Tarea III

Rayan García Fabián 144424, Bernardo Mondragon Brozon 143743, Karen DElgado Curiel 142252, Diego Garcia 14xxxx

1 October 2018

Problema 1

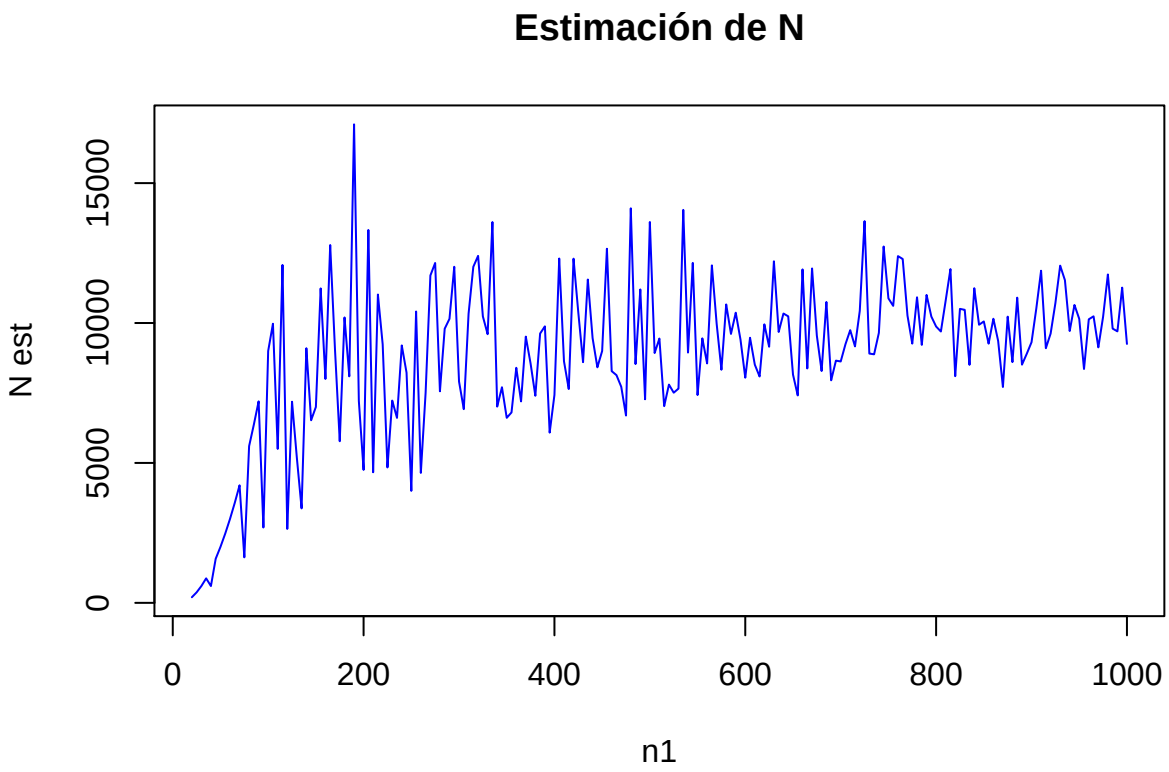
Un estadístico está interesado en el número N de peces en un estanque. Él captura 250 peces, los marca y los regresa al estanque. Unos cuantos días después regresa y atrapa peces hasta que obtiene 50 peces marcados, en ese punto también tiene 124 peces no marcados (la muestra total es de 174 peces).

- ¿Cuál es la estimación de N?
- Haga un programa, que permita simular el proceso de obtener la primera y segunda muestra considerando como parámetros el tamaño N de la población de interés, el tamaño de la primera y segunda muestra y como datos a estimar son: de qué tamaño debe ser n_1 y n_2 para obtener una buena aproximación y ver cómo se afecta por el tamaño N.

Solución:

Primero definamos la notación a usar. Consideremos N como el número de peces en la población, n_1 el número de peces marcados en la primer muestra, n_2 el número de peces capturados en la segunda muestra y r el número de animales de la segunda muestra que están marcados. Sabemos que $N = 174, n_1 = 250, n_2 = 174, r = 50$.Entonces por el método de Lincoln–Petersen, suponiendo que no cambió la población de peces entre el momento de la primera y segunda muestra, se estima que $\hat{N} = \frac{n_1 n_2}{r}$, sustituyendo valores $\hat{N} = 870$.

```
Simulacion_Peces<-function(N,n1,n2){ #Función para simular el proceso de obtener la primera y segunda m
  pecesM<-c(rep("M",n1))
  pecesT<-c(pecesM,c(rep("NM",N-n1)))
  muestra2<-sample(pecesT,n2)
  r<-length(subset(muestra2,muestra2=="M"))
  N_est<-(n1*n2)/(r+1) #Ojo aquí consideramos r+1 ya que como no asignamos probabilidades a la muestra,
  return(N_est)
}

#Supongamos N=10000, vamos a ver de qué tamaño deben ser n1 y n2 para obtener una buena estimación de N
N<-10000
n1<-c(seq(20,1000,by=5))
n2<-c(seq(10,990,by=5))

M_est<-c()
for (i in 1:length(n1)){
  M_est[i]<-Simulacion_Peces(N,n1[[i]],n2[[i]])
}

plot(n1[1:length(n1)],M_est[1:length(n1)],type = "l",col="blue",main = "Estimación de N", xlab = "n1",yl
```



■

Problema 2

Este problema es una versión simplificada de dos problemas comunes que enfrentan las compañías de seguros: calcular la probabilidad de quebrar y estimar cuánto dinero podrán hacer.

Supongan que una compañía de seguros tiene activos (todo en dólares) por \$1000000. Tienen $n = 1000$ clientes que pagan individualmente una prima anual de \$5500 al principio de cada año. Basándose en experiencia previa, se estima que la probabilidad de que un cliente haga un reclamo en el año es de $p = 0.1$, independientemente del número de reclamos previos de otros clientes. El tamaño X de los reclamos varía y tiene la siguiente distribución de probabilidad:

$$f_X(x) = \frac{\alpha\beta^\alpha}{(x+\beta)^{\alpha+1}} I_{[0,\infty)}(x)$$

con $\alpha = 5$ y $\beta = 125000$ (Tal X tiene una distribución Pareto, la cual es frecuentemente usada para modelar el monto de un siniestro). Suponemos las fortunas de la compañía aseguradora sobre un horizonte de 5 años. Sea $Z(t)$ el valor de los activos de la compañía al final del t-ésimo año, de manera que que $Z(0) = 1000000$ y

$$Z(t) = \max(Z(t-1) + P(t) - S(t), 0)$$

donde $P(t)$ es el monto de las primas pagadas durante el t-ésimo año y $S(t)$ es el monto de los siniestros pagados durante el t-ésimo año. Notar que si $Z(t)$ cae bajo 0, entonces la compañía se va a la bancarrota y deja de operar.

1. Calcular $F_X(x)$, $E(X)$ y $Var(X)$. Obtener por simulación una muestra de X y hallar los valores estimados de las cantidades anteriores y compararlos con los valores teóricos.
2. Escriban una función para simular los activos de la compañía por cinco años y estimar lo siguiente: (1) La probabilidad de que la compañía se vaya a la bancarrota. (2) Los activos esperados al final del quinto año.
3. Si el valor de los activos rebasan la cantidad de \$1000000, entonces el excedente se reparte entre los accionistas como dividentos de manera que si $D(t)$ son los dividendos pagados al final del t-ésimo año, entonces
$$D(t) = \begin{cases} 1000000 - Z(t) & \text{si} \quad Z(t) > 1000000 \\ 0 & \text{si} \quad Z(t) \leq 1000000 \end{cases}.$$
Bajo este nuevo esquema, estimar (1) la probabilidad de irse a la quiebra, (2) los activos esperados después de 5 años, y (3) las ganacias totales esperadas después de 5 años de operación.

Solución:

$$F_X = \int_0^x \frac{\alpha\beta^\alpha}{(s+\beta)^{\alpha+1}} ds = \alpha\beta^\alpha \int_0^x \frac{1}{(+\beta)^{\alpha+1}} ds = \alpha\beta^\alpha \int_\beta^{\beta+x} \frac{1}{u^{\alpha+1}} du = \alpha\beta^\alpha \left(\frac{-u^{-\alpha}}{\alpha}\right)\Big|_\beta^{\beta+x} = 1 - \left(\frac{\beta}{\beta+x}\right)^\alpha$$

Para el cálculo de la esperanza notemos que X es una v.a no negativa, entonces se puede proceder de manera más sencilla por medio de la función de supervivencia $S(x) = 1 - F_X(x)$

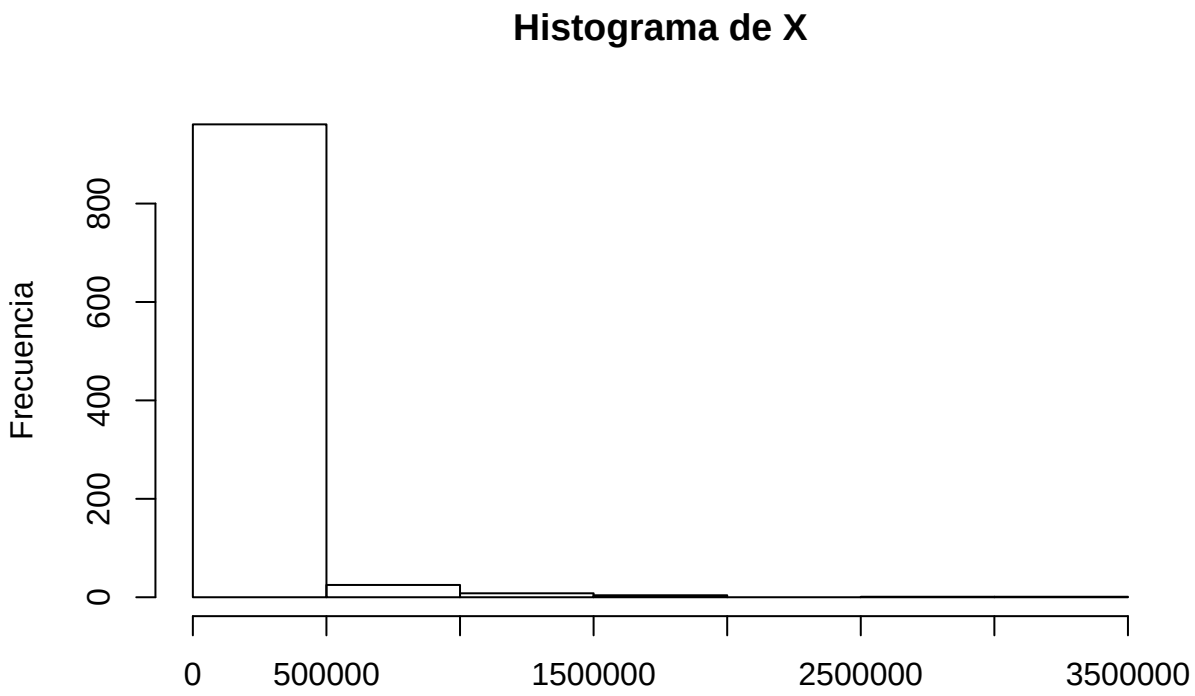
$$E[X] = \int_0^\infty x(1 - F_X(x))dx = \int_0^\infty \frac{\beta^\alpha}{(x+\beta)^\alpha} dx = \frac{\beta}{\alpha - 1}$$

De manera alternativa podemos considerar a Y distribuida Pareto del tipo I, sabemos que $E[Y] = \frac{\alpha\beta}{\alpha-1}$. Entonces sea $X = Y - \beta$, se tiene que $E[X] = E[Y] - \beta = \frac{\alpha\beta}{\alpha-1} - \beta$.

