Hoja 5 de problemas y prácticas con R

Estadística Computacional I. Grado en Estadística

Marta Venegas Pardo

Contents

1 Ejercicio 1

Generar una muestra de tamaño 200 de una ley N(0,1). Obtener histogramas con 10 y con 30 intervalos, y superponerles la representación gráfica de la función de densidad de la N(0,1). Explorar la estructura del objeto resultante de hist. Repetir el experimento con la ley Exp(1). Construir una función cuyos argumentos de entrada sean un valor x y un histograma, y devuelva la estimación de la función de densidad en x.

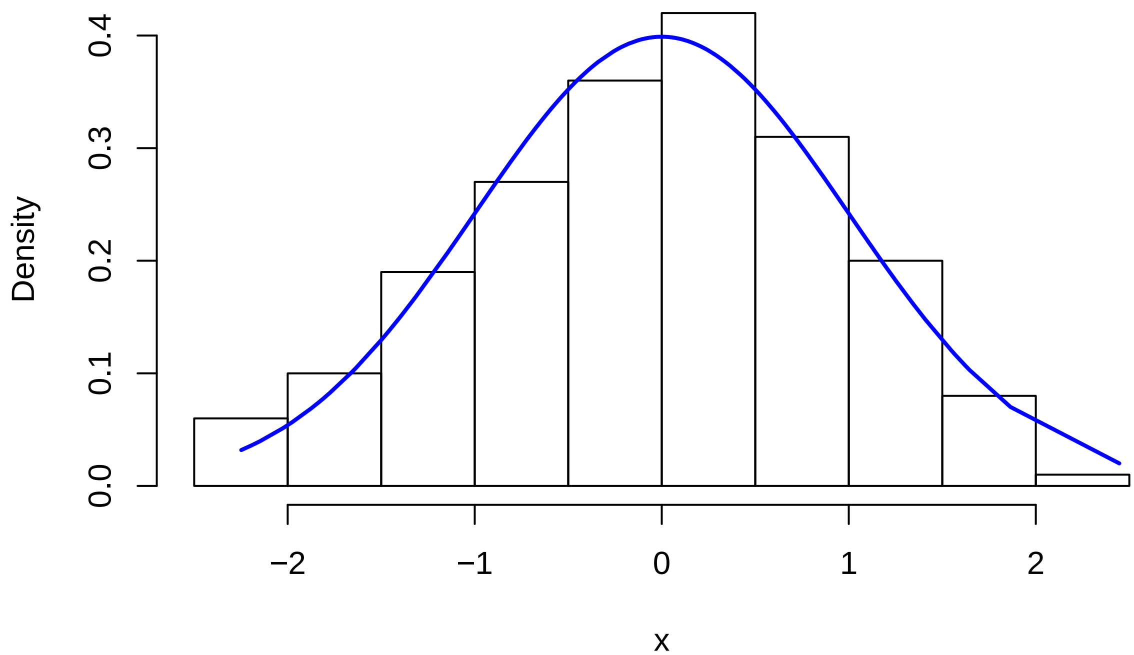
1.1 Distribución normal N(0,1)

```
set.seed(135)
x=rnorm(200)
xord=sort(x)
head(xord)
```

```
## [1] -2.248484 -2.194688 -2.147550 -2.028679 -2.019921 -2.012717
```

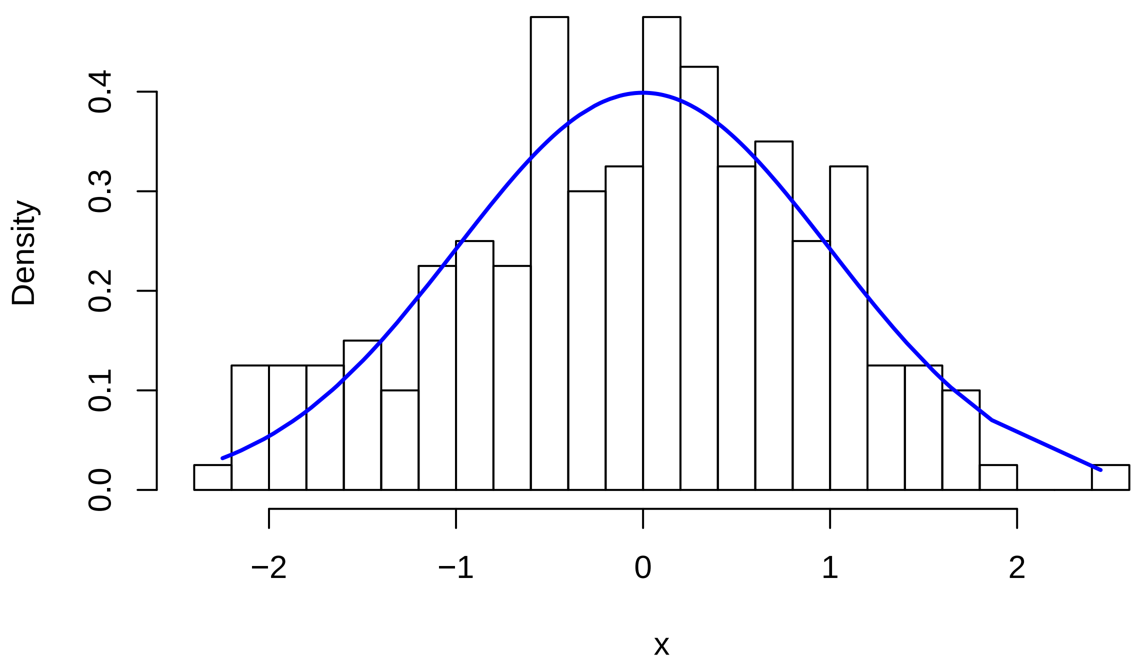
Vamos a construir el histograma

```
hist(x,breaks = 10,probability = TRUE,main = "")
lines(xord,dnorm(xord),lwd=2,col="blue")
```

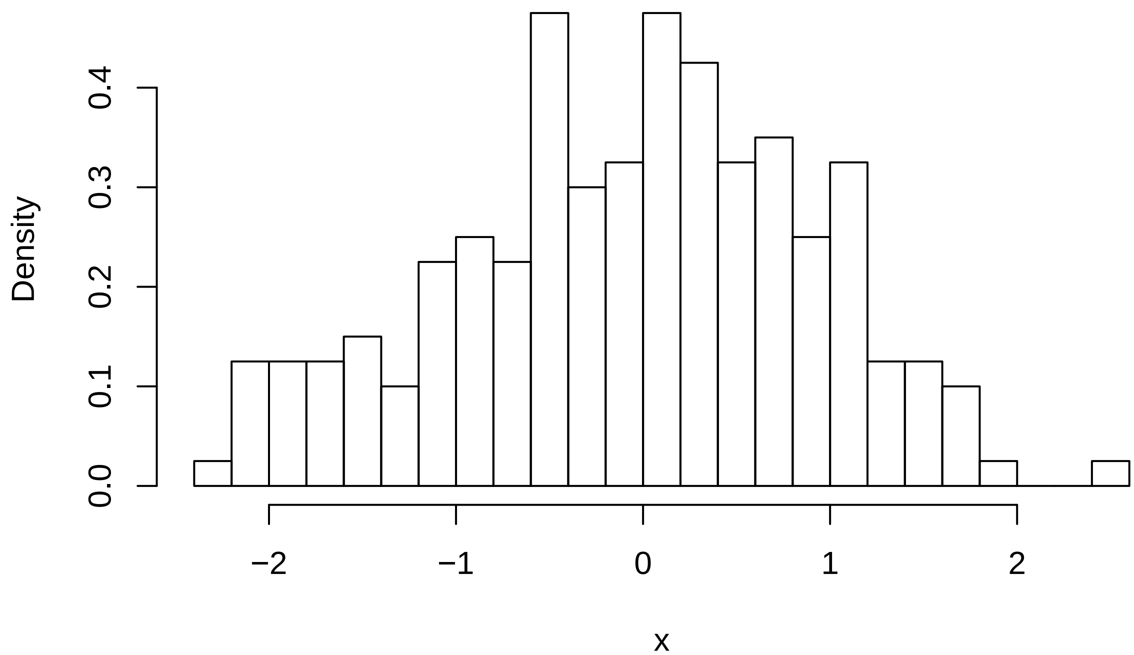


Podemos determinar los intervalos con la función pretty (ver help(pretty))

```
hist(x,breaks = 30,probability = TRUE,main = "")
lines(xord,dnorm(xord),lwd=2,col="blue")
```




```
h=hist(x,breaks = 30,probability = TRUE,main = "")
```




```r
str(h)
```

```
## List of 6
##  $ breaks  : num [1:26] -2.4 -2.2 -2 -1.8 -1.6 -1.4 -1.2 -1 -0.8 -0.6 ...
##  $ counts  : int [1:25] 1 5 5 5 6 4 9 10 9 19 ...
##  $ density : num [1:25] 0.025 0.125 0.125 0.125 0.15 ...
##  $ mids    : num [1:25] -2.3 -2.1 -1.9 -1.7 -1.5 -1.3 -1.1 -0.9 -0.7 -0.5 ...
##  $ xname   : chr "x"
##  $ equidist: logi TRUE
##  - attr(*, "class")= chr "histogram"
```

Obtenemos mucha información:

- Intervalos

```r
h$breaks
```

```
##  [1] -2.4 -2.2 -2.0 -1.8 -1.6 -1.4 -1.2 -1.0 -0.8 -0.6 -0.4 -0.2  0.0  0.2  0.4
## [16]  0.6  0.8  1.0  1.2  1.4  1.6  1.8  2.0  2.2  2.4  2.6
```

- Amplitud de los intevalos

```r
diff(h$breaks) #amplitud constante
```

```
##  [1] 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2
## [20] 0.2 0.2 0.2 0.2 0.2 0.2
```

- Densidad

```r
h$density
```

```
##  [1] 0.025 0.125 0.125 0.125 0.150 0.100 0.225 0.250 0.225 0.475 0.300 0.325
## [13] 0.475 0.425 0.325 0.350 0.250 0.325 0.125 0.125 0.100 0.025 0.000 0.000
## [25] 0.025
```

- Densidad Frecuencia relativa / amplitud

```r
(h$counts/200)/0.2
```

```
##  [1] 0.025 0.125 0.125 0.125 0.150 0.100 0.225 0.250 0.225 0.475 0.300 0.325
## [13] 0.475 0.425 0.325 0.350 0.250 0.325 0.125 0.125 0.100 0.025 0.000 0.000
## [25] 0.025
```

- Área total 1:

```
sum(diff(h$breaks)*h$density)
```

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

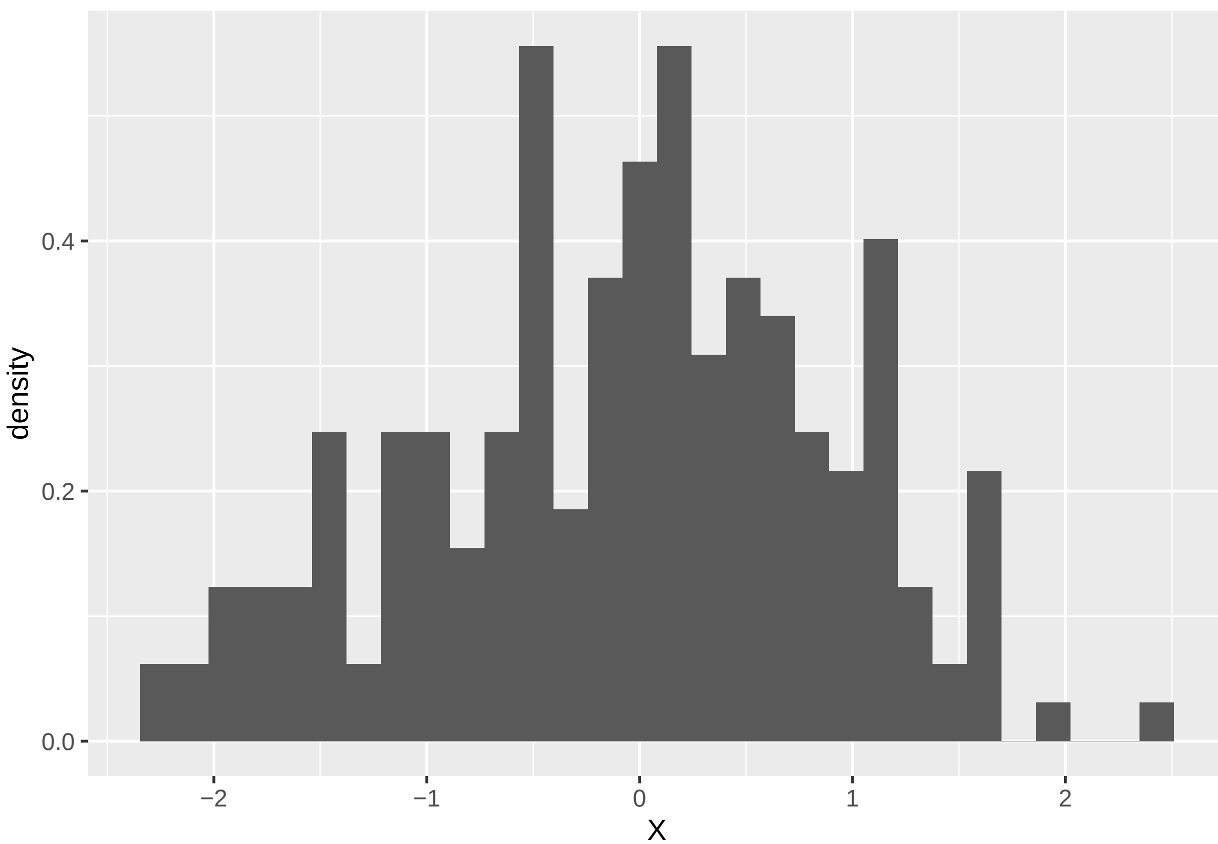
Perfecto.

Vamos a representarlo con la librería ggplot2

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.1.0     v dplyr   1.0.5
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

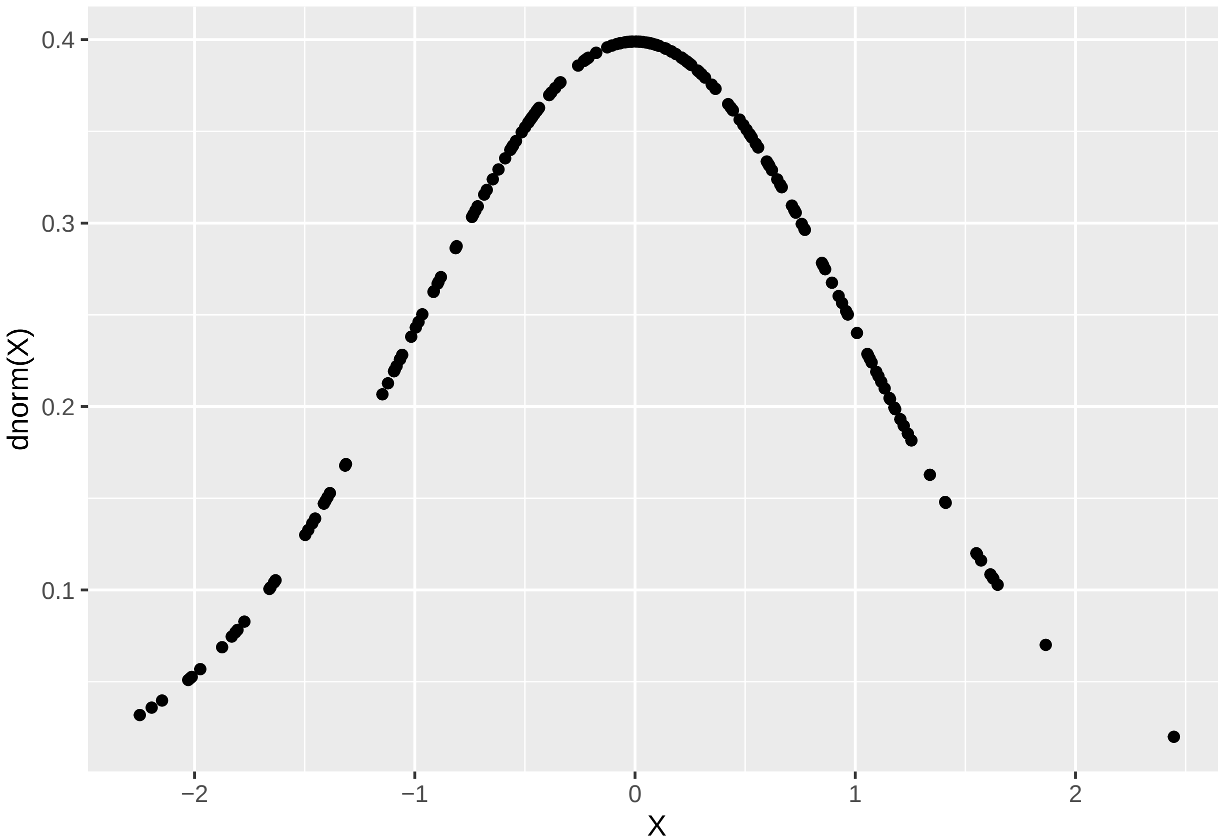
```
data.frame(X=x) %>%
  ggplot(aes(x=X))+
  geom_histogram(aes(y=..density..),bins = 30)->hg
hg
```



```r
head(layer_data(hg),10)
```

```
##              y count          x       xmin       xmax    density    ncount
## 1  0.06176609     2 -2.2666158 -2.3475663 -2.1856652 0.06176609 0.1111111
## 2  0.06176609     2 -2.1047146 -2.1856652 -2.0237641 0.06176609 0.1111111
## 3  0.12353219     4 -1.9428135 -2.0237641 -1.8618629 0.12353219 0.2222222
## 4  0.12353219     4 -1.7809124 -1.8618629 -1.6999618 0.12353219 0.2222222
## 5  0.12353219     4 -1.6190113 -1.6999618 -1.5380607 0.12353219 0.2222222
## 6  0.24706437     8 -1.4571101 -1.5380607 -1.3761596 0.24706437 0.4444444
## 7  0.06176609     2 -1.2952090 -1.3761596 -1.2142584 0.06176609 0.1111111
## 8  0.24706437     8 -1.1333079 -1.2142584 -1.0523573 0.24706437 0.4444444
## 9  0.24706437     8 -0.9714068 -1.0523573 -0.8904562 0.24706437 0.4444444
## 10 0.15441523     5 -0.8095056 -0.8904562 -0.7285551 0.15441523 0.2777778
##     ndensity flipped_aes PANEL group ymin       ymax colour   fill size
## 1  0.1111111       FALSE     1    -1    0 0.06176609     NA grey35  0.5
## 2  0.1111111       FALSE     1    -1    0 0.06176609     NA grey35  0.5
## 3  0.2222222       FALSE     1    -1    0 0.12353219     NA grey35  0.5
## 4  0.2222222       FALSE     1    -1    0 0.12353219     NA grey35  0.5
## 5  0.2222222       FALSE     1    -1    0 0.12353219     NA grey35  0.5
## 6  0.4444444       FALSE     1    -1    0 0.24706437     NA grey35  0.5
## 7  0.1111111       FALSE     1    -1    0 0.06176609     NA grey35  0.5
## 8  0.4444444       FALSE     1    -1    0 0.24706437     NA grey35  0.5
## 9  0.4444444       FALSE     1    -1    0 0.24706437     NA grey35  0.5
## 10 0.2777778       FALSE     1    -1    0 0.15441523     NA grey35  0.5
##    linetype alpha
## 1         1    NA
## 2         1    NA
## 3         1    NA
## 4         1    NA
## 5         1    NA
## 6         1    NA
## 7         1    NA
## 8         1    NA
## 9         1    NA
## 10        1    NA
```

```r
data.frame(X=x) %>%
  ggplot(aes(x=X,dnorm(X)))+
 layer(geom = "point", stat = "identity", position = "identity",
    params = list(na.rm = FALSE)
  )
```



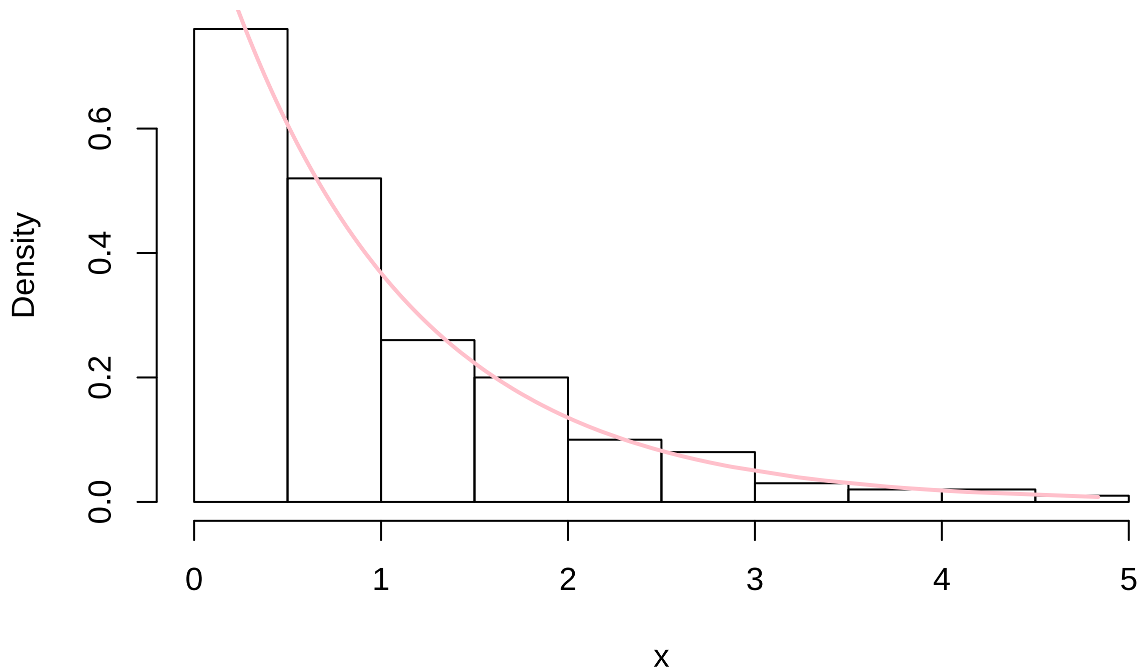
Exponencial 1 Exp(1)

```
x=rexp(200)
xord=sort(x)
head(xord,10)
```

```
##  [1] 0.003879447 0.011029475 0.015492738 0.021789749 0.024934943 0.033050073
##  [7] 0.049659100 0.049868454 0.053093340 0.057174360
```

