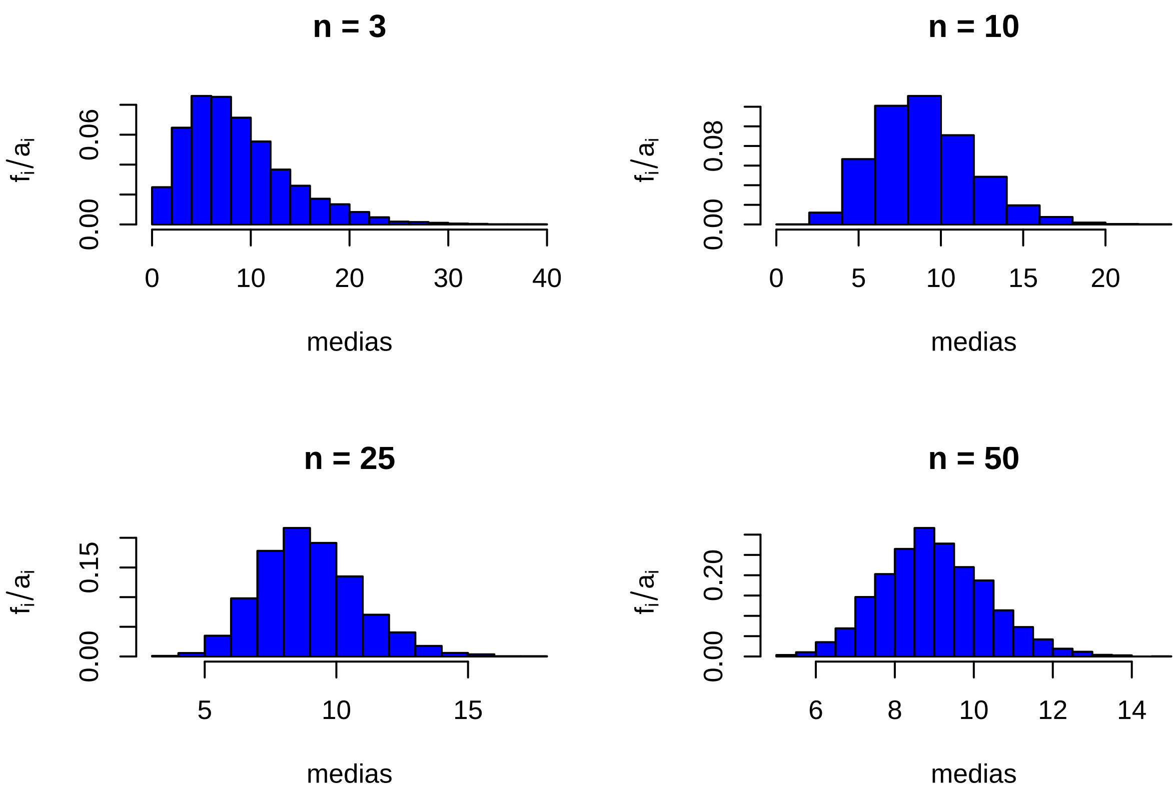
Ejercicio 3

Teorema central del límite

3. Ilustrar el Teorema Central del Límite con muestras de una ley Geométrica con parámetro 0.1 para tamaños muestrales 3, 10, 25, 50 y número de muestras 5000.

```r
M<-5000 #Núm. de muestras
p<-0.1
vn<-c(3,10,25,50) #Tamaños de cada muestra
paranti<- par(no.readonly=T)
par(mfrow=c(2,2))
for (n in vn) #Ejercicio: completar
{ x<- matrix(rgeom(M*n,p), # Cuantas genero (5000*3), con parametro p=0.1
             nrow=M,ncol=n)
  medias<- rowMeans(x)
```

```
  hist(medias,freq=FALSE, col="blue",
     main=paste("n =",n),
     ylab=expression(f[i] /a[i]))
}
```




```
par(paranti)
```

Ejercicio 4

Intervalo de confianza con una normal

4. Ilustrar el concepto de intervalo de confianza mediante 100 muestras de tamaño 10 de una ley N(0,1), siendo la media poblacional el parámetro de interés.

