Density Estimation

Bandwidht choice by leave-one-out maximum likelihood

Caballero Vergés Biel, Menzenbach Svenja and Reyes Illescas Kleber Enrique

2023-09-27

Histogram

1. At the slides we have seen the following relationship

$$\hat{f}_{h,(-i)}(x_i) = \frac{n}{n-1}\left(\hat{f}_h(x_i) - \frac{K(0)}{nh}\right)$$

between the leave-one-out kernel density estimator $\hat{f}_{h,(-i)}(x)$ and the kernel density estimator using all the observations $\hat{f}_h(x)$, when both are evaluated at x_i, one of the observed data. Find a similar relationship between the histogram estimator of the density function $\hat{f}_{hist}(x)$ and its leave-one-out version, $\hat{f}_{hist,(-i)}(x)$, when both are evaluated at x_i.

Starting from the formula for the histogram seen in the slides:

$$\hat{f}_{hist}(x) = \sum_{j=1}^{m} \frac{n_j}{n}\frac{1}{b} I_{B_j}(x)$$

And knowing the following equalities for the single point x_i

$$\hat{f}_{hist}(x_i) = \frac{n_j}{n}\frac{1}{b} \qquad \hat{f}_{hist,(-i)}(x_i) = \frac{n_j - 1}{n-1}\frac{1}{b}$$

We can transform the equation on the left to $n_j = nb\hat{f}_{hist}(x_i)$. Then, we can replace this value of n_j into the equation on the left (loo-cv). This give us then following equations:

$$\hat{f}_{hist,(-i)}(x_i) = \frac{nb\hat{f}_{hist}(x_i) - 1}{n-1}\frac{1}{b}$$

$$\hat{f}_{hist,(-i)}(x_i) = \frac{n\hat{f}_{hist}(x_i)b}{(n-1)b} - \frac{1}{(n-1)b}$$

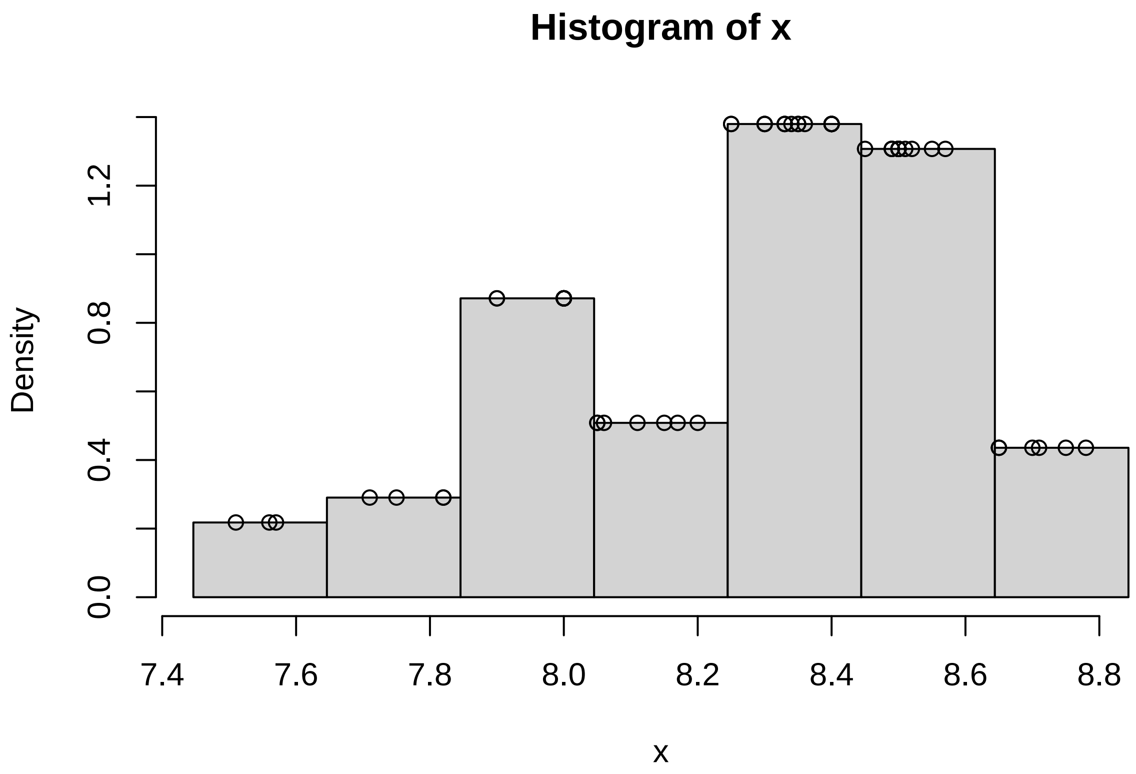
$$\hat{f}_{hist,(-i)}(x_i) = \frac{n}{n-1}\hat{f}_{hist}(x_i) - \frac{1}{(n-1)b}$$

$$\hat{f}_{hist,(-i)}(x_i) = \frac{1}{n-1}\left(n\hat{f}_{hist}(x_i) - \frac{1}{b}\right)$$

2. Read the CD rate data set and call x the first column. Then define A, Z and nbr and plot the histogram of x

```
cdrate.df <-read.table("./cdrate.dat.txt")
x <- cdrate.df[,1]
```

```
A <- min(x)-.05*diff(range(x))
Z <- max(x)+.05*diff(range(x))
nbr <- 7

hx <- hist(x,breaks=seq(A,Z,length=nbr+1),freq=F)
hx_f <- stepfun(hx$breaks,c(0,hx$density,0))
points(x, hx_f(x))
```



3. Use the formula you have found before relating $\hat{f}_{hist}(x_i)$ and $\hat{f}_{hist,(-i)}(x_i)$ to compute $\hat{f}_{hist,(-i)}(x), i = 1, ..., n,$. Then, add the points $(x_i, \hat{f}_{hist,(-i)}(x_i)), i = 1, ..., n,$ to the previous plot.