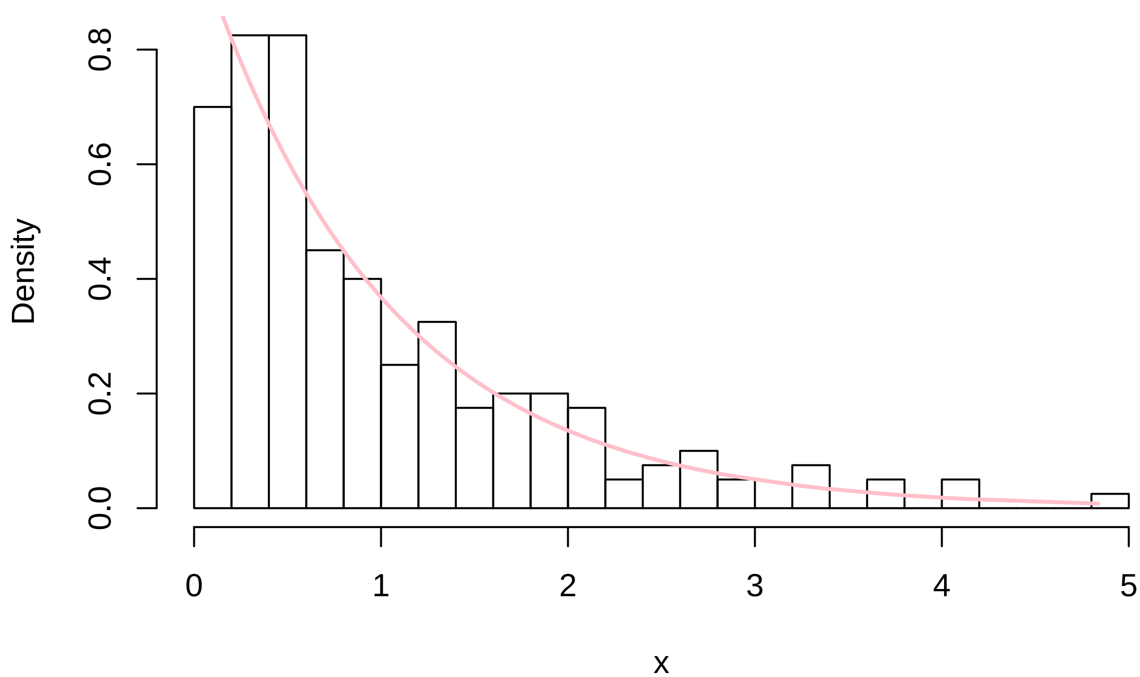
Construímos el histograma

```
hist(x,breaks = 10,probability = TRUE,main = "")
lines(xord,dexp(xord),lwd=2,col="pink")
```

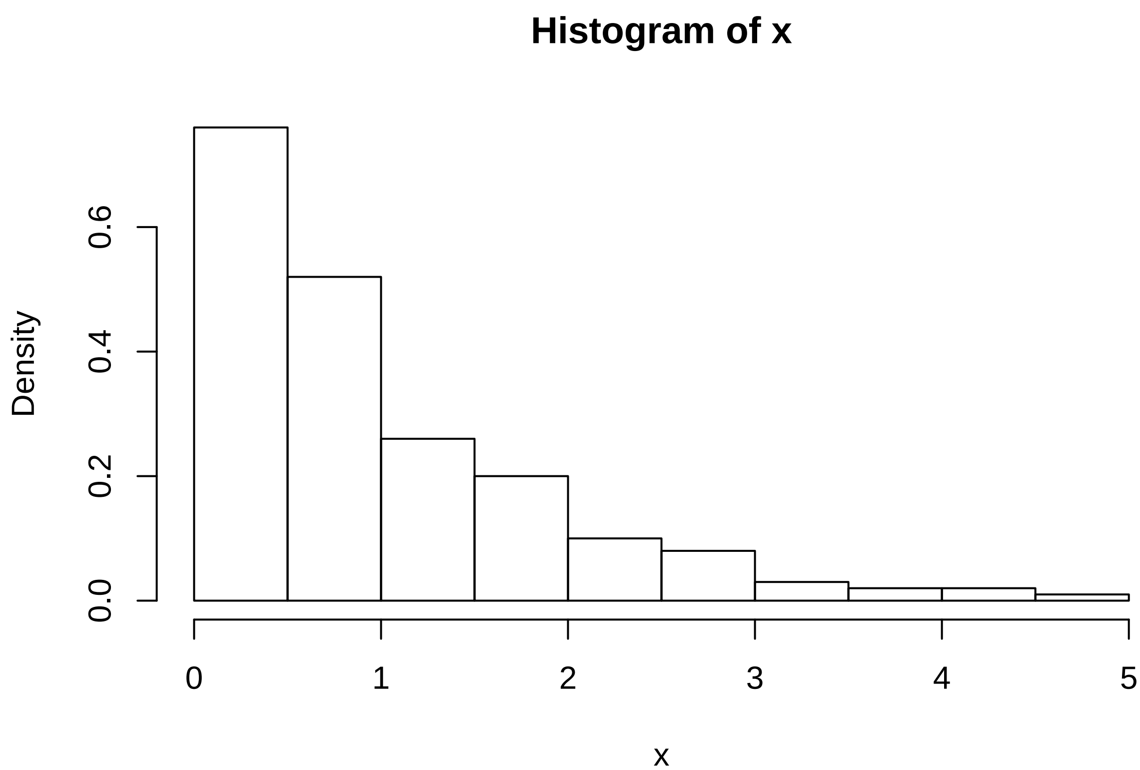


Ahora con 30 intervalos:

```
hist(x,breaks = 30,probability = TRUE,main = "")
lines(xord,dexp(xord),lwd=2,col="pink")
```




```
h=hist(x,breaks=10,freq=FALSE,plot=TRUE)
```



Función para calcular la estimación de f(x) dado un histograma

```
#x=0.8
fg=function(x,h){
  if (x<h$breaks[1] | x> h$breaks[length(h$breaks)] # x está entre el primer y
      # último intervalo (valores)
      ) resul=0
  else{
    interv=which(x<h$breaks)[1] -1
    resul=h$density[interv]
  }
  resul
}
```

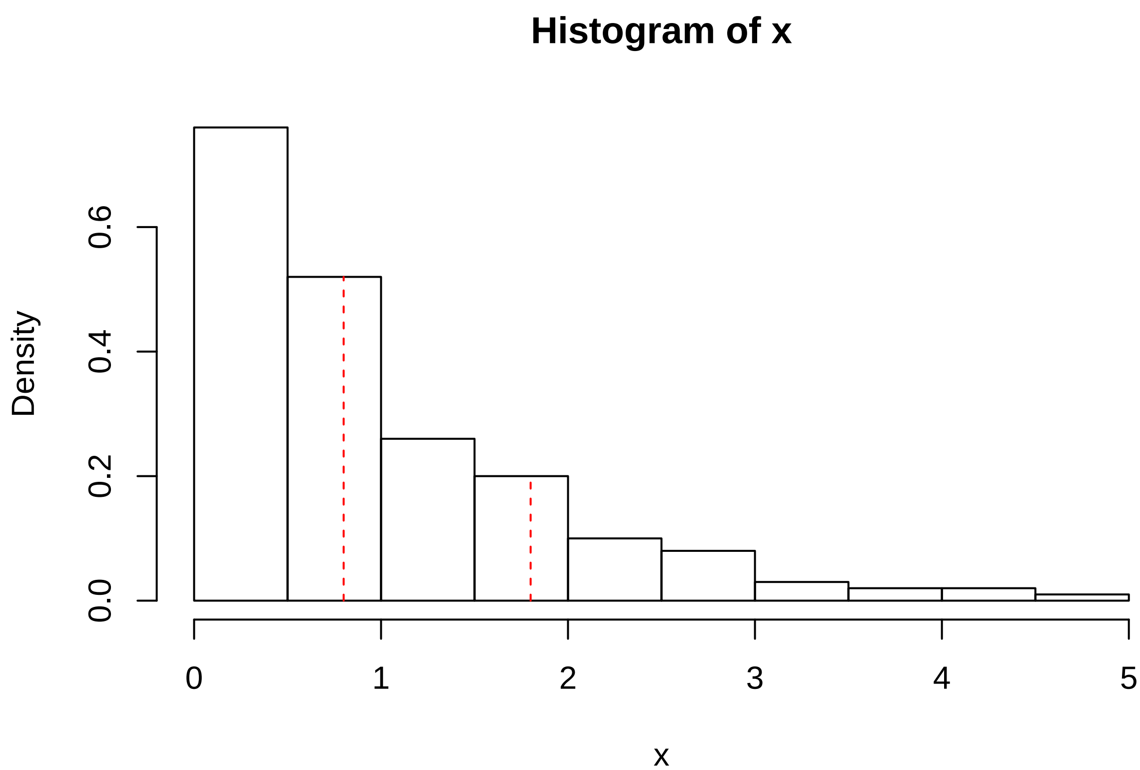
```
fg(-2,h)
```

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

```
fg(20,h)
```

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

```
h=hist(x,breaks=10,freq=FALSE,plot=TRUE)
points(0.8,fg(0.8,h),type="h",lty=2,col="red")
points(1.8,fg(1.8,h),type="h",lty=2,col="red")
```



2 Ejercicio 2

Implementar el método ASH, para ello realizar las siguientes tareas.

2.1 Generar una muestra de tamaño 50 de la ley N(0,1).

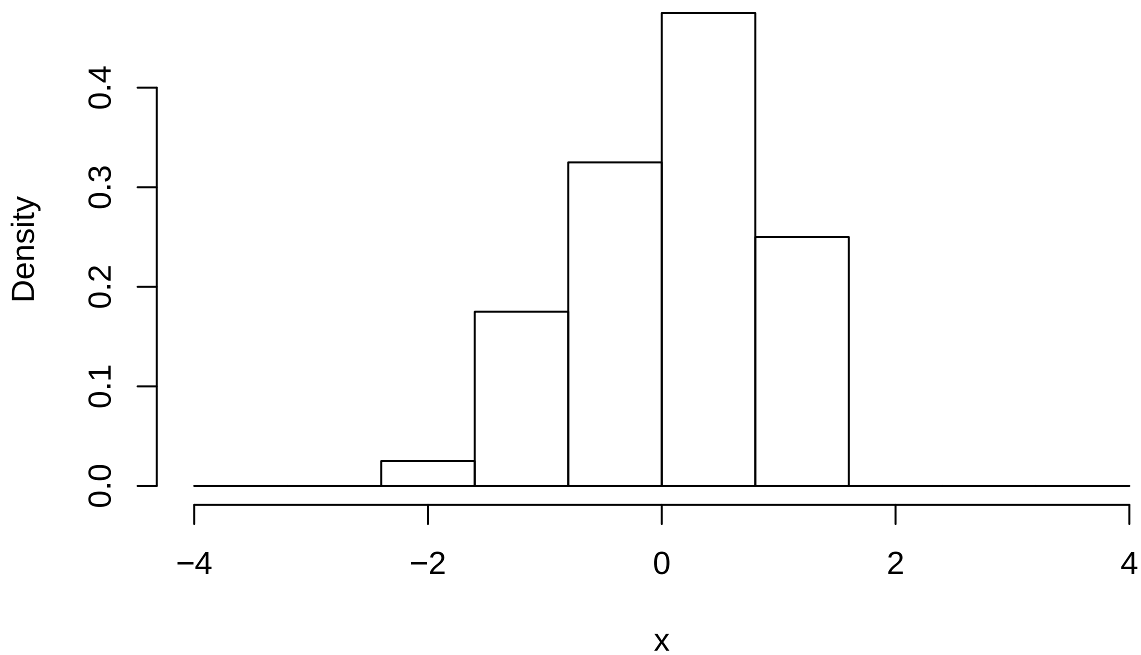
```
set.seed(135)
x=rnorm(50)
```

2.2 Definir 10 intervalos entre -4 y 4. Obtener el histograma con esos intervalos.

```
(breaks=seq(-4,4,length=11))
```

```
##  [1] -4.0 -3.2 -2.4 -1.6 -0.8  0.0  0.8  1.6  2.4  3.2  4.0
```

```
h=hist(x = x,breaks = breaks,main="",freq = FALSE,plot = TRUE)
```




```
str(h)
```

```
## List of 6
##  $ breaks  : num [1:11] -4 -3.2 -2.4 -1.6 -0.8 0 0.8 1.6 2.4 3.2 ...
##  $ counts  : int [1:10] 0 0 1 7 13 19 10 0 0 0
##  $ density : num [1:10] 0 0 0.025 0.175 0.325 0.475 0.25 0 0 0
##  $ mids    : num [1:10] -3.6 -2.8 -2 -1.2 -0.4 0.4 1.2 2 2.8 3.6
##  $ xname   : chr "x"
##  $ equidist: logi TRUE
##  - attr(*, "class")= chr "histogram"
```

2.3 Construir una función R que calcule las estimaciones mediante el método ASH. Aceptará como entradas: una muestra x, un conjunto de puntos donde obtener las estimaciones, unos intervalos iniciales y un número B de histogramas.

Método ASH= "Average Shifted Histogram" La idea es que los B histogramas se basan en el mismo número de intervalos y amplitud h, pero desplazando el extremo inferior del primer intervalo.

```
fg_ash=function(x,puntos,breaks,B){
  histogramas=vector(mode="list",length = B) # Genero lista de histogramas en cada iteración
  amplitud=breaks[2]-breaks[1] # amplitud de intervalos CTE
  for(b in 0:(B-1)) # Desplazamos los breaks
  {
    nbreaks=breaks+b*amplitud/B #número de intervalos
    nh=hist(x,breaks=nbreaks,plot=FALSE)
    histogramas[[b+1]]=nh
}
n=length(puntos)
fgorros=matrix(NA,n,B)
for (i in 1:n)
for (j in 1:B)
  fgorros[i,j]=fg(puntos[i],histogramas[[j]])# para cada punto apligo fg en cada histograma
apply(fgorros,1,mean)# calculo la media de las B estimaciones para cada punto
```

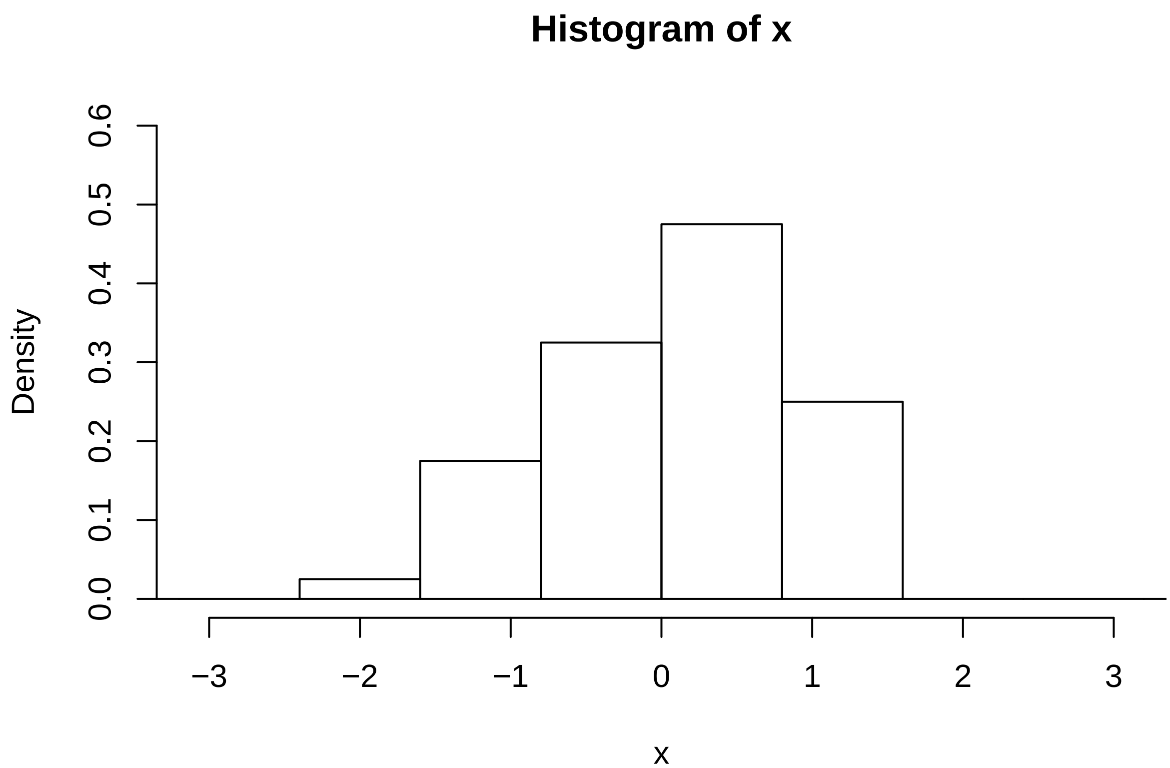
```
}
```

2.4 Aplicar esta función en una secuencia de 100 puntos entre -3 y 3, con varios valores de B, y representar gráficamente las estimaciones.

```
head((puntos=seq(-3,3,length=100)),10)
```

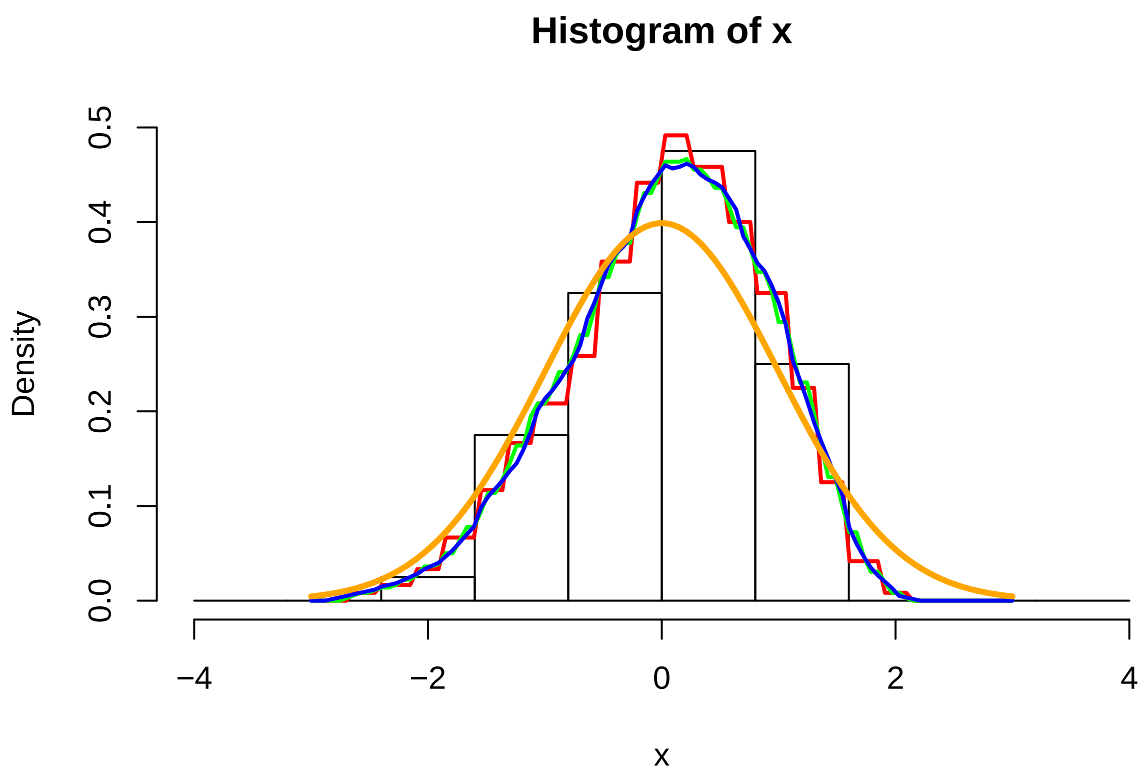
```
##  [1] -3.000000 -2.939394 -2.878788 -2.818182 -2.757576 -2.696970 -2.636364
##  [8] -2.575758 -2.515152 -2.454545
```

```
h=hist(x,breaks = breaks,probability = TRUE,plot = TRUE,ylim = c(0,0.6),xlim=c(-3.1,3.1))
```



Usamos la función

```
h=hist(x,breaks=breaks,prob=TRUE,plot=TRUE, ylim=c(0,0.5))
#ylim dependerá de los datos
colores=c("red","green","blue")
listaB=c(3,9,15)
for (i in 1:length(listaB))
{
estimaciones=fg_ash(x,puntos,breaks,B=listaB[i])
lines(puntos,estimaciones,type="l",col=colores[i],lwd=2)
}
lines(puntos,dnorm(puntos),col="orange",lwd=3)
```



Podría requerir perfeccionamiento, ya que se podría dar el caso de que al desplazar los intervalos queden fuera de algunos valores de la muestra

3 Ejercicio 3

3. Leer el fichero "nhanes.txt" y obtener estimaciones de la función de densidad mediante el histograma y la función ash1 de la librería ash para la variable LDL.

```
# install.packages("ash")
library(ash)
datos<-read.table("datos/nhanes.txt",header=TRUE)
dim(datos)
```

```
## [1] 3026    3
```

```
summary(datos)
```

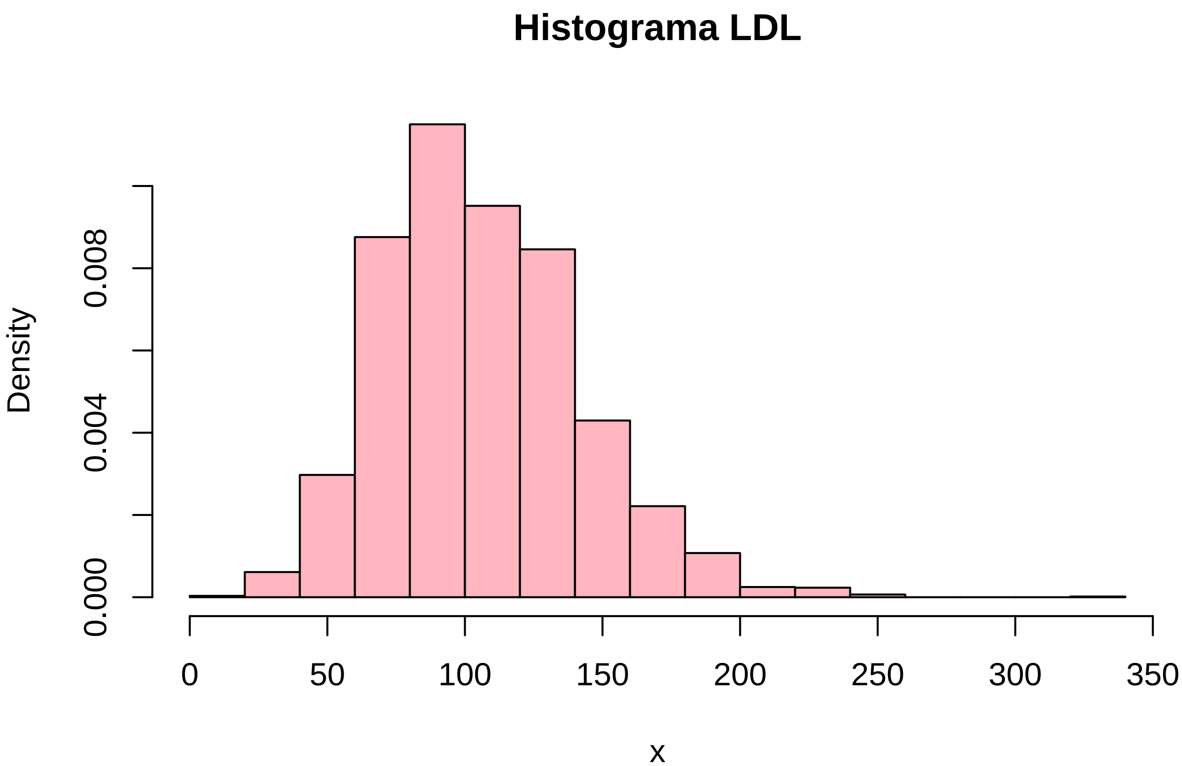
```
##       TRG             LDL             APB
##  Min.   : 19.0   Min.   : 19.0   Min.   : 24.00
##  1st Qu.: 68.0   1st Qu.: 81.0   1st Qu.: 74.00
##  Median : 98.0   Median :103.0   Median : 91.00
##  Mean   :116.9   Mean   :106.8   Mean   : 93.67
##  3rd Qu.:147.0   3rd Qu.:130.0   3rd Qu.:111.00
##  Max.   :399.0   Max.   :328.0   Max.   :220.00
```

