Density Estimation

Bandwidth Choice by Leave-one-out Maximum Likelihood

Azzarito Domenico, Daniel Reverter, Alexis Vendrix

24 September, 2025

Histogram

1. In the slides we have seen the following relationship (b_Density_estimation p.24)

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

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.

We want to find a similar relationship between the histogram estimator of the density function $\hat{f}_{\text{hist}}(x)$ and its leave-one-out version, $\hat{f}_{\text{hist},(-i)}(x)$, when both are evaluated at x_i.

Let the histogram be defined by bins B_j of common width b. The density estimator using all n observations is given by:

$$\hat{f}_{\text{H}}(x_i) = \frac{N_{j(x)}}{nb}$$

where $N_{j(x)}$ is the count of data points in bin $B_{j(x)}$.

The leave-one-out estimator, built using $n-1$ data points, evaluates at x_i as:

$$\hat{f}_{\text{H},(-i)}(x_i) = \frac{N_{j(x)} - 1}{(n-1)b}$$

By substituting $N_{j(x)} = nb \cdot \hat{f}_{\text{hist}}(x_i)$, we derive the relationship:

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

This is the desired relationship between the full histogram estimator and the leave-one-out version, both evaluated at an observation x_i.

---

2. Application to CD Rate Data

Read the CD rate data set and call x the first column.

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

Then define

$$A < -min(x) - 0.05 * diff(range(x))Z < -max(x) + 0.05 * diff(range(x))nbr < -7$$

```
# Define the range for the histogram
A <- min(x) - 0.05 * diff(range(x))
Z <- max(x) + 0.05 * diff(range(x))
nbr <- 7

cat("A =", A, "\n")
```

```
## A = 7.4465
```

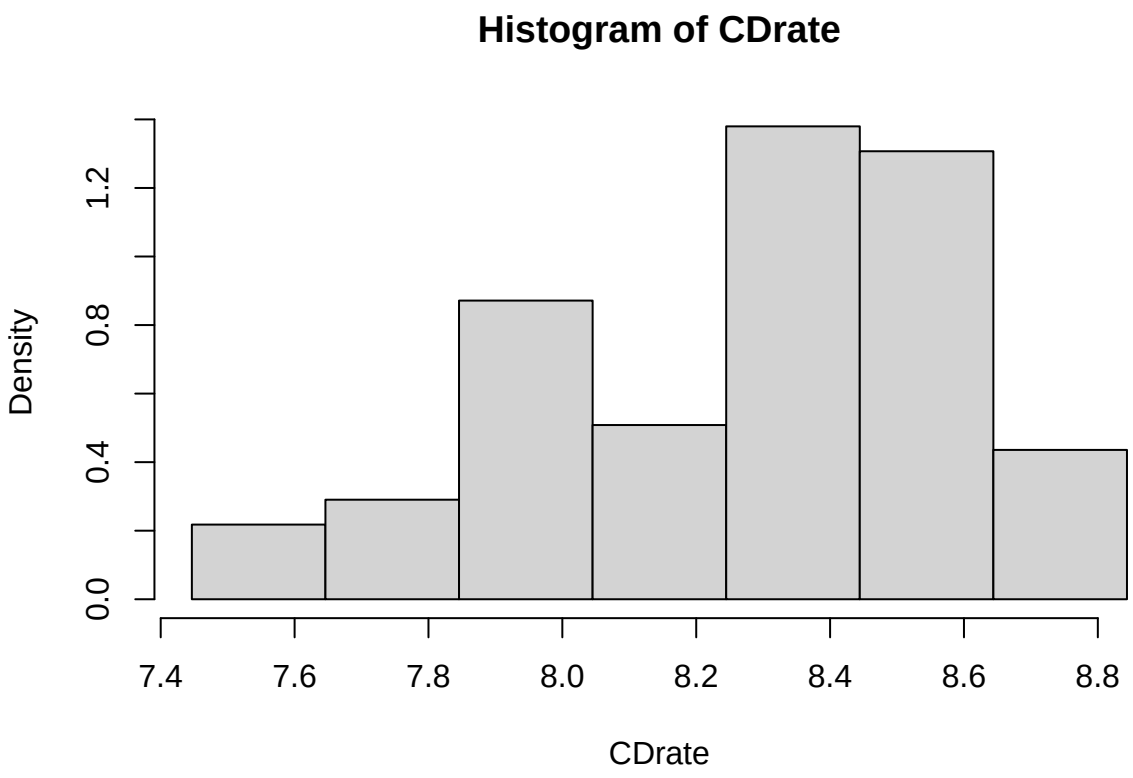
```
cat("Z =", Z, "\n")
```

```
## Z = 8.8435
```

and plot the histogram of x as

$$hx < -hist(x, breaks = seq(A, Z, length = nbr + 1), freq = F)$$

```
# Plot the histogram
hx <- hist(x, breaks = seq(A, Z, length = nbr + 1), freq = FALSE,
           main = "Histogram of CDrate", xlab = "CDrate")
```

