Density Estimation

Bandwidth Choice by Leave-one-out Maximum Likelihood

Azzarito Domenico, Daniel Reverter, Alexis Vendrix

03 October, 2025

Histogram

1.

We want to 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.

Let the histogram be defined by bins B_j of common width b and let $j(x)$ be the index function indicating the interval containing x. The density estimator using all n observations is given by:

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

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

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

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

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

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

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

2.

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

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

```r
# 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
```

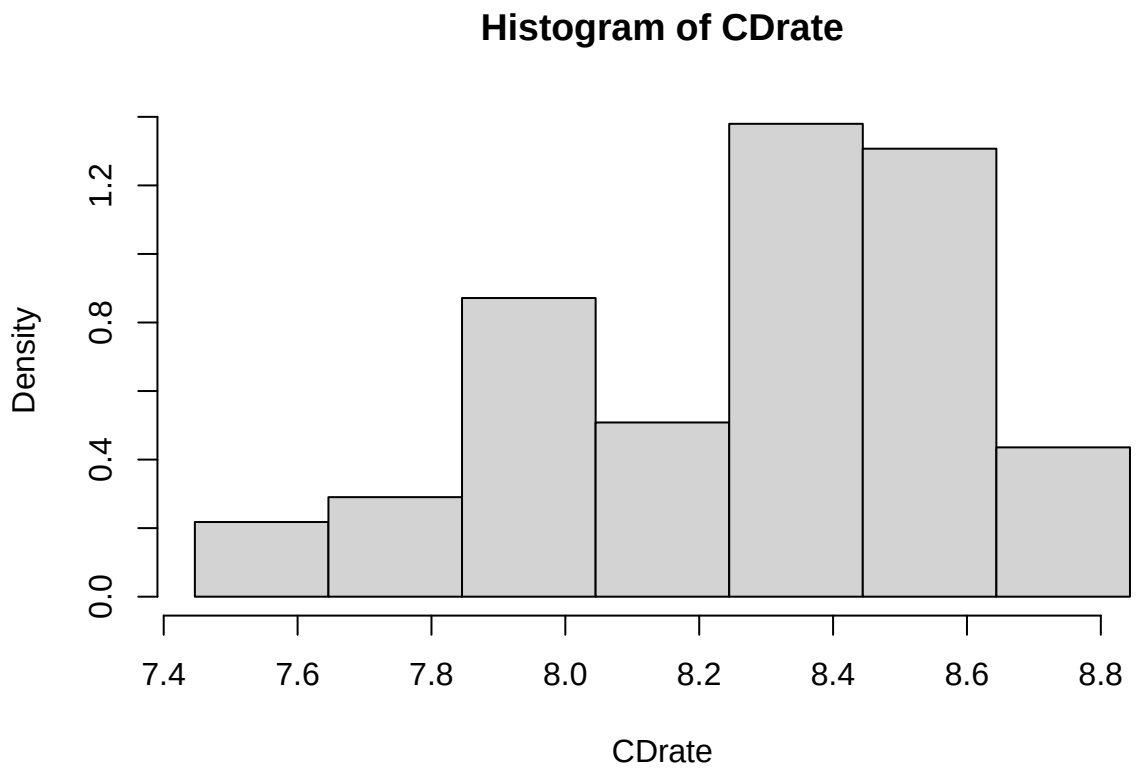
```r
cat("Z =", Z, "\n")
```

```
## Z = 8.8435
```

and plot the histogram of x as

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

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

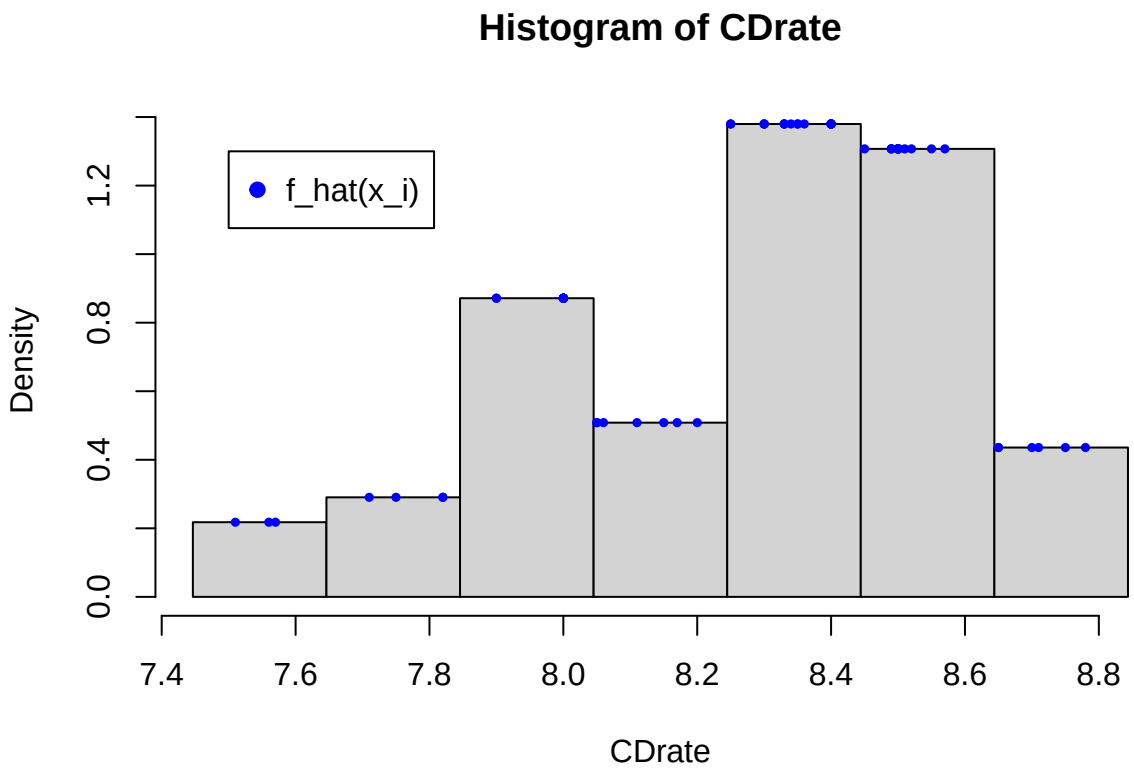


The following sentence converts this histogram into a function that can be evaluated at any point of $\mathbb{R}$, or at a vector of real numbers:

```r
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.