Variables:

- TRG: Triglicéridos
- LDL: Colesterol (Malo)
- APB: Alipoproteina B

3.1 Estimaciones de la función de densidad (histograma)

```
x<-datos$LDL
hist(x,prob = TRUE, col="lightpink", main="Histograma LDL")
```



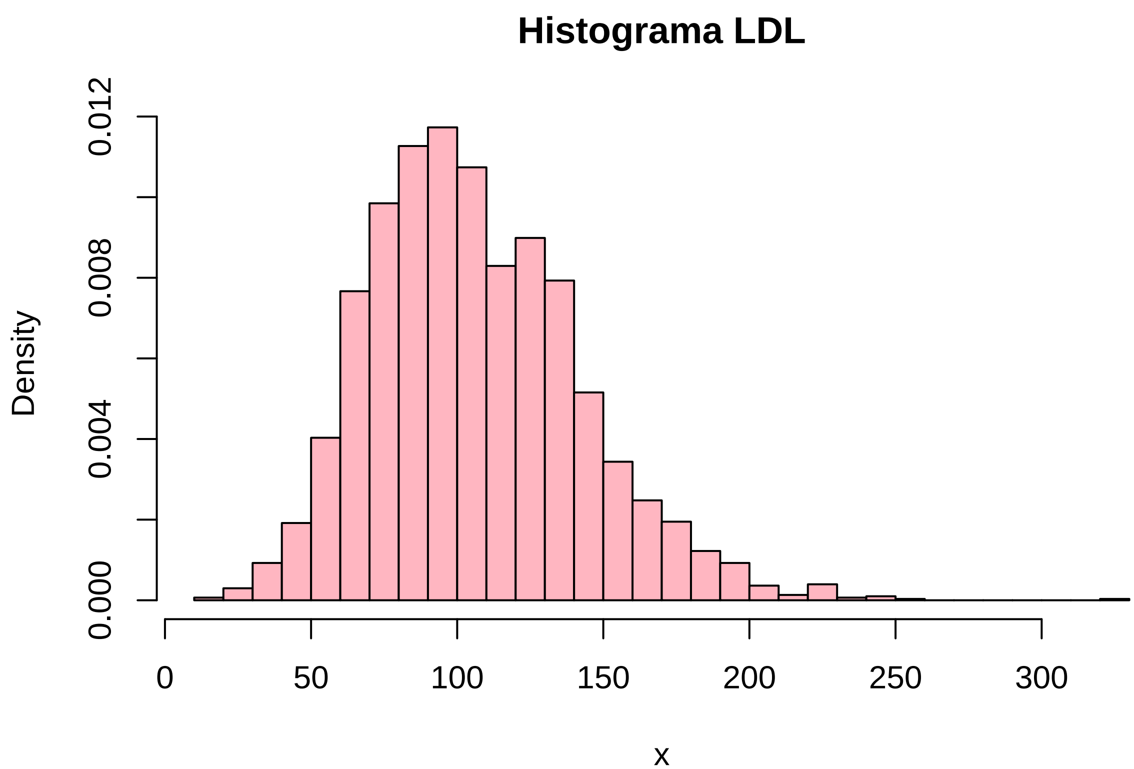
3.1.1 Método STURGES para determinar el número de intervalos

```
round(1+log(length(x)[1],base = 2))
```

```
## [1] 13
```

El histograma anterior ya tenía 13 intervalos.

```
hist(x,prob = TRUE, col="lightpink", main="Histograma LDL",breaks = 28)
```

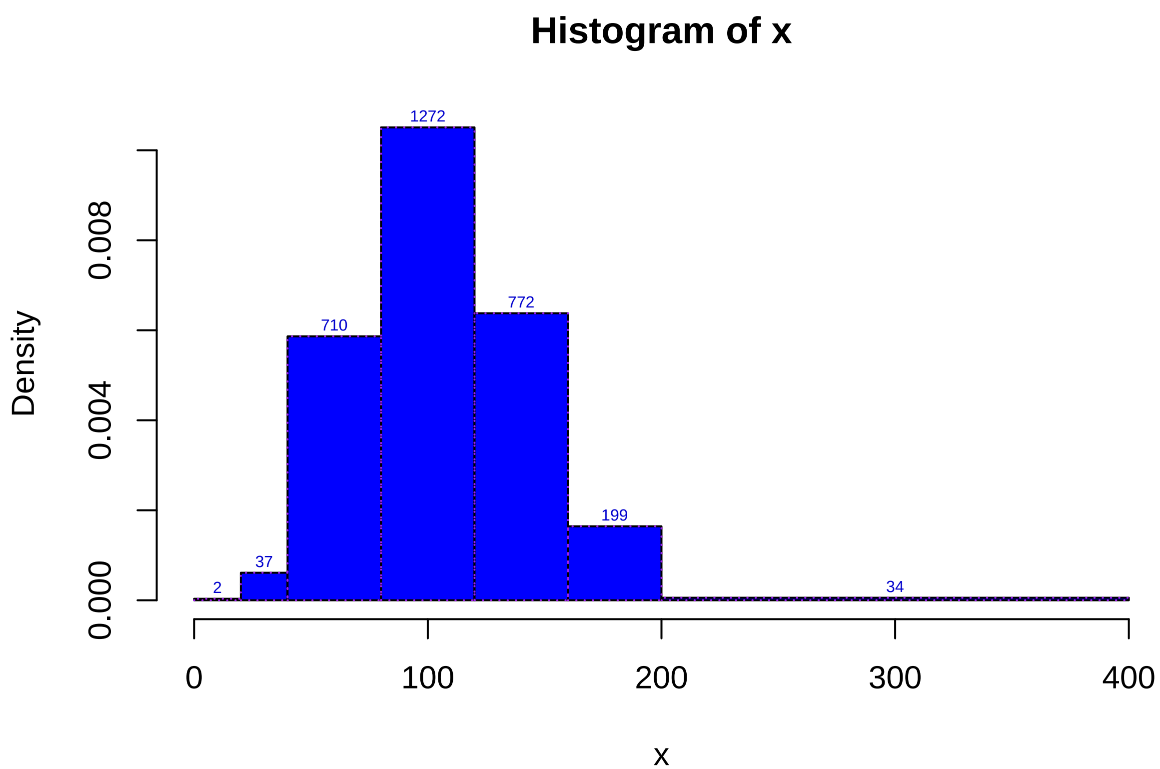


3.1.2 Histograma con intervalos desiguales

```
r <- hist(x, breaks = c(0,20,40,80,120,160,200,400),col = "blue1")

# Vamos a colocar en las posiciones (minds, counts) las frecuendas absoludas (adj: ajuste x e y)

str(r)
```

```
## List of 6
##  $ breaks  : num [1:8] 0 20 40 80 120 160 200 400
##  $ counts  : int [1:7] 2 37 710 1272 772 199 34
##  $ density : num [1:7] 0.000033 0.000611 0.005866 0.010509 0.006378 ...
##  $ mids    : num [1:7] 10 30 60 100 140 180 300
##  $ xname   : chr "x"
##  $ equidist: logi FALSE
##  - attr(*, "class")= chr "histogram"
```

```
text(r$mids,
     r$density,
     r$counts,
     adj = c(.5, -.5),
     col = "blue3",
     cex=0.5)
lines(r, lty = 3, border = "purple")
```



El area total es:

```
sum(r$density * diff(r$breaks))
```

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

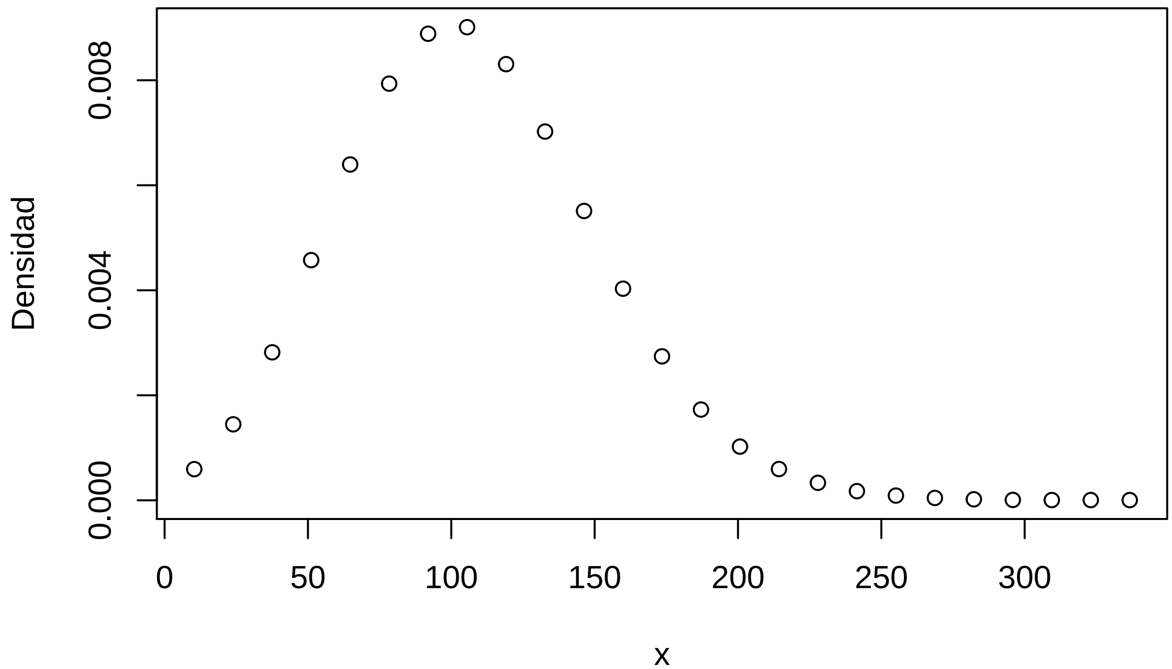
3.2 Función ash1: librería ash para una variable

```
library(ash)
f<-ash1(bin1(x,nbin=25)) # Número de intervalos
```

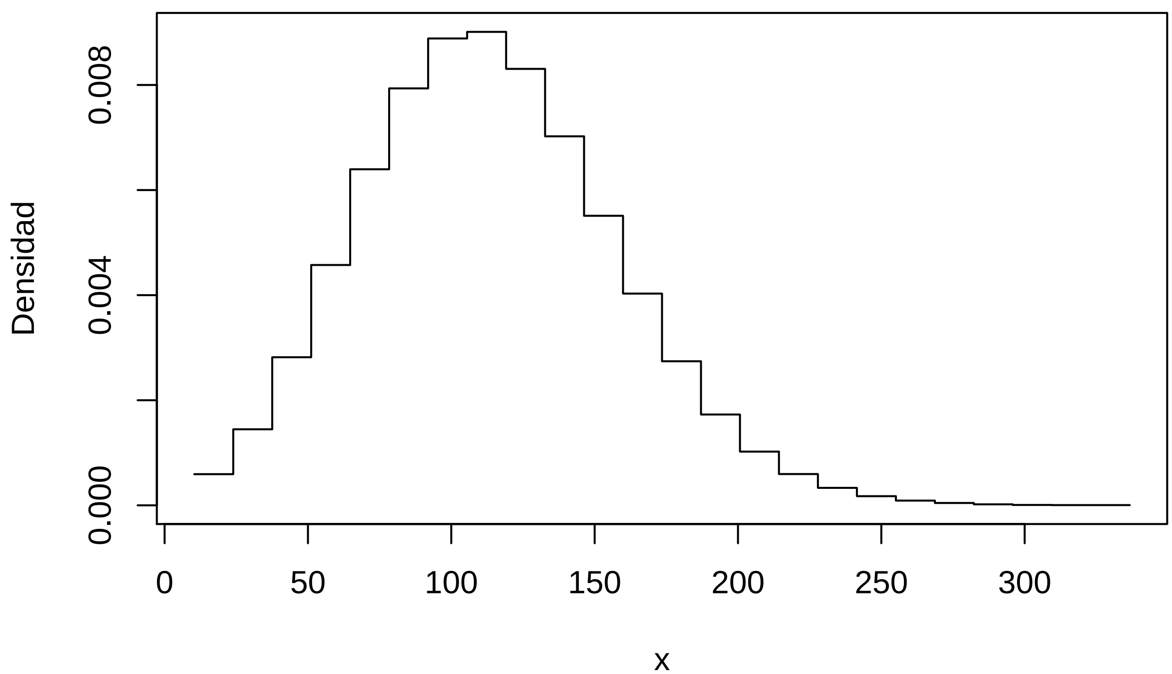
```
## [1] "ash estimate nonzero outside interval ab"
```

Representamos la función de densidad:

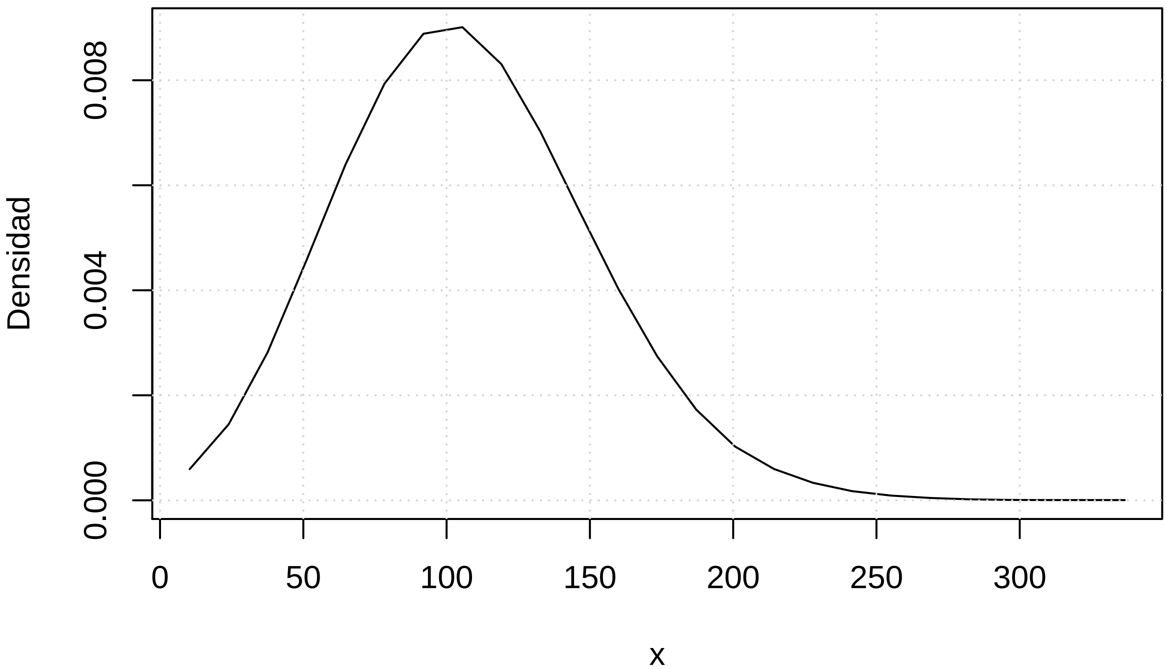
```
plot(f,xlab = "x",ylab = "Densidad")
```




```r
plot( f, type="s",xlab="x",ylab="Densidad" )
```




```r
plot( f, type="l", # Suavización
      xlab="x",ylab="Densidad" ) # Etiquetas
grid()
```




```
str(f)
```

```
## List of 6
##  $ x    : num [1:25] 10.3 23.9 37.5 51.1 64.7 ...
##  $ y    : num [1:25] 0.000593 0.001447 0.002818 0.004574 0.006396 ...
##  $ m    : num 5
##  $ ab   : num [1:2] 3.55 343.45
##  $ kopt: num [1:2] 2 2
##  $ ier : int 1
```

Se tiene:

- m: Por defecto es 5. **En las transparencias es B**.
- kopt: determina la forma de w de transparencias.

4 Ejercicio 4

4.1 Dibujar las funciones núcleo Normal, Epanechnikov, Triangular y Biweight, y comprobar que son funciones de densidad.

4.1.1 Normal

```
normal<- function(u)
{exp(-(u^2)/2)/sqrt(2*pi)}
```

4.1.2 Epanechnikov

```
epanec<- function(u)
{(3/4)*(1-u^2)*(abs(u)<1)}
```

4.1.3 Triangular

```
triangular<- function(u)
{(1-abs(u))*(abs(u)<1)}
```

4.1.4 Biweight

```
biweight<- function(u)
{
(15/16)*((1-u^2)^2)*(abs(u)<1)
}
```

4.1.5 Comprobamos que son veraderas función de densidad

```
integrate(normal, -Inf,Inf)
```

```
## 1 with absolute error < 9.4e-05
```

```
integrate(epanec,-Inf,Inf)
```

```
## 1 with absolute error < 6e-08
```

```
integrate(triangular,-Inf,Inf)
```

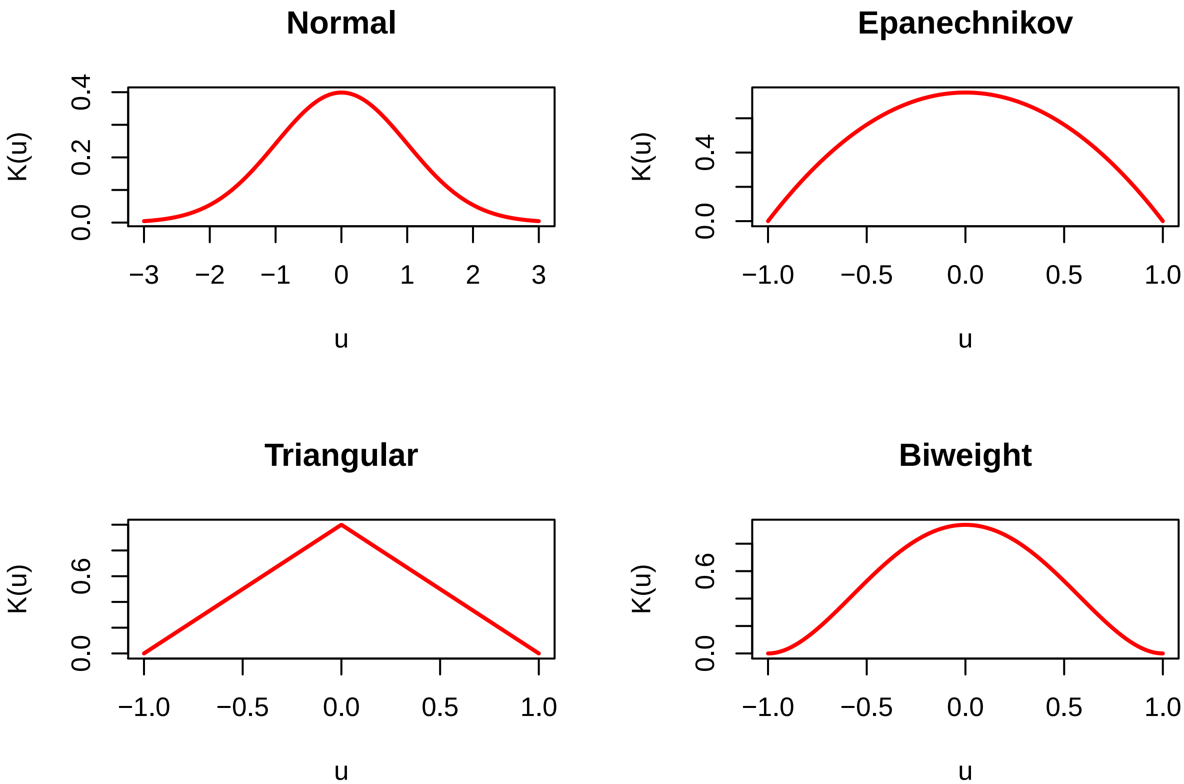
```
## 1 with absolute error < 4.7e-09
```

```
integrate(biweight,-Inf,Inf)
```

```
## 1 with absolute error < 5.2e-06
```

4.2 Dibujar en la misma gráfica las funciones núcleo Normal, Uniforme y Triangular.

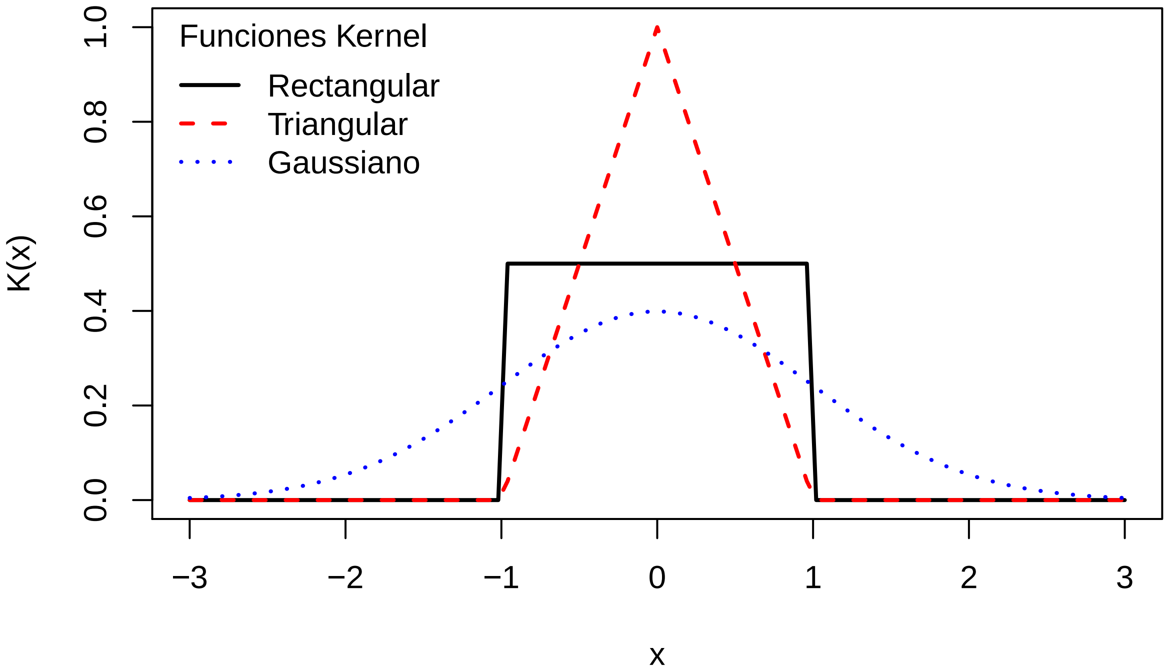
```
par(mfrow=c(2,2))
curve(normal(x),-3,3,1000,lwd=2,col="red",main="Normal",xlab="u",
ylab="K(u)")
curve(epanec(x),-1,1,1000,lwd=2,col="red",main="Epanechnikov",xlab="u",
ylab="K(u)")
curve(triangular(x),-1,1,1000,lwd=2,col="red",main="Triangular",xlab="u",
ylab="K(u)")
curve(biweight(x),-1,1,1000,lwd=2,col="red",main="Biweight",xlab="u",
ylab="K(u)")
```