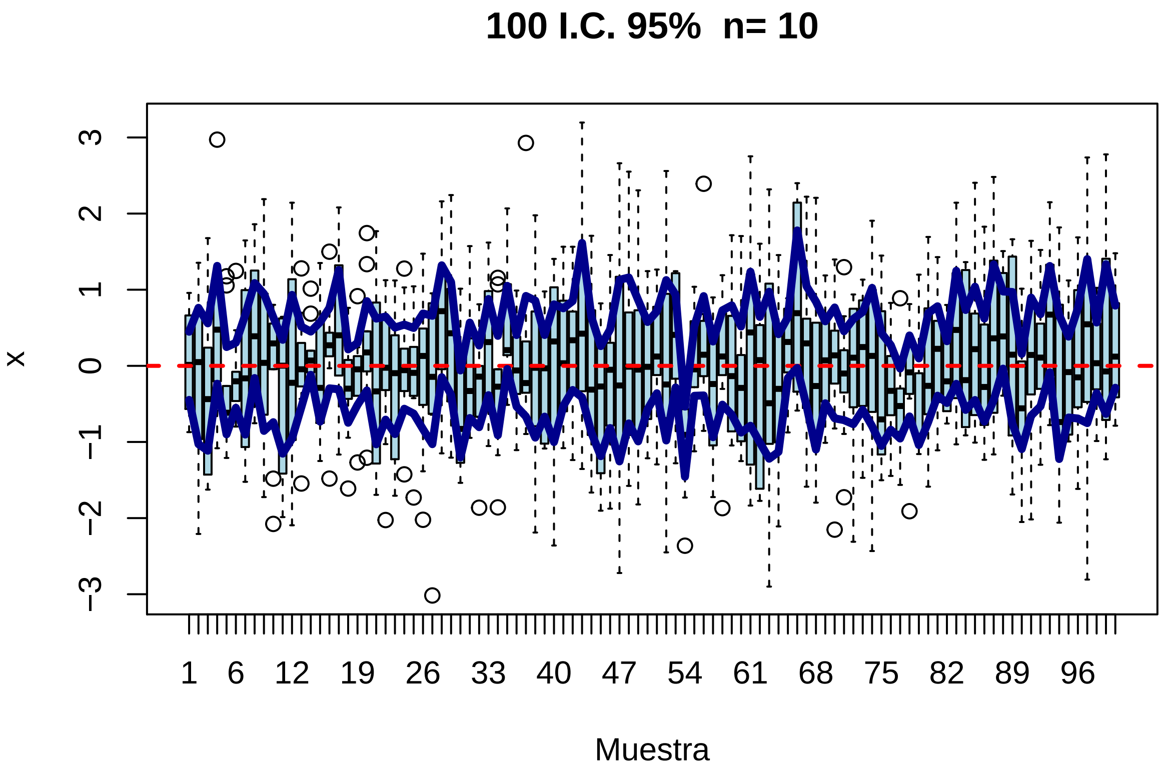
```
n<-10 #Tamaño muestral
M<-100 #M muestras
alfa<-0.05
media<-0
desvtip<-1
muestras<-matrix(rnorm(M*n,mean=media,sd=desvtip),
                 nrow=M,ncol=n,byrow=TRUE)
```

```
pv<-array(0,M) #M pvalores
exinf<-numeric(M)
exsup<-numeric(M)
```

```
for (i in 1:M)
{ #Ejercicio: completar
  test<-t.test(muestras[i,],conf.level=1-alfa)
```

```
  exinf[i]<-test$conf.int[1]
  exsup[i]<-test$conf.int[2]
}

plot(gl(M,n),as.vector(t(muestras)),main=paste("100 I.C. 95% ","n=",n), xlab="Muestra", ylab="x",col="li
abline(h=media,col="red",lwd=2,lty=2)
lines(exinf,col="darkblue",lwd=4)
lines(exsup,col="darkblue",lwd=4)
```




```
cat("Cobertura observada =",
100* mean((exinf<= media) & (exsup>=media)), "% \n")
```

```
## Cobertura observada = 97 %
```

```
cat("Longitud media =", mean(exsup-exinf),"\n")
```

```
## Longitud media = 1.4
```

Ejercicio 5

5. Escribir y probar una función R para generar muestras de una Weibull:

$$f(t) = \lambda\alpha(\lambda t)^{\alpha-1}e^{-(\lambda t)} \;,\; F(t) = 1 - e^{-(\lambda t)^{\alpha}} \;,\; \lambda, \alpha > 0$$

Generar muestras de tamaño 200 para las configuraciones siguientes, donde:

- λ es el parámetro de escala
- α es el parámetro de forma
- $\alpha = 0.5$, $\lambda = 1$