```r
# 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(x = 7.5, y = 1.3, "f_hat(x_i)", col = "blue", pch = 19)
```

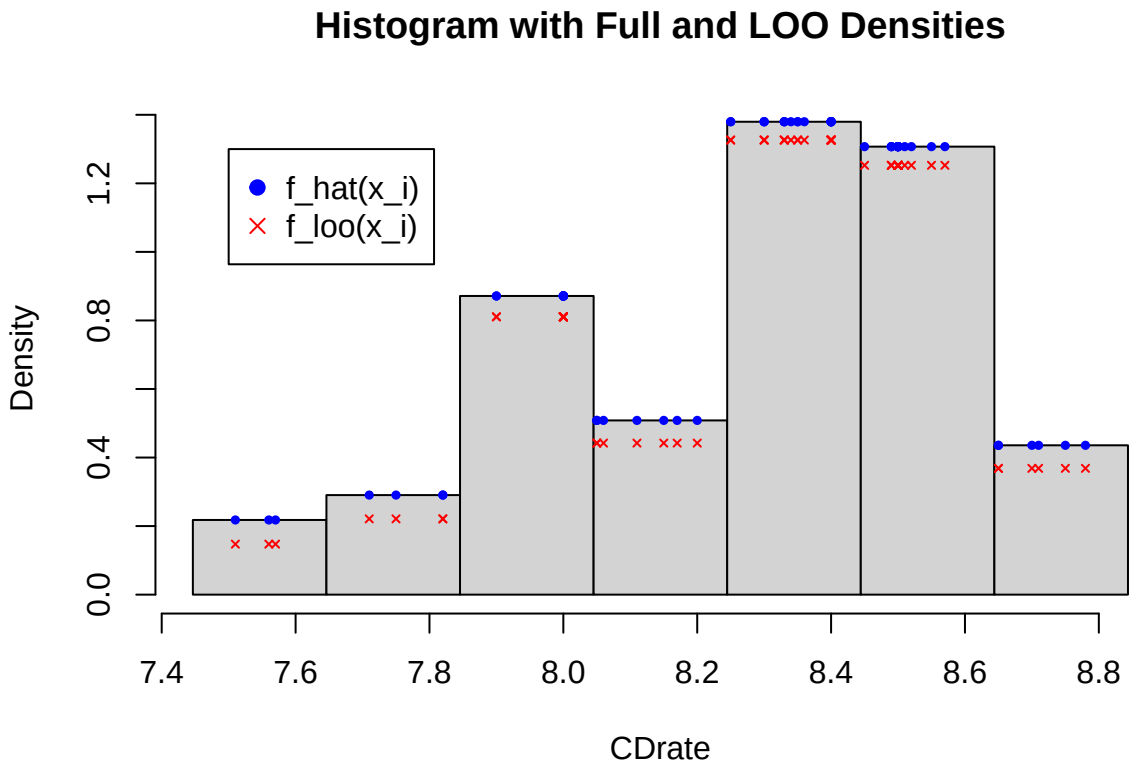


3.

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)
```

```r
# 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(x = 7.5, y = 1.3, c("f_hat(x_i)", "f_loo(x_i)"), col = c("blue", "red"), pch = c(19, 4))
```



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

The red crosses show the histogram estimator of the density function for each data point if that point had been excluded from the calculation.

The red crosses 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.

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

Before correction:

```
# 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
```

After correction:

```
# We take the log of all values.
looCV_log_lik_7 <- sum(log(f_loo))
cat("Leave-one-out log-likelihood for nbr=7:", looCV_log_lik_7)
```

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

After correction, we observe the same result as before meaning that all our values are positives and that our model will be able to make possible prediction for every points.