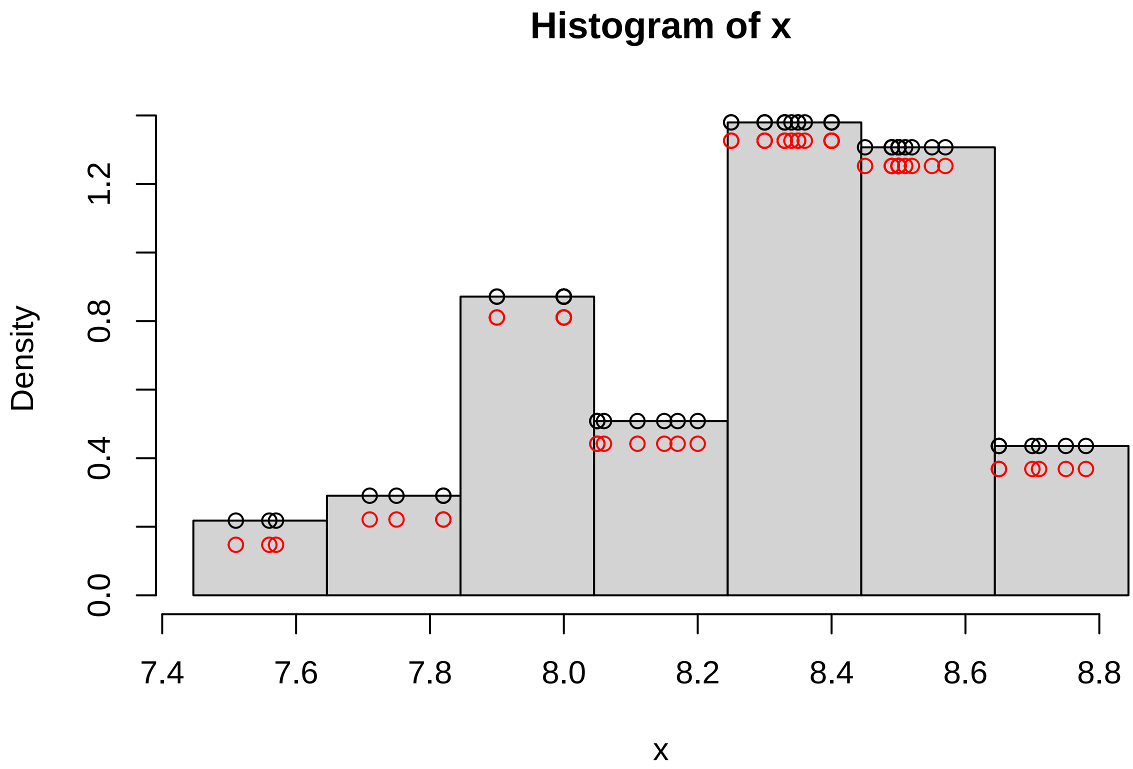
   In the question 2 we have obtained the next formula:

$$\hat{f}_{hist,(-i)}(x_i) = \frac{n}{n-1}\hat{f}_{hist}(x_i) - \frac{1}{(n-1)b}$$

   We can use it to generate new points that we can compare with the previous plot.

```
n <- length(x)
b <- hx$breaks[2]-hx$breaks[1]
hx_f2 <- (n/(n-1)* hx_f(x))- 1/((n-1)*b)
```

```
hx_f2[abs(hx_f2) < 1e-05] <- 0

A <- min(x)-.05*diff(range(x))
Z <- max(x)+.05*diff(range(x))
nbr <- 7

hx <- hist(x,breaks=seq(A,Z,length=nbr+1),freq=F)
hx_f <- stepfun(hx$breaks,c(0,hx$density,0))
points(x, hx_f(x))
points(x, hx_f2, col="red")
```



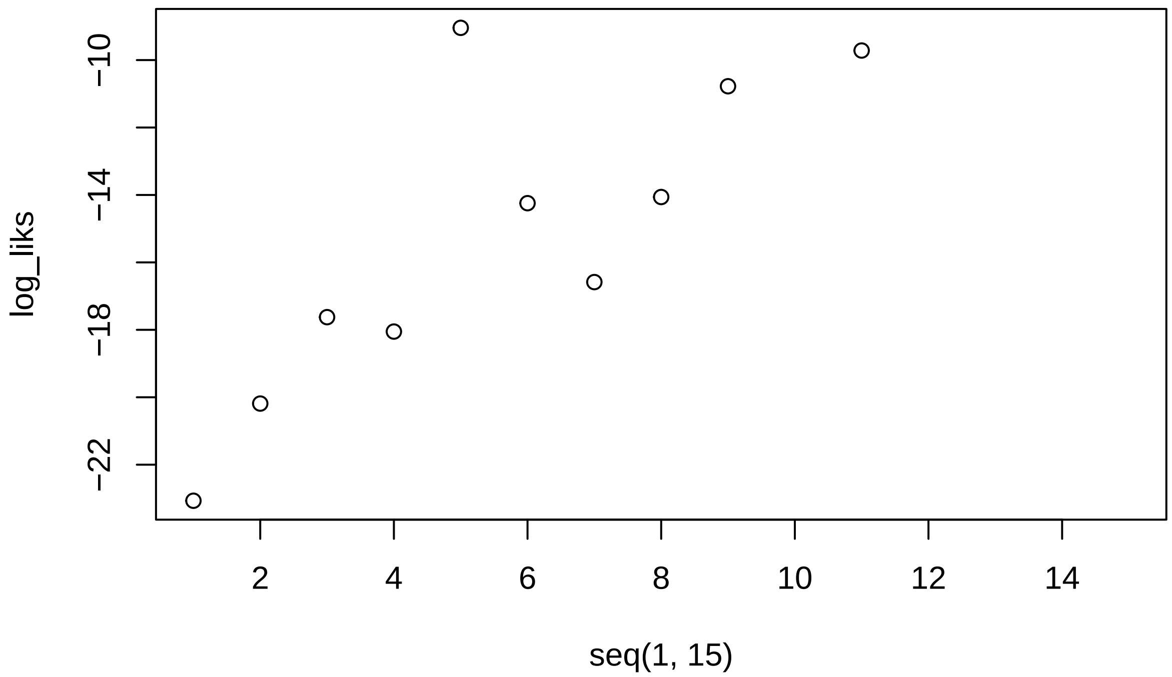
4. Compute the leave-one-out log-likelihood function corresponding to the previous histogram, at which *nbr=7* has been used

```
if (any(hx_f2 == 0)) {
  looCV_log_lik_7 <- -Inf
} else {
  # Calculate the log-likelihood if hx_f2 is not zero
  looCV_log_lik_7 <- sum(log(hx_f2))
}
looCV_log_lik_7
```