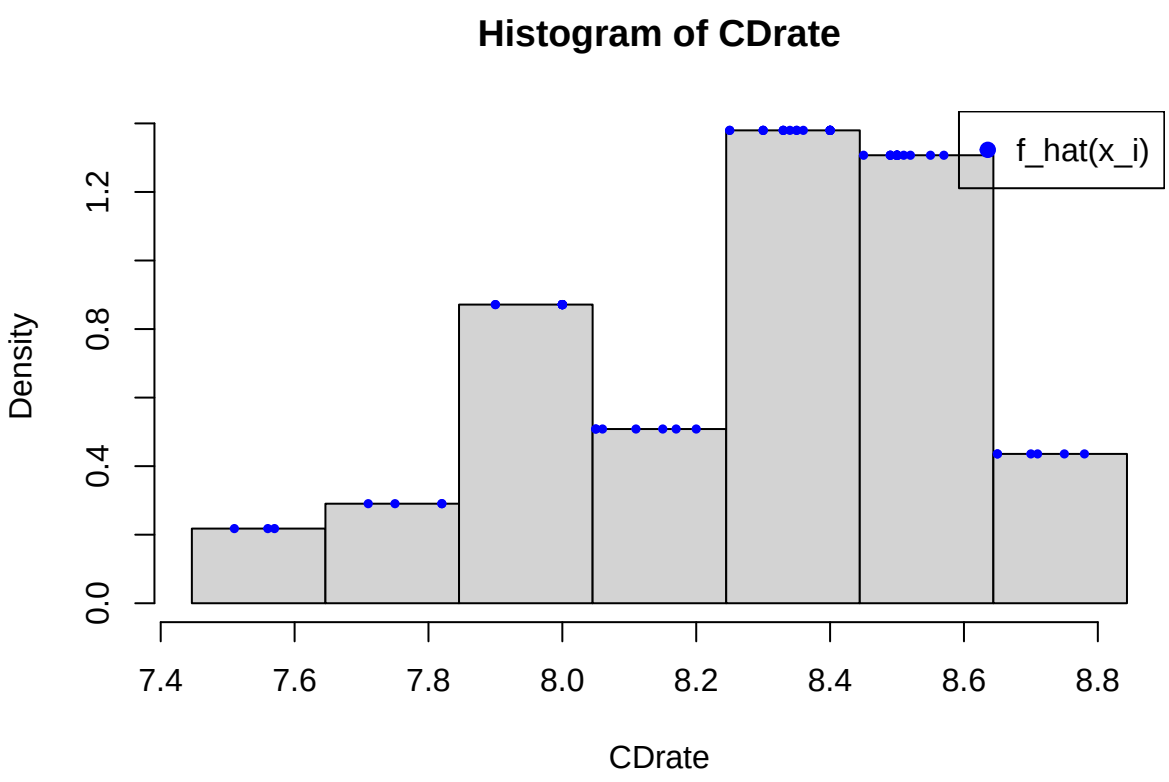


The following sentence converts this histogram into a function that can be evaluated at any point of $\mathbb{R}$, or at a vector of real numbers:

```
hx_f <- stepfun(hx$breaks, c(0, hx$density, 0))
```

Use `hx_f` to evaluate the histogram at the vector of observed data x. Then add the points $(x_i, \hat{f}_{\text{hist}}(x_i))$, $i = 1, \ldots, n$, to the histogram you have plotted before.

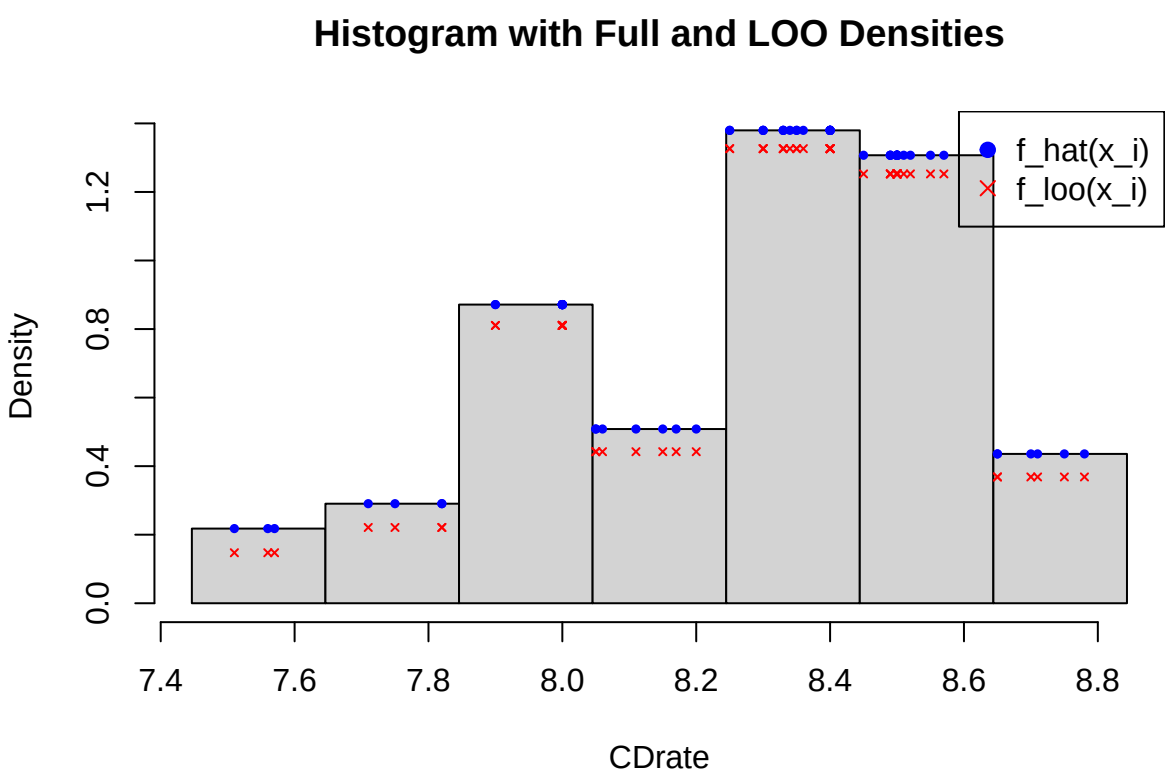
```
# Evaluate the histogram estimator at each data point
f_hat <- hx_f(x)

# Add the points to the plot
plot(hx, freq = FALSE, main = "Histogram of CDrate", xlab = "CDrate")
points(x, f_hat, col = "blue", pch = 19, cex = 0.5)
legend("topright", "f_hat(x_i)", col = "blue", pch = 19)
```



3.

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

```r
# Calculate bin width
b <- (Z - A) / nbr
n <- length(x)

# Calculate the leave-one-out estimates
f_loo <- (n / (n - 1)) * f_hat - 1 / ((n - 1) * b)

# Add the points to the plot
plot(hx, freq = FALSE, main = "Histogram with Full and LOO Densities", xlab = "CDrate")
points(x, f_hat, col = "blue", pch = 19, cex = 0.5)
points(x, f_loo, col = "red", pch = 4, cex = 0.5)
legend("topright", c("f_hat(x_i)", "f_loo(x_i)"), col = c("blue", "red"), pch = c(19, 4))
```



The blue Dots show the density estimate for each data point using the full dataset. As you can see, all dots within the same bin are at the same height—the top of that bin's bar.