Partiendo de la transformación anterior se tiene que $Var(X) = Var(Y) = \frac{\alpha^2}{(\alpha-1)^2(\alpha-2)}; \alpha > 2$.

Ahora obtendremos por medio de simulación una muestra de X y su función de distribución

```
Pareto_Lomax<-function(a,b,n){
  u<-runif(n)
  Lomax_MA<-c()
  for (i in 1:n) {
  Lomax_MA[i]<-(b*(1-(1-u[[i]])^(1/a)))/((1-u[[i]])^(1/a))
  }
  return(Lomax_MA)
}

hist(Pareto_Lomax(2,125000,1000),main = "Histograma de X",xlab = "",ylab = "Frecuencia")
```



Realizamos la simulación de los activos de la compañía en un horizonte de cinco años.S upondremos además que la aseguradora siempre renueva la póliza con el cliente, esto es que una vez que un cliente reclama se le vuelve a vender la póliza al año siguiente.

```
registro<-function(activos,prima,n_clientes,prob_reclamo){
registro_activos<-c()
registro_activos[1]<-activos
for (i in 2:6) {
reclamos<-rbinom(n_clientes,1,prob_reclamo)
monto_reclamos<-Pareto_Lomax(5,125000,sum(reclamos))
registro_activos[i]<-registro_activos[i-1]-sum(monto_reclamos)+n_clientes*prima
if(registro_activos[i-1]<=0){
  registro_activos[i]<-0
  }
}
return(registro_activos)
}

activos<-10e6
prima<-5500
n_clientes<-1000
prob_reclamo<-0.1

huella<-matrix(,100,6)
for (i in 1:100) {
  huella[i,]<-registro(activos =activos, prima = prima, n_clientes = n_clientes,prob_reclamo = prob_rec
}
```

```r
colnames(huella)<-c("z(0)","z(1)","z(2)","z(3)","z(4)","z(5)")
huella<-as.data.frame(huella)
head(huella)
```

```
##    z(0)     z(1)     z(2)     z(3)     z(4)     z(5)
## 1 1e+07 11376414 14084025 16253591 18346613 20439616
## 2 1e+07 11763781 14022049 16059061 17665593 19902378
## 3 1e+07 12177850 14621978 17785280 19951475 22841644
## 4 1e+07 11244458 13877150 16706456 18780172 20692663
## 5 1e+07 12008646 14723090 17263693 18788213 20768207
## 6 1e+07 12196318 13736852 16022083 18076134 20348058
```

Notemos que hasta el momento no hemos visto ningún caso de ruina. Estudiemos un esquema en el que si el registro de activos en el periodo excede cierta cantidad, entonces se recompensa a los accionistas.

```r
registro_acc<-function(activos,prima,n_clientes,prob_reclamo){
registro_activos<-c()
registro_activos[1]<-activos
for (i in 2:6) {
reclamos<-rbinom(n_clientes,1,prob_reclamo)
monto_reclamos<-Pareto_Lomax(5,125000,sum(reclamos))
if(registro_activos[i-1]<=0){
  registro_activos[i:6]<-0
}
else if(registro_activos[i-1]>1000000){
registro_activos[i]<-registro_activos[i-1]-(10e6)-sum(monto_reclamos)+n_clientes*prima
}
else{registro_activos[i]<-registro_activos[i-1]-sum(monto_reclamos)+n_clientes*prima
}
}
return(registro_activos)
}

activos<-10e6
prima<-5500
n_clientes<-1000
prob_reclamo<-0.1

huella<-matrix(,100,6)
for (i in 1:100) {
  huella[i,]<-registro_acc(activos =activos, prima = prima, n_clientes = n_clientes,prob_reclamo = prob
}

colnames(huella)<-c("z(0)","z(1)","z(2)","z(3)","z(4)","z(5)")
huella<-as.data.frame(huella)
head(huella)
```

```
##    z(0)    z(1)     z(2) z(3) z(4) z(5)
## 1 1e+07 2336582 -5584003    0    0    0
## 2 1e+07 1907752 -5500353    0    0    0
## 3 1e+07 2616197 -4016336    0    0    0
## 4 1e+07 2501759 -5447880    0    0    0
## 5 1e+07 2976290 -4626018    0    0    0
## 6 1e+07 3150258 -4455708    0    0    0
```

Notemos que bajo este esquema en el que los accionistas son recompensados, la probabilidad de quiebra es 1.

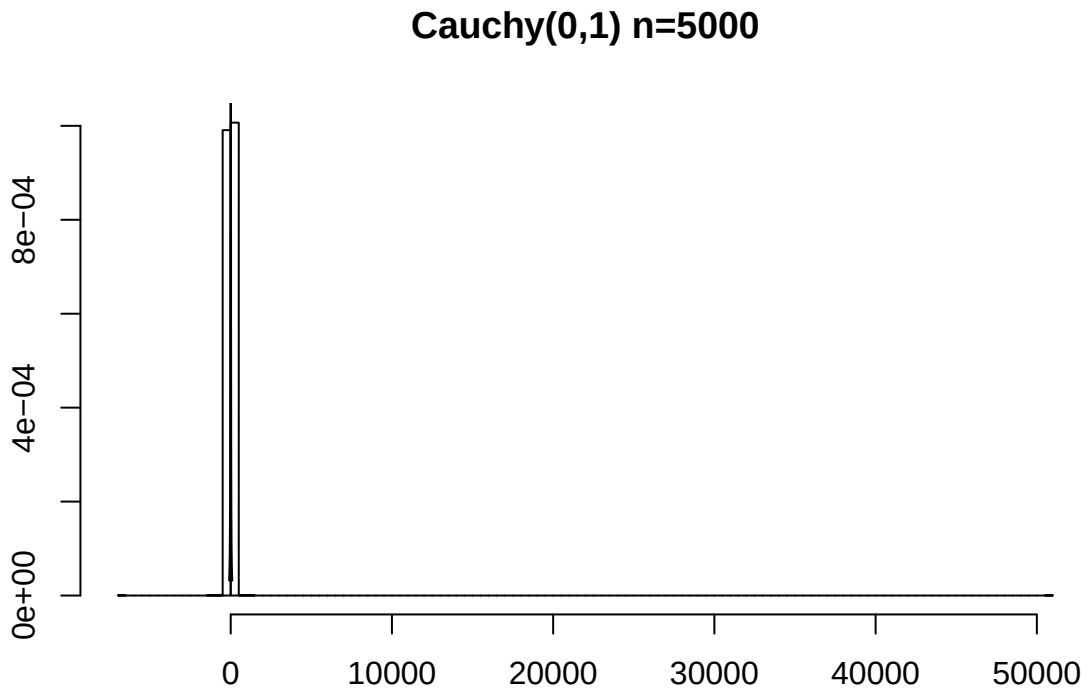
Problema 3

Proponer algoritmos (método y pseudocódigo o código, así como una corrida) para generar muestras de las siguientes densidades.

*Cauchy $f(x) = \frac{1}{\pi\beta\left[1+(\frac{x-\gamma}{\beta})^2\right]}; \gamma, x \in \mathbb{R}; \beta > 0$

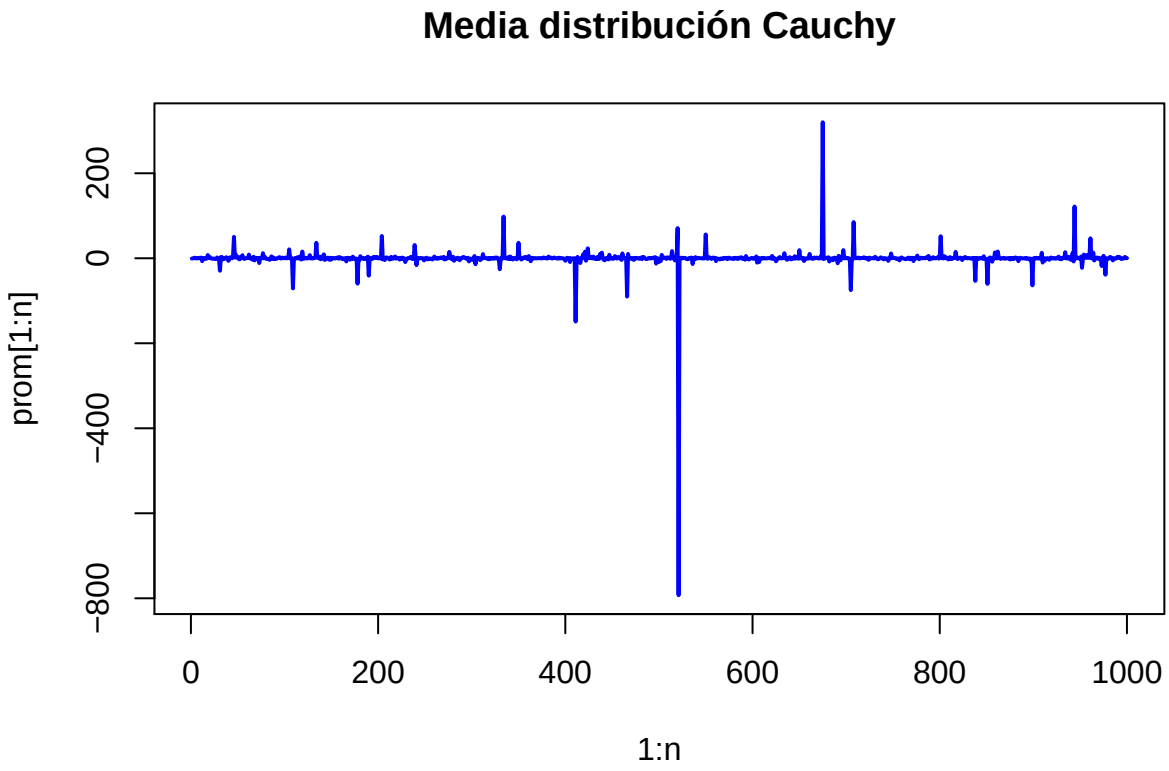
Solución: Podemos encontrar la distribución de X como $F_X(x) = \frac{arctan(\frac{x-\gamma}{\beta})}{\pi} + \frac{1}{2}$, entonces $F^{-1}(u) = tan(\pi(u - \frac{1}{2}))$. Usamos el método de la transformada inversa

```
cauchy<-function(gamma,beta,n){
u<-runif(n)
u<-tan(pi*u)*beta+gamma
return(u)
}
x<-1:100
hist(cauchy(0,1,5000),probability=T,breaks=100,main = "Cauchy(0,1) n=5000",ylab = "",xlab = "")
curve(dcauchy(x),add=T,from=-100,to=100)
```




```
U<- numeric(1000)
n<-1000
prom<-numeric(n)
y<-c()
```