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

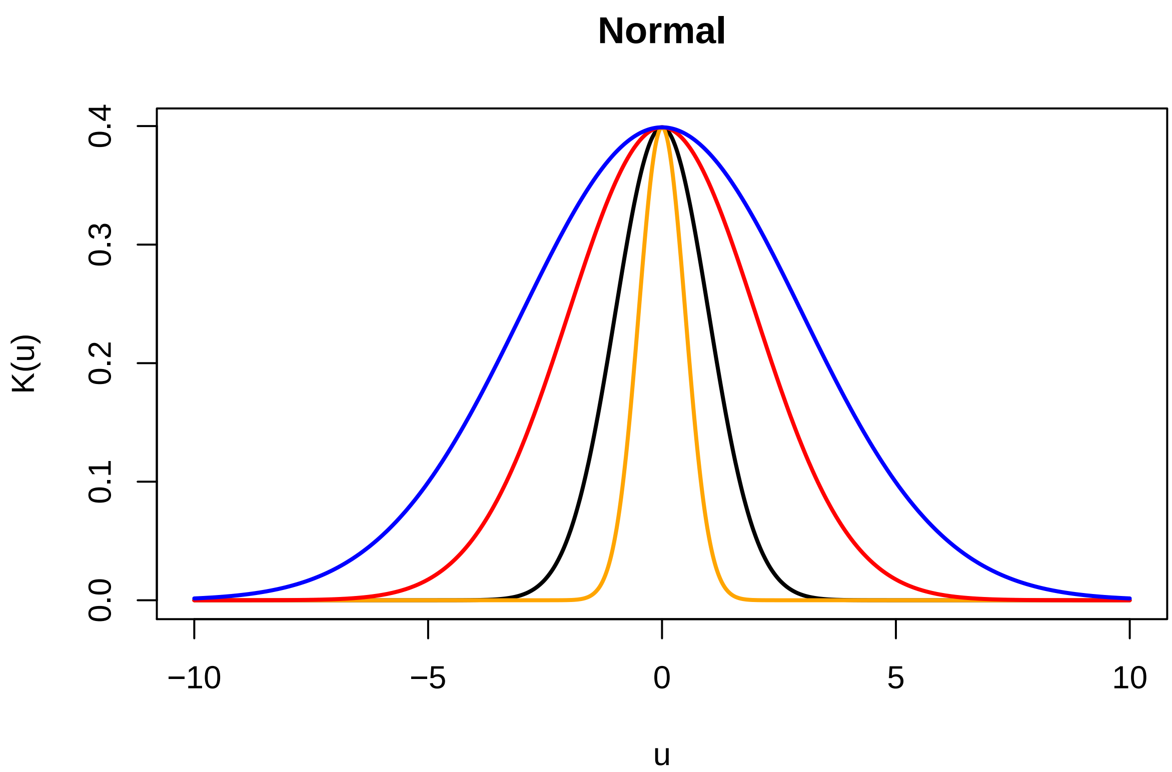
Ahora en la misma gráfica:

```
rec <- function(x) (abs(x) < 1) * 0.5
tri <- function(x) (abs(x) < 1) * (1 - abs(x))
gauss <- function(x) 1/sqrt(2*pi) * exp(-(x^2)/2)
curve(rec(x), -3,3,type = "l", ylim = c(0,1),
lty = 1,ylab = "K(x)",lwd=2)
curve(tri(x), -3,3,lty = 2,lwd=2,col="red",add=TRUE)
curve(gauss(x),-3,3, lty = 3,lwd=2,col="blue",add=TRUE)
legend("topleft",
       legend = c("Rectangular", "Triangular","Gaussiano"),
       lty = 1:3,
       lwd=2,
       title = "Funciones Kernel",
       bty = "n",
       col=c("black","red","blue"))
```



Comorobamos el efecto del parámetro h. Probar co 1, 1/2, 2 y 3.

```
curve(normal(x),-10,10,1000,lwd=2,
col="black",main="Normal",xlab="u",
ylab="K(u)")
curve(normal(2*x),-10,10,1000,lwd=2,
col="orange",add=TRUE)
curve(normal(x/2),-10,10,1000,lwd=2,
col="red",add=TRUE)
curve(normal(x/3),-10,10,1000,lwd=2,
col="blue",add=TRUE)
```



5 Ejercicio 5

5. El fichero "migracionballenas.dat" contiene los tiempos en horas, desde la medianoche del 5 de Abril de 2001, en los que fueron vistas 121 ballenas al pasar por Point Barrow, Alaska, durante la emigración de primavera.

```
datos<-read.table("datos/migracionballenas.dat", header=TRUE)
```

```
str(datos)
```

```
## 'data.frame':    121 obs. of  1 variable:
##  $ Tiempo: num  1007 1009 1025 1046 1057 ...
```

5.1 Dibujar un histograma y superponer estimaciones de la función de densidad con

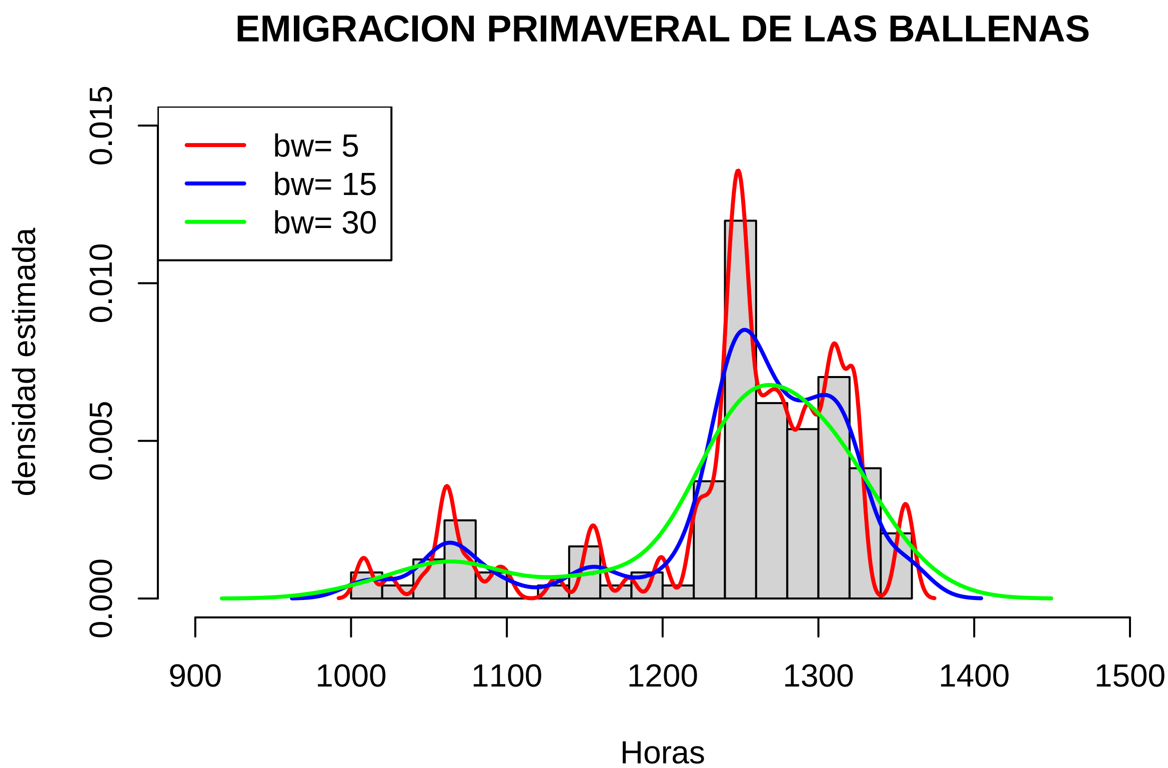
h=5, 15 y 30.

5.1.1 DEPENDENCIA DEL PARAMETRO bw.

bw is the smoothing bandwidth to be used. The kernels are scaled such that this is the standard deviation of the smoothing kernel. (Note this differs from the reference books cited below, and from S-PLUS.)

```
attach(datos)
hist(Tiempo,
     prob=TRUE,
     br=20,
     main="EMIGRACION PRIMAVERAL DE LAS BALLENAS",
     col="lightgray",
```

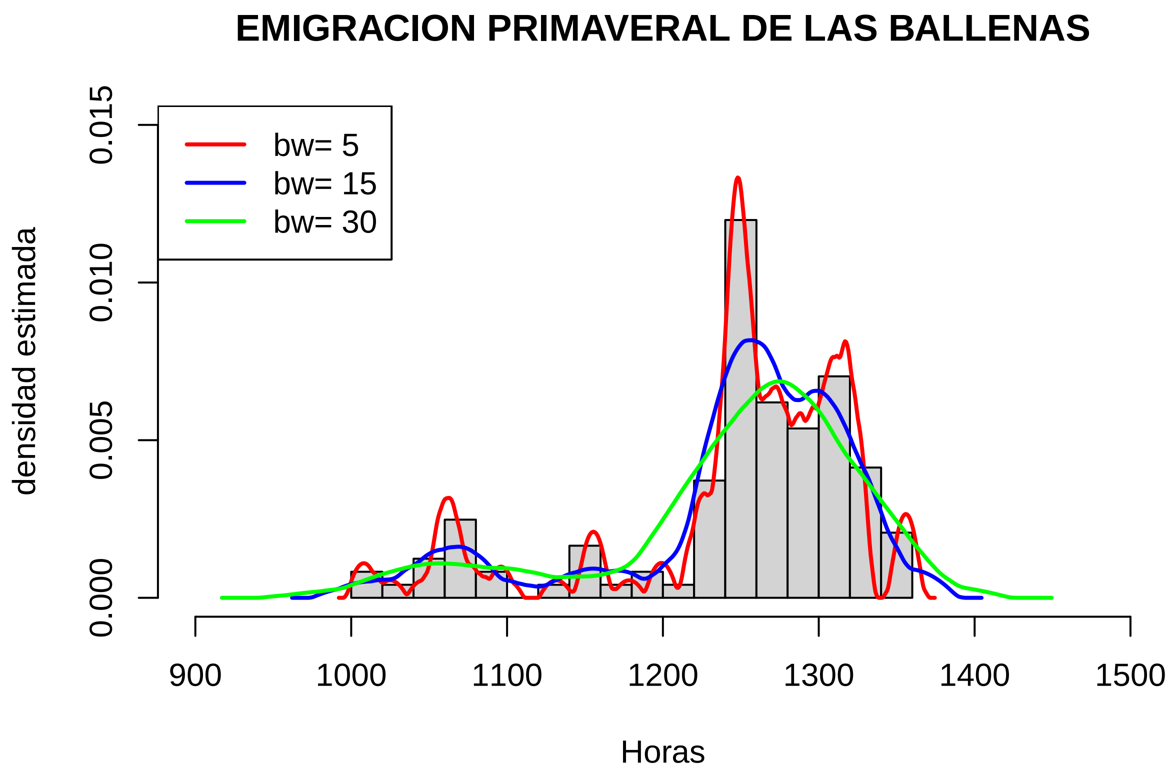
```
     xlab="Horas",ylab="densidad estimada",
     ylim = c(0,0.015),xlim = c(900,1500))

lines(density(Tiempo,bw=5),col="red",lwd=2)

lines(density(Tiempo,bw=15),col="blue",lwd=2)

lines(density(Tiempo,bw=30),col="green",lwd=2)

legend("topleft",
       col=c("red","blue","green"),
       lwd=2,
       legend=paste("bw=",c(5,15,30))
       )
```



5.1.2 Núcleo "epanechnikov"

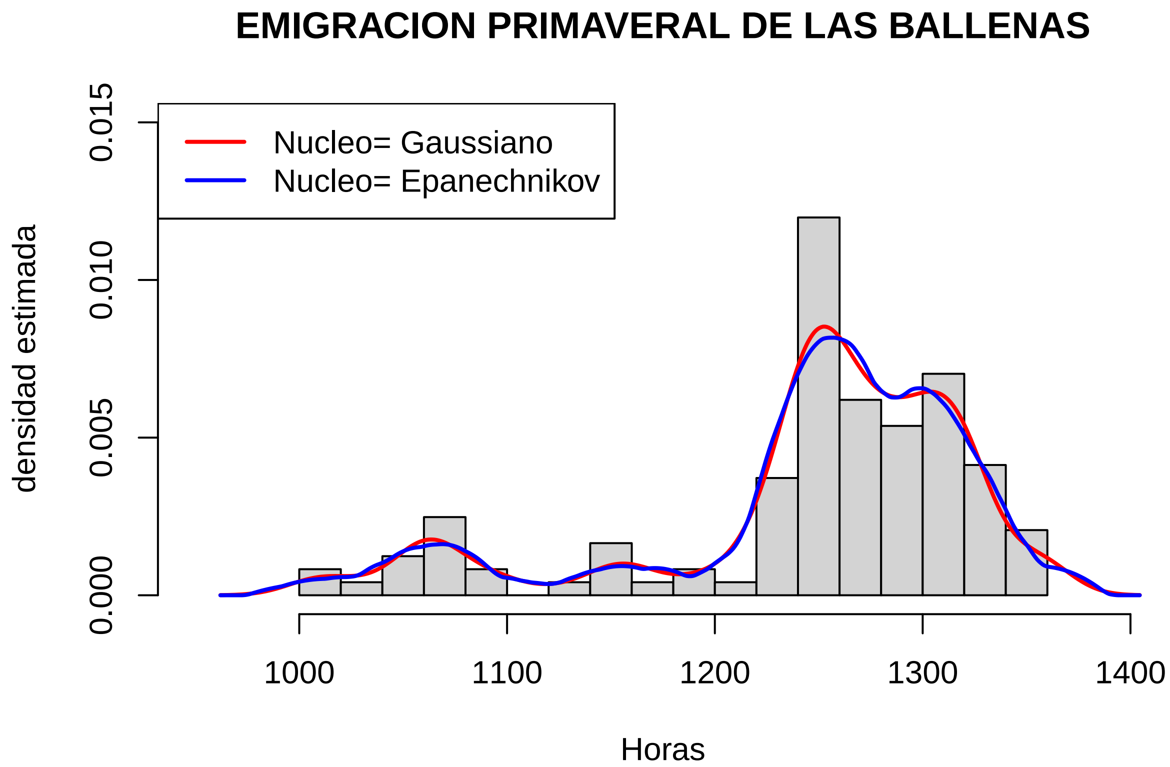
```
hist(Tiempo, prob=TRUE,
     br=20,
     main="EMIGRACION PRIMAVERAL DE LAS BALLENAS",
     col="lightgray",
     xlab="Horas",
     ylab="densidad estimada",
     ylim = c(0,0.015),xlim = c(900,1500))
lines(density(Tiempo,bw=5,kernel = "epanechnikov"),col="red",lwd=2)
lines(density(Tiempo,bw=15,kernel = "epanechnikov"),col="blue",lwd=2)
lines(density(Tiempo,bw=30,kernel = "epanechnikov"),col="green",lwd=2)
legend("topleft",
       col=c("red","blue","green"),
```

```
       lwd=2,
       legend=paste("bw=",c(5,15,30)
                    ))
```



5.1.3 Núcleo “epanechnikov” y gaussiano

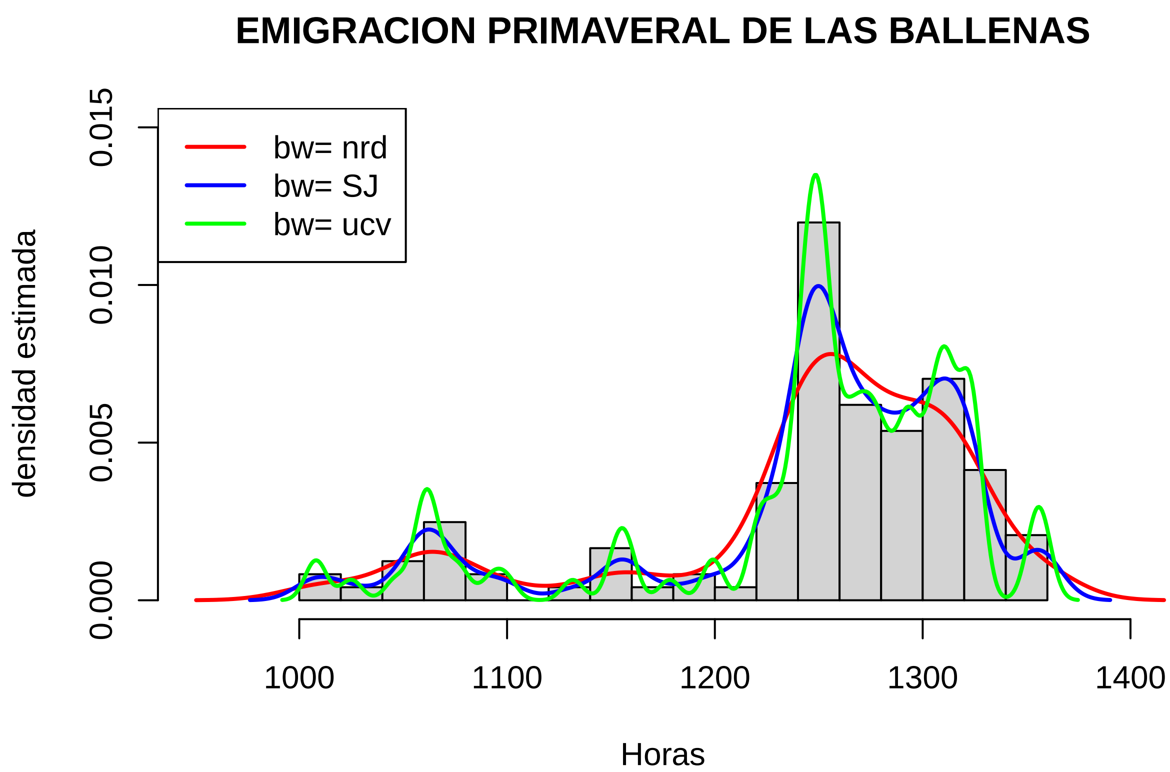
```
hist(Tiempo, prob=TRUE,br=20,
main="EMIGRACION PRIMAVERAL DE LAS BALLENAS",
col="lightgray",xlab="Horas",
ylab="densidad estimada",
ylim = c(0,0.015),xlim = c(950,1400))
lines(density(Tiempo,bw=15),col="red",lwd=2)
# "epanechnikov", "rectangular"
lines(density(Tiempo,bw=15,kernel = "epanechnikov"),col="blue",lwd=2)
legend("topleft",col=c("red","blue"),lwd=2,
legend=paste("Nucleo=",c("Gaussiano","Epanechnikov")))
```



5.2 Repetir el apartado anterior pero eligiendo h según los métodos nrd, SJ y UCV.

5.2.1 Elegir bw de forma automática

```
hist(Tiempo,
     prob=TRUE,
     br=20,
     main="EMIGRACION PRIMAVERAL DE LAS BALLENAS",
     col="lightgray",
     xlab="Horas", ylab="densidad estimada",
     ylim = c(0,0.015),xlim = c(950,1400))
lines(density(Tiempo,bw="nrd"),col="red",lwd=2)
lines(density(Tiempo,bw="SJ"),col="blue",lwd=2)
lines(density(Tiempo,bw="ucv"),col="green",lwd=2)
legend("topleft",col=c("red","blue","green"),lwd=2,
legend=paste("bw=",c("nrd","SJ","ucv")))
```



Nota: *bw: the bandwidth used.*

```
density(Tiempo, bw="nrd")
```

```
##
## Call:
##  density.default(x = Tiempo, bw = "nrd")
##
## Data: Tiempo (121 obs.); Bandwidth 'bw' = 18.89
##
##        x                 y
##  Min.   : 950.4   Min.   :3.463e-06
##  1st Qu.:1066.8   1st Qu.:5.154e-04
##  Median :1183.3   Median :8.930e-04
##  Mean   :1183.3   Mean   :2.145e-03
##  3rd Qu.:1299.7   3rd Qu.:3.170e-03
##  Max.   :1416.2   Max.   :7.810e-03
```

```
density(Tiempo,nw="SJ")
```

```
## Warning: In density.default(Tiempo, nw = "SJ") :
##  extra argument 'nw' will be disregarded
```

```
##
## Call:
##  density.default(x = Tiempo, nw = "SJ")
##
## Data: Tiempo (121 obs.); Bandwidth 'bw' = 16.04
##
##        x                 y
##  Min.   : 958.9   Min.   :3.938e-06
##  1st Qu.:1071.1   1st Qu.:5.502e-04
```

```
##   Median :1183.3   Median :9.498e-04
##   Mean   :1183.3   Mean   :2.226e-03
##   3rd Qu.:1295.4   3rd Qu.:3.285e-03
##   Max.   :1407.6   Max.   :8.300e-03
```

```
density(Tiempo,nw="ucv")
```

```
## Warning: In density.default(Tiempo, nw = "ucv") :
##  extra argument 'nw' will be disregarded
```

```
##
## Call:
##  density.default(x = Tiempo, nw = "ucv")
##
## Data: Tiempo (121 obs.); Bandwidth 'bw' = 16.04
##
##        x                 y
##   Min.   : 958.9   Min.   :3.938e-06
##   1st Qu.:1071.1   1st Qu.:5.502e-04
##   Median :1183.3   Median :9.498e-04
##   Mean   :1183.3   Mean   :2.226e-03
##   3rd Qu.:1295.4   3rd Qu.:3.285e-03
##   Max.   :1407.6   Max.   :8.300e-03
```

5.3 Utilizando el método SJ, dibujar la estimación núcleo y las contribuciones de cada observación.

Dibujar las contribuciones a la función de densidad

```
xgrid <- seq(from = min(Tiempo)-1,
             to = max(Tiempo) + 1,
             by = 0.5) #709 puntos

h <- density(Tiempo,bw="SJ")$bw

n<- length(Tiempo)
```