The red 'X' show the density estimate for each data point if that point had been excluded from the calculation.

The red 'X's are always slightly lower than the blue dots. Because when we "leave out" a data point x_i, the count of points in its bin (N_k) decreases by one. Since the density is calculated as (count / total), reducing the count naturally leads to a lower density estimate for that bin.

4. Histogram score for 7 bins

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

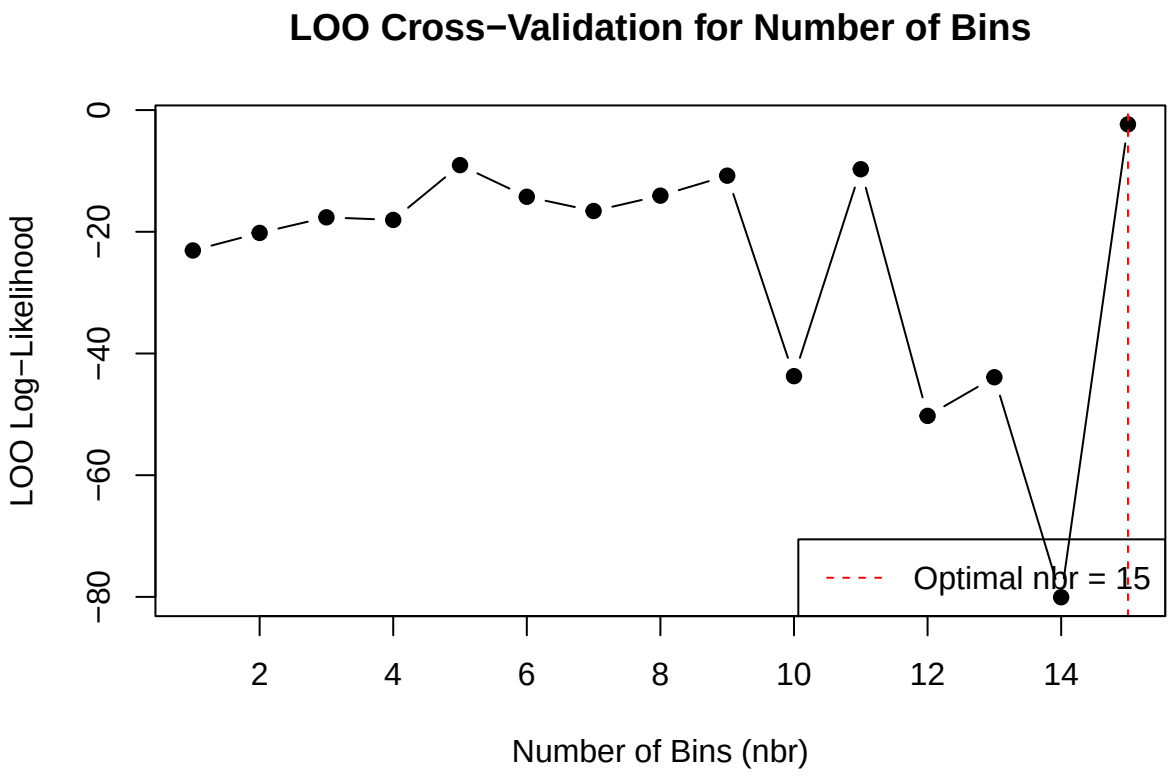
```
# We only take the log of positive values. If f_loo is 0, log(f_loo) is -Inf.
# This happens when a point is the only one in its bin.
looCV_log_lik_7 <- sum(log(f_loo[f_loo > 0]))
cat("Leave-one-out log-likelihood for nbr=7:", looCV_log_lik_7)
```

```
## Leave-one-out log-likelihood for nbr=7: -16.58432
```

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.

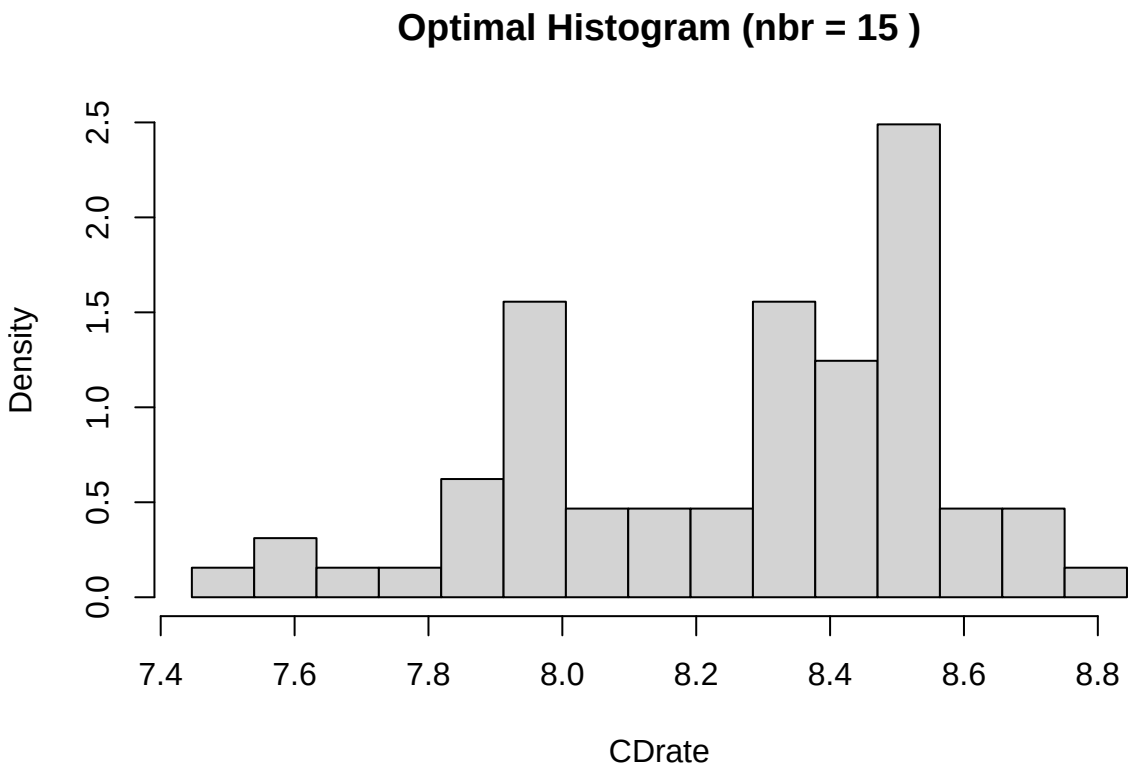
```r
n <- length(x)
nbr_values <- 1:15
looCV_log_lik_nbr <- numeric(length(nbr_values))

for (i in seq_along(nbr_values)) {
  current_nbr <- nbr_values[i]
  b <- (Z - A) / current_nbr

  # Create histogram object
  hx <- hist(x, breaks = seq(A, Z, length = current_nbr + 1), plot = FALSE)

  # Create step function
  hx_f <- stepfun(hx$breaks, c(0, hx$density, 0))

  # Calculate f_hat and f_loo
  f_hat <- hx_f(x)
  f_loo <- (n / (n - 1)) * f_hat - 1 / ((n - 1) * b)

  # Calculate and store looCV log-likelihood
  looCV_log_lik_nbr[i] <- sum(log(f_loo[f_loo > 0]))
}

# Plot the results
plot(nbr_values, looCV_log_lik_nbr, type = "b", pch = 19,
     xlab = "Number of Bins (nbr)", ylab = "LOO Log-Likelihood",
     main = "LOO Cross-Validation for Number of Bins")

# Find the optimal nbr
optimal_nbr <- nbr_values[which.max(looCV_log_lik_nbr)]
abline(v = optimal_nbr, col = "red", lty = 2)
legend("bottomright", legend = paste("Optimal nbr =", optimal_nbr), col = "red", lty = 2)
```