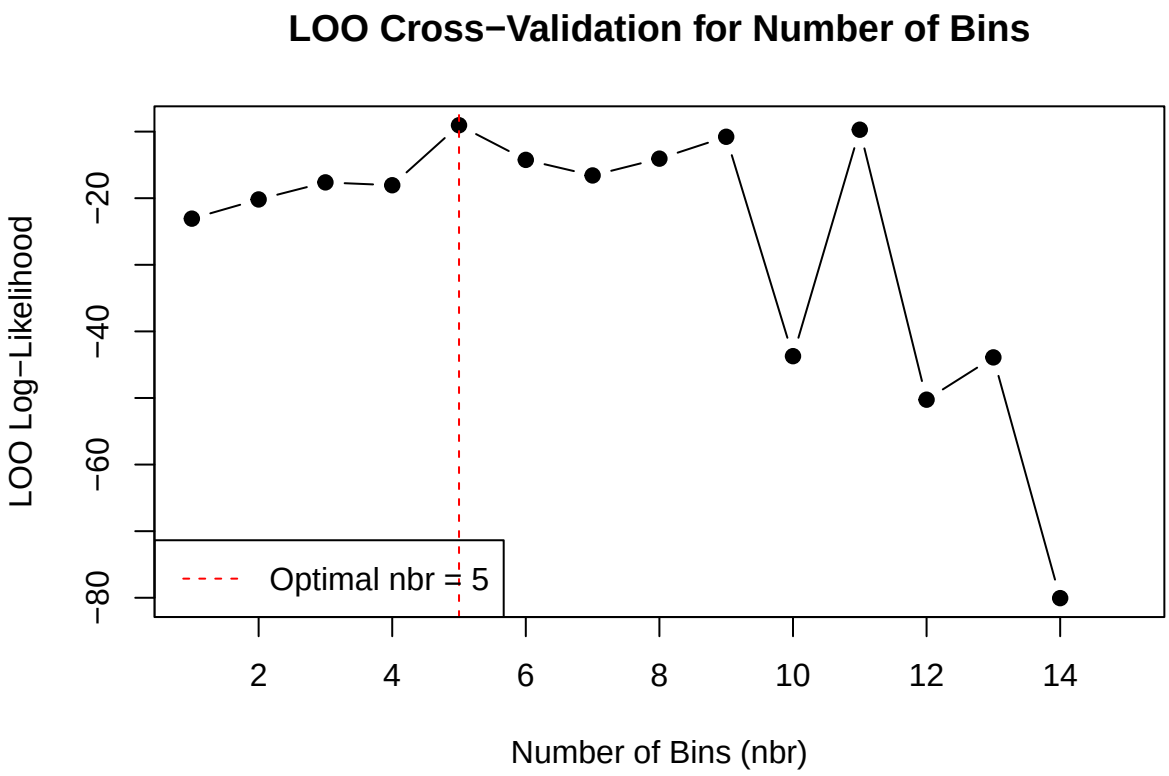
It's better to keep all values because a zero prediction ($f\_loo = 0$) means the model considered that data point impossible. Including these zeros results in a log-likelihood of negative infinity, which correctly penalizes and disqualifies a model that makes impossible predictions.

5.

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.

```
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))
```

```
}

# 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("bottomleft", legend = paste("Optimal nbr =", optimal_nbr), col = "red", lty = 2)
```

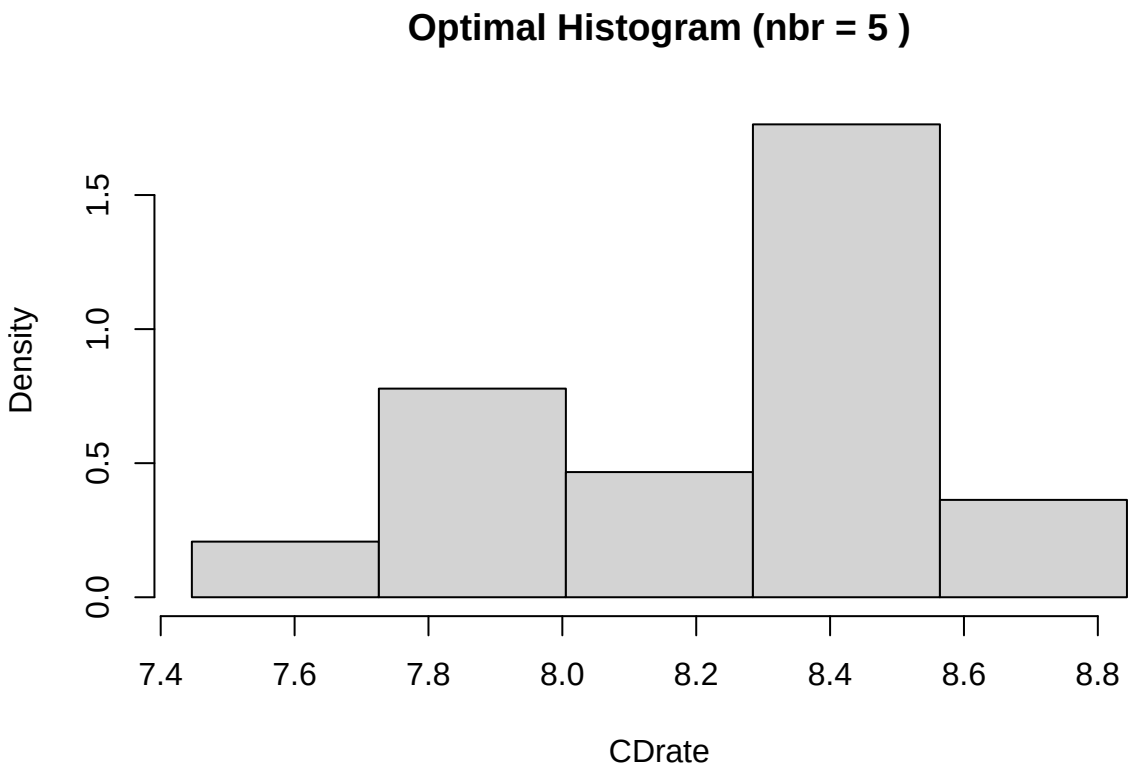


The plot displays the leave-one-out log-likelihood score for histograms with a varying number of bins (nbr). The score increases to a distinct peak at nbr = 5, which is the value that maximizes the log-likelihood. After this point, the performance becomes more erratic and generally decreases.

Therefore, according to the looCV method, the optimal choice for the number of bins is 5 for this dataset. Unlike the previous analysis, there is a clear peak, and choosing a higher nbr would likely result in overfitting.