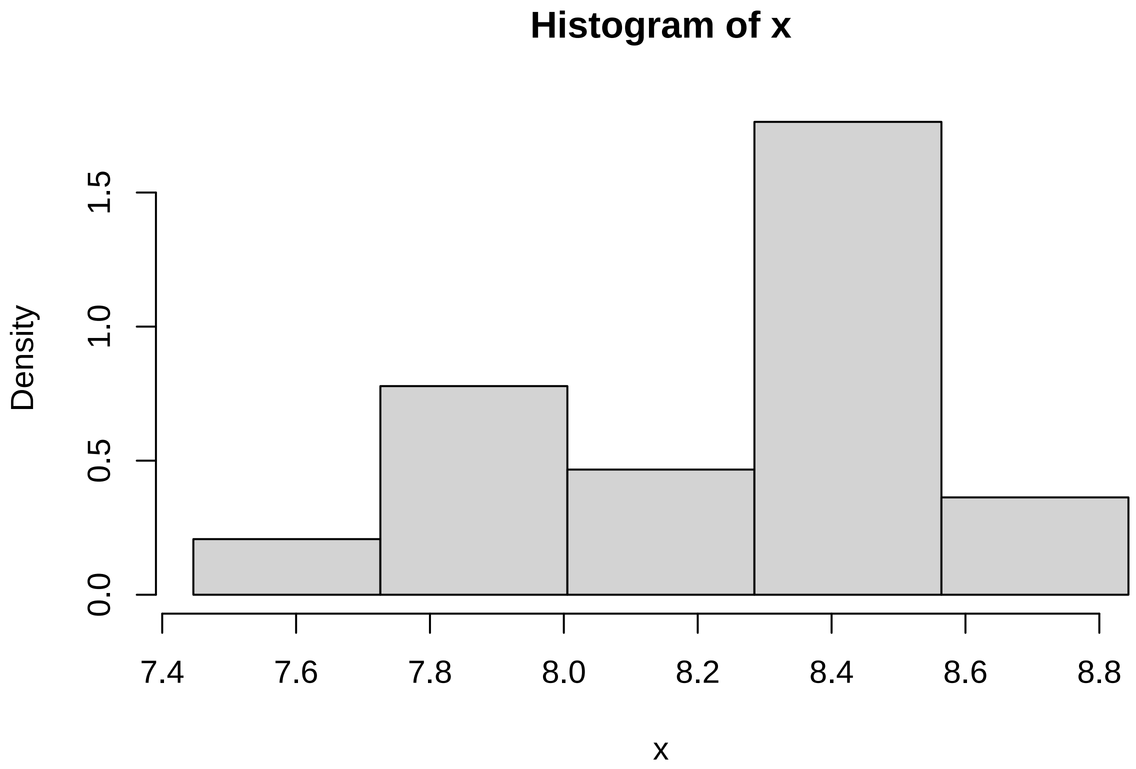
```
## [1] -16.58432
```

5. **Choosing *nbr* by leave-one-out Cross Validation (looCV)**. Consider now the set *seq(1,15)* as possible values for *nbr*, the number of intervals of the histogram. For each of them compute the leave-one-out log-likelihood function (*looCv_log_lik*) for the corresponding histogram. Then plot the values of *looCv_log_lik* against the values of *nbr* and select the optimal value of *nbr* as that at which *looCv_log_lik* takes its maximum. Finally, plot the histogram of x using the optimal value of *nbr*

```r
#sum of the product of the hx_f2 vector plot histograms for different number of breaks nbr

log_liks = list()

A <- min(x)-.05*diff(range(x))
Z <- max(x)+.05*diff(range(x))
n <- length(x)

for (nbr in c(1:15)){
  hx_i <- hist(x,breaks=seq(A,Z,length=nbr+1), plot = FALSE)
  hx_f_i <- stepfun(hx_i$breaks,c(0,hx_i$density,0))
  b <- hx_i$breaks[2]-hx_i$breaks[1]
  hx_f2_i <- (n/(n-1)* hx_f_i(x)) - 1/((n-1)*b)
  hx_f2_i[abs(hx_f2_i) < 1e-05] <- 0

  if (any(hx_f2_i == 0)) {
    looCV_log_lik <- -Inf
  } else {
  # Calculate the log-likelihood if hx_f2 is not zero
    looCV_log_lik <- sum(log(hx_f2_i))
  }
  #print(paste("nbr", nbr,"->", looCV_log_lik))
  log_liks <- append(log_liks, looCV_log_lik)
}

plot(seq(1, 15), log_liks, type = "p")
```



```r
nbr_opt <- which.max(log_liks)
hist(x,breaks=seq(A,Z,length=nbr_opt+1),freq=F)
```