The plot shows the optimal value of nbr as that at which looCV_log_lik takes its maximum is nbr = 15, which is therefore considered the optimal choice for this dataset. One could argue that the Log-likelihood steadily increases from nbr=1 to nbr=9, and then from nbr=10 onward the Log-likelihood changes erratically, probably due to the high variance nature of choosing a high bandwidth, making nbr=15 the maximum due to randomness. It could be wise to choose nbr=9 as a good tradeoff, but basing our choice purely on the maximization of the Log-likelihood we would have to choose nbr=15.

Finally, plot the histogram of x using the optimal value of `nbr`.

```
hist(x, breaks = seq(A, Z, length = optimal_nbr + 1), freq = FALSE,
     main = paste("Optimal Histogram (nbr =", optimal_nbr, ")"),
     xlab = "CDrate")
```



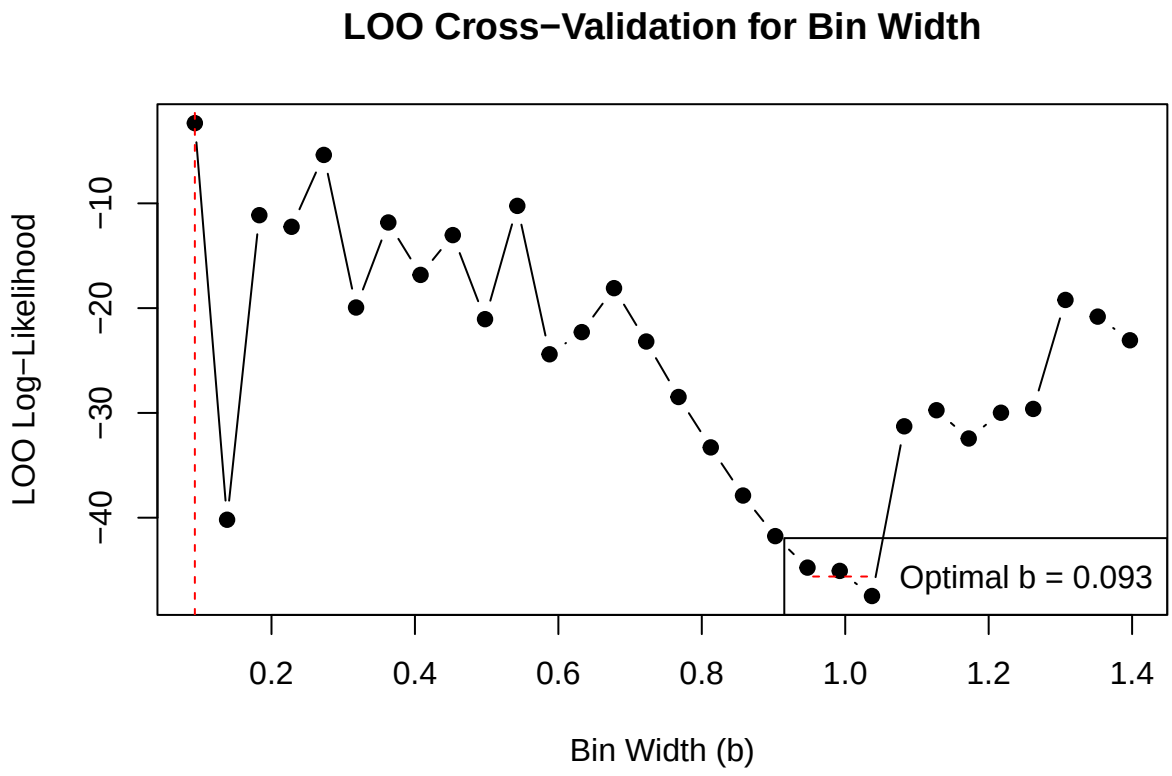
---

6. Choosing `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.

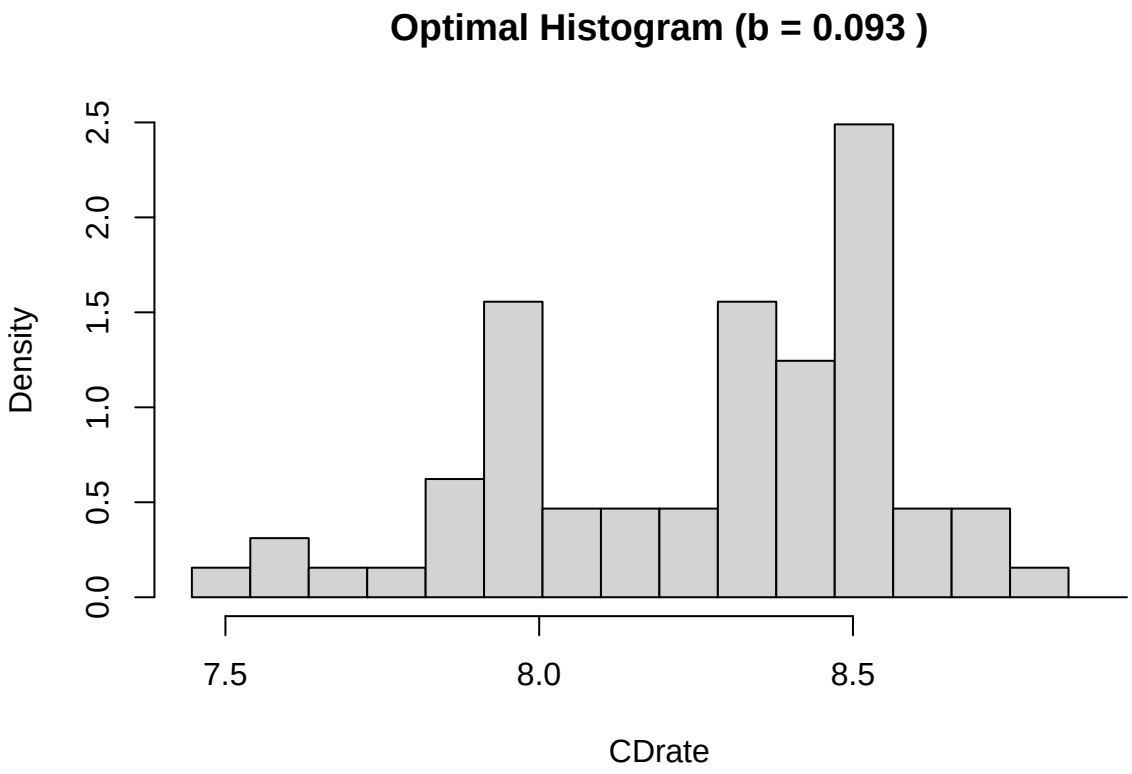
```
b_values <- seq((Z - A) / 15, (Z - A) / 1, length = 30)
looCV_log_lik_b <- numeric(length(b_values))

for (i in seq_along(b_values)) {
  current_b <- b_values[i]

  # Create histogram object with specified bin width
  hx <- hist(x, breaks = seq(A, Z + current_b, by = current_b), plot = FALSE)

  # Create step function
  hx_f <- stepfun(hx$breaks, c(0, hx$density, 0))