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")
```



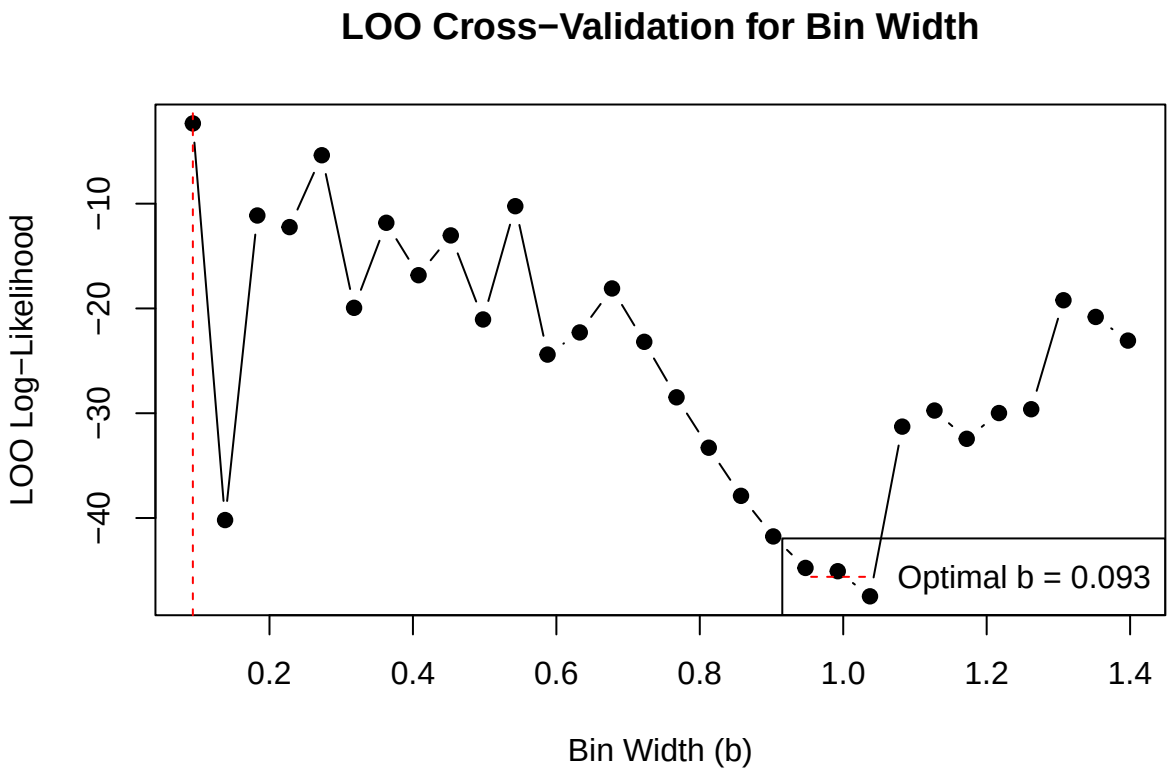
6.

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.

Before correction:

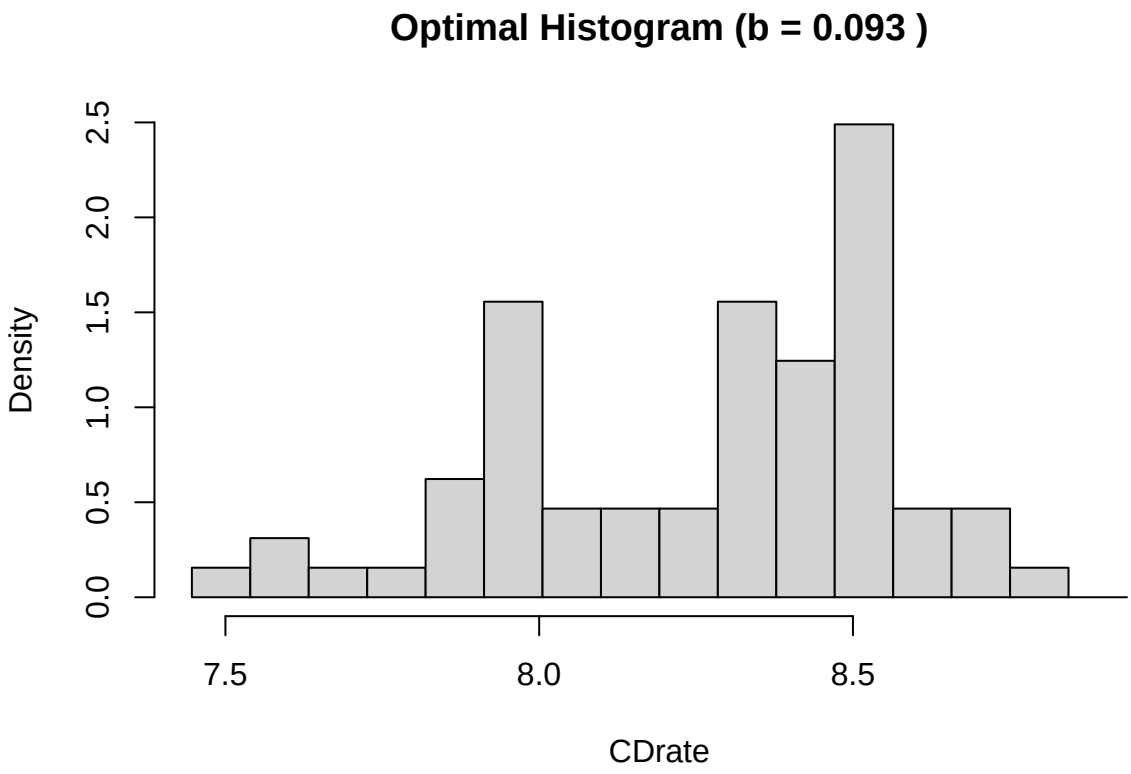
```
b_values <- seq((Z - A) / 15, (Z - A) / 1, length = 30)
looCV_log_lik_b <- numeric(length(b_values))

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)
```

```r
  # 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.