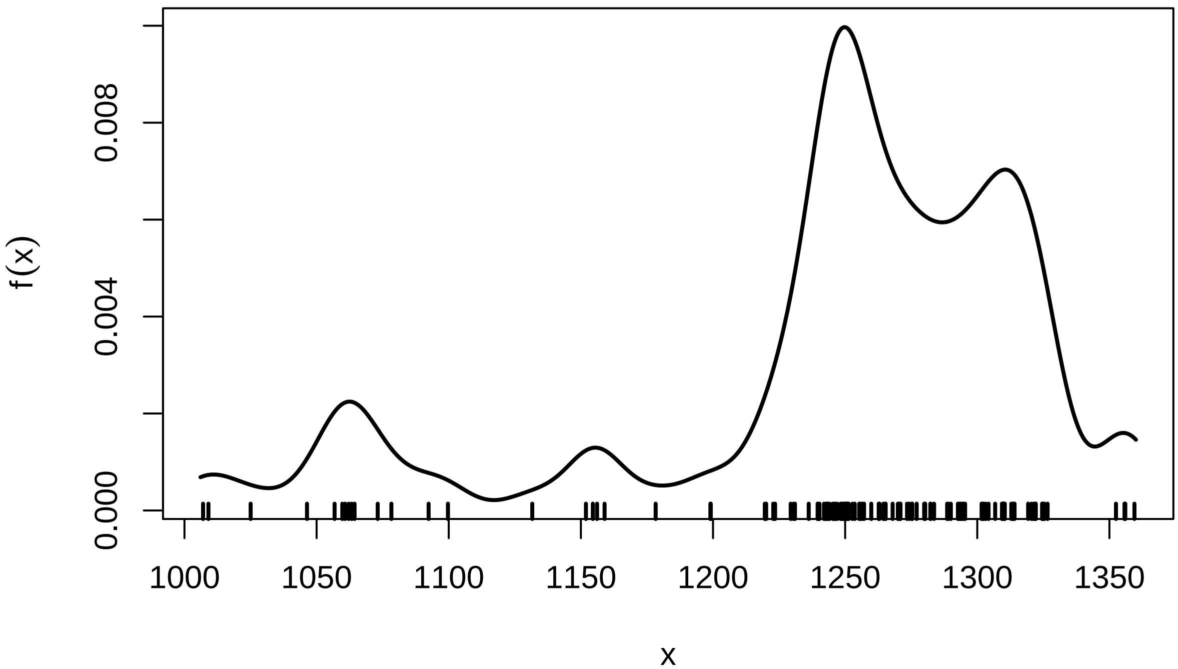
Nota: K_i contiene en cada columna los $\frac{\frac{x_{grid}-X_i}{h}}{nh}$ para $i = 1, 2, ..., n$.

```
Ki <- sapply(Tiempo, function(a) gauss((xgrid - a)/h)/(n * h))
dim(Ki)
```

```
## [1] 709 121
```

Lo dibujamos

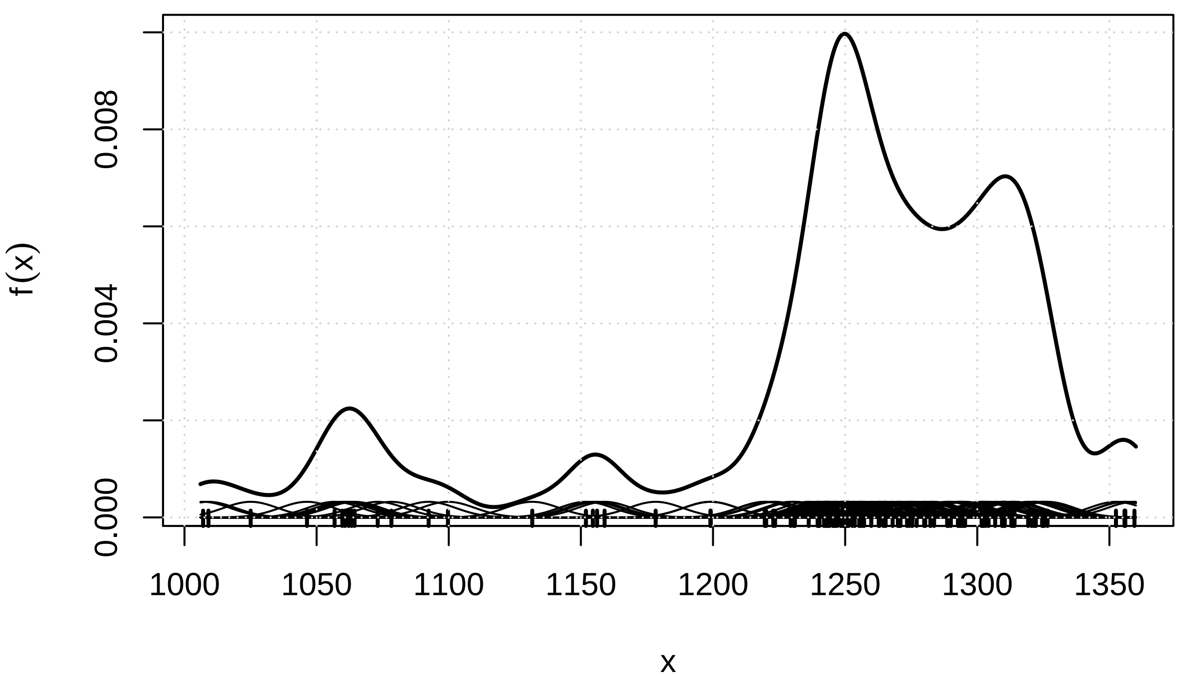
```
plot(xgrid,
     rowSums(Ki),
     ylab = expression(hat(f)(x)),
     type = "l", xlab = "x", lwd = 2)
rug(Tiempo, lwd = 2)
```




```
## Se han dibujado 121 líneas. Cada línea i está formada por los puntos:
## {(xgrid_j,K((xgrid_i-Xi)/h)/(nh)), j=1,...,709}
```

La suma de todas las líneas da la estimación núcleo:

```
plot(xgrid,
     rowSums(Ki),
     ylab = expression(hat(f)(x)),
     type = "l", xlab = "x", lwd = 2)
rug(Tiempo, lwd = 2)
out <- apply(Ki, 2, function(b) lines(xgrid, b))
grid()
```



5.4 Librería KernSmooth

Repetir el apartado ii) añadiendo la estimación que ofrece la función bkde de la librería KernSmooth.

```
library(KernSmooth)
```

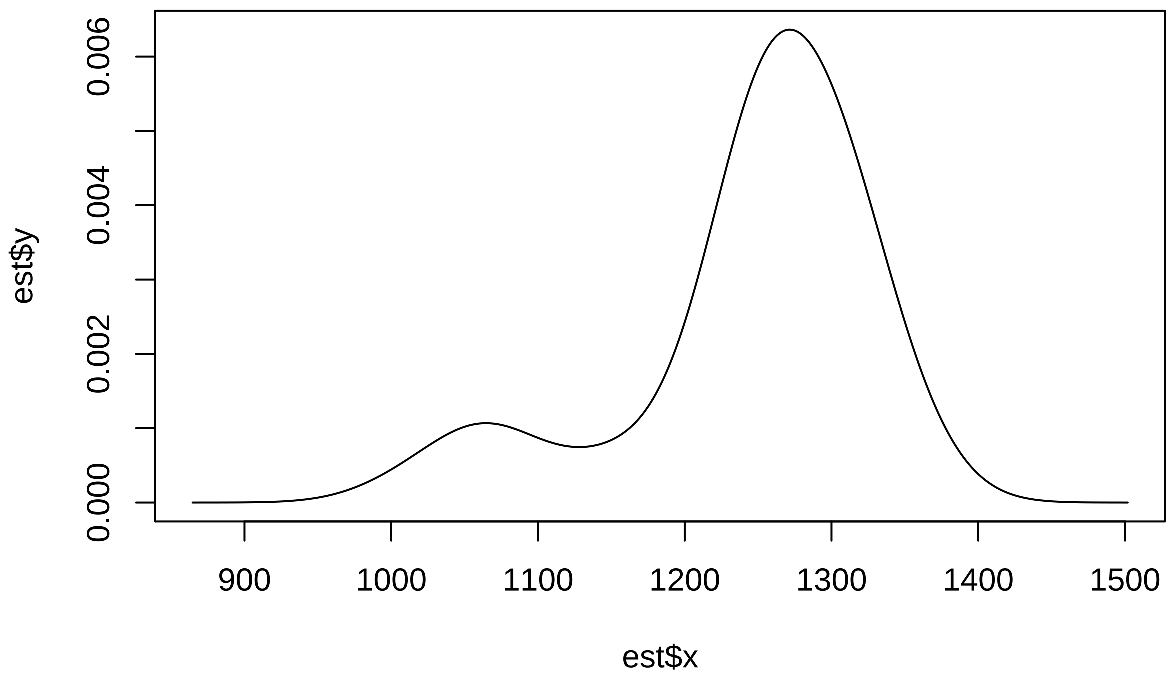
```
## KernSmooth 2.23 loaded
## Copyright M. P. Wand 1997-2009
```

```
est <- bkde(Tiempo)
str(est)
```

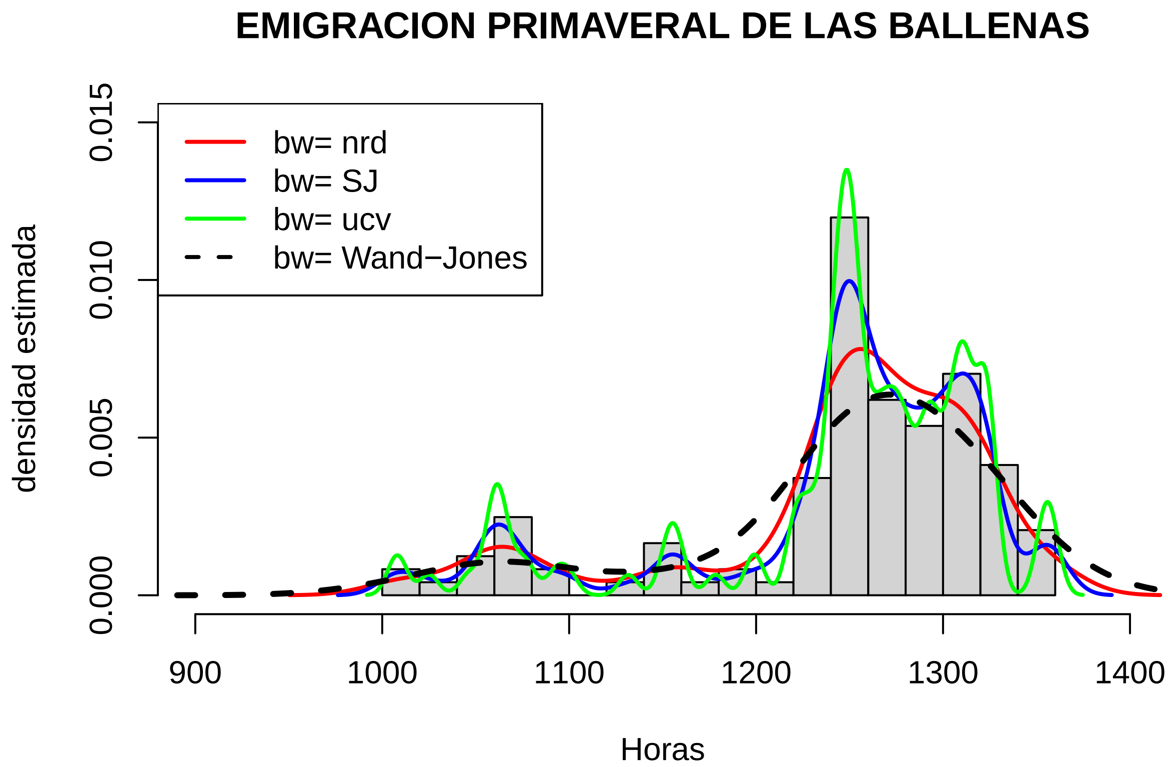
```
## List of 2
##  $ x: num [1:401] 865 866 868 869 871 ...
##  $ y: num [1:401] 2.05e-08 4.86e-08 7.99e-08 9.52e-08 1.13e-07 ...
```

Dibujamos la densidad:

```
plot(est, type="l")
```




```
hist(Tiempo,
     prob=TRUE,
     br=20,
     main="EMIGRACION PRIMAVERAL DE LAS BALLENAS",
     col="lightgray", xlab="Horas",
     ylab="densidad estimada",
     xlim = c(900,1400) , ylim=c(0,0.015))
lines(density(Tiempo,bw="nrd"),col="red",lwd=2)
lines(density(Tiempo,bw="SJ"),col="blue",lwd=2)
lines(density(Tiempo,bw="ucv"),col="green",lwd=2)
lines(est,col="black",lty=2,lwd=3)
legend("topleft",
       col=c("red","blue","green","black"),
       lwd=2,
       lty=c(1,1,1,2),legend=paste("bw=",c("nrd","SJ","ucv","Wand-Jones")))
```



6 Ejercicio 6

6. Responder a los siguientes apartados:

6.1 Mixtura univariante

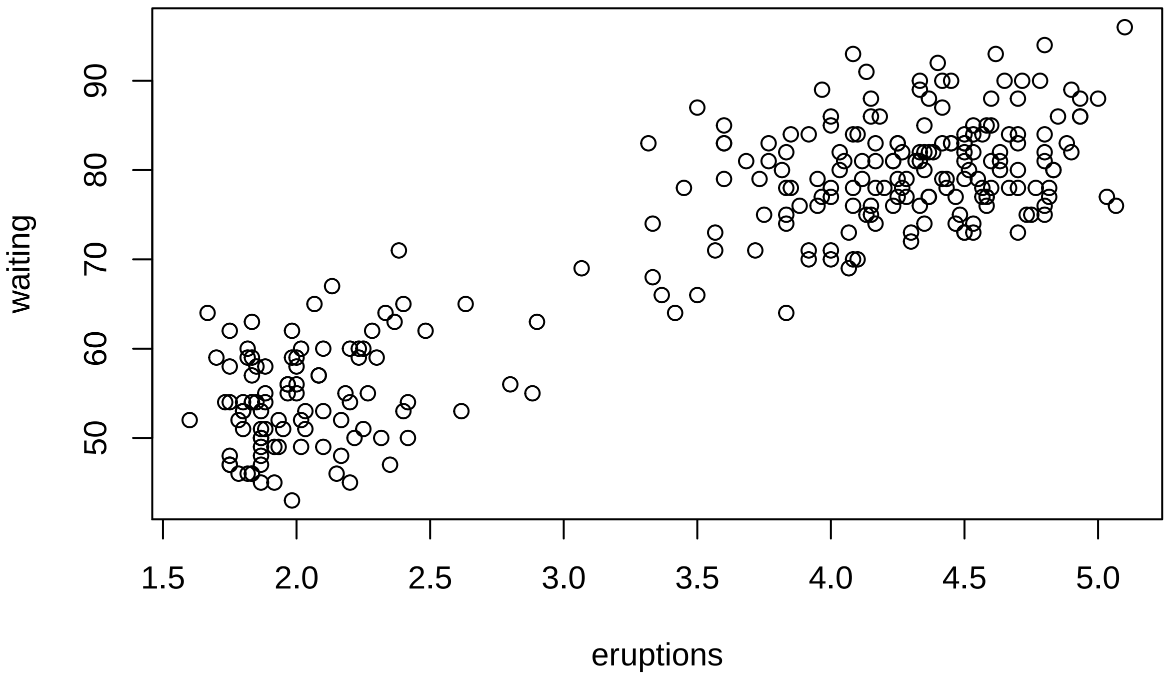
Cargar el fichero faithful de R y comprobar mediante estimaciones no paramétricas de la función de densidad que las variables eruption y waiting parecen seguir sendas mixturas.

```
data("faithful")
summary(faithful)
```

```
##    eruptions        waiting
##  Min.   :1.600   Min.   :43.0
##  1st Qu.:2.163   1st Qu.:58.0
##  Median :4.000   Median :76.0
##  Mean   :3.488   Mean   :70.9
##  3rd Qu.:4.454   3rd Qu.:82.0
##  Max.   :5.100   Max.   :96.0
```

Vamos a dibujarlo

```
plot(faithful)
```



- Eruptions: Duración de las erupciones

Test de Shapiro de normalidad

```
attach(faithful)
shapiro.test(eruptions)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  eruptions
## W = 0.84592, p-value = 9.036e-16
```

Rechazo la normalidad de esta variable.

```
summary(eruptions)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.600   2.163   4.000   3.488   4.454   5.100
```

```
fivenum(eruptions)
```

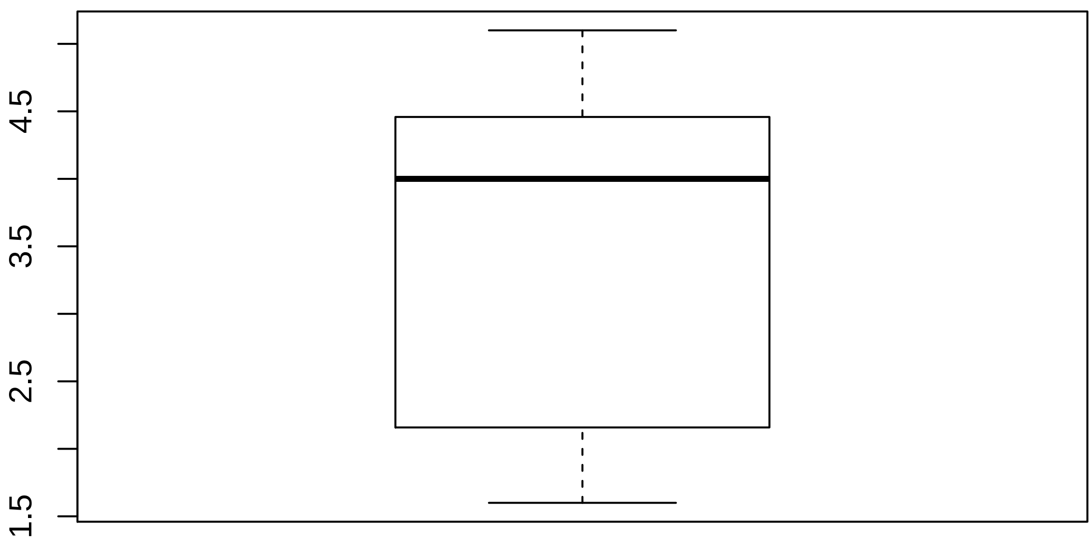
```
## [1] 1.6000 2.1585 4.0000 4.4585 5.1000
```

```
stem(eruptions)
```

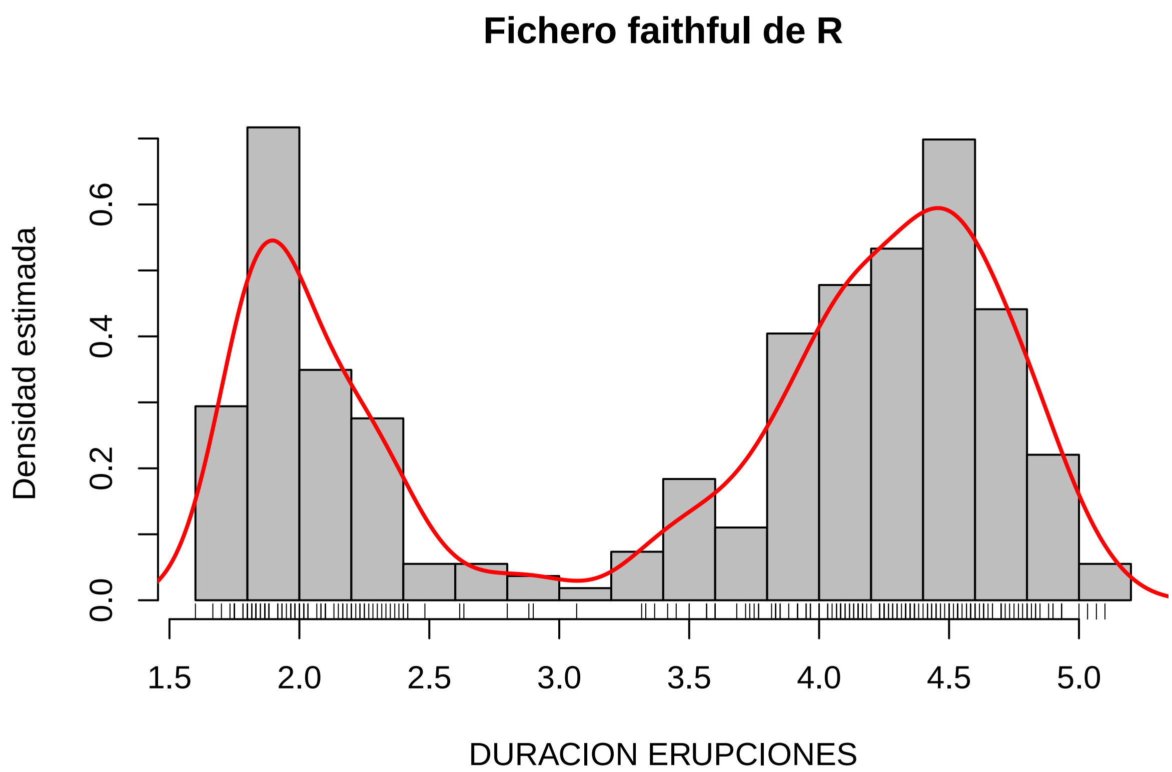
```
##
##   The decimal point is 1 digit(s) to the left of the |
##
##   16 | 070355555588
##   18 | 000022233333335577777777888822335777888
##   20 | 00002223378800035778
##   22 | 0002335578023578
##   24 | 00228
##   26 | 23
##   28 | 080
```

```
##    30 | 7
##    32 | 2337
##    34 | 250077
##    36 | 0000823577
##    38 | 2333335582225577
##    40 | 0000003357788888002233555577778
##    42 | 03335555778800233333555577778
##    44 | 02222335557780000000023333357778888
##    46 | 0000233357700000023578
##    48 | 00000022335800333
##    50 | 0370
```

```r
boxplot(eruptions)
```




```r
hist(eruptions, seq(1.6,5.2, 0.2), prob=TRUE,
main="Fichero faithful de R", col="gray", xlab="DURACION ERUPCIONES",
ylab="Densidad estimada")
lines(density(eruptions, bw="SJ"),lwd=2,col="red")
rug(eruptions)
```




```r
estimaf<-density(eruptions, bw="SJ")
estimaf
```

```
##
## Call:
##  density.default(x = eruptions, bw = "SJ")
##
## Data: eruptions (272 obs.);  Bandwidth 'bw' = 0.14
##
##        x               y
##  Min.   :1.180   Min.   :0.0001834
##  1st Qu.:2.265   1st Qu.:0.0422638
##  Median :3.350   Median :0.1709243
##  Mean   :3.350   Mean   :0.2301726
##  3rd Qu.:4.435   3rd Qu.:0.4134348
##  Max.   :5.520   Max.   :0.5945634
```