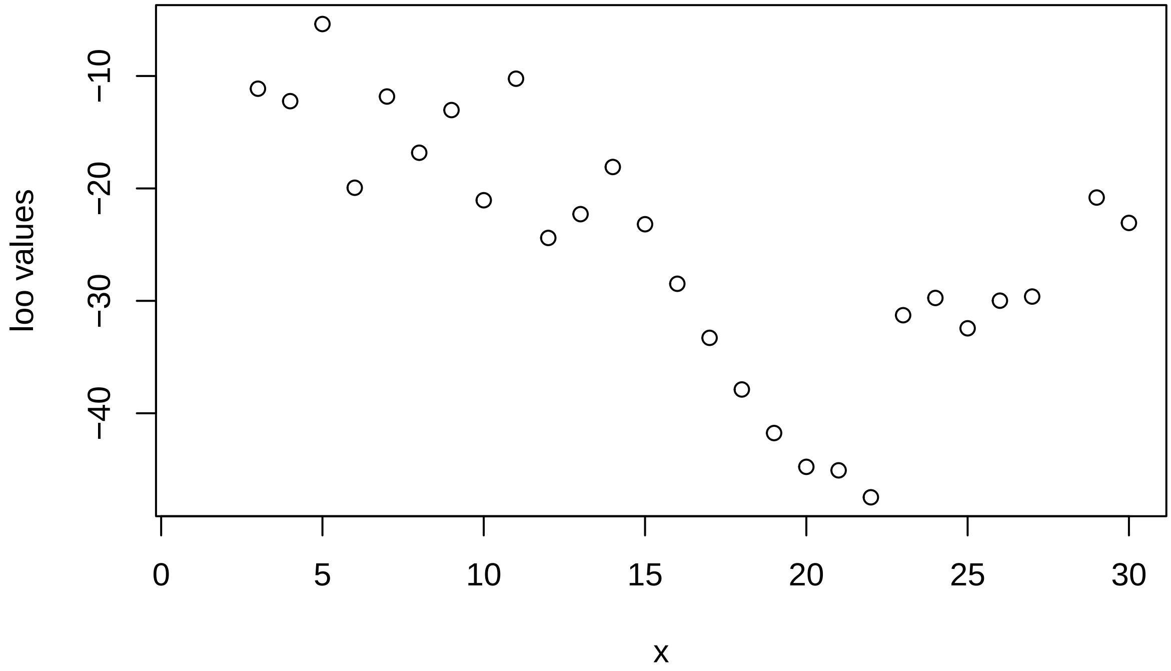


In the range from 1 to 15, the leave-one-out log-likelihood achieves its highest value at nbr= 5 (We should mention that nbr=11 has also a nice score). The computed value is *-9.044528*, which is greather than the *-16.58432* previously calculated in exercise 4 with a fixed nbr=7.

6. **Chossing b by looCV**. Let b be the common width of the bins of a histogram. Consider the set $seq((Z - A)/15, (Z - A)/1, length = 30)$ as possible values for b. Select the value of b maximizing the leave-one-out log-likelihood function, and plot the corresponding histogram

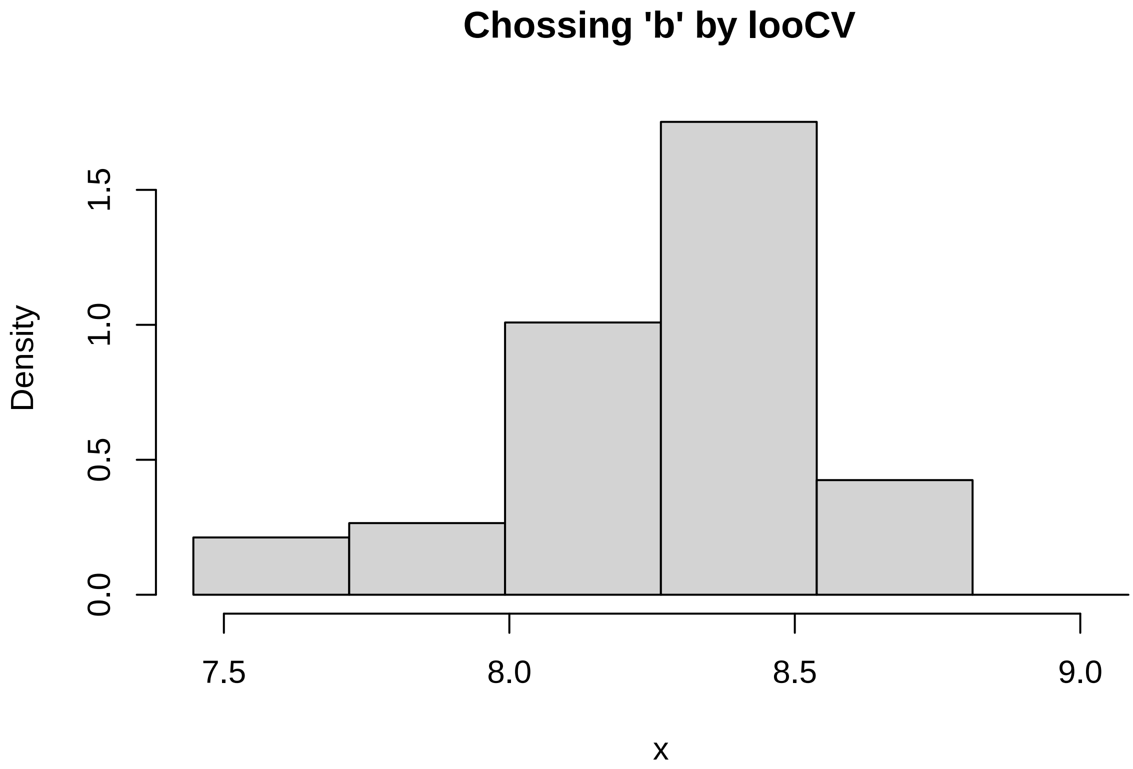
```r
b_set <- seq((Z-A)/15, (Z-A)/1, length=30)

b_max_log_lik <- function(b_set, x){

  log_liks_b = list()
  n <- length(x)

  for(b in b_set){
    hx <- hist(x, breaks=seq(A, Z+b, by=b), plot=F)

    # compute histogram estimator
    hx_f <- stepfun(hx$breaks,c(0,hx$density,0))
    hx_f2 <- (n/(n-1)* hx_f(x))- 1/((n-1)*b)
    hx_f2[abs(hx_f2) < 1e-05] <- 0

    if (any(hx_f2 == 0)) {
      looCV_log_lik_b <- -Inf
    } else {
```

```
    # Calculate the log-likelihood if hx_f2 is not zero
      looCV_log_lik_b <- sum(log(hx_f2))
    }
    #looCV_log_lik_b <- sum(log(hx_f2))
    log_liks_b <- append(log_liks_b, looCV_log_lik_b)

  }

  plot(x=seq(1,30),log_liks_b, type = "p", xlab="x", ylab="loo values")
  b_opt <- which.max(log_liks_b)

  b <- b_set[b_opt]
  print(b)
  return(b)
}

b <- b_max_log_lik(b_set, x)
```



```
## [1] 0.272977
```

```
hx <- hist(x, breaks=seq(A, Z+b, by=b), plot=F)
plot(hx, freq = FALSE, main="Chossing 'b' by looCV")
```