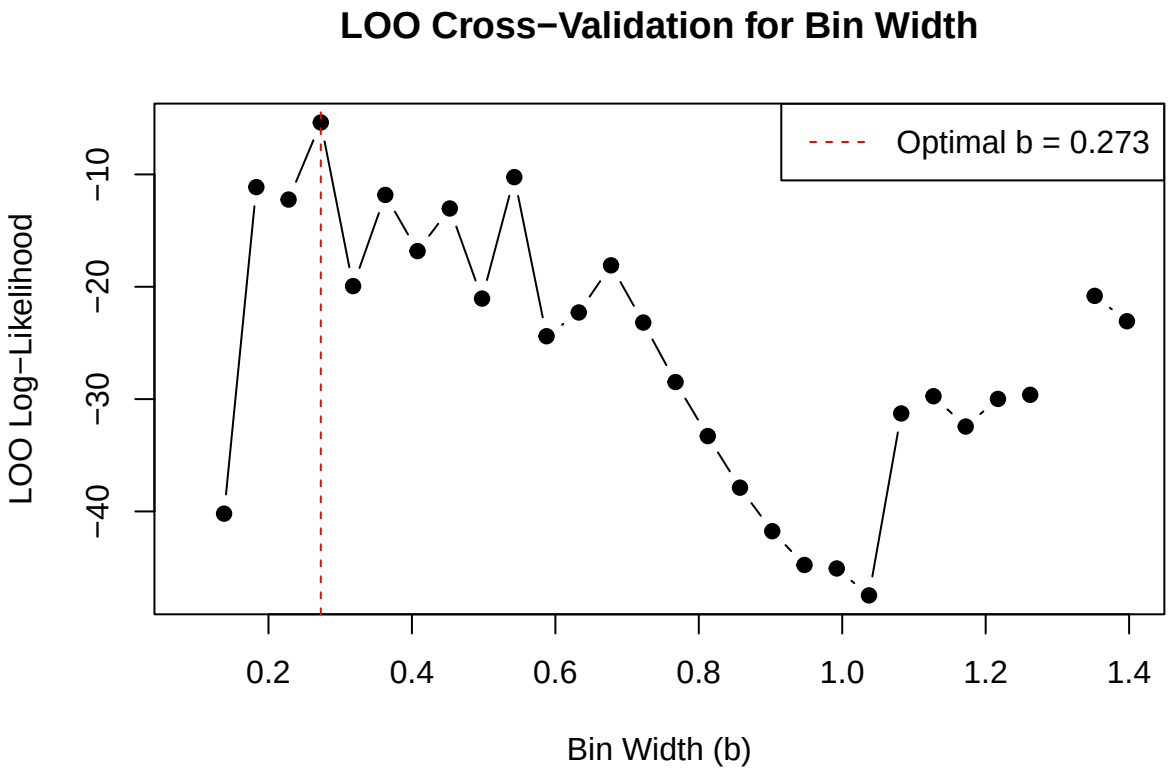
```r
hx_optimal_b <- hist(x, breaks = seq(A, Z + optimal_b, by = optimal_b), plot = FALSE)
plot(hx_optimal_b, freq = FALSE,
     main = paste("Optimal Histogram (b =", round(optimal_b, 3), ")"),
     xlab = "CDrate")
```



After correction:

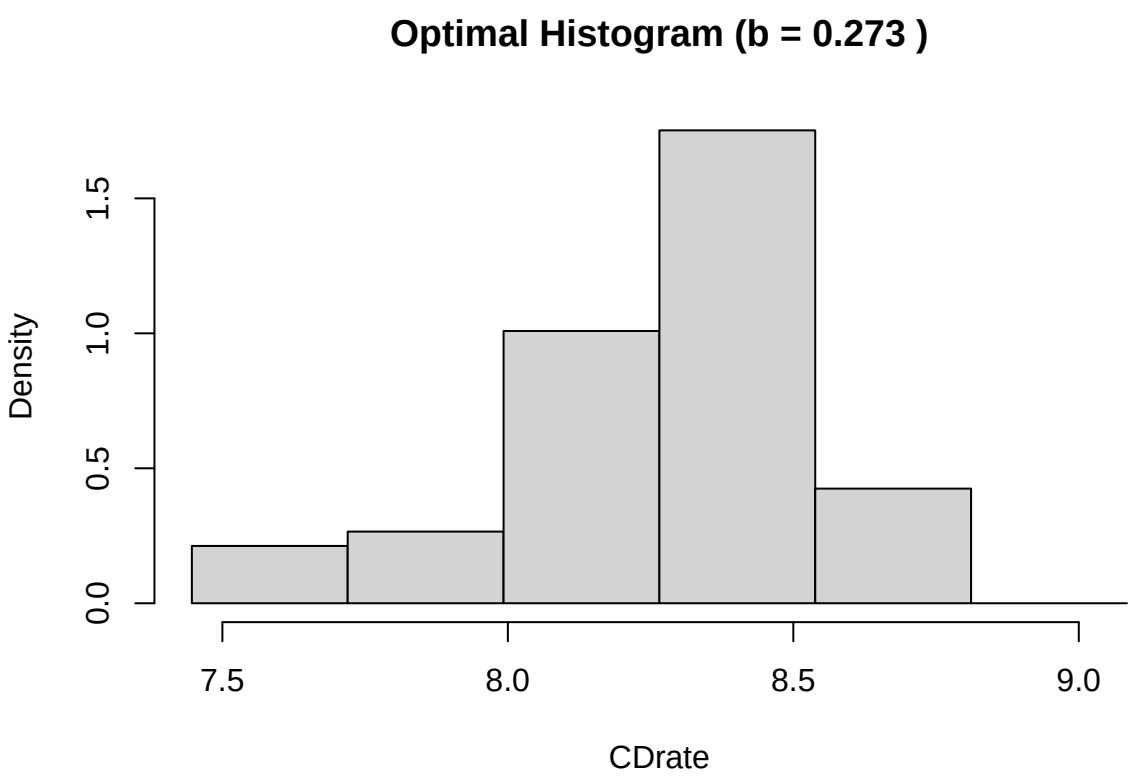
```r
b_values <- seq((Z - A) / 15, (Z - A) / 1, length = 30)
looCV_log_lik_b <- numeric(length(b_values))

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))
}

# 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")
```

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



Plot the corresponding histogram.

```r
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")
```



As we can see from the new plots after correction, both optimization methods—choosing the number of bins (nbr) and choosing the bin width (b) directly—converge on the same optimal histogram structure. The analysis for nbr found that 5 bins were optimal, and the analysis for b found an optimal width of 0.273, which also results in a 5-bin histogram. This confirms that the model with 5 bins provides the most robust predictive fit for this dataset.