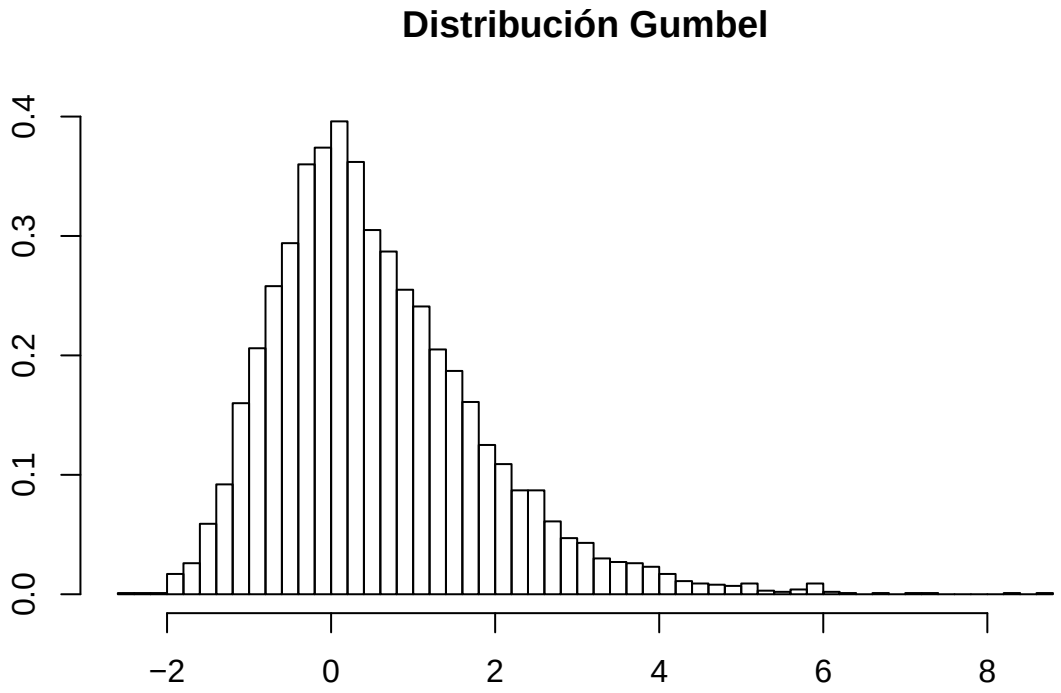
```r
for (i in 1:n) {
  u<-runif(1000)
  x<-tan(pi*(u-0.5))
  prom[i]<-mean(x)
}
plot(1:n,prom[1:n],type="l",lwd=2,col="blue",main = "Media distribución Cauchy")
```



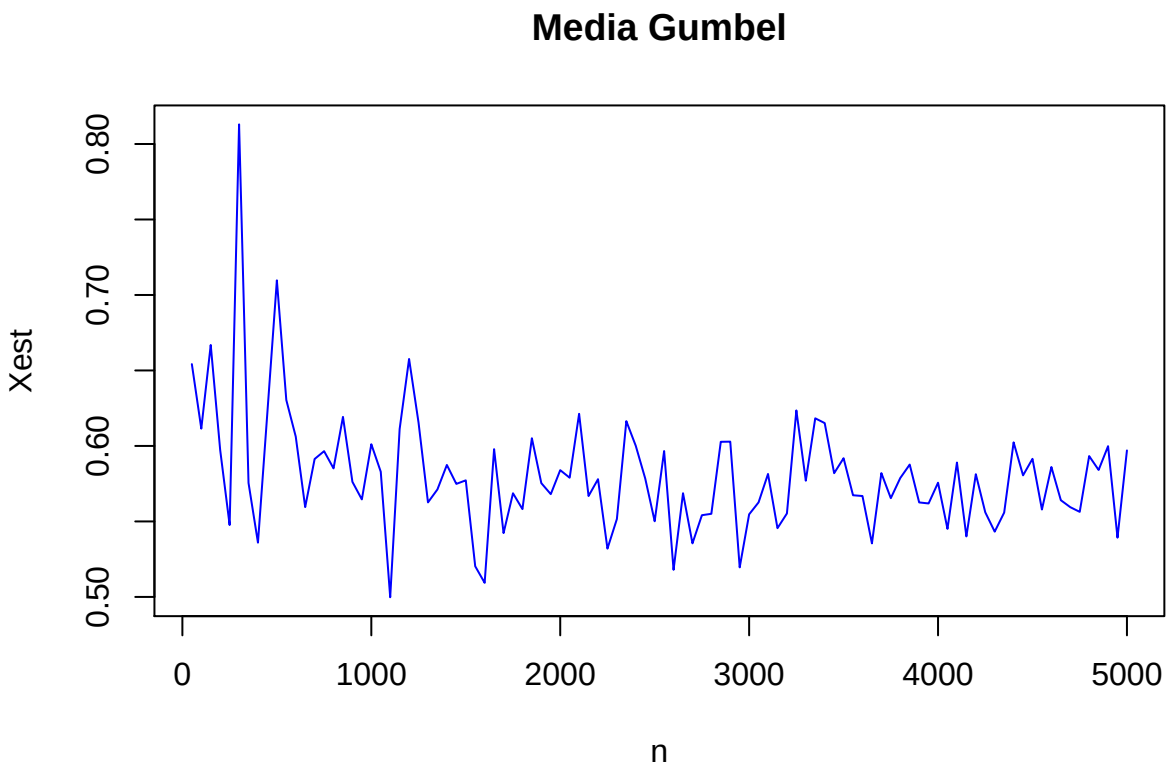
*Gumbel $f(x) = \frac{1}{\beta} exp\left[-e^{-\frac{(x-\gamma)}{\beta}} - \frac{x-\gamma}{\beta}\right]; \gamma, x \in \mathbb{R}; \beta > 0.$

Solución:

```r
gumbel<-function(gamma,beta,n){
unif<-runif(n)
unif<--beta*log(-log(unif))+gamma
return(unif)
}
hist(gumbel(0,1,5000),probability =T,breaks=60,xlab = "",ylab="",main = "Distribución Gumbel")
```




```r
n_s<-c(seq(50,5000,by=50))
X_est<-c()
for (i in 1:length(n_s)) {
  X_est[i]<-sum(gumbel(0,1,n_s[i]))/n_s[i]
}

plot(n_s[1:length(n_s)],X_est[1:length(n_s)],type = "l",col="blue", xlab = "n",ylab = "Xest",main="Media
```



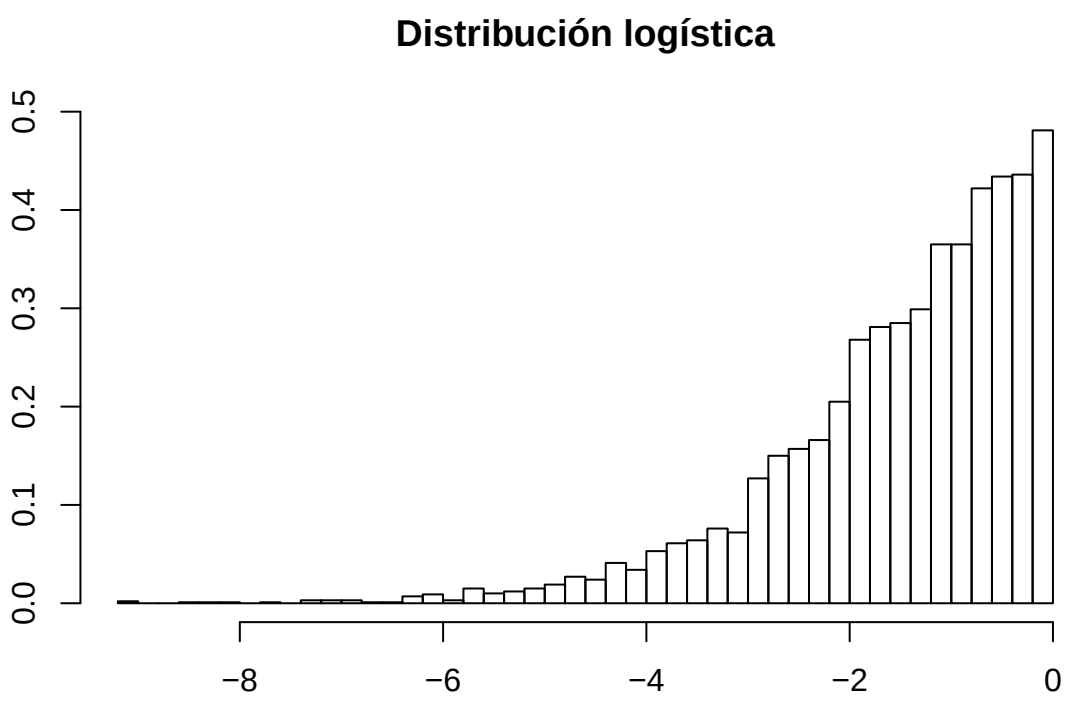
*Logística

$$f_X(x) = \frac{e^{-\frac{x-\gamma}{\beta}}}{\beta\left(1+e^{-\frac{x-\gamma}{\beta}}\right)^2} \quad \text{con} \quad \gamma, x \in \mathbb{R} \quad \text{y} \quad \beta > 0$$

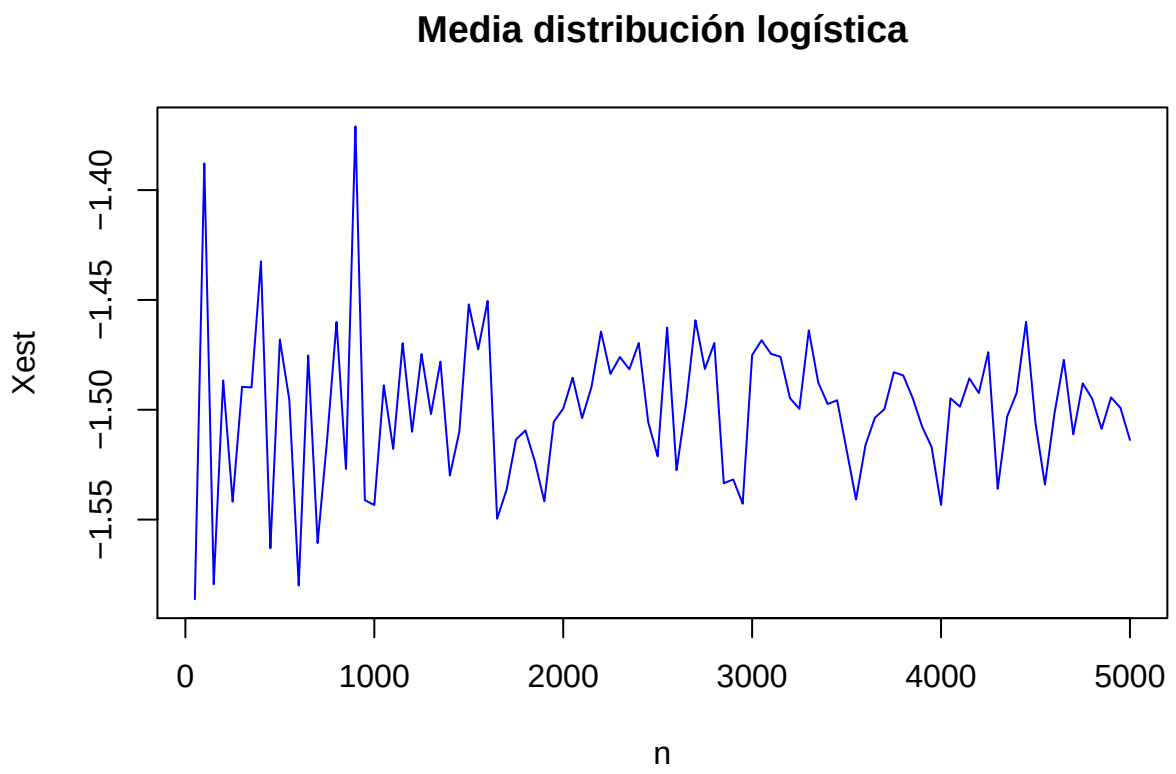
Solución:

Para esto utilizaremos el hecho de que si $X \sim U(0,1)$, entonces $\gamma + \beta(log(X) - (1-X)) \sim log(\gamma, \beta)$

```
Dist_logistica<-function(n,g,b){
u<-runif(n)
x<-g+b*(log(u)-1+u)
return(x)
}

hist(Dist_logistica(5000,0,1),probability =T,breaks=60,main = "Distribución logística",xlab = "",ylab =
```




```
n_s<-c(seq(50,5000,by=50))
X_est<-c()
for (i in 1:length(n_s)) {
  X_est[i]<-sum(Dist_logistica(n_s[i],0,1))/n_s[i]
}

plot(n_s[1:length(n_s)],X_est[1:length(n_s)],type = "l",col="blue", xlab = "n",ylab = "Xest",main="Media
```



*Pareto $f(x) = \frac{\alpha_2 c^{\alpha_2}}{x^{\alpha_x+1}}; c > 0, \alpha_2 > 0, x > c.$

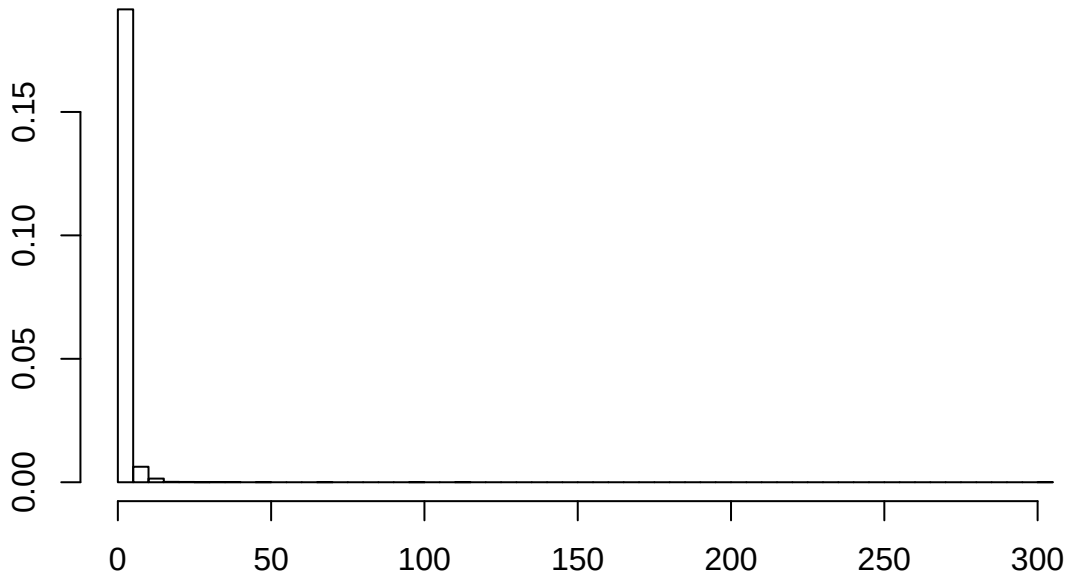
Solución:

$$F_X = \int_c^x \frac{\alpha_2 c^{\alpha_2}}{s^{\alpha_x+1}} ds = \alpha_2 c^{\alpha_2} \frac{s^{-\alpha_2}}{-\alpha_2}\Big|_c^x = 1 - (\frac{c}{x})^{\alpha_2}$$

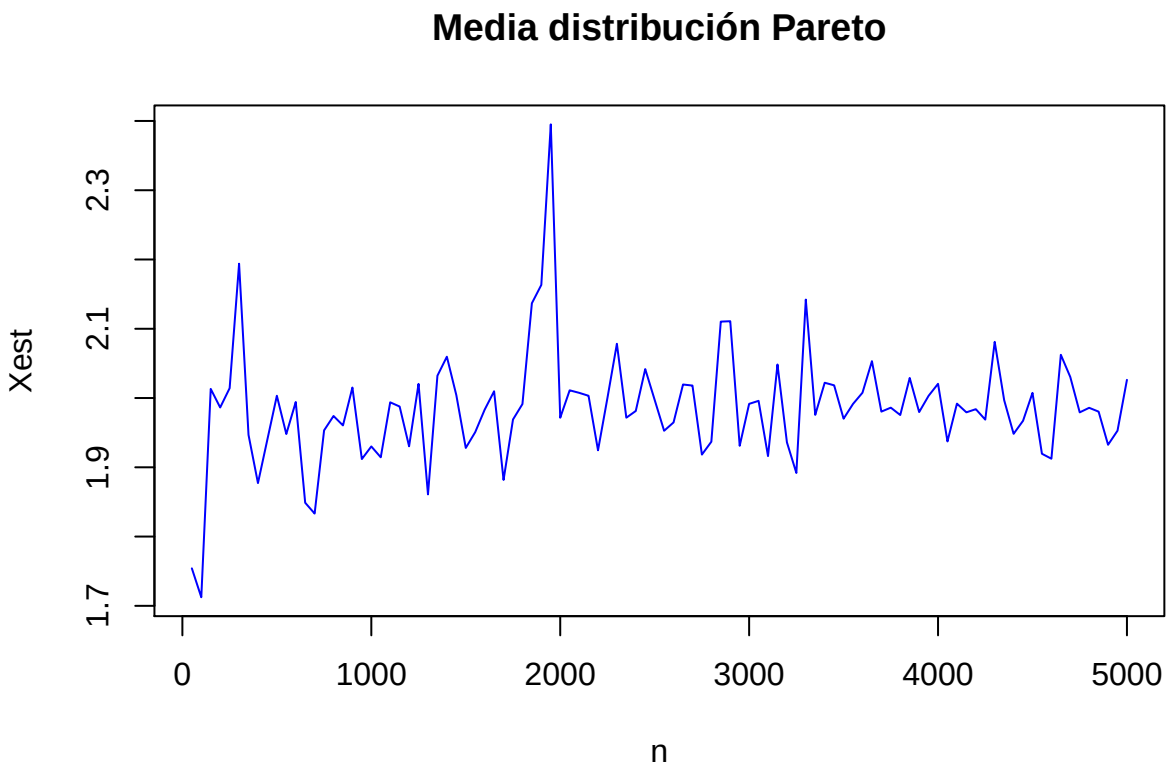
.

$$F^{-1}(u) = \frac{c}{(1-u)^{\frac{1}{\alpha_2}}}$$

```
S_pareto<-function(n,c,alfa){
  u<-runif(n)
  x<-c/(1-u)^(1/alfa)
  return(x)
}

hist(S_pareto(5000,1,2),probability =T,breaks=60,main = "Distribución Pareto",xlab = "",ylab = "")
```




```
n_s<-c(seq(50,5000,by=50))
X_est<-c()
for (i in 1:length(n_s)) {
  X_est[i]<-sum(S_pareto(n_s[i],1,2))/n_s[i]
}

plot(n_s[1:length(n_s)],X_est[1:length(n_s)],type = "l",col="blue", xlab = "n",ylab = "Xest", main="Medi
```



Problema 4

Grafiquen las siguientes densidades. Dar los algoritmos de transformación inversa, composición y aceptación-rechazo para cada una de las siguientes densidades. Discutir cuál algoritmo es preferible para cada densidad.

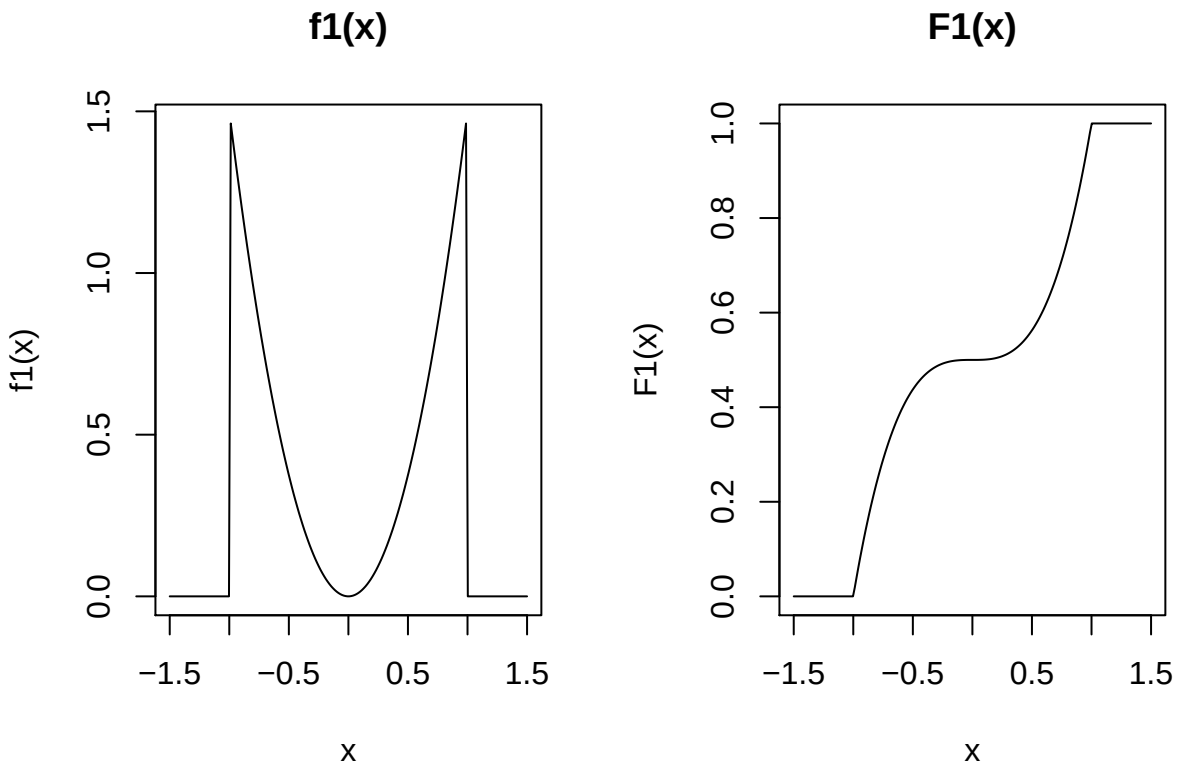
$$f(x) = \frac{3x^2}{2} I(x)_{[-1,1]}$$

$$f(x) = \begin{cases} 0, x \leq 0 \\ \frac{x}{a(1-a)}, 0 \leq x \leq a \\ \frac{1}{1-a}, a \leq x \leq 1-a \\ \frac{1-x}{a(1-a)}, 1-a \leq x \leq 1 \\ 0, x \geq 1 \end{cases}$$

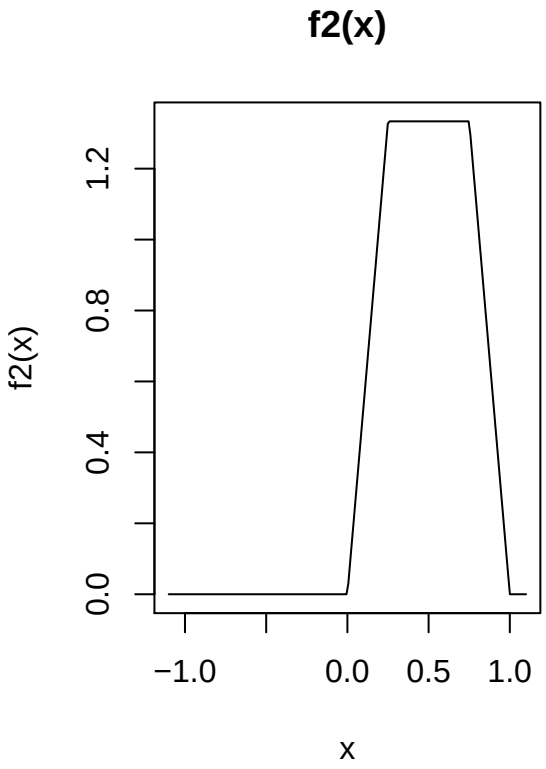
Solución:

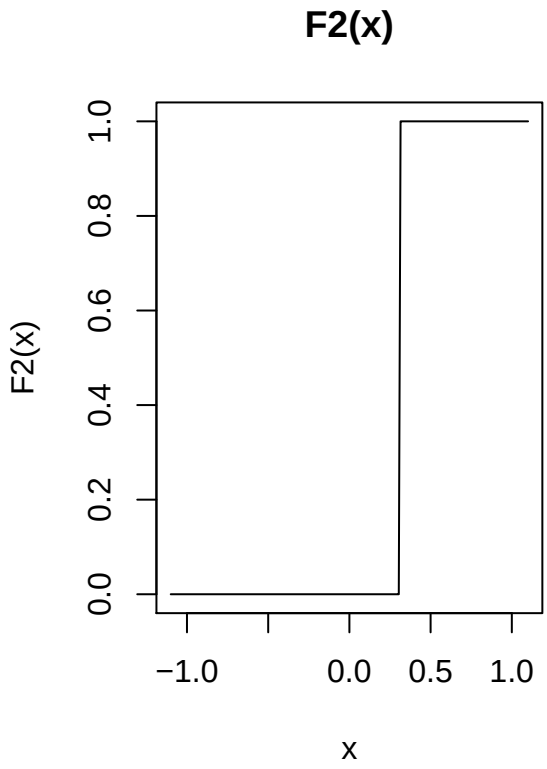
```
ind<-function(x,a,b){
ifelse(x<=b & x>= a,1,0)
}

f1<-function(x){
(3*(x^2)/2)*ind(x,-1,1)
}
```

```r
F1<-function(x){
ifelse(x<=-1,0,ifelse(x<=1,0.5*(x^3+1),1))
}

x<-seq(-1.5,1.5,length=200)
par(mfrow=c(1,2))
plot(x,f1(x),type="l",main="f1(x)")
plot(x,F1(x),type = "l",main = "F1(x)")
```




```r
f2<-function(x,a=0.25){
ind(x,-1,1)*(ind(x,0,a)*(x/(a*(1-a)))+ind(x,a,1-a)/(1-a)+ind(x,1-a,1)*((1-x)/(a*(1-a))))
}

F2<-function(x,a=0.25){
ind(x,0,a*x^2/(2*a*(1-a))+(x-a/2)/(1-a)*ind(x,a,1-a)+((1-3*a/2)/(1-a)+(x*(1-x/2)-(1-a)*(1+a)/2)/(a*(1-a
}

par(mfrow=c(1,2))
x<-seq(-1.1,1.1,length=200)
plot(x,f2(x),type="l",main="f2(x)")
plot(x,F2(x),type="l",main="F2(x)")
```





Problema 5

Considerando la transformación polar de Marsaglia para generar muestras de normales estándar, muestren que la probabilidad de aceptación de $S = V_1^2 + V_2^2$ en el paso 2 es $\frac{\pi}{4}$. Encuentre la distribución del número de rechazos de S antes de que ocurra una aceptación. ¿Cuál es el número esperado de ejecuciones del paso 1?

Solución:

Notemos que gráficamente estamos trabajando con un circulo unitario dentro de un cuadrado de 1x1, se acepta si $S = V_1^2 + V_2^2$ cae dentro del circulo, y se rechaza si cae en el área restante. Entonces la probabiidad de aceptación es el área del circulo unitario $A = \frac{\pi r^2}{2} = \frac{\pi}{4}$. Para modelar la distribución del número de rechazos de S antes de una aceptación, basta con definir $X \sim Geo(p)$ donde p es la probabilidad de aceptación, en este caso $\frac{\pi}{4}$. Finalmente $E[X] = \frac{1-p}{p} = 3\pi$.

Problema 6

Obtengan una muestra de 1,000 números de la siguiente distribución discreta, para $k = 100$.

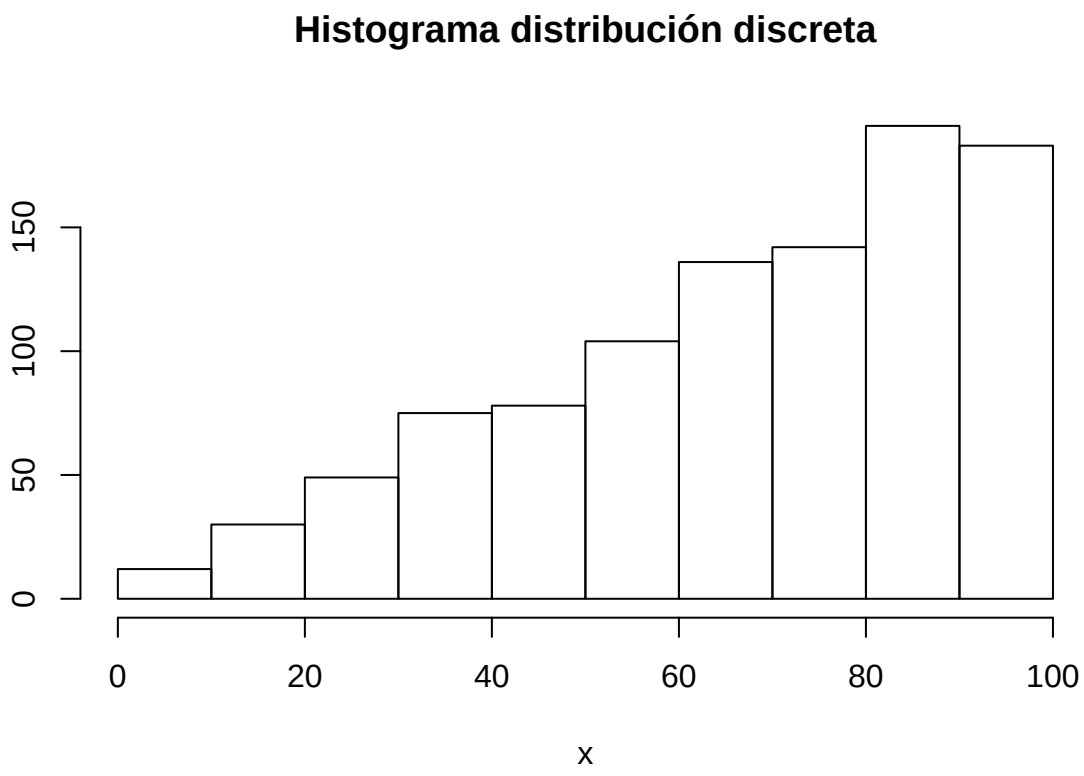
$$p(x) = \frac{2x}{k(k+1)}; x = 1, 2, ...k$$

Solución:

```r
x <-sample(1:100,size=1000,replace=T,prob=c(seq(1:100)*(2/10100)))
x[1:50]
```

```
##  [1] 35 99 61 47 74 81 30 35 84 62 64 61 60 69 63 90 85 67 47 14 28 15 79
## [24] 70 55 60 66 68 99 99 99 50 40 97 88 37 40 75 74 96 68 48 57 91 49 60
## [47] 69 85 65 50
```

```r
hist(x,main = "Histograma distribución discreta",ylab = "")
```



Problema 7

Desarrollen un algoritmo para generar una variable aleatoria binomial, usando la técnica de convolución (Hint: ¿cuál es la relación entre la distribución binomial y Bernoulli?). Generar una muestra de 100,000 números. ¿Qué método es más eficiente, el de convoluciones o la función rbinom en R?

```r
s_Binom<-function(n,t,p){
  esp_muestral<-c(0,1)
  muestra_Binom<-c()
  for (i in 1:n) {
  muestra_Binom[i]<-sum(sample(esp_muestral,t,replace = TRUE,prob = c(p,1-p)))
  }
  return(muestra_Binom)
}
```

```r
ptm <- proc.time()
prueba1<-s_Binom(1,100000,0.4)
```

```
proc.time()-ptm
```

```
##    user  system elapsed
##    0.02    0.00    0.02
```

```
ptm<-proc.time()
prueba2<-rbinom(1,100000,0.4)
proc.time()-ptm
```

```
##    user  system elapsed
##       0       0       0
```

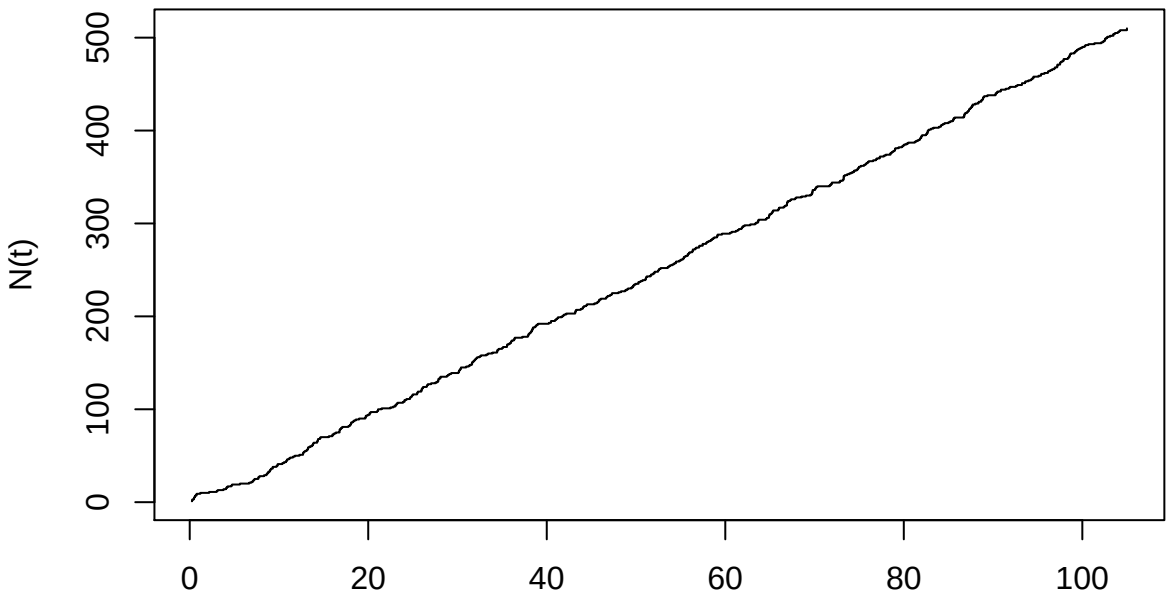
Resulta más eficiente realizar una muestra de 100000 números con la función rbinom.