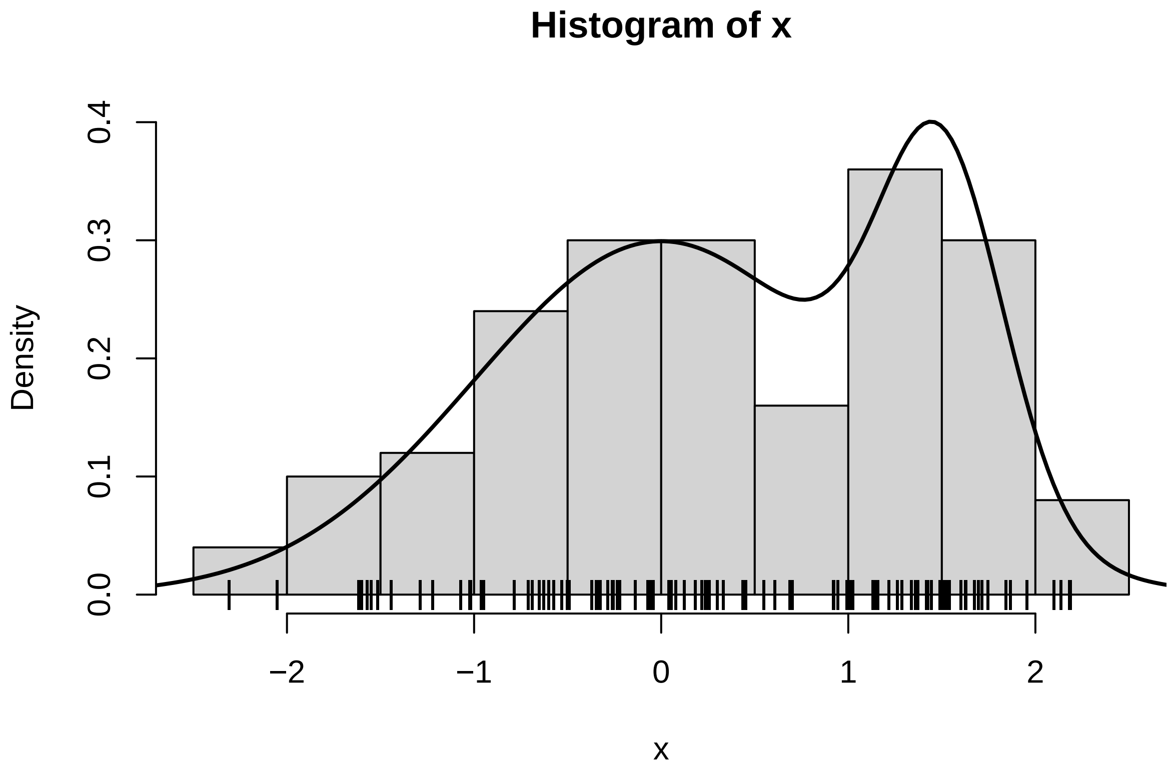


In this case we have a similar shape than in the previous exercise. Therefore, in practise, estimating the optimal binwidth b is equivalent to estimate the number of intervals *nbr*.