7.

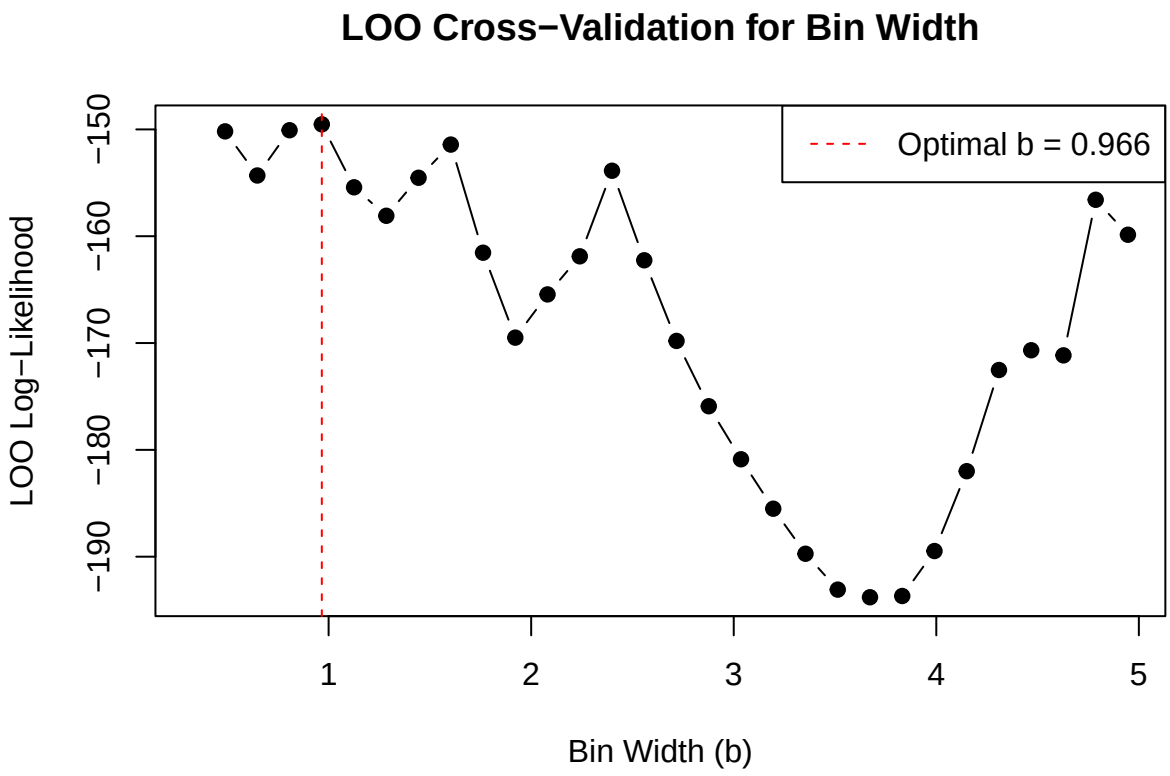
Recycle the functions `graph.mixt` and `sim.mixt` to generate $n = 100$ data from

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

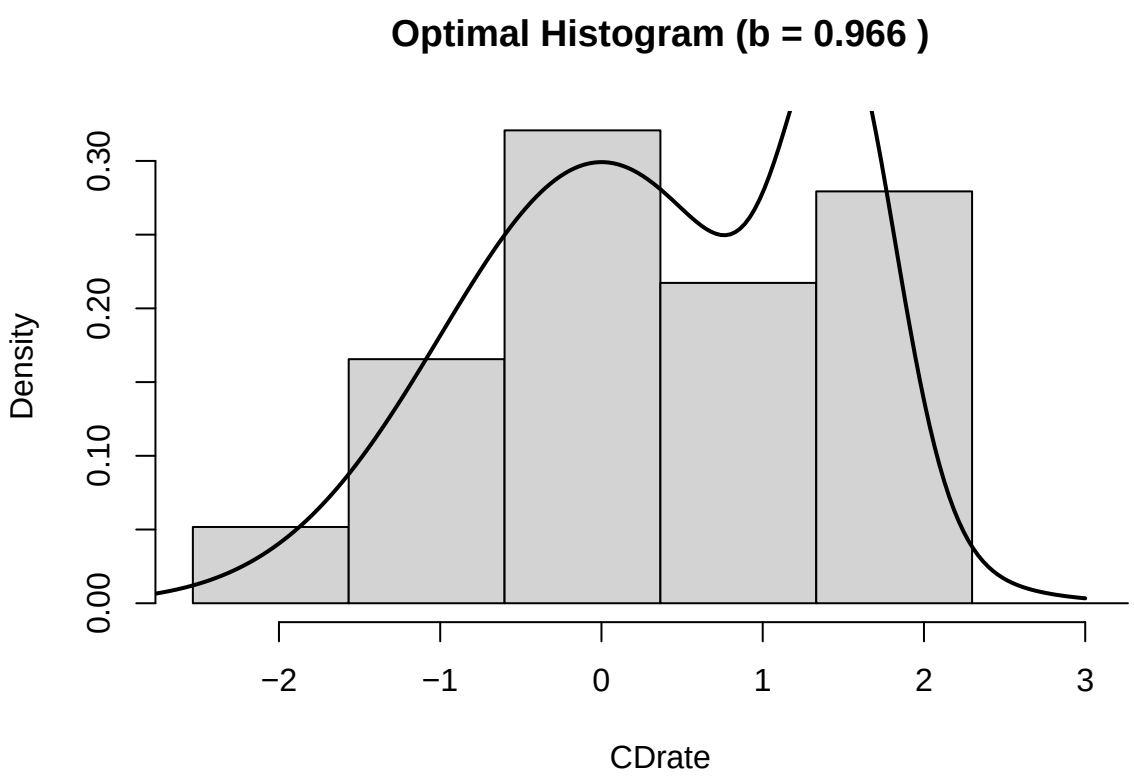
```
# Generate 100 observations from f(x) = (3/4)N(x; m = 0, s = 1) +(1/4) N(x; m = 3/2, s = 1/3)
set.seed(123)
n <- 100
mu <- c(0,3/2)
sigma <- c(1,1/3)
alpha <- c(3/4,1/4)
x <- sim.mixt(n=n, k=2, mu=mu, sigma=sigma, alpha=alpha)
f_x <- graph.mixt(k=2, mu=mu, sigma=sigma, alpha=alpha, graphic = F)
```