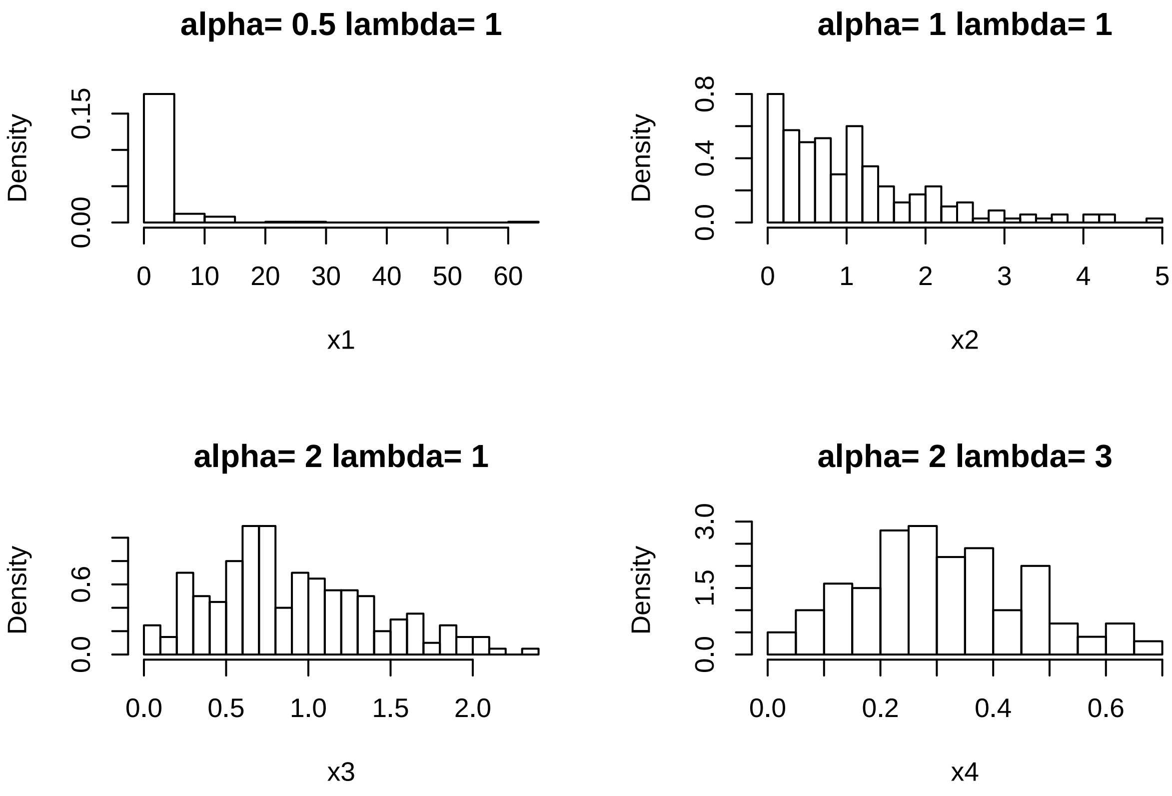
- $\alpha = 1$, $\lambda = 1$
- $\alpha = 2$, $\lambda = 1$
- $\alpha = 2$, $\lambda = 3$

```
generaweib<- function(n,alfa,landa){
  U<- runif(n)
  ((-log(1-U))^(1/alfa))/landa
}
#i)
n<-200
set.seed(129871)
x1<- generaweib(n,0.5,1)
x2<- generaweib(n,1,1)
x3<- generaweib(n,2,1)
x4<- generaweib(n,2,3)
```

Dibujar los histogramas

```
tit1<-paste("alpha=",0.5,"lambda=",1)
tit2<-paste("alpha=",1,"lambda=",1)
tit3<-paste("alpha=",2,"lambda=",1)
tit4<-paste("alpha=",2,"lambda=",3)

par(mfrow=c(2,2))
hist(x1,main=tit1,br=20,prob=TRUE)
hist(x2,main=tit2,br=20,prob=TRUE)
hist(x3,main=tit3,br=20,prob=TRUE)
hist(x4,main=tit4,br=20,prob=TRUE)
```



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

Representar las funciones de distribución empírica y superponer las teóricas

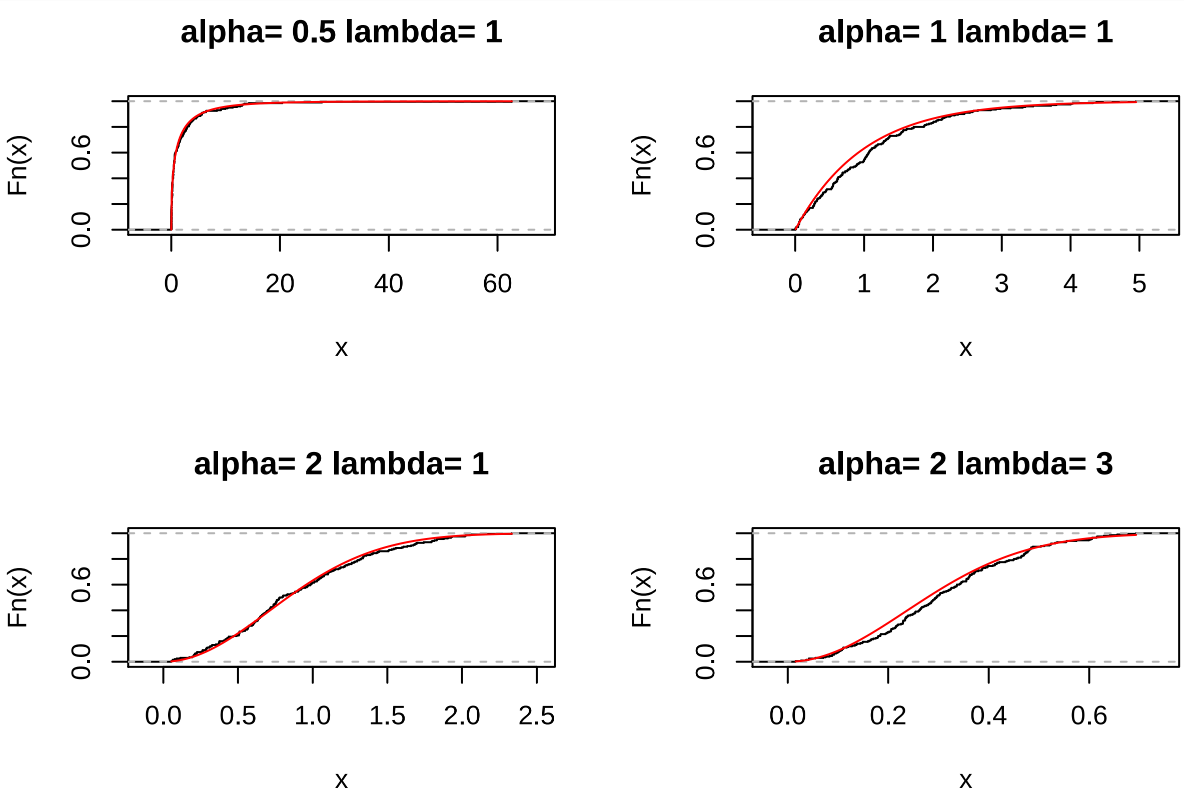
```
par(mfrow=c(2,2))
plot(ecdf(x1),main=tit1,do.points=FALSE,verticals=TRUE)
curve(pweibull(x,0.5,1),min(x1),max(x1),1000,add=TRUE,col="red")

# ecdf: EMpirical cumulative distribution Function

plot(ecdf(x2),main=tit2,do.points=FALSE,verticals=TRUE)
curve(pweibull(x,1,1),min(x2),max(x2),1000,add=TRUE,col="red")

plot(ecdf(x3),main=tit3,do.points=FALSE,verticals=TRUE)
curve(pweibull(x,2,1),min(x3),max(x3),1000,add=TRUE,col="red")

plot(ecdf(x4),main=tit4,do.points=FALSE,verticals=TRUE)
curve(pweibull(x,2,1/3),min(x4),max(x4),1000,add=TRUE,col="red")
```