Problema 8

Para un proceso Poisson no homogéneo con función de intensidad dada por:

$$\lambda(t) = \begin{cases} 5, t \in (1,2],(3,4],(5,6]... \\ 3, t \in (0,1],(2,3],(4,5]... \end{cases}$$

```
lambdat<-function(t){
x<-paste("","{",0,"<=t & t<",1,"}",sep="")
for(i in seq(2,100,2)){
x<-paste(x,paste("","{",i,"<=t & t <",1+i,"}",sep=""),sep="|")
}
return(ifelse(eval(parse(text=0)),3,5))
}
poissonnohomogeneo<-function(lambdat,n,pic=T){
    lambda<-5
    TT<-rexp(n,lambda)
    s<-cumsum(TT)
    u<-runif(n)
    ss<-s[u<=lambdat(s)/lambda]
    Ns<-1:length(ss)
    if(pic==T){
    plot(ss,Ns,type="s",xlab="",ylab="N(t)",main="Proceso Poisson no homogéneo")
    return(list(ss,cuenta=Ns))
}
}
poissonnohomogeneo(lambdat,510)
```




```
## [[1]]
##   [1]   0.2389606   0.2681236   0.3058456   0.4669504   0.4812516
##   [6]   0.5655090   0.5981991   0.7508498   0.7713021   1.1888583
##  [11]   2.1787040   3.0278807   3.1303432   3.7821406   4.0725394
##  [16]   4.1388032   4.2460304   4.6713587   4.7474088   5.6259563
##  [21]   6.6633753   6.9297404   7.0958258   7.1610806   7.2649376
##  [26]   7.6917747   7.6976167   7.8195287   8.3246207   8.5984321
##  [31]   8.6496277   8.7965578   8.8233652   8.9998075   9.0176676
##  [36]   9.1070172   9.2915402   9.3022863   9.7226385   9.8332172
##  [41]   9.8554361  10.3692437  10.5485509  10.7787954  10.8613904
##  [46]  10.9022101  11.0957032  11.2542817  11.5673652  11.7698972
##  [51]  12.2957908  12.6978516  12.7058223  12.7651085  12.8829077
##  [56]  13.0868342  13.1982633  13.2377971  13.2636475  13.4462296
##  [61]  13.6648368  13.7816585  13.8208363  13.8828592  14.2743062
##  [66]  14.3071591  14.3526900  14.3831288  14.6025346  14.6586507
##  [71]  15.5857076  15.9889020  16.0027465  16.1838300  16.3681641
##  [76]  16.7806814  16.7866675  16.7870786  16.8737342  16.9752899
##  [81]  17.0962669  17.7839496  17.9442276  18.0293394  18.0810950
##  [86]  18.1181707  18.2920068  18.4205537  18.6135535  18.9745871
##  [91]  19.6709521  19.7149604  19.7310946  19.9306521  20.1454774
##  [96]  20.1638161  20.2663743  20.9855476  21.1006351  21.1042972
## [101]  21.5004414  22.4959770  22.8075684  23.0494560  23.0679316
## [106]  23.2012327  23.2760044  23.9028273  24.0640182  24.0729267
## [111]  24.2795474  24.6777945  24.7105146  24.8413170  24.9144713
## [116]  24.9975421  25.4785346  25.5031428  25.5614336  25.9433791
## [121]  25.9735846  25.9941756  26.0707756  26.1498217  26.5499918
```

```
## [126]  26.5989195  26.6764300  27.0444792  27.5472734  27.7587678
## [131]  27.8561423  27.8563127  27.8856703  28.0810005  28.0891022
## [136]  28.8133774  28.9418895  29.0941095  29.3002019  30.1094979
## [141]  30.1627289  30.1914914  30.2908209  30.3556723  30.3875863
## [146]  30.9729427  31.2670628  31.4354746  31.6322970  31.6398258
## [151]  31.6660374  31.7976233  31.8955792  31.9854415  32.0918676
## [156]  32.2003935  32.4792676  32.6210049  33.2420261  33.4203362
## [161]  33.9103837  34.4368019  34.4643794  34.4820963  34.6204869
## [166]  34.9990779  35.1039668  35.5422015  35.5619695  35.6008237
## [171]  35.8193195  35.9664362  35.9987075  36.0845013  36.3061803
## [176]  36.3871149  36.4132754  37.2649173  37.8801233  37.9384953
## [181]  37.9608544  38.0616864  38.1827654  38.2740960  38.3155717
## [186]  38.3873980  38.4191748  38.4522211  38.6586837  38.7925233
## [191]  38.8695808  39.0283607  40.2346290  40.4642350  40.4769573
## [196]  40.9181261  41.0270833  41.1642087  41.2550027  41.6702249
## [201]  41.8192335  41.9513095  42.1799958  43.1818483  43.2092071
## [206]  43.2189384  43.2930217  43.8113561  44.0121998  44.0550082
## [211]  44.1682758  44.4120988  44.5077019  45.3417593  45.6128963
## [216]  45.7489417  45.8715842  45.8796672  46.0517267  46.6365998
## [221]  46.7095130  46.8334505  47.0932755  47.2593577  47.3119578
## [226]  48.0388614  48.3142524  48.7764823  48.9373465  49.0995654
## [231]  49.4757411  49.6038572  49.6976785  49.7516052  49.9276431
## [236]  50.2770304  50.2789694  50.5044878  50.7310334  51.0521178
## [241]  51.0891341  51.1257553  51.2002394  51.6311805  51.6743039
## [246]  51.8770114  52.0498921  52.0875008  52.4643384  52.5213240
## [251]  52.5829099  52.7511146  53.5492867  53.6633555  53.8188869
## [256]  54.0761848  54.2569057  54.4412989  54.4931883  54.8472189
## [261]  55.0127846  55.2311016  55.3434266  55.3649429  55.5371280
## [266]  55.7778886  55.7982168  55.9142680  56.0016288  56.3012846
## [271]  56.3530114  56.3637727  56.6177186  56.7414931  57.0665898
## [276]  57.0899720  57.4650916  57.4676502  57.8579698  57.9272806
## [281]  58.1090542  58.3525794  58.4806728  58.6035115  58.7599629
## [286]  59.0774125  59.1227332  59.1391176  59.6320527  60.5366314
## [291]  60.7256588  61.1851906  61.4019575  61.5243230  61.8113935
## [296]  61.8895903  61.9593395  62.1485663  62.7995637  63.3477112
## [301]  63.4770570  63.6850645  63.6975310  63.7240400  64.5663011
## [306]  64.6681214  64.8888292  64.8915918  64.9428342  64.9619751
## [311]  65.1514637  65.2577759  65.2959742  65.3654819  65.9356234
## [316]  65.9851081  66.0381848  66.5499088  66.6697996  66.8669805
## [321]  66.9083172  66.9144073  66.9601278  66.9950879  67.2319894
## [326]  67.3649529  67.8085713  67.9339554  68.5402266  69.0323148
## [331]  69.5467456  69.6856509  69.7445252  69.7505446  69.7871045
## [336]  69.7916795  70.0499466  70.1223663  70.1493257  70.3063638
## [341]  71.6884220  71.7959419  71.9138055  71.9371392  72.8110621
## [346]  72.8833252  73.1725216  73.2566486  73.2595542  73.2652705
## [351]  73.2676444  73.3819327  73.6498269  73.9098383  74.1515062
## [356]  74.3803381  74.4607673  74.7193655  74.8472420  74.9137620
## [361]  74.9765171  75.1873429  75.5430407  75.7054399  75.8058839
## [366]  75.9582715  76.0631773  76.5628392  76.8723632  76.9650881
## [371]  77.2976863  77.3142723  77.7610766  77.9647954  78.5013373
## [376]  78.5240189  78.6820148  78.8233377  78.9780254  79.0183294
## [381]  79.0569179  79.4194987  79.7860210  79.9243984  80.0500247
## [386]  80.3102809  80.4895562  81.2326106  81.4047834  81.6275965
## [391]  81.8050891  81.8434867  81.9529466  81.9599755  82.0480051
```

```
## [396]  82.4820198  82.5697991  82.6644369  82.6650221  82.6874636
## [401]  82.7982385  83.0437080  83.3035888  83.9278720  84.1402059
## [406]  84.1958835  84.3705036  84.5637025  85.0098968  85.2080250
## [411]  85.4879079  85.5610802  85.5698704  85.6806142  86.7064333
## [416]  86.7733232  86.7807484  86.8056421  86.8989535  87.1038219
## [421]  87.1732381  87.2990067  87.3091183  87.4194339  87.5365761
## [426]  87.6089905  87.7003779  87.7127029  88.0073895  88.3480028
## [431]  88.3949571  88.6734383  88.7099977  88.8389536  88.8858732
## [436]  88.9039583  89.0038284  89.3680858  90.2802753  90.2822197
## [441]  90.4105663  90.5064465  90.8510575  90.8745069  91.4211627
## [446]  91.7728790  91.9075862  92.5484769  92.7212224  93.2293239
## [451]  93.2379088  93.5769948  93.6384436  93.9607543  94.2323860
## [456]  94.3573261  94.5164782  94.5184303  95.1037230  95.3653494
## [461]  95.4002079  95.7500189  96.1451934  96.1895378  96.4050386
## [466]  96.6142614  96.8554979  96.9332096  97.1857263  97.2056505
## [471]  97.2127109  97.5103886  97.5105009  97.5855657  97.7741958
## [476]  97.9028038  97.9097851  98.3671247  98.4628519  98.5438689
## [481]  98.5763222  98.6141117  98.6963576  99.0999585  99.1829376
## [486]  99.3193133  99.4154890  99.5782326  99.7912871  99.9976404
## [491] 100.2602470 100.3492881 100.6786659 101.3514636 102.1270078
## [496] 102.3245904 102.4747233 102.5479537 102.5553589 102.7052451
## [501] 102.8145081 103.1215935 103.4794014 103.4837287 103.6685413
## [506] 103.9714118 104.1064122 104.2183721 104.9940341 104.9986306
##
## $cuenta
##   [1]   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17
##  [18]  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34
##  [35]  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51
##  [52]  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68
##  [69]  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85
##  [86]  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100 101 102
## [103] 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119
## [120] 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136
## [137] 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153
## [154] 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170
## [171] 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187
## [188] 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204
## [205] 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221
## [222] 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238
## [239] 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255
## [256] 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272
## [273] 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289
## [290] 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306
## [307] 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323
## [324] 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340
## [341] 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357
## [358] 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374
## [375] 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391
## [392] 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408
## [409] 409 410 411 412 413 414 415 416 417 418 419 420 421 422 423 424 425
## [426] 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442
## [443] 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459
## [460] 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476
## [477] 477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493
```

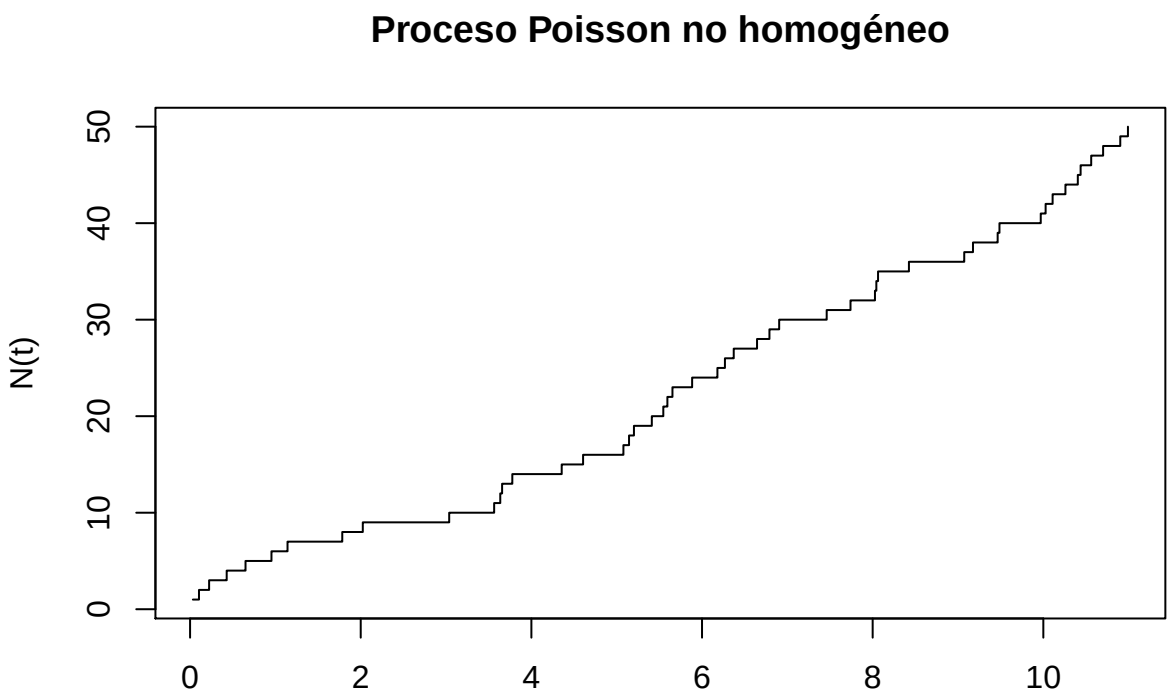
```
## [494] 494 495 496 497 498 499 500 501 502 503 504 505 506 507 508 509 510
```

Problema 9

Simular un proceso Poisson no homogéneo con función de intensidad dada por $\lambda(t) = sin(t)$.