  # Calculate f_hat and f_loo
  f_hat <- hx_f(x)
  f_loo <- (n / (n - 1)) * f_hat - 1 / ((n - 1) * current_b)

  # Calculate and store looCV log-likelihood
```

```
  looCV_log_lik_b[i] <- sum(log(f_loo[f_loo > 0]))
}

# Plot the results
plot(b_values, looCV_log_lik_b, type = "b", pch = 19,
     xlab = "Bin Width (b)", ylab = "LOO Log-Likelihood",
     main = "LOO Cross-Validation for Bin Width")

# Find the optimal b
optimal_b <- b_values[which.max(looCV_log_lik_b)]
abline(v = optimal_b, col = "red", lty = 2)
legend("bottomright", legend = paste("Optimal b =", round(optimal_b, 3)), col = "red", lty = 2)
```



Plot the corresponding histogram.

```
hx_optimal_b <- hist(x, breaks = seq(A, Z + optimal_b, by = optimal_b), plot = FALSE)
plot(hx_optimal_b, freq = FALSE,
     main = paste("Optimal Histogram (b =", round(optimal_b, 3), ")"),
     xlab = "CDrate")
```



---

7. Mixture of Normals Example

Recycle the functions `graph.mixt` and `sim.mixt` to generate $n = 100$ data from

$$f(x) = (3/4)N(x; m = 0, s = 1) + (1/4)N(x; m = 3/2, s = 1/3)$$

```
# 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),
```

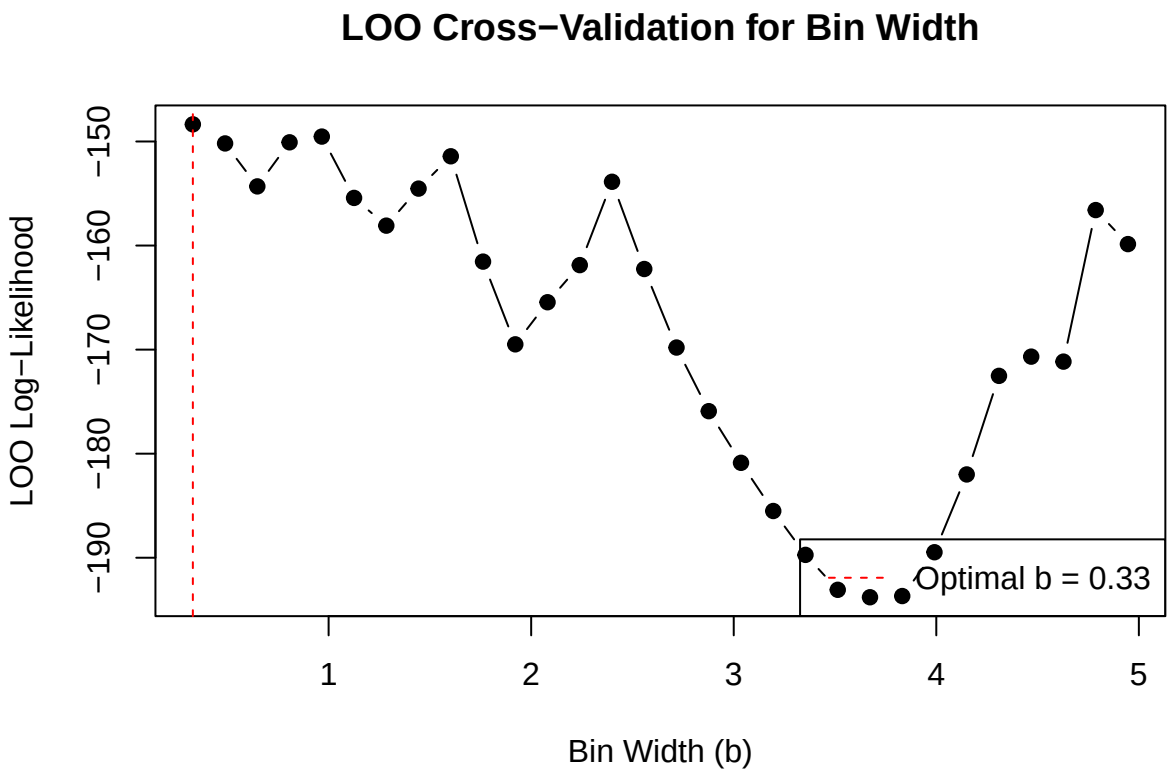
```
{
   L<-min(mu-3*sigma)
   U<-max(mu+3*sigma)

   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)
```

```
}
```