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

Contrastes de bondad de ajuste (fitdistrplus)

SIGUE UNA WEIBULL FRENTE A NO. ESCRIBIR.

Debemos estimar los parámetros por el método de la máxima verosimilitud.

```
library(fitdistrplus)
```

```r
print(mv1<-mledist(x1,"weibull"))
```

```
## $estimate
## shape scale
##  0.49  1.07
##
## $convergence
## [1] 0
##
## $value
## [1] 236
##
## $hessian
##       shape scale
## shape  1551   -79
## scale   -79    41
##
## $optim.function
## [1] "optim"
##
## $optim.method
## [1] "Nelder-Mead"
##
## $fix.arg
## NULL
##
## $fix.arg.fun
## NULL
##
## $weights
## NULL
##
## $counts
## function gradient
##       43       NA
##
## $optim.message
## NULL
##
## $loglik
## [1] -236
```

Test de Kolmogorov-Smirnov de bondad de ajuste

Es el mejor.

```r
ks.test(x1,"pweibull",mv1$estimate[1], 1/mv1$estimate[2])
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  x1
## D = 0.05, p-value = 0.8
## alternative hypothesis: two-sided
```

```
#R usa 1/landa
```

Cuidado al dar los parámetros.

Conclusión, p-valor mayor que alpha, no tenemos evidencias para negar que la muestra x1 siga una distribución pweibull.

Otra opción: ajustar los datos mediante fitdist y luego aplicar gofstat: Aquí lo que se muestra es el p-valor del test chi-cuadrado de bondad de ajuste.

```
fitweib1 <- fitdist(x1, "weibull")
summary(fitweib1)
```

```
## Fitting of the distribution ' weibull ' by maximum likelihood
## Parameters :
##       estimate Std. Error
## shape     0.49      0.027
## scale     1.07      0.165
## Loglikelihood:  -236   AIC:  476   BIC:  483
## Correlation matrix:
##       shape scale
## shape  1.00  0.31
## scale  0.31  1.00
```

Obtenemos los estimadores, que coinciden con los anteriores. También calcula el log de la verosimilitud y calcula los criterios AIC y BIC.

MEDIANTE el test Chi-Cuadrado

```
gofstat(fitweib1)$chisqpvalue
```

```
## [1] 0.37
```

Acepto H0, no encontramos evidencias de que no la sigue.

```
fitweib2 <- fitdist(x2, "weibull")
summary(fitweib2)
```

```
## Fitting of the distribution ' weibull ' by maximum likelihood
## Parameters :
##       estimate Std. Error
## shape      1.1      0.060
## scale      1.1      0.078
## Loglikelihood:  -218   AIC:  440   BIC:  447
## Correlation matrix:
##       shape scale
## shape  1.00  0.31
## scale  0.31  1.00
```

```
gofstat(fitweib2)$chisqpvalue
```

```
## [1] 0.25
```

```
fitweib3 <- fitdist(x3, "weibull")
summary(fitweib3)
```

```
## Fitting of the distribution ' weibull ' by maximum likelihood
## Parameters :
##       estimate Std. Error
```

```
## shape      1.9       0.11
## scale      1.0       0.04
## Loglikelihood:  -131   AIC:  267   BIC:  273
## Correlation matrix:
##       shape scale
## shape  1.00  0.31
## scale  0.31  1.00
```

```
gofstat(fitweib3)$chisqpvalue
```

```
## [1] 0.32
```

```
fitweib4 <- fitdist(x4, "weibull")
summary(fitweib4)
```

```
## Fitting of the distribution ' weibull ' by maximum likelihood
## Parameters :
##       estimate Std. Error
## shape     2.20      0.124
## scale     0.35      0.012
## Loglikelihood:  103   AIC:  -202   BIC:  -195
## Correlation matrix:
##       shape scale
## shape  1.00  0.31
## scale  0.31  1.00
```

```
gofstat(fitweib4)$chisqpvalue
```