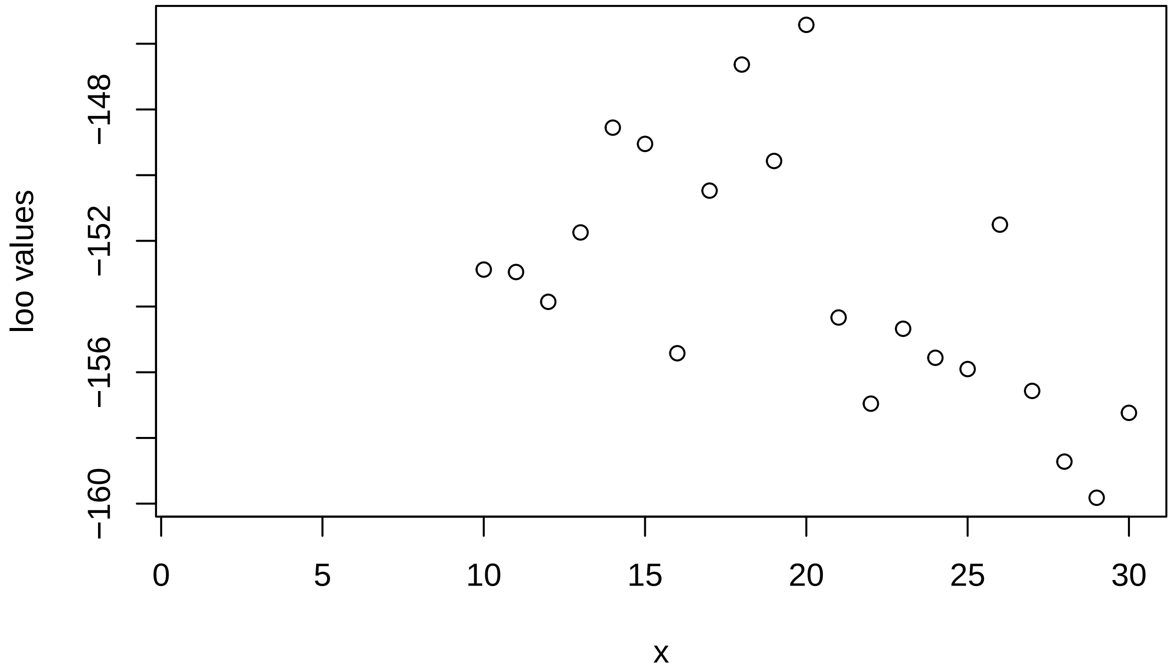
7. Recycle the functions *graph.mixt* and *sim.mixt* defined at *density_estimation.Rmd* to generate n = 100 data from
$$f(x) = (3/4)N(x; m = 0, s = 1) + (1/4)N(x; m = 3/2, s = 1/3)$$
   Let b be the bin width of a histogram estimator of f(x) using the generated data. Select the value of b maximizing the leave-one-out log-likelihood function, and plot the corresponding histogram. Compare with the results obtained using the Scott's formula:
$$b_{Scott} = 3.49 St.Dev(X)_n^{-1/3}$$
   .

```
# graph.mixt
# Input:
#    k: number mixture components
#    mu: vector of length k with the mean values of the k normals
#    sigma: vector of length k with the st.dev. values of the k normals
#    alpha: vector of length k with the weights of each normal
#    graphic: logical value indicating if the mixture density must be plotted
#    ...: Other parameters passed to plot()
#
# Output:
#    L, U: extremes of the interval where the mixture density is plotted
#    x: points at which the mixture density is evaluated
#    fx: value of the mixture density at x
#
graph.mixt<-
function(k=1, mu=seq(-2*(k-1),2*(k-1),length=k), sigma=seq(1,1,length=k), alpha=seq(1/k,1/k,length=k), g
{
   L<-min(mu-3*sigma)
   U<-max(mu+3*sigma)

   x<- seq(from=L,to=U,length=200)
   fx<- 0*x
   Salpha<-sum(alpha)
   for(i in 1:k){
    p<-alpha[i]/Salpha
#       fx <- fx + p*exp(-.5*((x-mu[i])/sigma[i])^2)/(sqrt(2*pi)*sigma[i])
    fx <- fx + p*dnorm(x,mu[i],sigma[i])
   }
   if (graphic){
      plot(x,fx,type="l",...)
   }
   return(list(L = L, U = U, x = x, fx = fx))
}

# sim.mixt
# Input:
#    n: number of simulated data
#    k: number mixture components
#    mu: vector of length k with the mean values of the k normals
#    sigma: vector of length k with the st.dev. values of the k normals
#    alpha: vector of length k with the weights of each normal
#    graphic: logical value indicating if the mixture density and the
#              histogram of the simulated data must be plotted
#    ...: Other parameters passed to plot()
#
# Output:
#    x: simulated data
#
# Requires:
#    graph.mixt
sim.mixt <- function(n=1,k=1,
         mu=seq(-2*(k-1),2*(k-1),length=k),
         sigma=seq(1,1,length=k),
         alpha=seq(1/k,1/k,length=k), graphic=FALSE,...)
{
   csa<-cumsum(alpha)
   x<-runif(n)

   for (i in 1:n){
      comp<-sum(csa<=x[i])+1
      x[i]<-rnorm(1,mu[comp],sigma[comp])
   }
   if(graphic) {
      out<-graph.mixt(k, mu, sigma, alpha, gr=FALSE)
      hist(x,freq = FALSE,
           ylim=c(0,max(c(max(out$fx),max(hist(x,plot=FALSE)$density)))))
      lines(out$x,out$fx,lty=1,lwd=2)
   }
   return(x)
}
```

```
set.seed(123)
n <- 100
mu <- c(0,3/2)
sigma <- c(1,1/3)
alpha <- c(3/4,1/4)
x <- sim.mixt(n=n, k=2, mu=mu, sigma=sigma, alpha=alpha, gr=T)
points(x,0*x,pch="|")
```

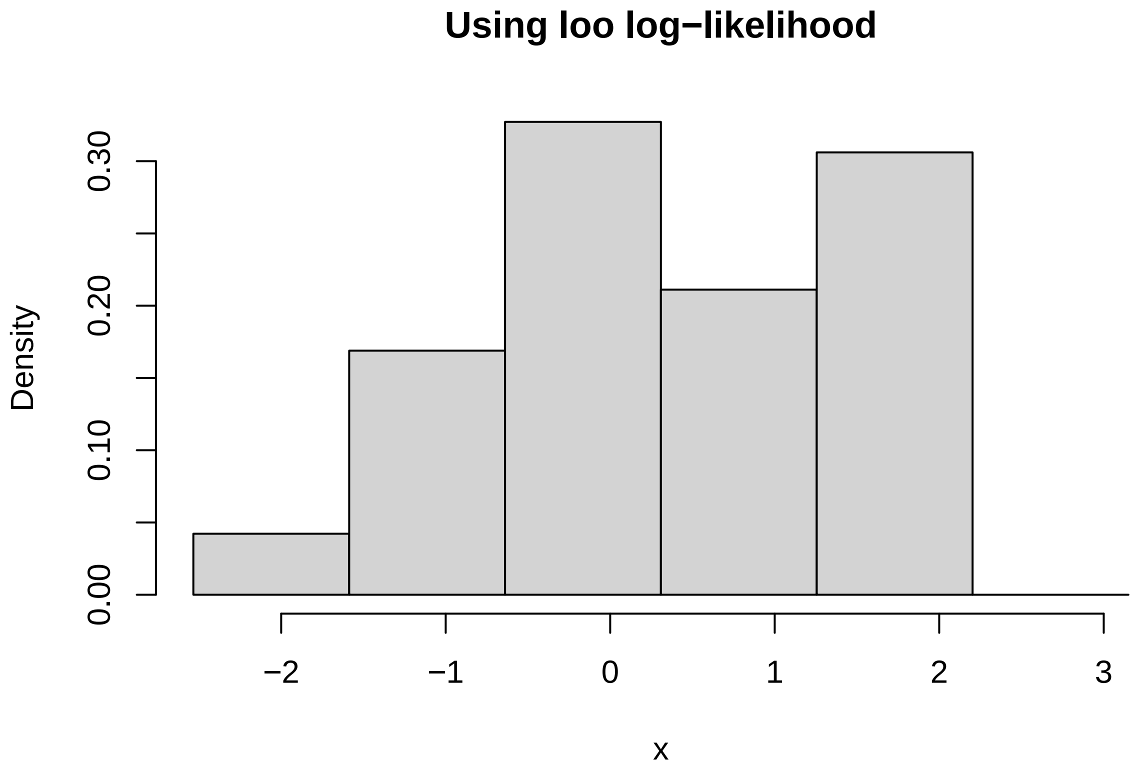


```
b_scott <- function(X){
  return(3.49 * sd(X) * length(X)**(-1/3))
}

A <- min(x)-.05*diff(range(x))
Z <- max(x)+.05*diff(range(x))

b <- b_max_log_lik(b_set, x)
```