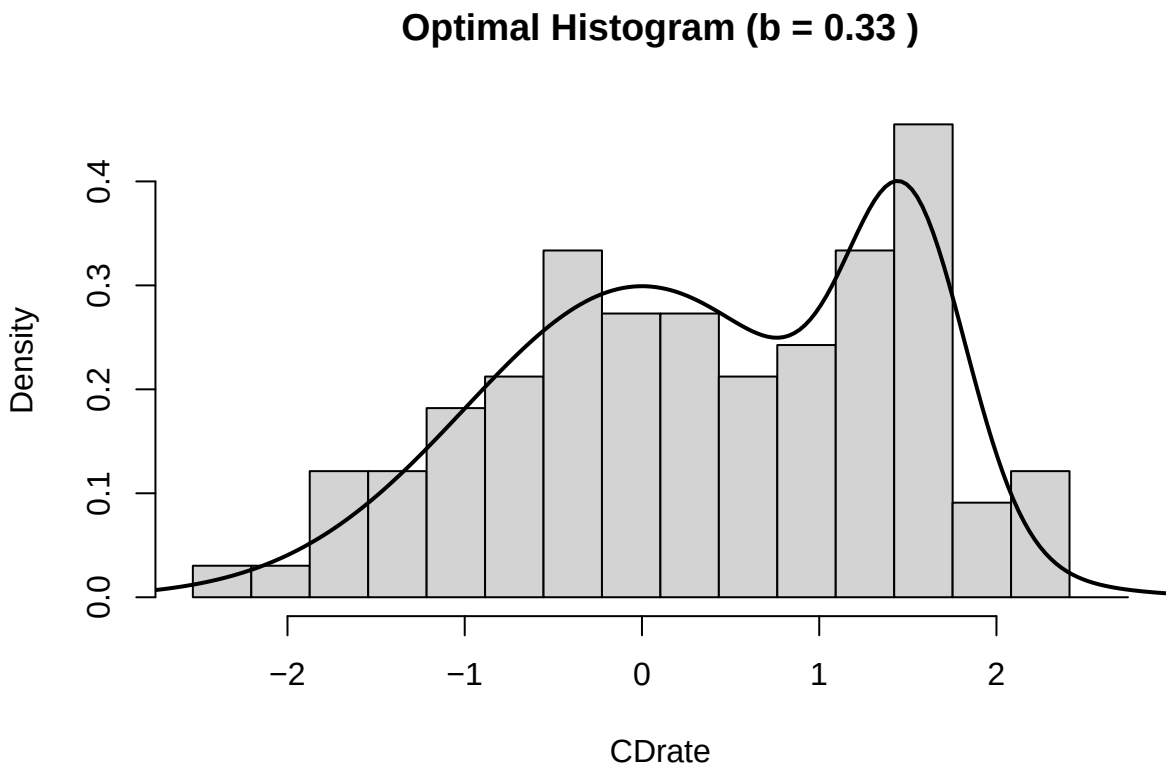
```
# Generate 100 observations from f(x) = (3/4)N(x; m = 0, s = 1) +(1/4) N(x; m = 3/2, s = 1/3)
set.seed(123)
n <- 100
mu <- c(0,3/2)
sigma <- c(1,1/3)
alpha <- c(3/4,1/4)
x <- sim.mixt(n=n, k=2, mu=mu, sigma=sigma, alpha=alpha)
f_x <- graph.mixt(k=2, mu=mu, sigma=sigma, alpha=alpha, graphic = F)
```

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.

```
#We will use the same method as before
A <- min(x) - 0.05 * diff(range(x))
Z <- max(x) + 0.05 * diff(range(x))

b_values <- seq((Z - A) / 15, (Z - A) / 1, length = 30)
looCV_log_lik_b <- numeric(length(b_values))

for (i in seq_along(b_values)) {
  current_b <- b_values[i]

  # Create histogram object with specified bin width
  hx <- hist(x, breaks = seq(A, Z + current_b, by = current_b), plot = FALSE)

  # Create step function
  hx_f <- stepfun(hx$breaks, c(0, hx$density, 0))

  # Calculate f_hat and f_loo
  f_hat <- hx_f(x)
  f_loo <- (n / (n - 1)) * f_hat - 1 / ((n - 1) * current_b)

  # Calculate and store looCV log-likelihood
  looCV_log_lik_b[i] <- sum(log(f_loo[f_loo > 0]))
}

# Plot the results
plot(b_values, looCV_log_lik_b, type = "b", pch = 19,
     xlab = "Bin Width (b)", ylab = "LOO Log-Likelihood",
     main = "LOO Cross-Validation for Bin Width")

# Find the optimal b
optimal_b <- b_values[which.max(looCV_log_lik_b)]
abline(v = optimal_b, col = "red", lty = 2)
legend("bottomright", legend = paste("Optimal b =", round(optimal_b, 3)), col = "red", lty = 2)
```