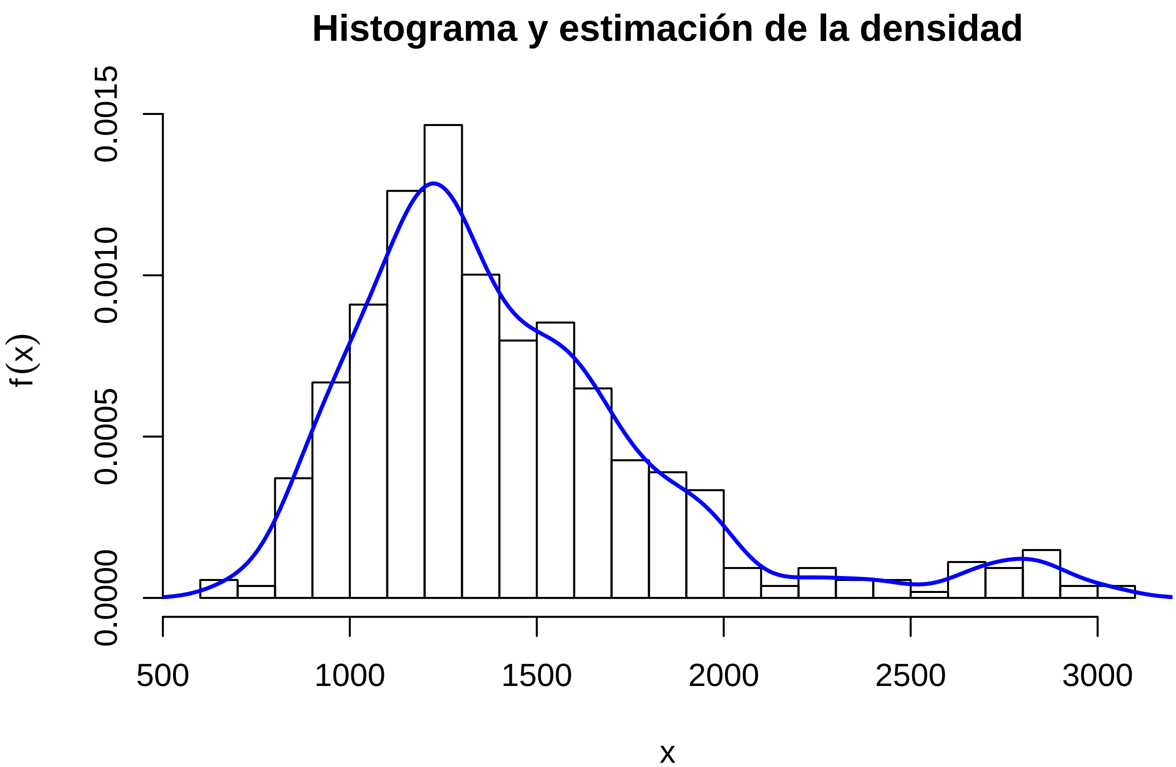
```
## [1] 0.43
```

Ejercicio 6 "Pesos.RData"

Leer el fichero datos en "Pesos.RData", y a continuación:

Estimar la densidad por el método del núcleo.

```
load("datos/Pesos.RData")
hist(datos,
     br=30,
     prob=TRUE,
     main="Histograma y estimación de la densidad",
     ylab = expression(hat(f)(x)),xlab="x")
lines(density(datos,bw="SJ"),col="blue",lwd=2) # Estimación por el método del núcleo con density
```



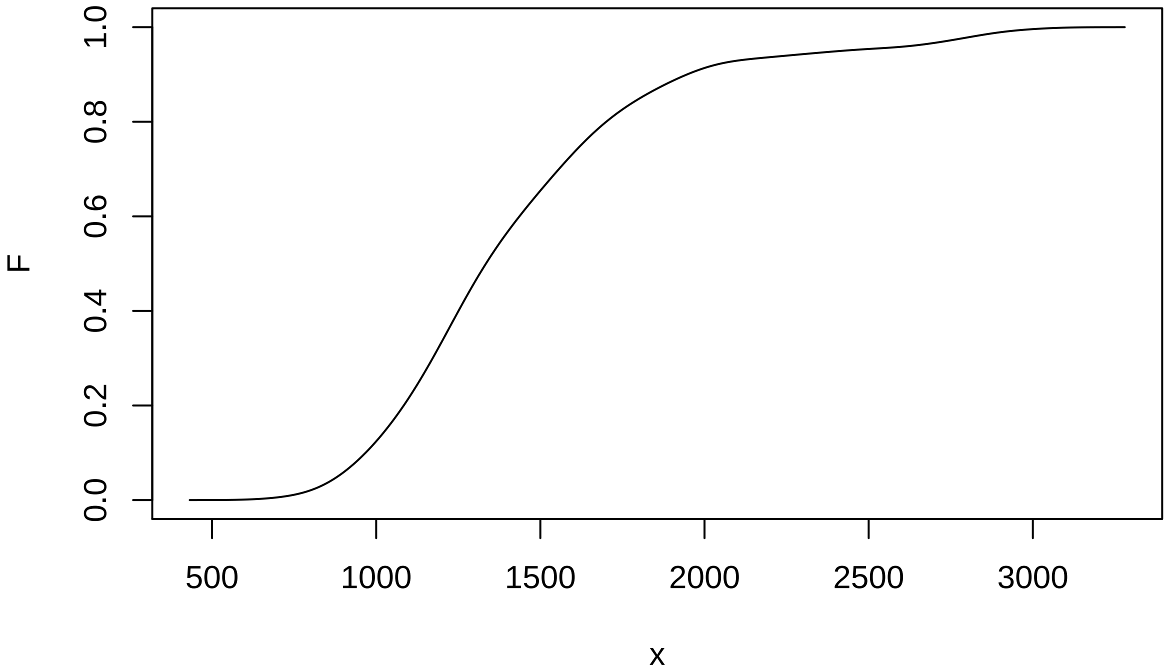

```
## SJ es el método que mejor se porta
```