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.

```r
#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))
}

# 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("topright", legend = paste("Optimal b =", round(optimal_b, 3)), col = "red", lty = 2)
```



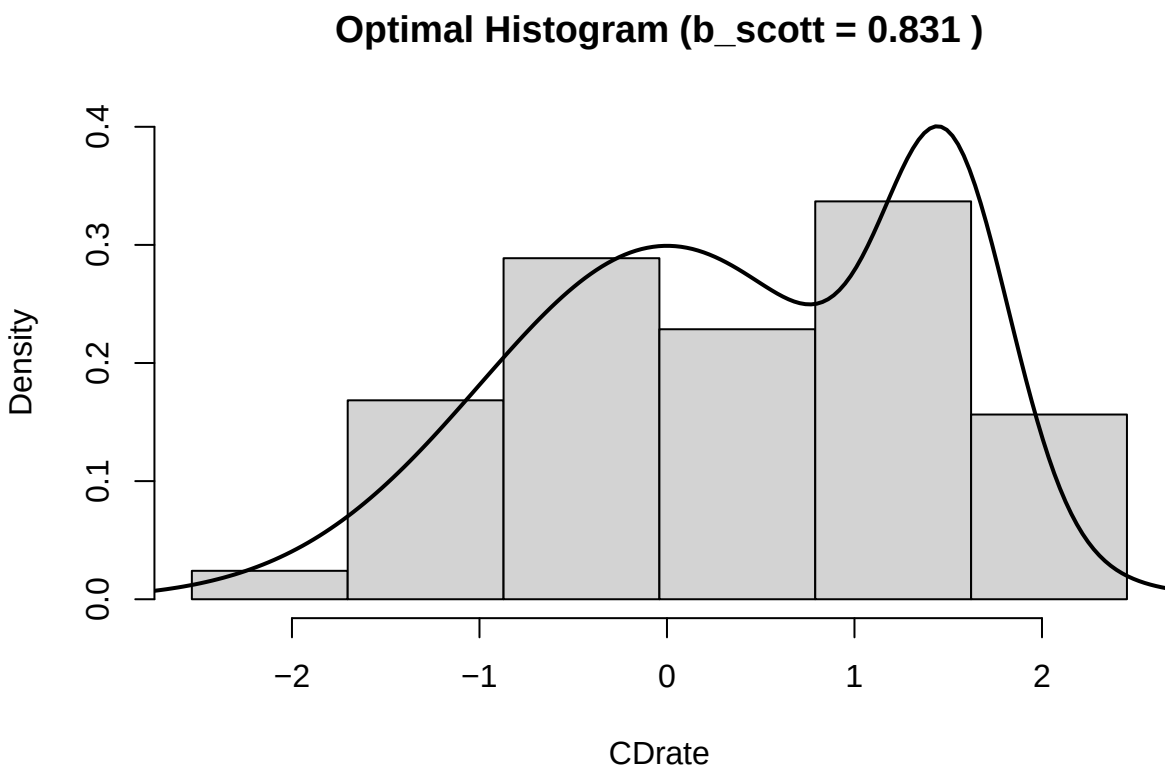

```
hx_optimal_b <- hist(x, breaks = seq(A, Z + optimal_b, by = optimal_b), plot = FALSE)
plot(hx_optimal_b, freq = FALSE,
     main = paste("Optimal Histogram (b =", round(optimal_b, 3), ")"),
     xlab = "CDrate")
lines(f_x$x, f_x$fx, lwd = 2)
```



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. This can be proved by calculating the MSE of the estimators.

```
# Compute histogram densities as step functions
hx_scott_f <- stepfun(hx_scott_b$breaks, c(0, hx_scott_b$density, 0))
hx_optimal_f <- stepfun(hx_optimal_b$breaks, c(0, hx_optimal_b$density, 0))

# Evaluate histogram estimates at the grid of f_x
f_hat_scott <- hx_scott_f(f_x$x)
f_hat_optimal <- hx_optimal_f(f_x$x)

# Compute MSE for Scott's rule and Optimal b
mse_scott <- mean((f_hat_scott - f_x$fx)^2)
mse_optimal <- mean((f_hat_optimal - f_x$fx)^2)

# Print results
cat("MSE (Scott's rule):", mse_scott, "\n")
```

```
## MSE (Scott's rule): 0.002943868
```

```
cat("MSE (Optimal b):", mse_optimal, "\n")
```

```
## MSE (Optimal b): 0.00459882
```

Kernel Density Estimator

8.