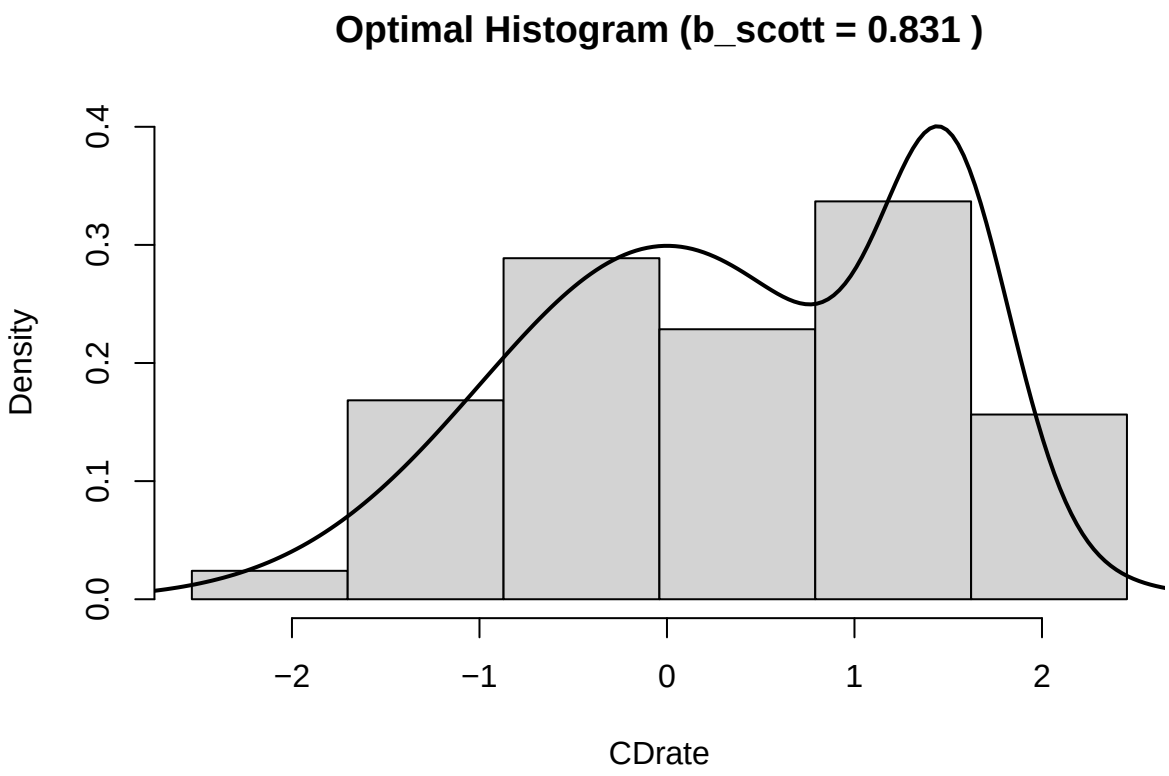

```
hx_optimal_b <- hist(x, breaks = seq(A, Z + optimal_b, by = optimal_b), plot = FALSE)
plot(hx_optimal_b, freq = FALSE,
     main = paste("Optimal Histogram (b =", round(optimal_b, 3), ")"),
     xlab = "CDrate")
lines(f_x$x, f_x$fx, lwd = 2)
```



Compare with the results obtained using the Scott's formula:

$$b_{Scott} = 3.49 St.Dev(X) n^{-1/3}$$

```
scott_b <- 3.49 * sd(x) * (length(x)^(-1/3))
hx_scott_b <- hist(x, breaks = seq(A, Z + scott_b, by = scott_b), plot = FALSE)
ymax <- max(c(hx_scott_b$y, f_x$fx))
plot(hx_scott_b, freq = FALSE, ylim = c(0, ymax),
     main = paste("Optimal Histogram (b_scott =", round(scott_b, 3), ")"),
     xlab = "CDrate")
lines(f_x$x, f_x$fx, lwd = 2)
```



In this case, the b that maximized the leave-one-out log-likelihood provided a better fit than Scott's formula for choosing b.

Kernel density estimator

8. Selecting the best h

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

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

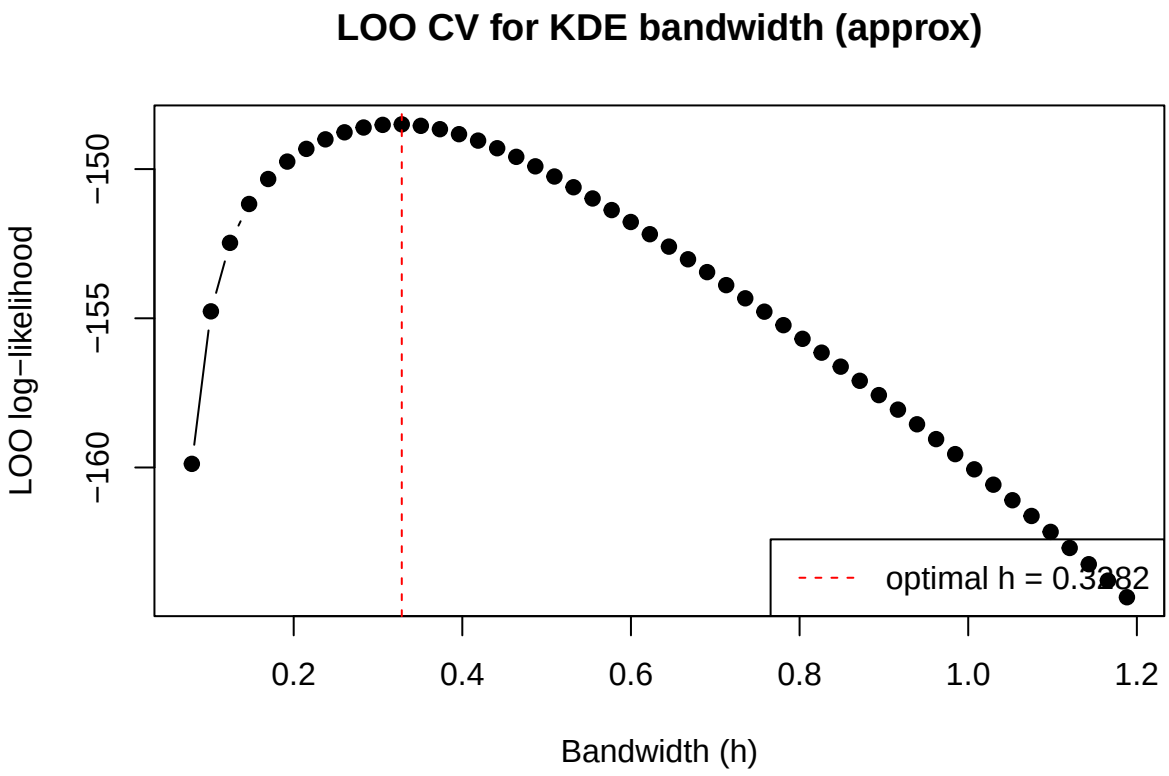
to select the value of h maximizing the leave-one-out log-likelihood function, and plot the corresponding kernel density estimator.

```
n <- length(x)
K0 <- dnorm(0)

kx0 <- density(x)
base_bw <- kx0$bw
h_values <- seq(base_bw/5, base_bw*3, length.out = 50)

loo_loglik_h <- numeric(length(h_values))
```