Vamos a crear una muestra de datos a partir de la función de densidad estimada, serán estimaciones.

```
estimnuc<- density(datos,bw="SJ",n=5000) #Para tener más puntos
# Devuelve los puntos sobre los que ha hecho las estiamciones

distrib<-data.frame(x=estimnuc$x,
                    F=cumsum(estimnuc$y)/sum(estimnuc$y))
# Aquí tenemos una tabla de valores de la función de distribución, no de densidad

plot(distrib,type="l") # Función de distribución
```



Escribir una función para generar valores según dicha densidad estimada.

```
generarx<- function(n,distrib) {
  U<-runif(n)
  sapply(U, function(u) min(distrib$x[distrib$F>=u]) ) }
## Mínimo x que supera al valor que busco en la FDD
## Función inversa de la función tabulada
## Sapply: acada elemento del vector que le paso, le aplica la función
generarx(10,distrib)
```

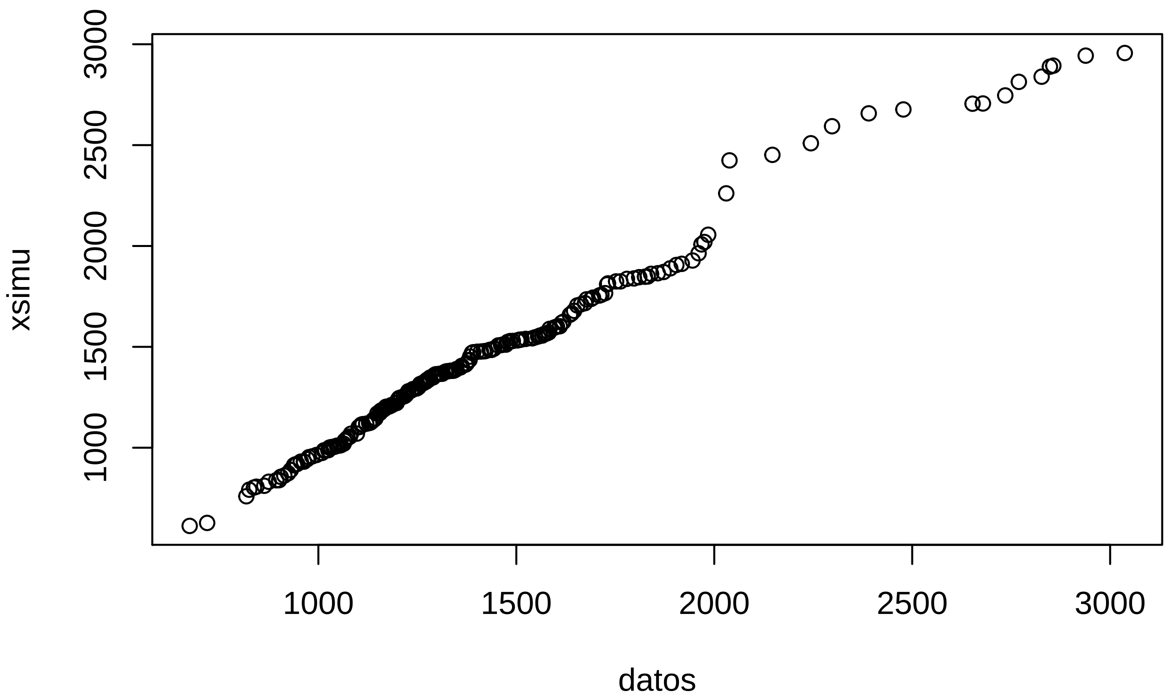
```
##  [1] 1284 1688 1524 1264 1053  870 1369 1310 1476 1257
```

Simulación de valores:

```
xsimu<- generarx(200,distrib)
```

Lo dibujo

```
qqplot(datos,xsimu)
```




```
summary(datos)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     675    1144    1327    1428    1612    3037
```

```
summary(xsimu)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     613    1144    1379    1457    1607    2956
```

Vemos que la mediana con el segundo método es mayor, al igual que la media. Sin embardo, estas evidencias no son significativas para rechazar el test que realizamos a continuación:

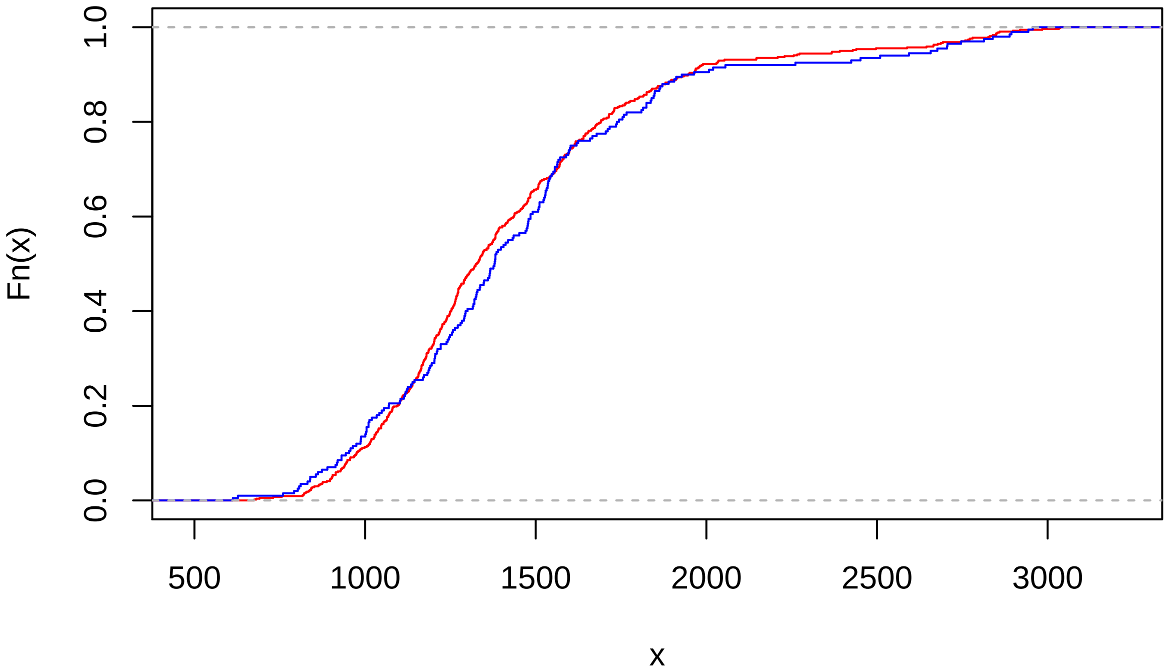
Comparar las distribuciones de una muestra generada de tamaño 200 y el conjunto de datos original.

```
ks.test(datos,xsimu)
```

```
## Warning in ks.test(datos, xsimu): p-value will be approximate in the presence of
## ties
```

```
##
##  Two-sample Kolmogorov-Smirnov test
##
## data:  datos and xsimu
## D = 0.08, p-value = 0.3
## alternative hypothesis: two-sided
```

```
plot(ecdf(datos),main="",do.points=FALSE, verticals=TRUE,col="red")
lines(ecdf(xsimu),do.points=FALSE, verticals=TRUE,col="blue")
```



Aquí vemos porque K-S no rechaza.

Ejercicio 7

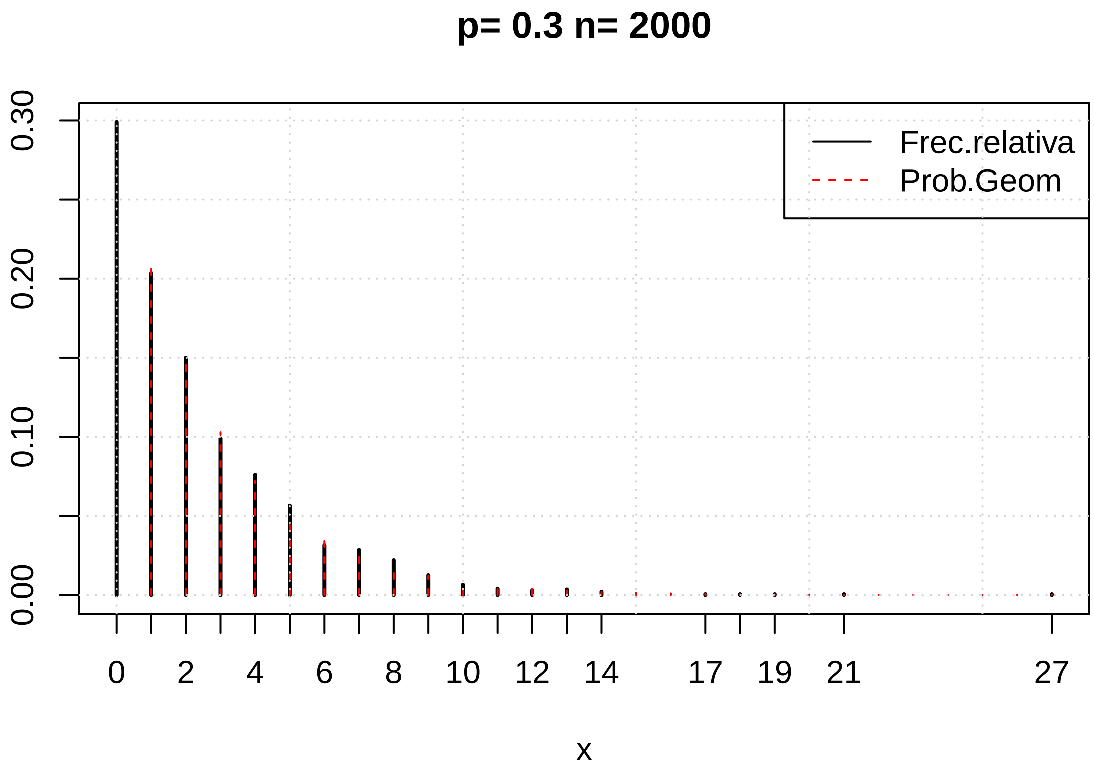
7. Diseñar una función para generar realizaciones de una ley Geométrica simulando el proceso de conteo del número de fracasos antes del primer éxito en la repetición de ensayos Bernouilli. Probar la función y analizar los resultados.

```
#para generar muestra de tamaño n de Ge(p)
#
Geom<- function(n,p) {
 #Inicializaciones
X<- integer(n)
#algoritmo
for (i in 1:n) {
s<- -1
repeat
{
s<- s+1
U<- runif(1)
if (U <= p) break
}
X[i]<- s
}
X
}
```

```
p<- 0.3
n<- 2000
set.seed(12345)
x<- Geom(n,p)
(tabla<-table(x))
```

```
## x
##   0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  17  18  19  21  27
## 598 407 300 198 152 113  63  57  44  25  13   8   6   7   4   1   1   1   1   1
```

```
vx<- 1:max(x)
plot(prop.table(tabla),
     type="h",
     ylab="",
     main=paste("p=",p,"n=",n))
lines(vx,dgeom(vx,p),type="h",col="red",lty=2)
legend("topright",
       lty=1:2,
       col=c("black","red"),
       legend=c("Frec.relativa","Prob.Geom"))
grid()
```