- Waiting: Muestra el tiempo entre dos erupciones seguidas

```r
shapiro.test(waiting)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  waiting
## W = 0.92215, p-value = 1.015e-10
```

Rechazo la normalidad.

```r
summary(waiting)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
```

```
##    43.0    58.0    76.0    70.9    82.0    96.0
```

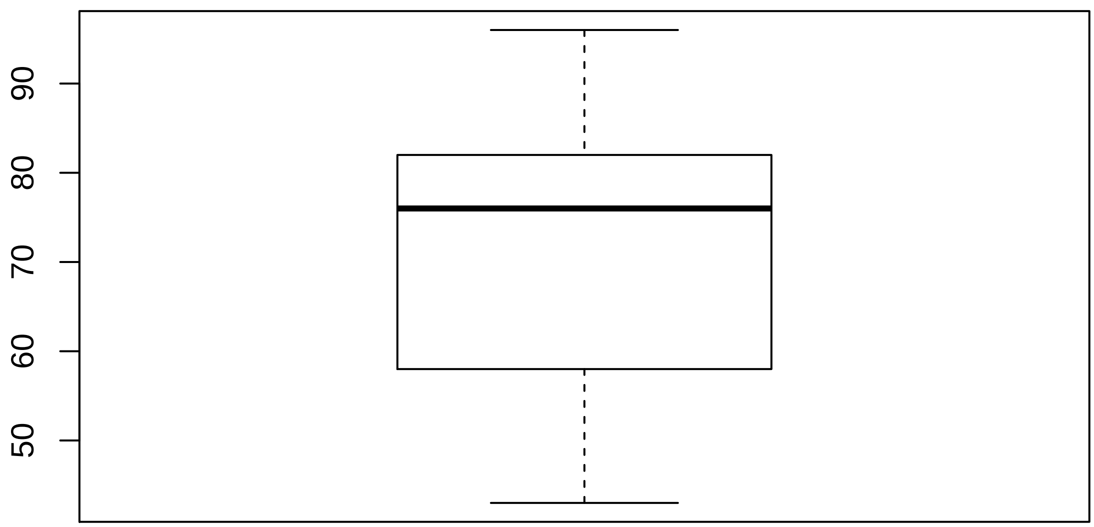
```r
fivenum(waiting)
```

```
## [1] 43 58 76 82 96
```

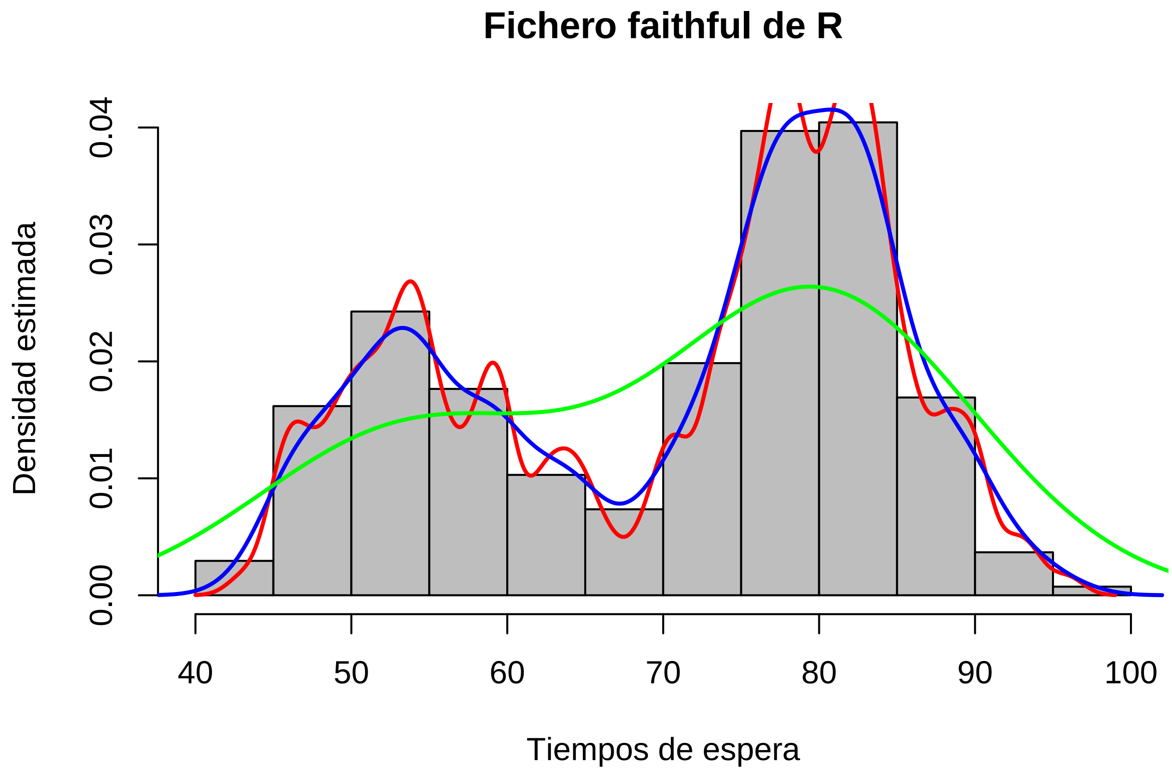
```r
stem(waiting)
```

```
##
##   The decimal point is 1 digit(s) to the right of the |
##
##   4 | 3
##   4 | 55566666777788899999
##   5 | 00000111111222223333333444444444
##   5 | 555555666677788889999999
##   6 | 00000022223334444
##   6 | 555667899
##   7 | 00001111123333333444444
##   7 | 555555556666666667777777777778888888888888889999999999
##   8 | 000000001111111111111222222222222333333333333334444444444
##   8 | 55555566666677888888999
##   9 | 00000012334
##   9 | 6
```

```r
boxplot(waiting)
```

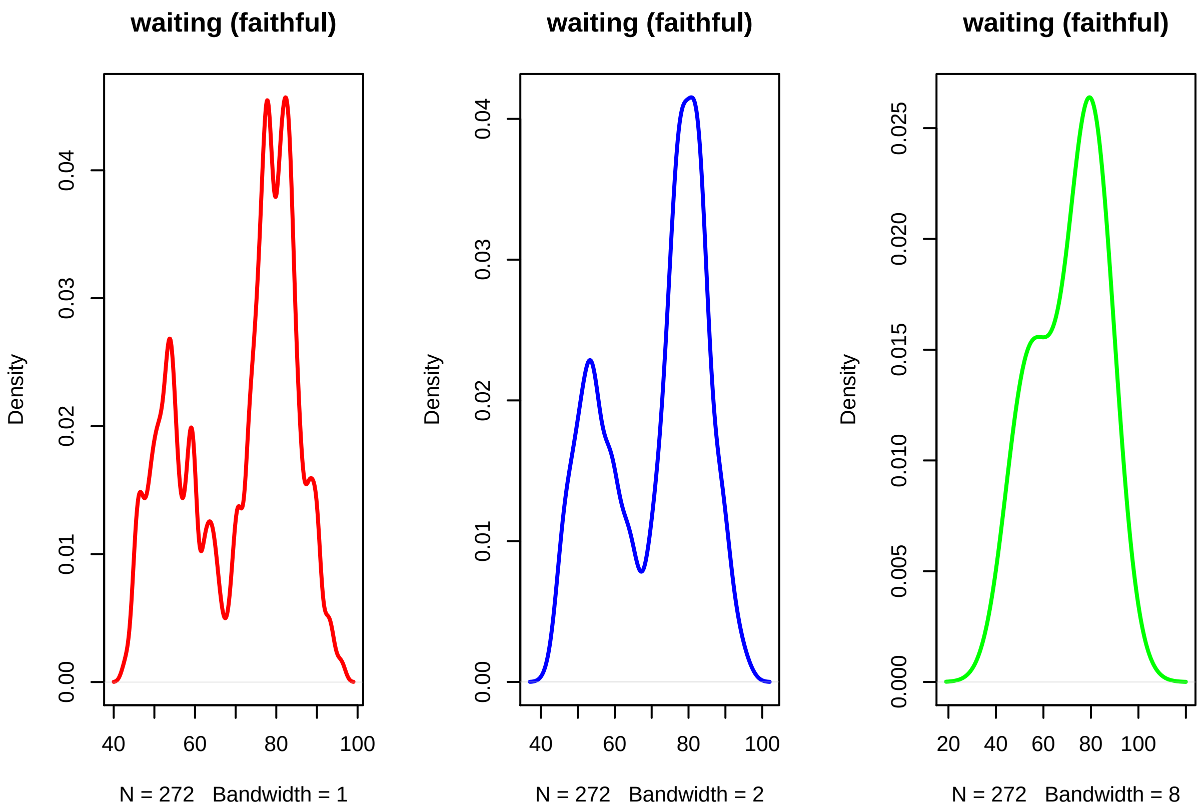



```r
hist(waiting, seq(40,100, 5), prob=TRUE,
main="Fichero faithful de R", col="gray", xlab="Tiempos de espera",
ylab="Densidad estimada")
lines(density(waiting,bw=1),col="red",lwd=2)
lines(density(waiting,bw=2),col="blue",lwd=2)
lines(density(waiting,bw=8),col="green",lwd=2)
```

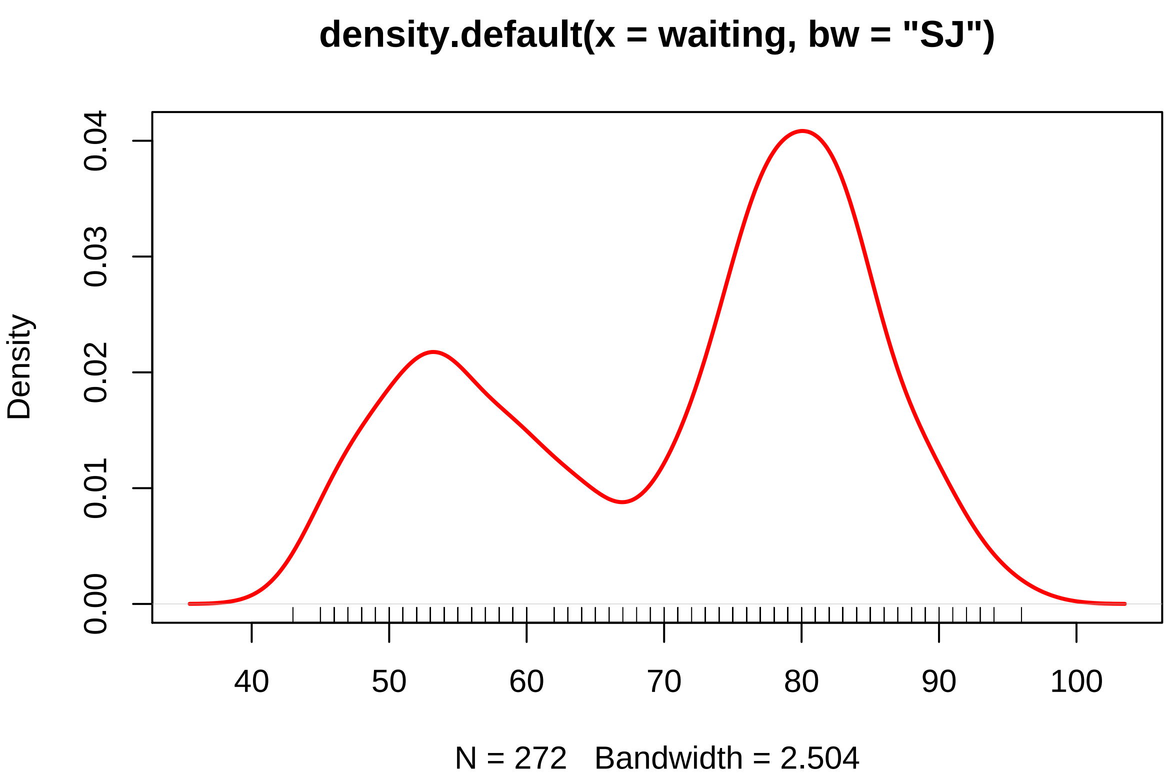


Vamos a dibujar las esitmaciones de la densidas

```
par(mfrow=c(1,3))
plot(density(waiting,bw=1),col="red",
lwd=2,main="waiting (faithful)")
plot(density(waiting,bw=2),col="blue",
lwd=2,main="waiting (faithful)")
plot(density(waiting,bw=8),col="green",
lwd=2,main="waiting (faithful)")
```




```r
par(mfrow=c(1,1))
plot(density(waiting, bw="SJ"),lwd=2,col="red")
rug(waiting)
```



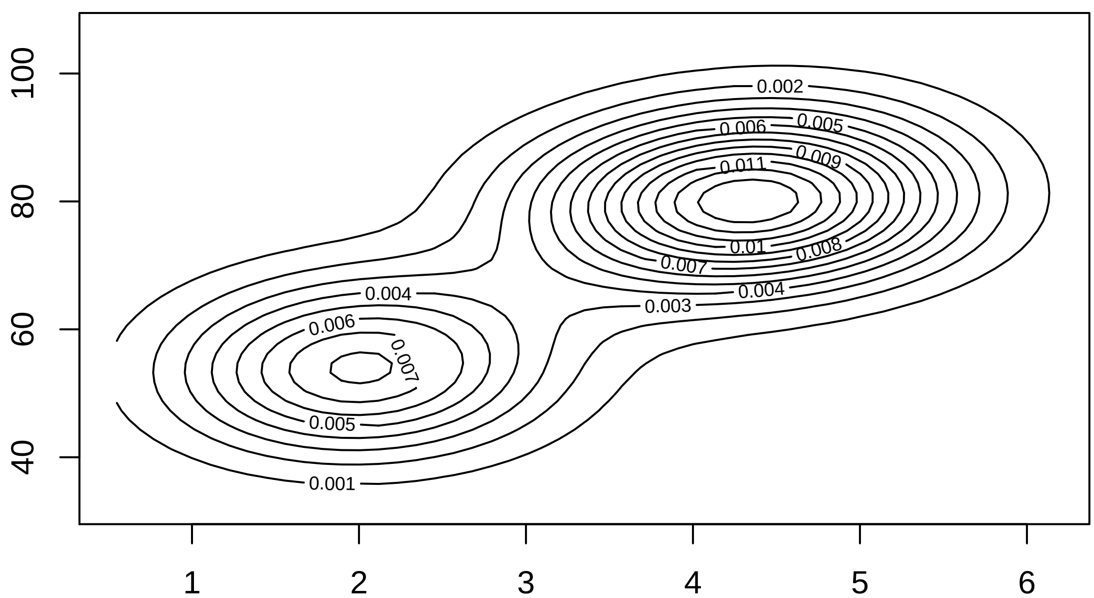
```r
estimaf<-density(waiting, bw="SJ")
estimaf
```

```
##
## Call:
##  density.default(x = waiting, bw = "SJ")
##
## Data: waiting (272 obs.);    Bandwidth 'bw' = 2.504
##
##        x                 y
##  Min.   : 35.49   Min.   :7.240e-06
##  1st Qu.: 52.49   1st Qu.:4.301e-03
##  Median : 69.50   Median :1.302e-02
##  Mean   : 69.50   Mean   :1.469e-02
##  3rd Qu.: 86.51   3rd Qu.:2.088e-02
##  Max.   :103.51   Max.   :4.085e-02
```

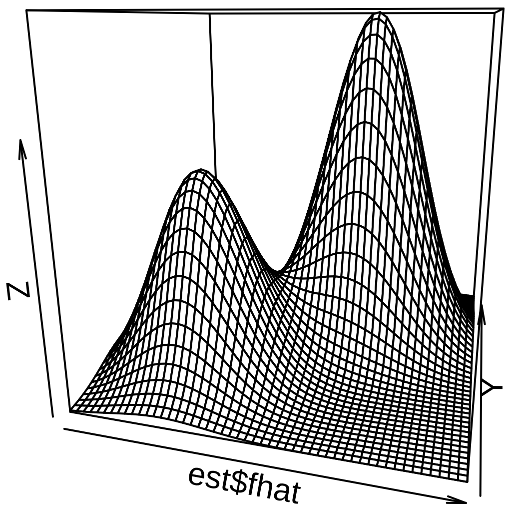
6.2 Mixtura bivariante

Con la ayuda de la función bkde2D de la librería KernSmooth, estimar la densidad bivariante de estas dos variables.

```r
library(KernSmooth)
est <- bkde2D(faithful, bandwidth=c(0.7, 7))
#una anchura de ventana para cada dimensión
#se usa el núcleo gaussiano bivariante
contour(est$x1, est$x2, est$fhat)
```



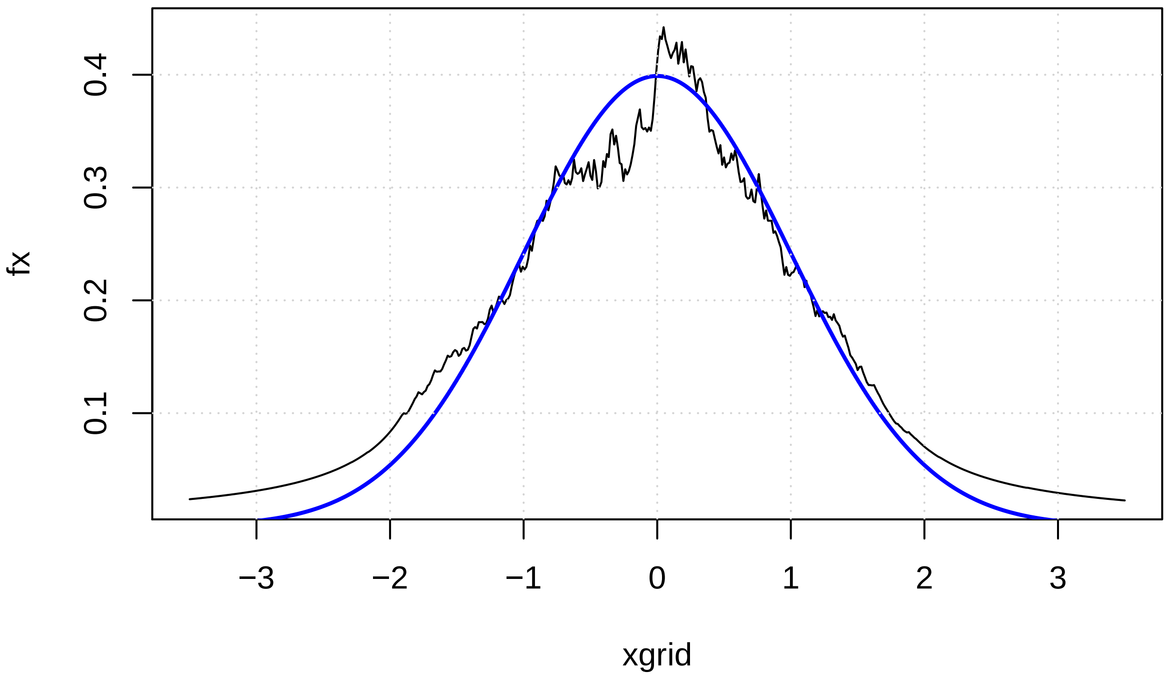

```r
persp(est$fhat,theta=15)
```



7 Ejercicio 7

7.1 Ilustrar con una simulación el método de los k vecinos más próximos.

```
n=1000
X=rnorm(n)
xgrid=seq(from=-3.5,to=3.5,length=512)
ng=length(xgrid)
k=100
dk=numeric(ng)
for (i in 1:ng)
{
distancias=abs(xgrid[i]-X)
dk[i]=distancias[which(rank(distancias)==k)]
}
fx=(k-1)/(2*n*dk)
plot(xgrid,fx,type="l")
lines(xgrid,dnorm(xgrid),col="blue",lwd=2)
grid()
```



8 Ejercicio 8 fichero "migracionballenas.dat"

Leemos los datos:

```
datos<- read.table("datos/migracionballenas.dat",header=TRUE)
summary(datos)
```

```
##      Tiempo
##  Min.   :1007
##  1st Qu.:1240
##  Median :1260
##  Mean   :1246
##  3rd Qu.:1302
##  Max.   :1359
```

8.1 Estimar la función de densidad mediante logsplines.

```
attach(datos)
```

```
## The following object is masked from datos (pos = 5):
##
##     Tiempo
```

```
library(logspline)
ajuste <- logspline(Tiempo)
ajuste # 7nudos, criterio BIC
```

```
##  knots A(1)/D(2)  loglik     AIC minimum penalty maximum penalty
##      4         2 -684.67 1383.73           35.01             Inf
##      5         2 -667.17 1353.52            5.69           35.01
##      6         2 -666.29 1356.56              NA              NA
##      7         2 -661.47 1351.72            1.45            5.69
##      8         2 -660.75 1355.06            0.06            1.45
```

```
##      9          2 -660.72 1359.81            0.01            0.06
##     10          2 -660.71 1364.59            0.01            0.01
##     11          1 -660.71 1369.37            0.00            0.01
## the present optimal number of knots is  7
## penalty(AIC) was the default: BIC=log(samplesize): log( 121 )= 4.8
```

```
# Ver: help(logspline)
```

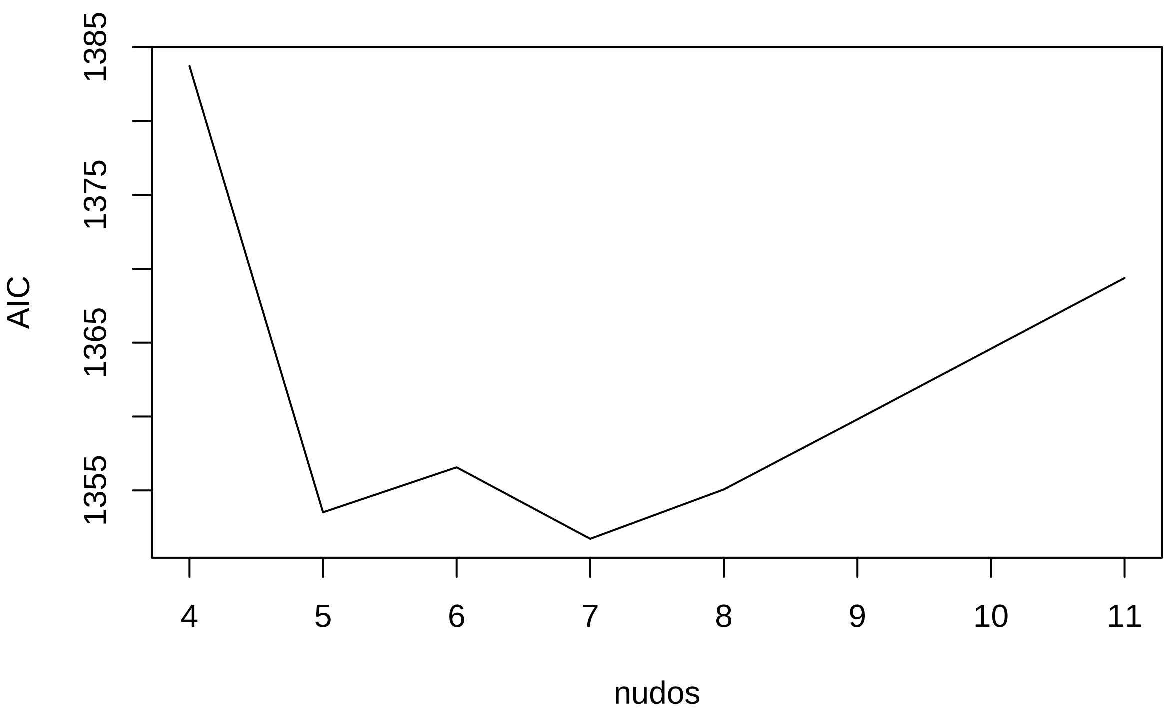
Vamos a comprobar los valores AIC:

```
resul<- ajuste$logl
nudos<- resul[,1]
logL<- resul[,3]
AIC<- -2*logL+log(length(Tiempo))*(nudos-1)
AIC
```

```
## [1] 1383.729 1353.518 1356.563 1351.722 1355.065 1359.805 1364.587 1369.374
```