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

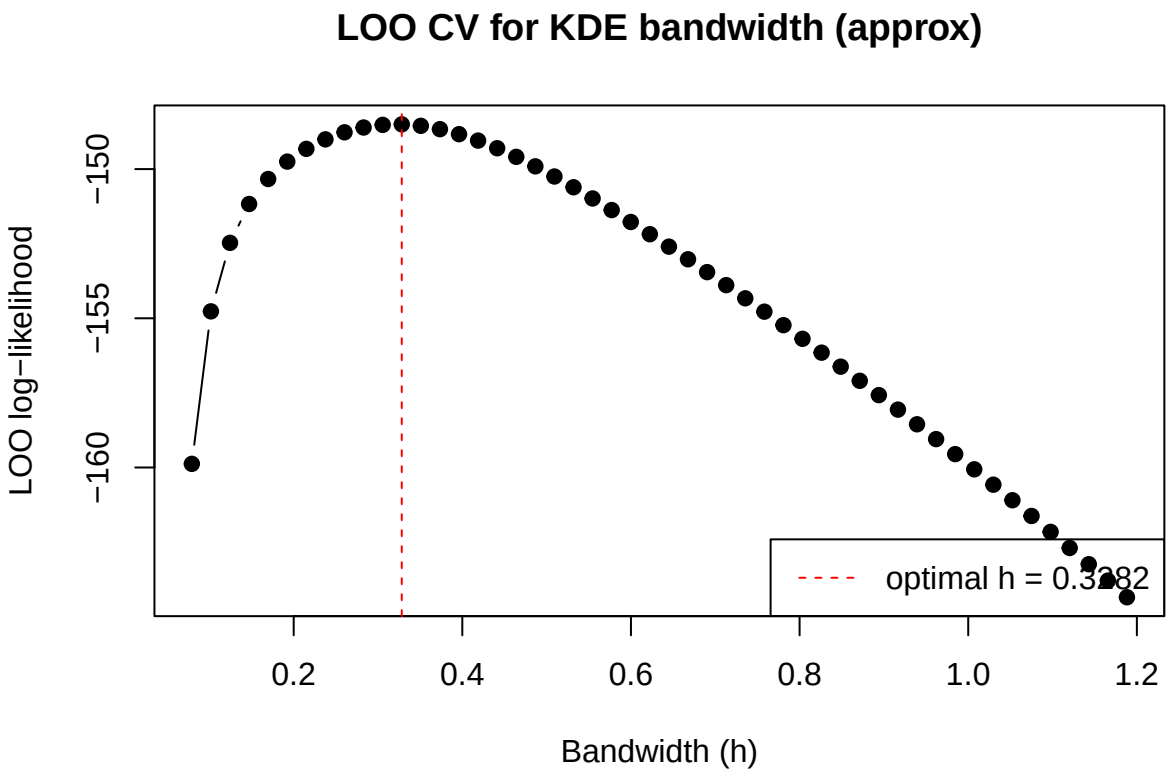
$$\hat{f}_{h,(-i)}(x_i) = \frac{n}{n-1}\left(\hat{f}_h(x_i) - \frac{K(0)}{nh}\right)$$

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

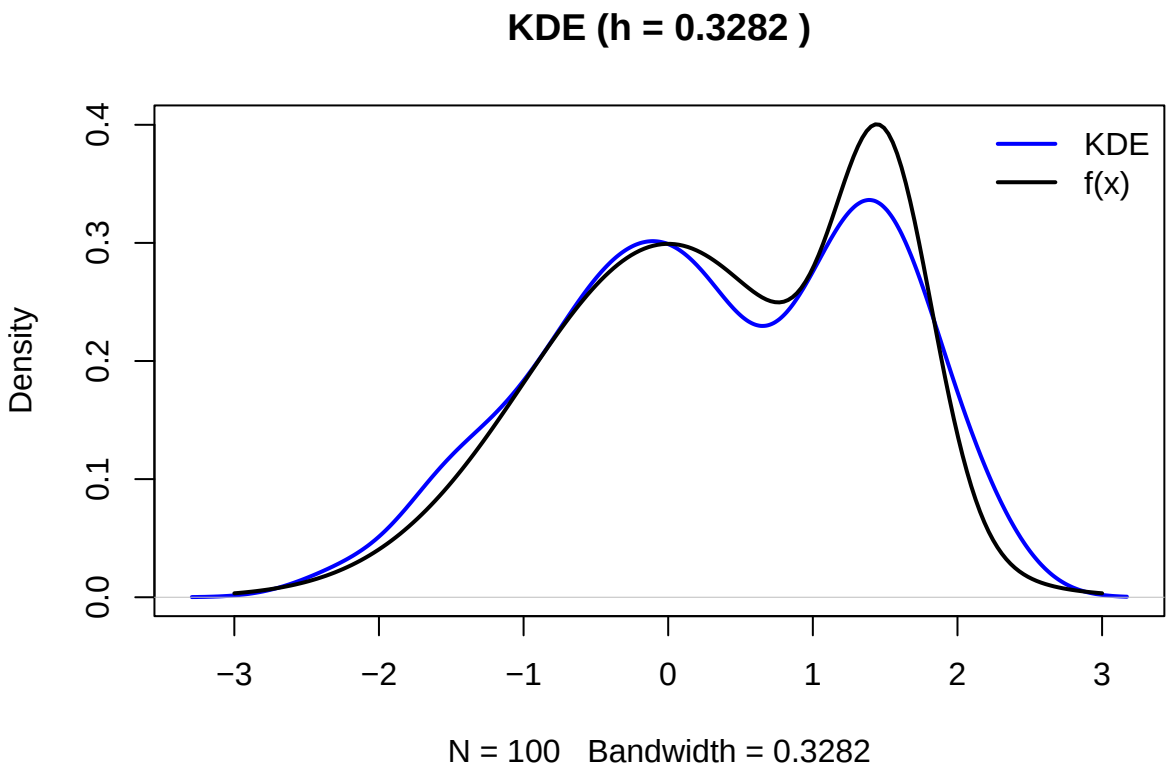
```
n <- length(x)
K0 <- dnorm(0)

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))

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
```

```
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