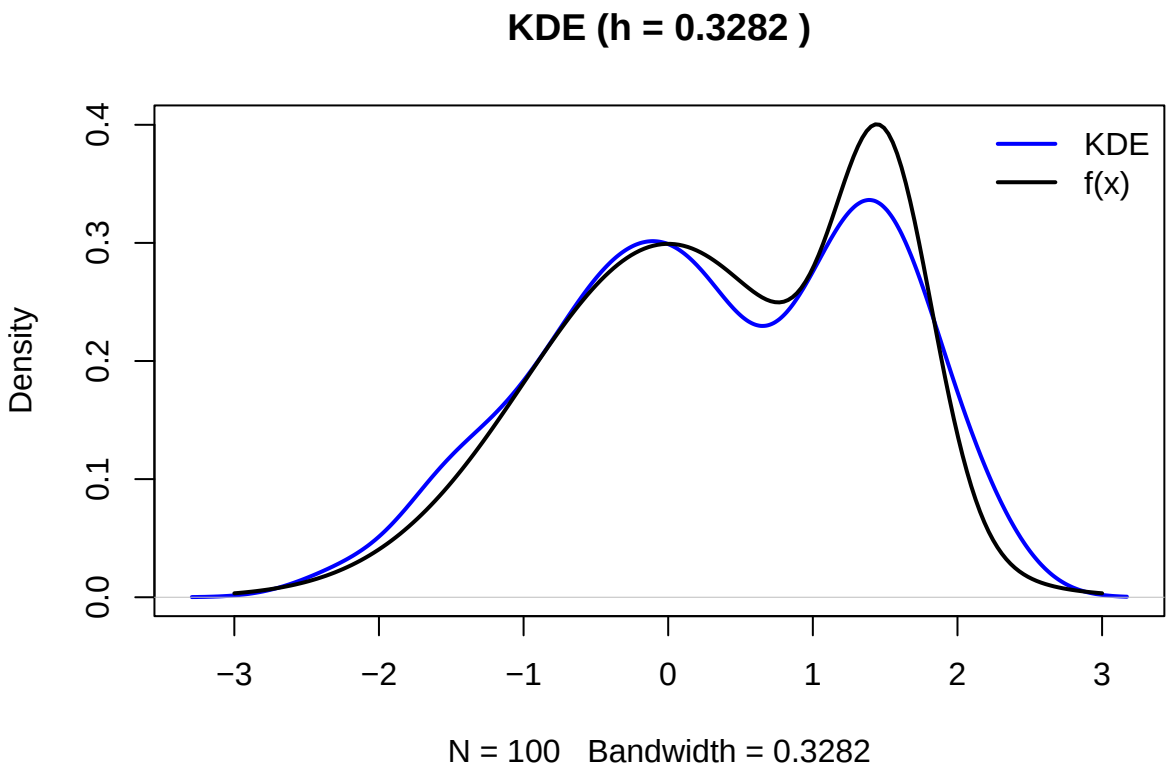
```r
for (i in seq_along(h_values)) {
  h <- h_values[i]
  kx <- density(x, bw = h, kernel = "gaussian", n = 1024,
                from = min(x) - 3*h, to = max(x) + 3*h)
  kx_f <- approxfun(x = kx$x, y = kx$y, rule = 2)
  f_hat <- kx_f(x)                      # approx f_hat at each xi
  f_loo <- (n / (n - 1)) * (f_hat - K0 / (n * h))
  pos <- f_loo > 0
  if (any(pos)) {
    loo_loglik_h[i] <- sum(log(f_loo[pos]))
  } else {
    loo_loglik_h[i] <- -Inf
  }
}

optimal_h_approx <- h_values[which.max(loo_loglik_h)]
cat("Optimal h (approx) =", optimal_h_approx, "\n")
```

```
## Optimal h (approx) = 0.3281848
```

```r
plot(h_values, loo_loglik_h, type = "b", pch = 19,
     xlab = "Bandwidth (h)", ylab = "LOO log-likelihood",
     main = "LOO CV for KDE bandwidth (approx)")
abline(v = optimal_h_approx, col = "red", lty = 2)
legend("bottomright", legend = paste("optimal h =", round(optimal_h_approx, 4)),
       col = "red", lty = 2)
```




```r
kx_opt <- density(x, bw = optimal_h_approx, kernel = "gaussian")

# Common ylim so both functions fit inside the plot
ymax <- max(c(kx_opt$y, f_x$fx))


plot(kx_opt,
     ylim = c(0, ymax),
     main = paste("KDE (h =", round(optimal_h_approx, 4), ")"),
     col = "blue",
     lwd = 2)

lines(f_x$x, f_x$fx, col = "black", lwd = 2)

legend("topright",
       legend = c("KDE", "f(x)"),
       col = c("blue", "black"),
       lwd = 2,
       bty = "n")
```


KDE (h = 0.3282)
KDE
f(x)
Density
0.0
0.1
0.2
0.3
0.4
−3
−2
−1
0
1
2
3
N = 100 Bandwidth = 0.3282