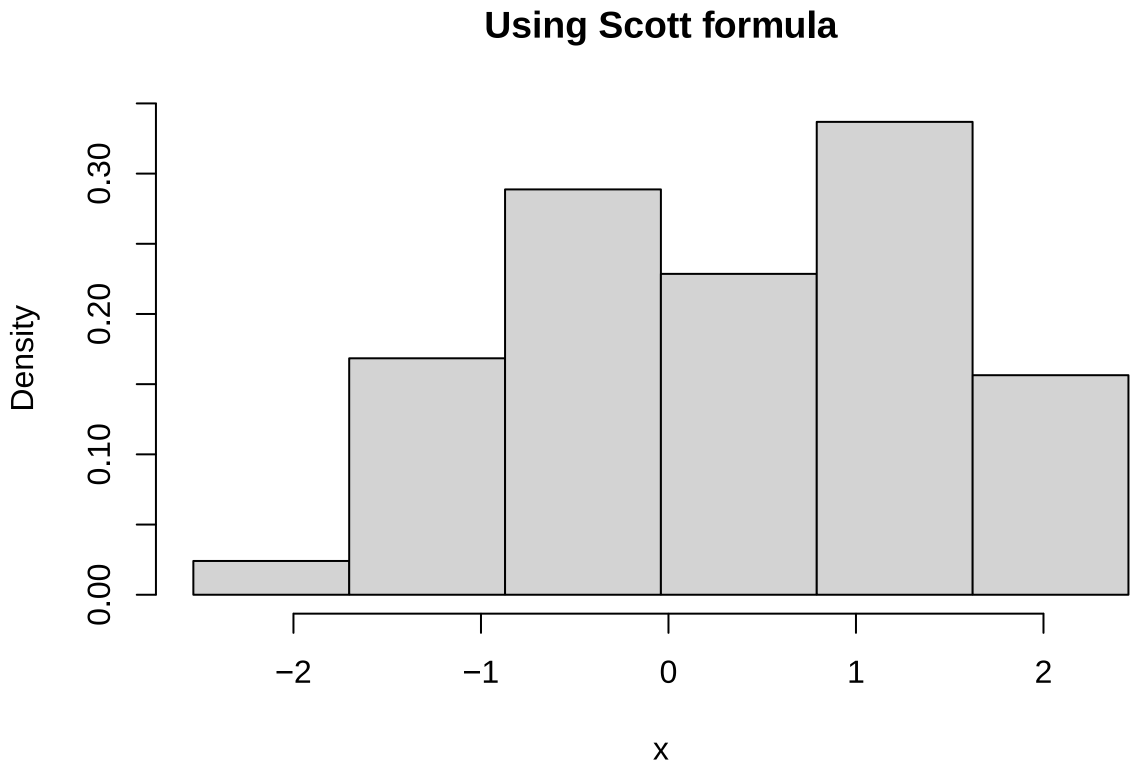


```
## [1] 0.9473908
```

```
hx <- hist(x, breaks=seq(A, Z+b, by=b), plot=F)
plot(hx, freq = FALSE, main="Using loo log-likelihood")
```



```
b_s <- b_scott(x)
hx_s <- hist(x, breaks=seq(A, Z+b_s, by=b_s), plot=F)
plot(hx_s, freq = FALSE, main="Using Scott formula")
```



While the shape of the original dataset is visible in both histograms, the bin width generated by the LOO histogram looks more similar than the Scoot histogram. We can conclude that the first one could give us a better estimation for the bin width.