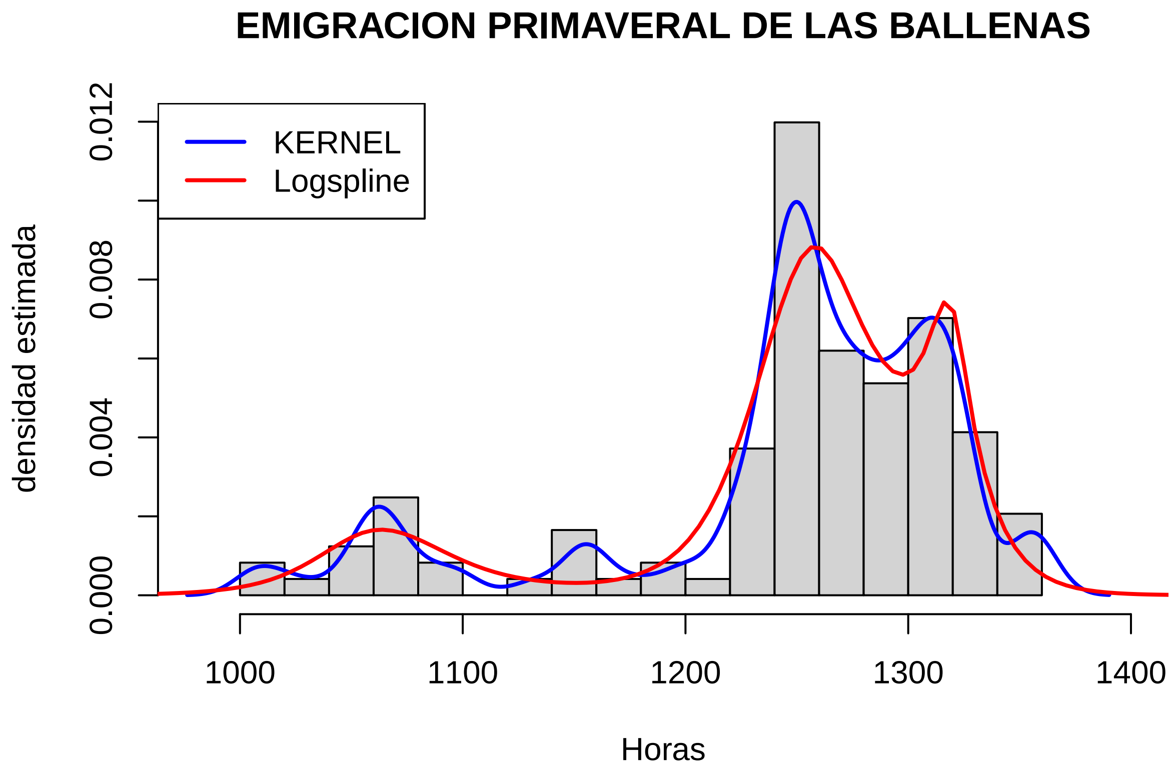
Hacemos un dibujo:

```
plot(nudos,AIC,type="l")
```



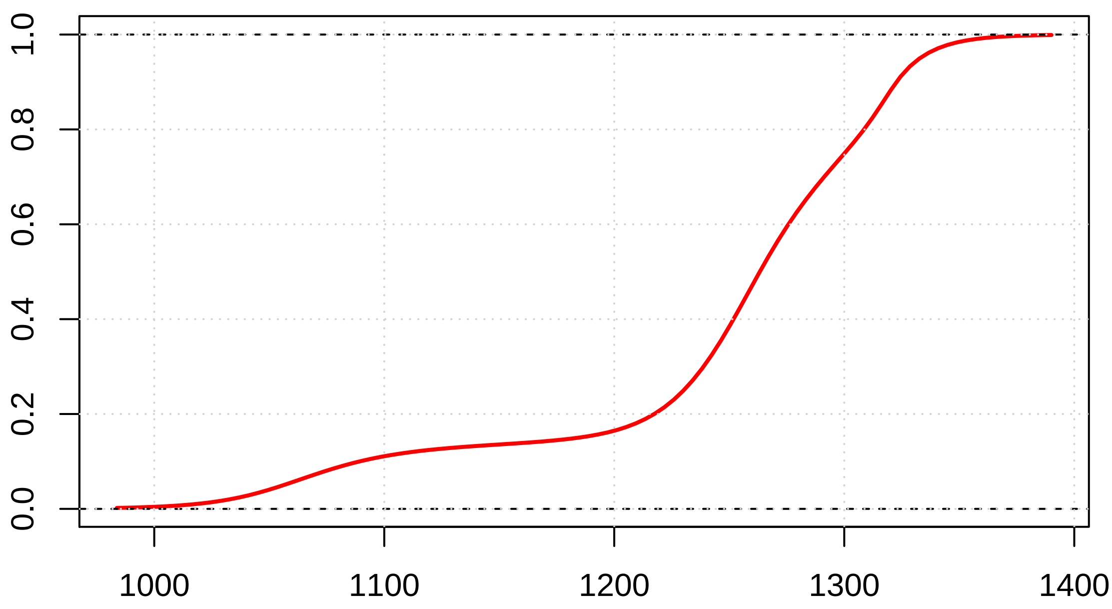
Dibujamos el histograma

```
hist(Tiempo,
     prob=TRUE,br=20,
     main="EMIGRACION PRIMAVERAL DE LAS BALLENAS",
     col="lightgray", xlab="Horas",ylab="densidad estimada",
     xlim = c(980,1400))
lines(density(Tiempo,bw="SJ"),col="blue",lwd=2)

plot(ajuste,col="red",lwd=2,add=TRUE)
legend("topleft",col=c("blue","red"),
       lwd=2,
       legend=c("KERNEL","Logspline"))
```



8.2 Dibujar la estimación de la función de distribución, y probar las funciones qlogspline y rlogspline.

```
plot(ajuste,what="p",col="red",lwd=2)
abline(h=c(0,1),lty=2)
grid()
```



Los cuantiles:

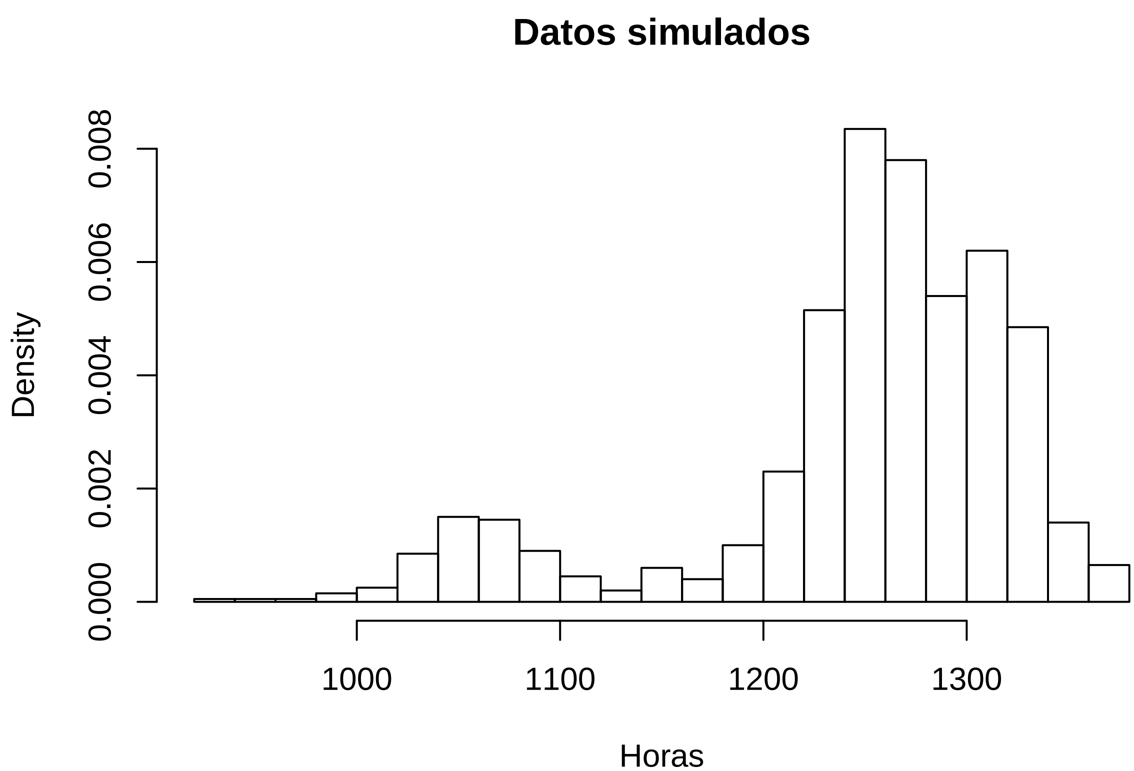
```r
qlogspline((1:99)/100, ajuste)
```

```
##  [1] 1017.580 1032.641 1042.164 1049.637 1056.150 1062.251 1068.294 1074.566
##  [9] 1081.388 1089.207 1098.849 1111.994 1132.682 1162.960 1184.297 1196.000
## [17] 1203.618 1209.225 1213.662 1217.349 1220.535 1223.301 1225.798 1228.073
## [25] 1230.164 1232.103 1233.917 1235.629 1237.257 1238.814 1240.302 1241.725
## [33] 1243.108 1244.455 1245.748 1247.020 1248.264 1249.478 1250.679 1251.855
## [41] 1253.020 1254.173 1255.316 1256.455 1257.587 1258.718 1259.850 1260.986
## [49] 1262.127 1263.276 1264.437 1265.610 1266.798 1268.002 1269.226 1270.472
## [57] 1271.742 1273.036 1274.362 1275.716 1277.104 1278.526 1279.984 1281.481
## [65] 1283.016 1284.590 1286.205 1287.858 1289.546 1291.267 1293.016 1294.787
## [73] 1296.571 1298.360 1300.143 1301.910 1303.648 1305.347 1307.000 1308.596
## [81] 1310.132 1311.612 1313.042 1314.432 1315.793 1317.136 1318.478 1319.834
## [89] 1321.226 1322.684 1324.248 1325.971 1327.915 1330.159 1332.814 1336.062
## [97] 1340.250 1346.153 1356.244
```

```r
qlogspline((1:3)/4, ajuste)
```

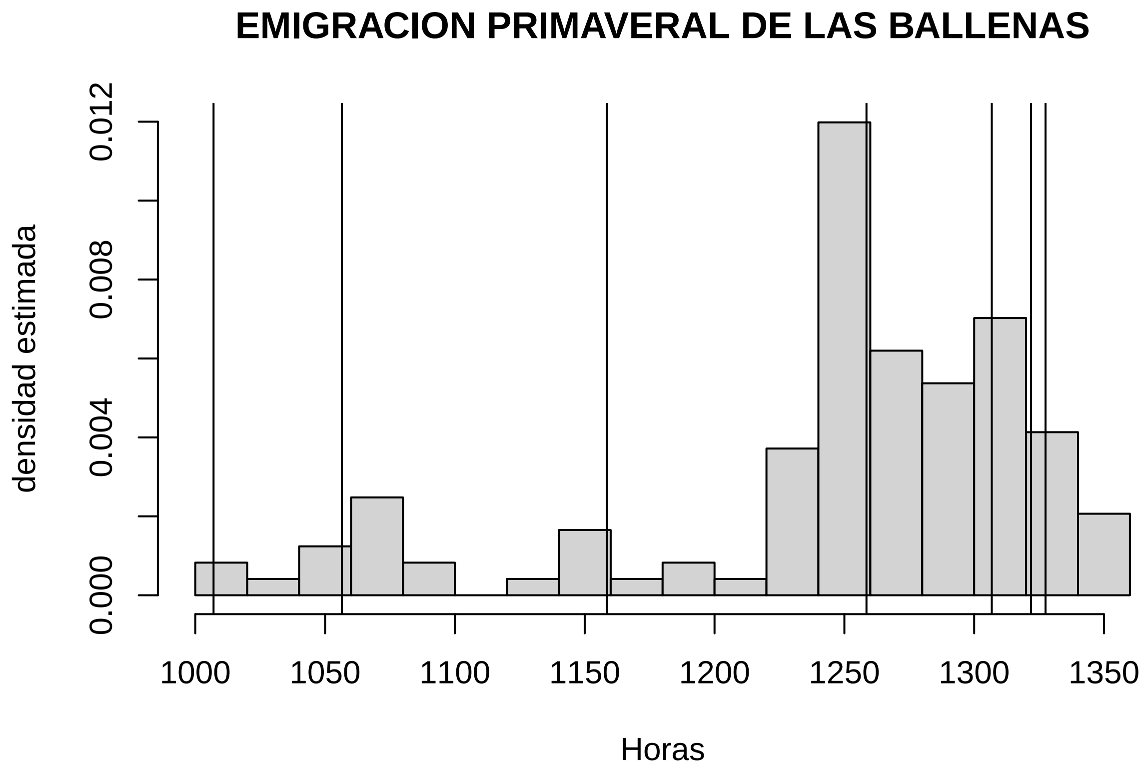
```
## [1] 1230.164 1263.276 1300.143
```

```r
hist(rlogspline(1000,ajuste),br=20,main="Datos simulados",xlab="Horas",
                prob=TRUE)
```



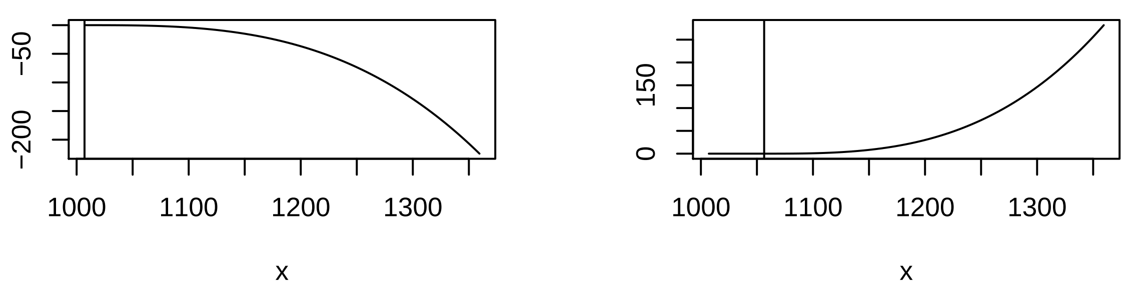
8.3 Dibujar las funciones base que forman el spline.

```r
hist(Tiempo, prob=TRUE,br=20,
     main="EMIGRACION PRIMAVERAL DE LAS BALLENAS",
     col="lightgray", xlab="Horas",ylab="densidad estimada")
abline(v=ajuste$knots) #posiciones de los nudos
```



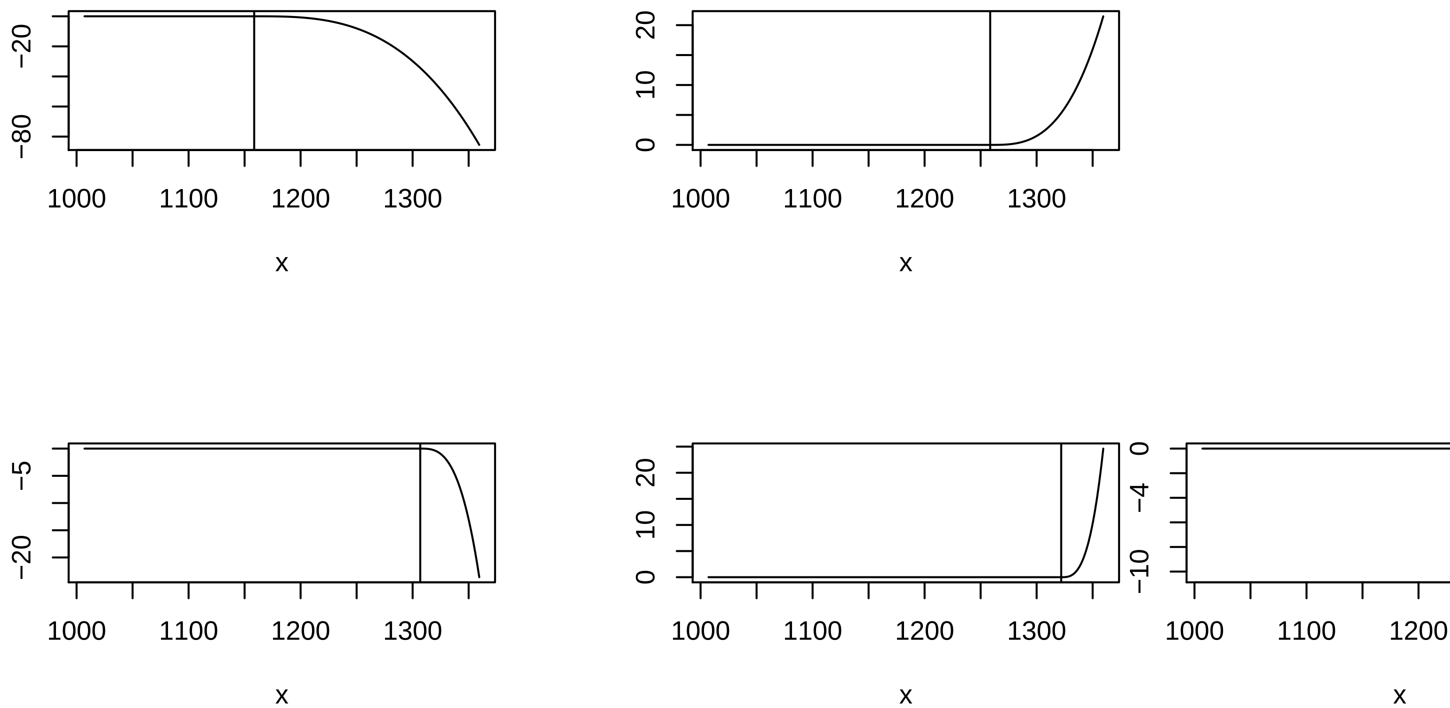
La forma de las componentes:

```
cubico<- function(x,nudo,coefi) {
  potencia<-(x-nudo)^3
  auxi<- cbind(potencia,rep(0,length(x)))
  positivo<- apply(auxi,1,max)
  coefi*positivo
}
par(mfrow=c(2,2))
for (i in 1:2)
{
  curve(cubico(x,
               nudo=ajuste$knots[i],
               coefi=ajuste$coef.kts[i]),
        min(Tiempo),max(Tiempo),1000,ylab="")
abline(v=ajuste$knots[i])
}
```




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

for (i in 3:ajuste$nknots) {
  curve(cubico(x,nudo=ajuste$knots[i],
```

```
              coefi=ajuste$coef.kts[i]),
        min(Tiempo),max(Tiempo),1000,ylab="")
 abline(v=ajuste$knots[i])
 }
```



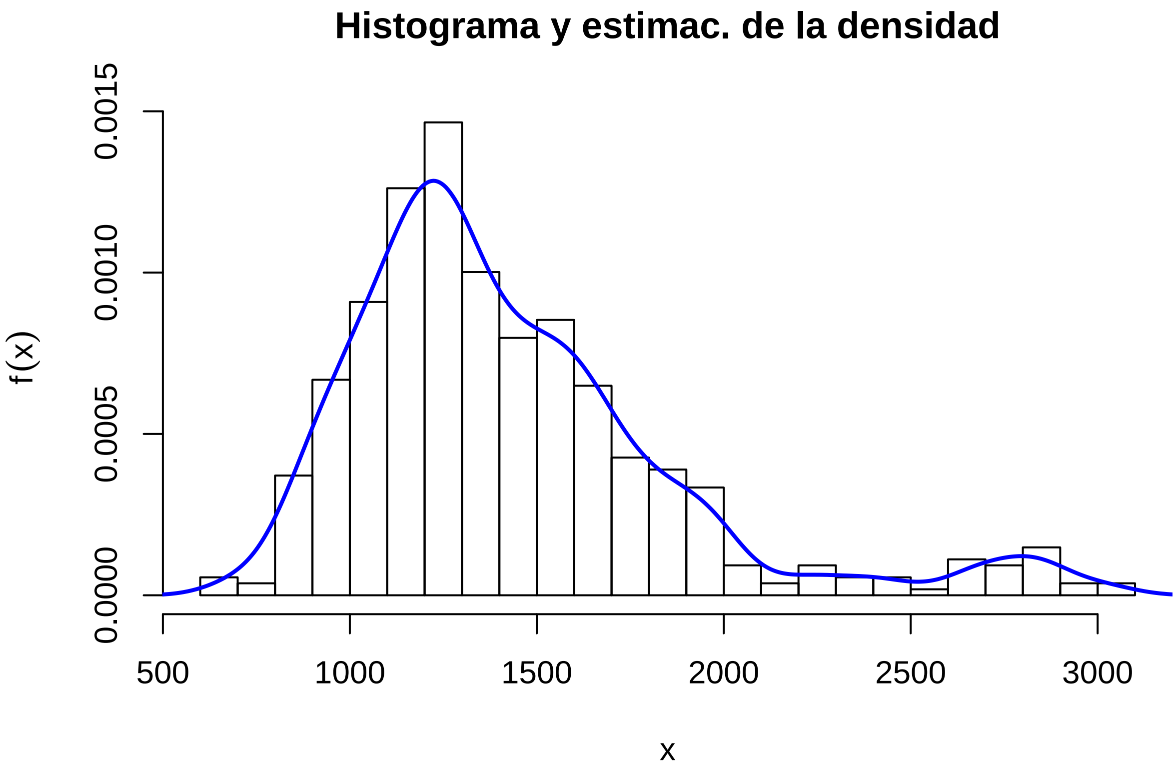

```
par(mfrow=c(1,1))
```

9 Ejercicio 9 fichero "Pesos.RData"

El fichero datos en "Pesos.RData" contiene los pesos en gramos de cierto animal:

9.1 Realizar una estimación no paramétrica de la función de densidad por el método del núcleo.