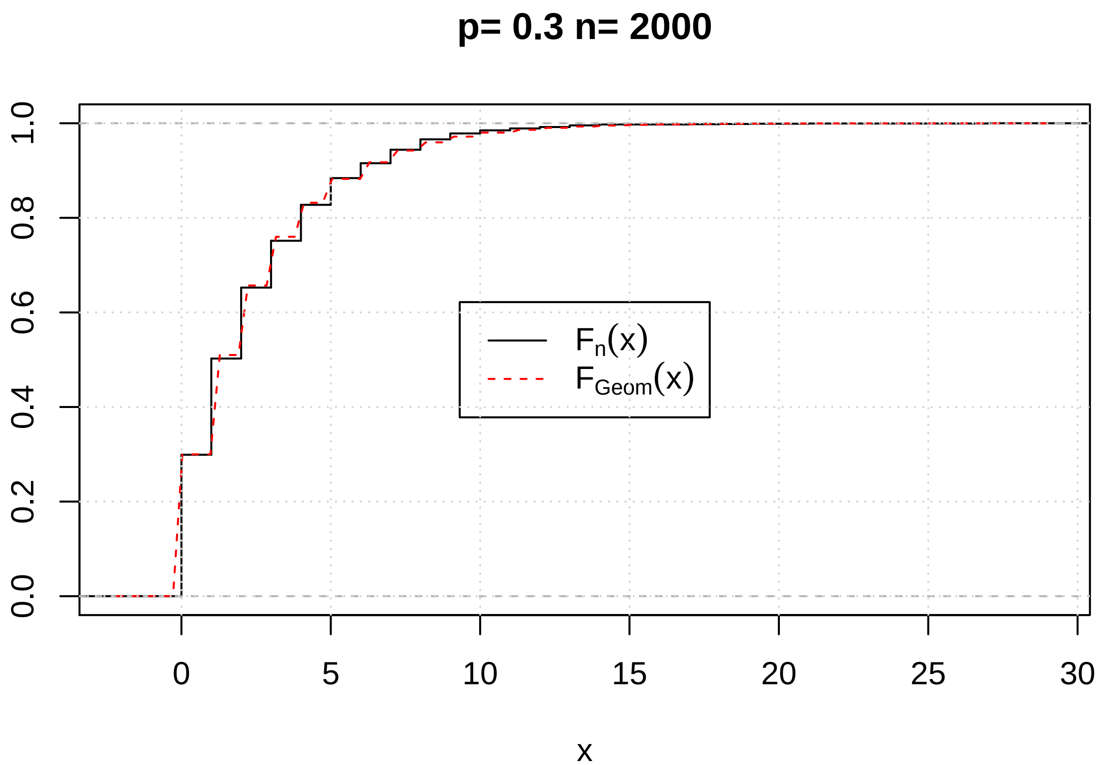


No poner valores de p muy pequeños.

```
plot(ecdf(x),
     do.points=FALSE,
     verticals=TRUE,
     ylab="",
     main=paste("p=",p,"n=",n))

curve(pgeom(x,p),
      add=TRUE,
      col="red",
      lty=2)
legend("center",
       lty=1:2,
       col=c("black","red"),
```

```
       legend=expression(F[n](x),F[Geom](x)))
grid()
```



Fin hoja