Solución:

```
lambdat2<-function(t){sin(t)}

poissonnohomogeneo2<-function(lambdat,n,pic=T){
    lambda<-1
    TT<-rexp(n,lambda)
    s<-cumsum(TT)
    u<-runif(n)
    ss<-s[u<=lambdat(s)/lambda]
    Ns<-1:length(ss)
    if(pic==T){
    plot(ss,Ns,type="s",xlab="",ylab="N(t)",main="Proceso Poisson no homogéneo")
    return(list(ss,cuenta=Ns))
}
}
poissonnohomogeneo(lambdat,50)
```




```
## [[1]]
##  [1]  0.03243882  0.10444436  0.22303450  0.42929799  0.64969382
```

```
##  [6]  0.95403750  1.14215802  1.78379619  2.02441428  3.03687648
## [11]  3.56381751  3.63528741  3.65694913  3.77661931  4.35557834
## [16]  4.60605535  5.07880427  5.14463507  5.20202031  5.41146719
## [21]  5.54623338  5.59415453  5.65326567  5.88403941  6.18034431
## [26]  6.26834619  6.37169850  6.64466353  6.79022593  6.90404294
## [31]  7.46101554  7.74029751  8.02678715  8.04313239  8.06294336
## [36]  8.42479753  9.07309330  9.17577032  9.46477508  9.48599162
## [41]  9.96964608 10.02698329 10.10874751 10.25923211 10.40417715
## [46] 10.43701876 10.56100235 10.70089818 10.90159455 10.99183860
##
## $cuenta
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
## [24] 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46
## [47] 47 48 49 50
```

Problema 10

Una compañía de seguros tiene 1000 asegurados, cada uno de los cuales presentará de manera independiente una reclamación en el siguiente mes con probabilidad $p = 0.09245$. Suponiendo que las cantidades de los reclamos hechos son variables aleatorias normales con media 7000 y desviación estándar 5000, hagan simulación para estimar la probabilidad de que la suma de los reclamos exceda $500,000.

Solución:

```
comp_seguros<-function(n_as,p,m,de){
  n_rec<-sum(rbinom(n,1,p))
  montos_rec<-rnorm(n_rec,m,de)
  tot_rec<-sum(montos_rec)
}

registro<-replicate(10000,comp_seguros(1000,0.09245,7000,5000))
p_exceder<-length(subset(registro,registro>500000))/10000
p_exceder
```

```
## [1] 0.9748
```

Problema 11

Escribir una función para generar una mezcla de una distribución normal multivariada con dos componentes con medias μ_1, μ_2 y matrices de covarianzas S_1, S_2 respectivamente. Con el programa, generar una muestra de tamaño $n = 1000$ observaciones de una mezcla 50% de una normal 4-dimensional con $\mu_1 = (0, 0, 0, 0)$, $\mu_2 = (2, 3, 4, 5)$, y matrices de covarianzas $S_1 = S_2 = I_4$. Obtener los histogramas de las 4 distribuciones marginales

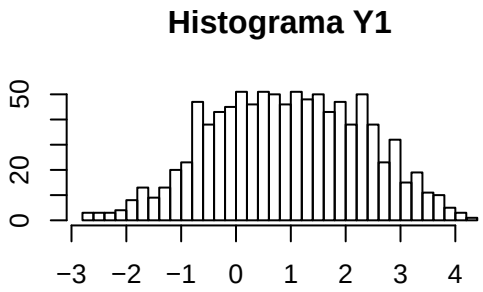
Solución:

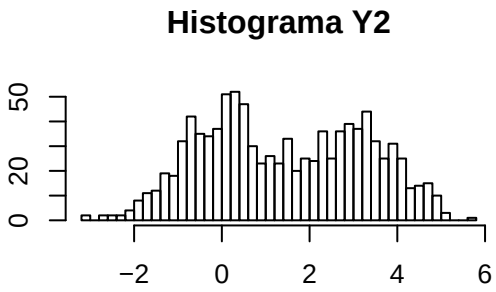
```
r_normal_multi <-function(n,mu,Sigma){
d <-length(mu)
S <-svd(Sigma)
Q <- S$u %*%diag(sqrt(S$d)) %*%t(S$v)
Z <-matrix(rnorm(n*d),nrow=n, ncol=d)
X <- Z %*% Q +matrix(mu,n,d,byrow=T)
X
```

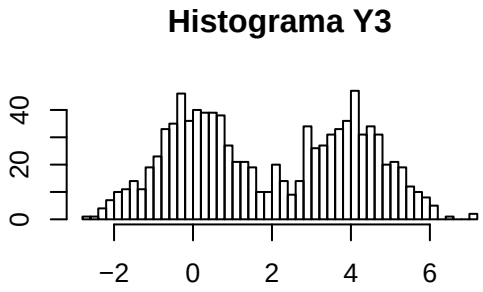
```
}

Sigma <-matrix(c(1, 0, 0,0, 0, 1, 0, 0, 0,0,1,0,0,0,0,1),byrow=T,nrow=4)
n<-1000

Y1<-r_normal_multi(n,c(0,0,0,0),Sigma = Sigma)
Y2<-r_normal_multi(n,c(2,3,4,5),Sigma = Sigma)
u <-runif(n)
k <-as.integer(u > 0.5)
Y <- k*Y1 + (1-k)*Y2

head(Y)
```

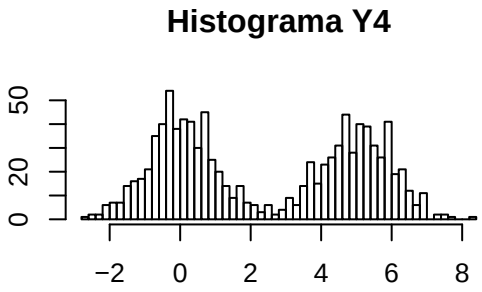
```
##            [,1]      [,2]       [,3]       [,4]
## [1,]  0.1177318 0.5337524 -1.7563726 -0.3487163
## [2,] -1.5234286 0.8580362 -0.8447488 -1.5608975
## [3,]  2.2898215 2.3339014  3.8243543  6.7080842
## [4,]  3.5170306 2.9112414  4.0137777  4.6279267
## [5,] -0.9886053 1.9004404  0.1412666  0.9310865
## [6,]  2.7165188 0.5727434  3.6752946  5.8403874
```

```
par(mfrow=c(2,2))

hist(Y[,1],xlab = "",ylab = "",main = "Histograma Y1",breaks = 50)
hist(Y[,2],xlab = "",ylab = "",main = "Histograma Y2",breaks = 50)
hist(Y[,3],xlab = "",ylab = "",main = "Histograma Y3",breaks = 50)
hist(Y[,4],xlab = "",ylab = "",main = "Histograma Y4",breaks = 50)
```









Problema 12

Distribución de Wishart. Suponer que $M = X^T X$, donde X es una matrix de n x d de una muestra aleatoria de una distribución $N_d(\mu, \Sigma)$. Entonces M tiene una distribución Wishart con matriz de escala Σ y n grados de libertad, y se denota $W \sim W_d(\Sigma, n)$. Cuando $d = 1$, los elementos de X son una muestra aleatoria de una $N(\mu, \sigma^2)$, por lo que $W_1(\sigma^2, n) \sim \sigma^2 \chi 2$. Una forma de generar observaciones de una distribución Wishart, es generar muestras de multivariadas normales y calcular la matriz producto XX^T. Programar este método. Noten que este método es muy costoso porque se tienen que generar nd valores aleatorios normales para determinar las $d(d+1)/2$ diferentes entradas de M.

```
Wishart_1<-function(n,mu,s){
  X<-r_normal_multi(n,mu,s)
  B<-X%*%t(X)
  return(B)

}

Wishart_ma<-function(tm,n,mu,s){
  replicate(tm,Wishart_1(n,mu,s))
}

#Ejemplo
ptm<-proc.time()
Wishart_ma(4,4,c(1,1),matrix(c(1, 0, 0,1),byrow=T,nrow=2))
```

```
## , , 1
##
##           [,1]      [,2]      [,3]      [,4]
## [1,]  1.935175 -1.833866 -3.394977 -3.364612
## [2,] -1.833866  1.819280  3.078101  3.279192
## [3,] -3.394977  3.078101  6.193778  5.747670
## [4,] -3.364612  3.279192  5.747670  5.951004
##
## , , 2
##
##           [,1]       [,2]      [,3]       [,4]
## [1,] 6.3998641 0.76207119 3.9568380 5.26160348
## [2,] 0.7620712 0.14308507 0.1297582 0.09962399
## [3,] 3.9568380 0.12975819 4.6733181 6.68999725
## [4,] 5.2616035 0.09962399 6.6899972 9.63011857
##
## , , 3
##
##            [,1]      [,2]      [,3]      [,4]
## [1,] 0.04122349 0.1146116 0.3977098 0.2495497
## [2,] 0.11461157 0.5338593 1.1523722 0.9270937
## [3,] 0.39770975 1.1523722 3.8470721 2.4581248
## [4,] 0.24954971 0.9270937 2.4581248 1.7635432
##
## , , 4
##
##           [,1]      [,2]      [,3]      [,4]
## [1,] 2.4592158 0.1412787 4.0174060 4.1564665
## [2,] 0.1412787 0.7307771 0.3541981 0.7398995
## [3,] 4.0174060 0.3541981 6.5839578 6.8756279
## [4,] 4.1564665 0.7398995 6.8756279 7.3725801
```

```
proc.time()-ptm
```

```
##    user  system elapsed
##       0       0       0
```

Un método más eficiente se basa en la descomposición de Bartlett: sea $T = (T_{ij})$ una matriz triangular inferior de d×d con entradas independientes que satisfacen: $T_{ij} \sim N(0,1)$ independientes para $i > j$, $T_{ii} \sim \sqrt{\chi^2_{n-i+1}}, i = 1, .., d$. Entonces la matrix $A = TT^{'}$ tiene una distribución $W_d(I_d, n)$. Para generar variables $W_d(\Sigma, n)$, obtener la descomposición de Cholesky $\Sigma = LL^{'}$, donde L es triangular inferior. Entonces $LAL^{'} \sim W_d(\Sigma, n)$.