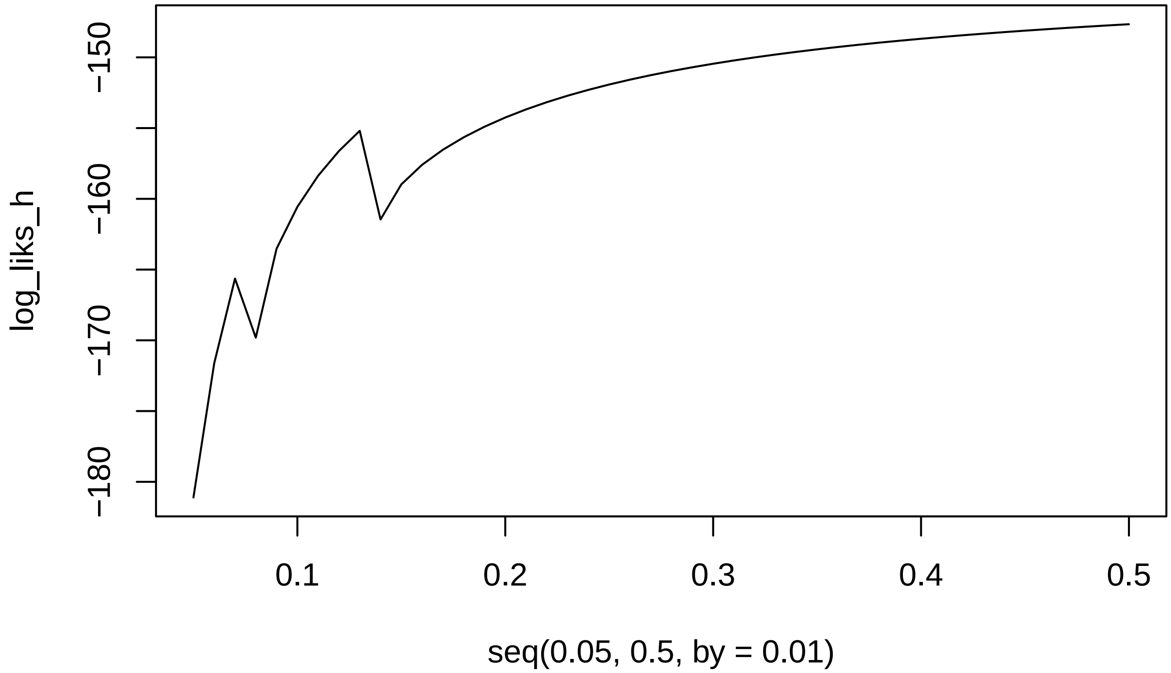
Kernel density estimator

8.Consider the vector x of data you have generated before from the mixture of two normals. Use the relationship

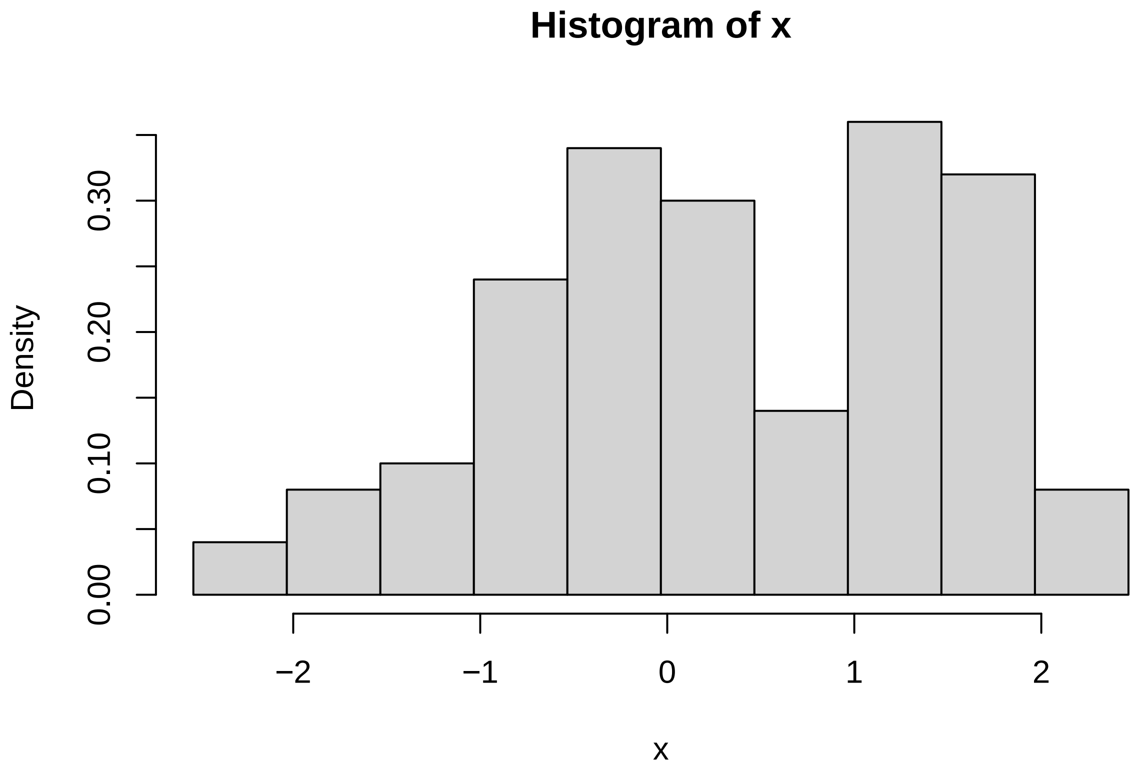
$$\hat{f}_{h,(-i)}(x_i) = \frac{n}{n-1}\left(\hat{f}_h(x_i) - \frac{K(0)}{nh}\right)$$

to select the value of **h** maximizing the leave-one-out log-likelihood function, and plot the corresponding kernel density estimator. NOTE: The following sentences converts the kernel density estimator obtained with the function *density* into a function that can be evaluated at any point of R or at a vector of real numbers:

```
kx <- density(x)
kx_f <- approxfun(x=kx$x, y=kx$y, method = 'linear', rule=2)

n <- length(x)
h_set <- seq(0.05,0.5, by=0.01)

log_liks_h = list()

for (h in h_set){
  kx_f2 <- (n/(n-1)) * (kx_f(x)- ((1/sqrt(2*pi))/(n*h))) # K(0) = 1/sqrt(2*pi)
  #kx_f2[abs(kx_f2) < 1e-05] <- 0

  looCV_log_lik_h <- sum(log(kx_f2[kx_f2>0]))
  log_liks_h <- append(log_liks_h, looCV_log_lik_h)
}

# plot kernel density estimator
plot(seq(0.05,0.5, by=0.01), log_liks_h, type = "l")
```



```
h <- h_set[which.max(log_liks_h)]

hx_s <-  hist(x, breaks=seq(A, Z+h, by=h), plot=F)
plot(hx_s, freq = FALSE)
```


Histogram of x
Density
0.00
0.10
0.20
0.30
−2
−1
0
1
2
x