```
load("datos/Pesos.RData")
hist(datos,br=30,prob=TRUE,
     main="Histograma y estimac. de la densidad",
     ylab = expression(hat(f)(x)),xlab="x")

lines(density(datos,bw="SJ"),col="blue",lwd=2)
```



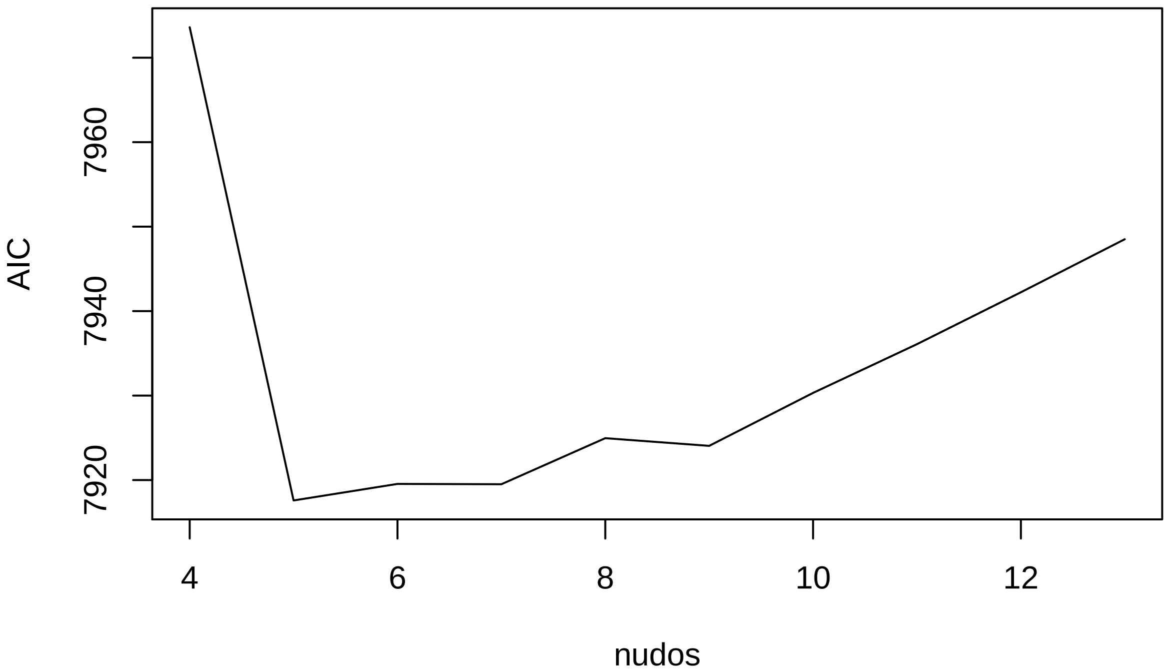
9.2 Realizar una estimación no paramétrica de la función de densidad por el método de los logsplines.

```
library(logspline)
ajuste<- logspline(datos)
ajuste
```

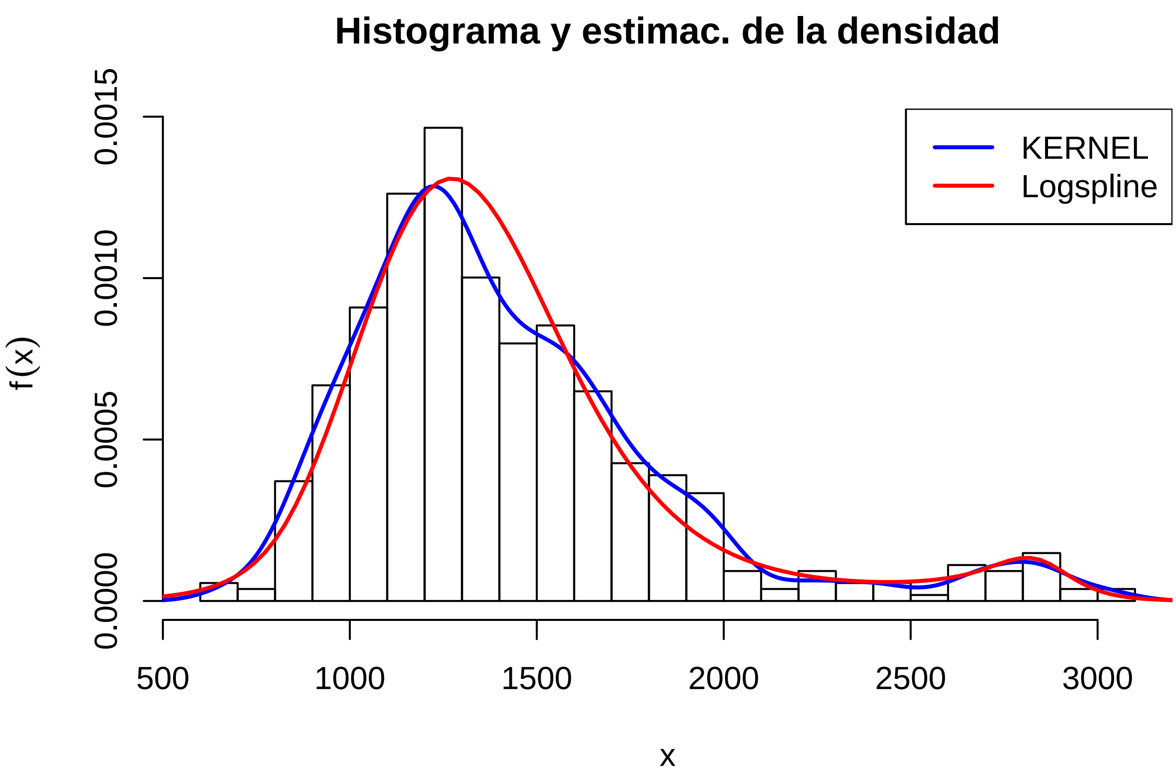
```
##   knots A(1)/D(2)    loglik     AIC minimum penalty maximum penalty
##       4         2 -3977.37 7973.61           62.31             Inf
##       5         2 -3946.22 7917.59            5.33           62.31
##       6         2 -3944.05 7919.55              NA              NA
##       7         2 -3940.88 7919.51            4.02            5.33
##       8         2 -3940.47 7924.96              NA              NA
##       9         2 -3936.87 7924.05            0.27            4.02
##      10         2 -3936.86 7930.32              NA              NA
##      11         2 -3936.60 7936.09            0.15            0.27
##      12         2 -3936.52 7942.23            0.01            0.15
##      13         1 -3936.52 7948.51            0.00            0.01
## the present optimal number of knots is  5
## penalty(AIC) was the default: BIC=log(samplesize): log( 539 )= 6.29
```

Dibujamos la evolución del AIC

```
resul<- ajuste$logl
nudos<- resul[,1]
logL<- resul[,3]
AIC<- -2*logL+log(length(datos))*(nudos-1)
plot(nudos,AIC,type="l")
```




```
hist(datos,br=30,
     prob=TRUE,
     main="Histograma y estimac. de la densidad",
     ylab = expression(hat(f)(x)),xlab="x")
lines(density(datos,bw="SJ"),col="blue",lwd=2)
plot(ajuste,col="red",lwd=2,add=TRUE)
legend("topright",col=c("blue","red"),
       lwd=2,legend=c("KERNEL","Logspline"))
```



9.3 Estimar $P[peso > 1600]$ y el cuantil 0.80

```
1-plogspline(1600, ajuste)
```

```
## [1] 0.2408163
```

```
qlogspline(0.8, ajuste)
```

```
## [1] 1662.793
```

Para estimar el núcleo

```
estimnuc<- density(datos,bw="SJ",n=3000)
```

```
#PARA TENER MAS PUNTOS
res1 = cbind(estimnuc$x,cumsum(estimnuc$y)/sum(estimnuc$y))
dim(res1)
```

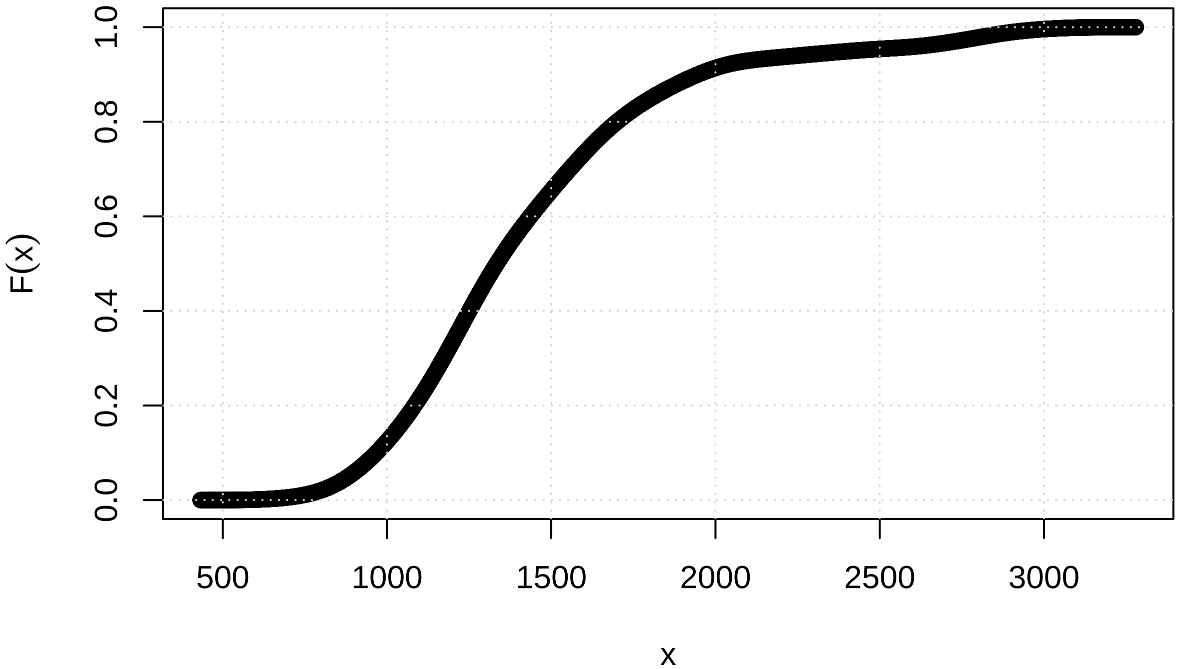
```
## [1] 3000    2
```

```
res1[1:8,]
```

```
##          [,1]         [,2]
## [1,] 431.9356 2.389487e-07
## [2,] 432.8853 4.867507e-07
## [3,] 433.8349 7.436971e-07
## [4,] 434.7846 1.010087e-06
## [5,] 435.7343 1.286237e-06
## [6,] 436.6840 1.572483e-06
## [7,] 437.6337 1.869154e-06
## [8,] 438.5834 2.176583e-06
```

Lo dibujamos:

```
plot(estimnuc$x,cumsum(estimnuc$y)/sum(estimnuc$y),
     xlab="x",
     ylab=expression(hat(F)(x)))
grid()
```




```
cbind(estimnuc$x,cumsum(estimnuc$y)/sum(estimnuc$y))[estimnuc$x>1600,][1,]
```

```
## [1] 1600.0577756    0.7334607
```

```
1-cbind(estimnuc$x,cumsum(estimnuc$y)/sum(estimnuc$y))[estimnuc$x>1600,][1,2]
```

```
## [1] 0.2665393
```

```
#Se puede interpolar:
x1<- rev(estimnuc$x[cumsum(estimnuc$y)/sum(estimnuc$y)<0.8])[1]
x2<-estimnuc$x[cumsum(estimnuc$y)/sum(estimnuc$y)>=0.8][1]
x1
```

```
## [1] 1699.776
```

```
x2
```

```
## [1] 1700.725
```

```
y1<- (cumsum(estimnuc$y)/sum(estimnuc$y))[estimnuc$x==x1]
y2<- (cumsum(estimnuc$y)/sum(estimnuc$y))[estimnuc$x==x2]
x1+((0.8-y1)/(y2-y1))*(x2-x1)
```

```
## [1] 1700.504
```

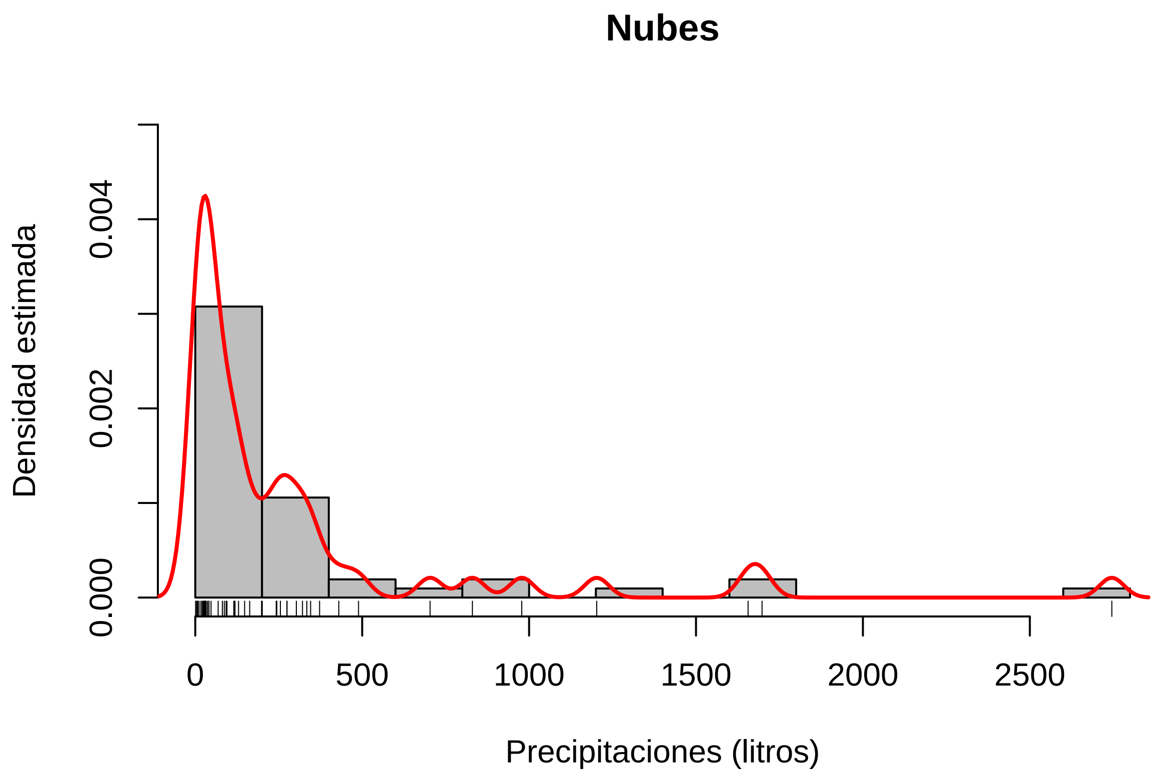
10 Ejercicio 10 fichero "clouds.txt"

El fichero "clouds.txt" contiene las precipitaciones resultantes de 26 nubes sembradas y 26 no sembradas. Trabajando con las precipitaciones de las 52 nubes:

10.1 Estimar la función de densidad con el método del núcleo. ¿Se obtienen estimaciones de la densidad no nulas para precipitaciones negativas?

```
nubes=read.table("datos/clouds.txt",header=TRUE)
x=c(nubes[,1],nubes[,2])
```

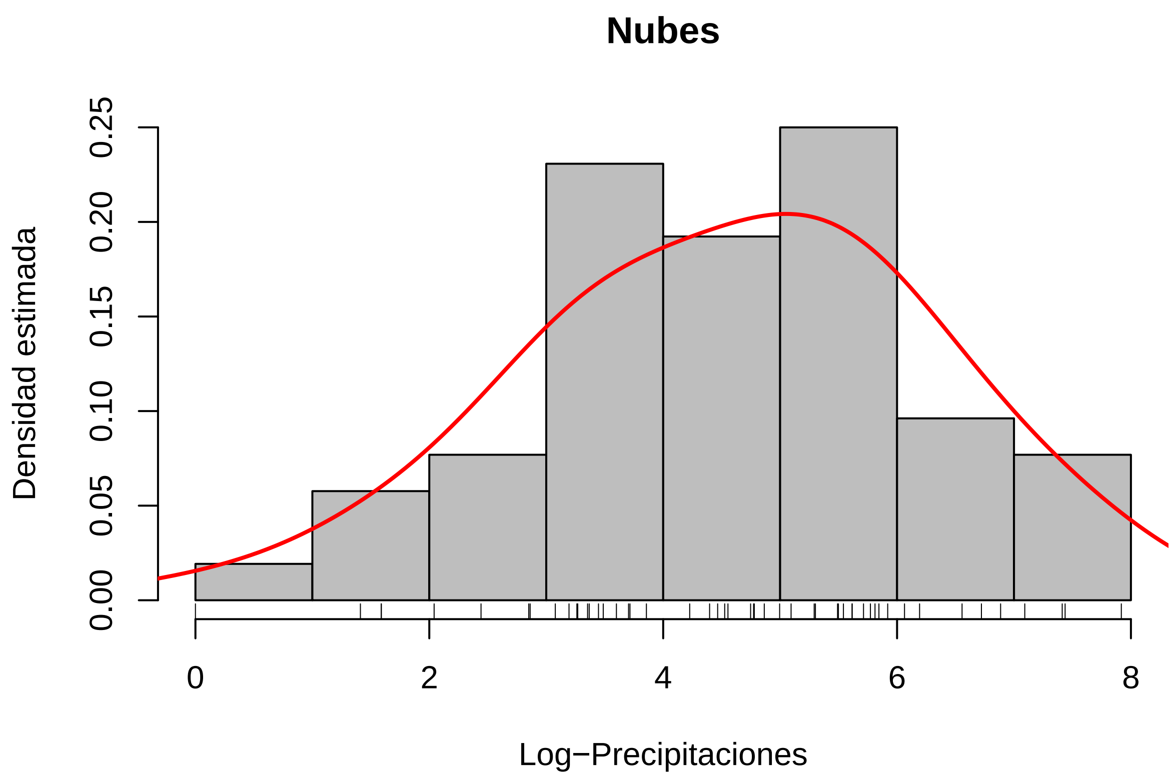
```
hist(x, br=10,prob=TRUE,
     main="Nubes",
     col="gray",
     xlab="Precipitaciones (litros)", ylab="Densidad estimada",
     ylim = c(0,0.005))
lines(density(x, bw="SJ"),lwd=2,col="red")
rug(x)
```



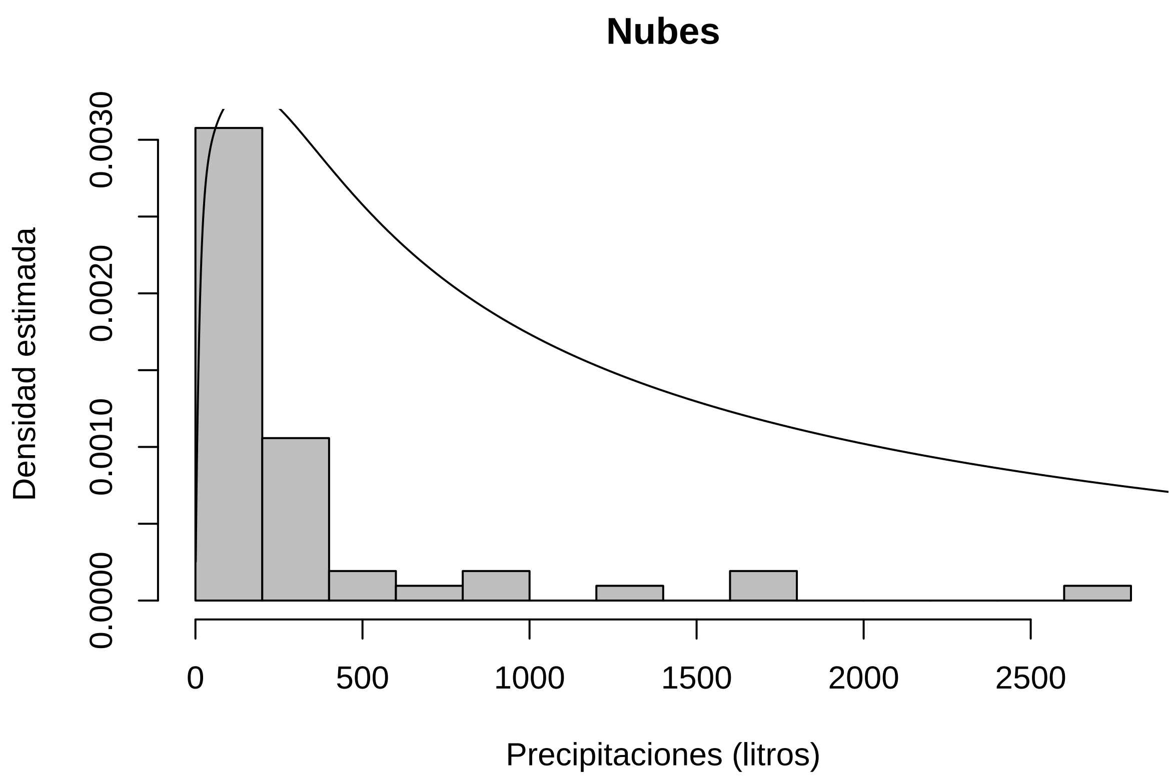
Hay un problema: La densidad es mayor que 0 para valores negativos, por lo que podemos trabajar con LOG.

10.2 Realizar la estimación de la densidad trabajando con el logaritmo de las precipitaciones.

```
y=log(x)
hist(y, br=10,prob=TRUE,
     main="Nubes", col="gray",
     xlab="Log-Precipitaciones", ylab="Densidad estimada")
lines(density(y, bw="SJ"),lwd=2,col="red")
rug(y)
```




```
estimnuc=density(y,bw="SJ",from=0,to=8) #VER ANTERIOR HIST
hist(x, br=10,prob=TRUE,
     main="Nubes", col="gray",
     xlab="Precipitaciones (litros)", ylab="Densidad estimada")
lines(exp(estimnuc$x),estimnuc$y/sum(estimnuc$y))
```



10.3 Estimar $P[precipitaciones > 4]$

```
res1 = cbind(estimnuc$x,cumsum(estimnuc$y)/sum(estimnuc$y))
dim(res1)
```

```
## [1] 512   2
```

```
res1[1:8,]
```

```
##             [,1]         [,2]
## [1,] 0.00000000 0.0002536577
## [2,] 0.01565558 0.0005109994
## [3,] 0.03131115 0.0007720452
## [4,] 0.04696673 0.0010368607
## [5,] 0.06262231 0.0013054773
## [6,] 0.07827789 0.0015779530
## [7,] 0.09393346 0.0018543308
## [8,] 0.10958904 0.0021346607
```

```
cbind(estimnuc$x,cumsum(estimnuc$y)/sum(estimnuc$y))[estimnuc$x>log(4),][1,]
```

```
## [1] 1.39334638 0.04493602
```

```
1-cbind(estimnuc$x,cumsum(estimnuc$y)/sum(estimnuc$y))[estimnuc$x>log(4),][1,2]
```

```
## [1] 0.955064
```