Problema 13

Las ocurrencias de huracanes que tocan tierra durante el fenómeno meteorológico "el Niño" se modelan como un proceso Poisson (ver Bove et al (1998)). Los autores aseguran que "durante un año del Niño, la probabilidad de dos o más huracanes haciendo contacto con tierra en los Estados Unidos es 28 %". Encontrar la tasa del proceso Poisson.

Solución:

Sea N el número de huracánes que tocan tierra en Estados Unidos, de manera que N tiene una distribución $Po(\lambda)$. Se sabe que $Pr\{N \geq 2\} = 0.28$, entonces

$$\begin{aligned} 0.28 &= Pr\{N \geq 2\} \\ &= 1 - Pr\{N < 2\} \\ &= 1 - Pr(N = 0) - Pr(N = 1) \\ &= 1 - \frac{\lambda^0 e^{-\lambda}}{0!} - \frac{\lambda^1 e^{-\lambda}}{1!}. \end{aligned}$$

Resolviendo para λ se tiene que $\lambda = 1.042284919$.

■

Problema 14

Comenzando a medio día, los comensales llegan a un restaurante de acuerdo a un proceso de Poisson a una tasa de $\lambda = 5$ clientes por minuto. El tiempo que cada cliente pasa comiendo en el restaurante tiene una distribución exponencial con media de $\frac{1}{\beta} = 40$ minutos y es independiente del tiempo de los otros clientes e independiente de los tiempos de arribo de los clientes. Encuentra la distribución así como la media y varianza, del número de comensales en el restaurante a las 2:00pm. Simular el restaurante para verificar los resultados obtenidos.

Solución:

Sea R_t el numero de clientes en el restaurante en el instante t y A_t el numero de clientes que han llegado hasta el tiempo t de manera que A_t es el proceso de Poisson con $\lambda = 5$ (A_t es el proceso de llegada de los clientes). Por el teorema de probabilidad total se tiene que la función de masa de probabilidad de de R_t está dada por

$$\begin{aligned} Pr(R_t = k) &= \sum_{n=0}^{\infty} Pr(R_t = k | A_t = n) Pr(A_t = n) \\ &= \sum_{n=0}^{k-1} Pr(R_t = k | A_t = n) Pr(A_t = n) + \sum_{n=k}^{\infty} Pr(R_t = k | A_t = n) Pr(A_t = n) \\ &= 0 + \sum_{n=k}^{\infty} Pr(R_t = k | A_t = n) Pr(A_t = n) \\ &= \sum_{n=k}^{\infty} P(R_t = k | A_t = n) \frac{e^{-\lambda t}(\lambda t)^n}{n!}. \end{aligned}$$

Suponiendo que al tiempo t se ha registrado la llegada de n clientes ($A_t = n$), sean S_i y L_i son los tiempos de arribo y de estancia del i-ésimo cliente respectivamente, para $i = 1, 2, \ldots, n$, de manera que los clientes saldrán del restaurante en los siguientes tiempos de salida:

$$S_1 + L_1, \quad S_2 + L_2, \quad \ldots, \quad S_n + L_n.$$

Se sigue que habrá k clientes en el restaurante al tiempo t si y sólo si k de los timepos de salida son superiores al tiempo t. Entonces

$$
\begin{aligned}
Pr(R_t = k | A_t = n) &= Pr(k \text{ de los tiempos } S_1 + L_1, \ldots, S_n + L_n \text{ exceden } t | A_t = n) \\
&= Pr(k \text{ de } U_{(1)} + L_1, \ldots, U_{(n)} + L_n \text{ exceden } t) \\
&= Pr(k \text{ de } U_1 + L_1, \ldots, U_n + L_n \text{ exceden } t) \\
&= \binom{n}{k} p^k (1-p)^{n-k},
\end{aligned}
$$

en donde

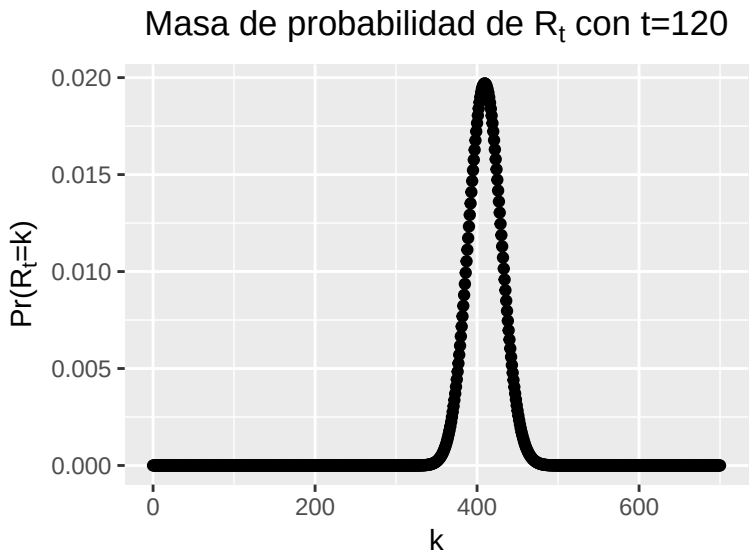
$$
\begin{aligned}
p &= Pr(U_1 + L_1 > t) \\
&= \int_0^t Pr(U_1 + L_1 > t | U_1 = x) Pr(U_1 = x) \, \mathrm{d}x \\
&= \frac{1}{t} \int_0^t Pr(L_1 > t - x) \, \mathrm{d}x \\
&= \frac{1}{t} \int_0^t 1 - e^{-\beta(t-x)} \, \mathrm{d}x \\
&= \frac{1}{t} \left[t - e^{-\beta t} \int_0^t e^{\beta x} \, \mathrm{d}x \right] \\
&= \frac{1}{t} \left[t - \frac{e^{-\beta t}}{\beta} e^{\beta x} \Big|_0^t \right] \\
&= 1 - \frac{1}{t\beta} + \frac{e^{-t\beta}}{t\beta}. \\
&= 1 - \frac{1}{120(\frac{1}{40})} + \frac{e^{-120(\frac{1}{40})}}{120(\frac{1}{40})} \approx 0.6832624
\end{aligned}
$$

Por lo que se tiene que

$$
\begin{aligned}
Pr(R_t = k) &= \sum_{n=0}^{\infty} Pr(R_t = k | A_t = n) Pr(A_t = n) \\
&= \sum_{n=k}^{\infty} \binom{n}{k} p^k (1-p)^{n-k} \frac{e^{-\lambda t} (\lambda t)^n}{n!} I^{(k)}_{\{0,1,\ldots\}} \\
&= \frac{p^k (\lambda t)^k}{k!} \sum_{n=k}^{\infty} \frac{(1-p)^{n-k} (\lambda t)^{n-k}}{(n-k)!} I^{(k)}_{\{0,1,\ldots\}} \\
&= \frac{p^k (\lambda t)^k}{k!} e^{\lambda(1-p)t} I^{(k)}_{\{0,1,\ldots\}} \\
&= \frac{e^{-\lambda p t} (\lambda p t)^k}{k!} I^{(k)}_{\{0,1,\ldots\}}.
\end{aligned}
$$

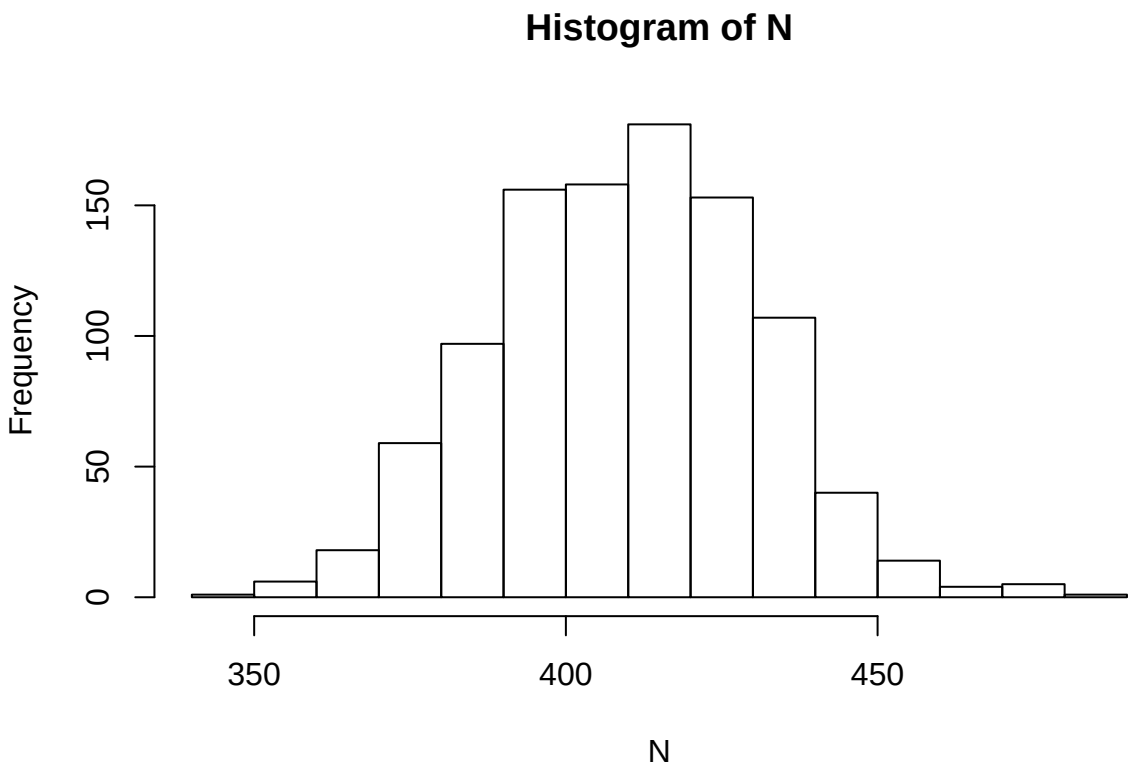
Por lo tanto R_t tiene una distribución Poisson con media λpt:

$$
\begin{aligned}
Pr(R_t = k) &= \frac{e^{-\lambda p t} (\lambda p t)^k}{k!} I^{(k)}_{\{0,1,\ldots\}} \\
&= \frac{e^{-(5)(0.6832624)(120)} [(5)(0.6832624)(120)]^k}{k!} I^{(k)}_{\{0,1,\ldots\}} \\
&= \frac{409.9574137^k e^{-409.9574137}}{k!} I^{(k)}_{\{0,1,\ldots\}}
\end{aligned}
$$



A continuación se muestra un algoritmo que realiza 1000 simulaciones del proceso de llegada y salida de los clientes:

```
N <- c()
for (i in 1:1000) {
  n <- 0
  inter_arrivals <- c()
  arrivals <- c()
  stays <- c()
  time <- 0
  while (time <= t) {
    inter_arrival <- rexp(1, lambda)
    time <- time + inter_arrival
    if (time <= t) {
      #inter_arrivals <- c(inter_arrivals, inter_arrival)
      arrival <- time
      #arrivals <- c(arrivals, arrival)
      stay <- rexp(1, beta)
      #stays <- c(stays, stay)
      if (arrival + stay <= 120) {
        n = n + 1
      }
    }
  }
  N <- c(N, n)
}
hist(N)
```



■

Problema 15