Density Estimation

Bandwidth Choice by Leave-one-out Maximum Likelihood

Pedro Delicado

23 September, 2025

Histogram

Relationship between $\hat{f}_{\text{hist}}(x_i)$ and $\hat{f}_{\text{hist},(-i)}(x_i)$

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.

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:

$$\hat{f}_{\text{hist}}(x) = \frac{N_j}{nb} \quad \text{for } x \in B_j$$

where N_j is the number of data points falling in bin B_j.

Let's consider a specific data point x_i which falls into a particular bin, say B_k. The value of the estimator at this point is:

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

The leave-one-out estimator, $\hat{f}_{\text{hist},(-i)}(x)$, is constructed using the dataset of size $n-1$ that excludes the point x_i. When evaluated at $x_i \in B_k$, the number of points in bin B_k from this reduced dataset is $N_k - 1$. Therefore, the leave-one-out estimator is:

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

To find the relationship, we can express N_k from the first equation as $N_k = nb \cdot \hat{f}_{\text{hist}}(x_i)$. Substituting this into the second equation gives:

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

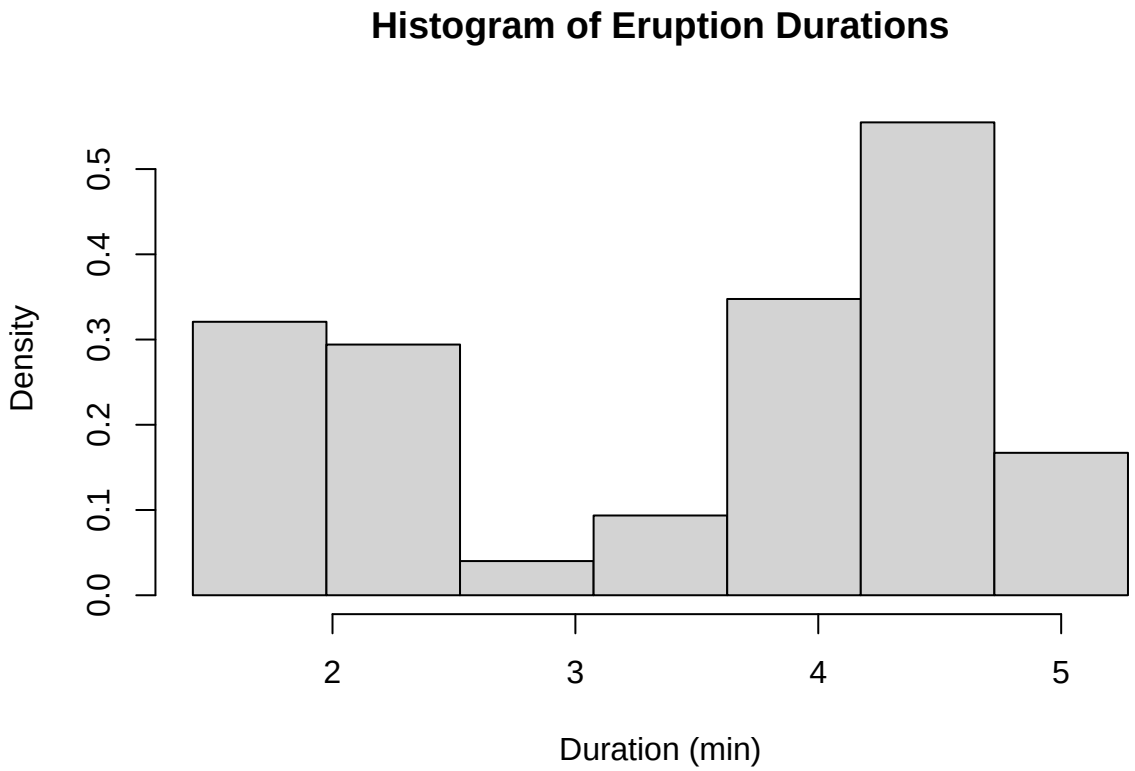
$$\hat{f}_{\text{hist},(-i)}(x_i) = \frac{n}{n-1}\hat{f}_{\text{hist}}(x_i) - \frac{1}{(n-1)b}$$

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

Application to CD Rate Data

Note: The "CD rate data set" is not a standard R dataset. For reproducibility, we will use the `eruptions` column from the `faithful` dataset, which is a common dataset for density estimation examples.

```r
# Using faithful$eruptions as a substitute for the "CD rate data"
x <- faithful$eruptions
n <- length(x)

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

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



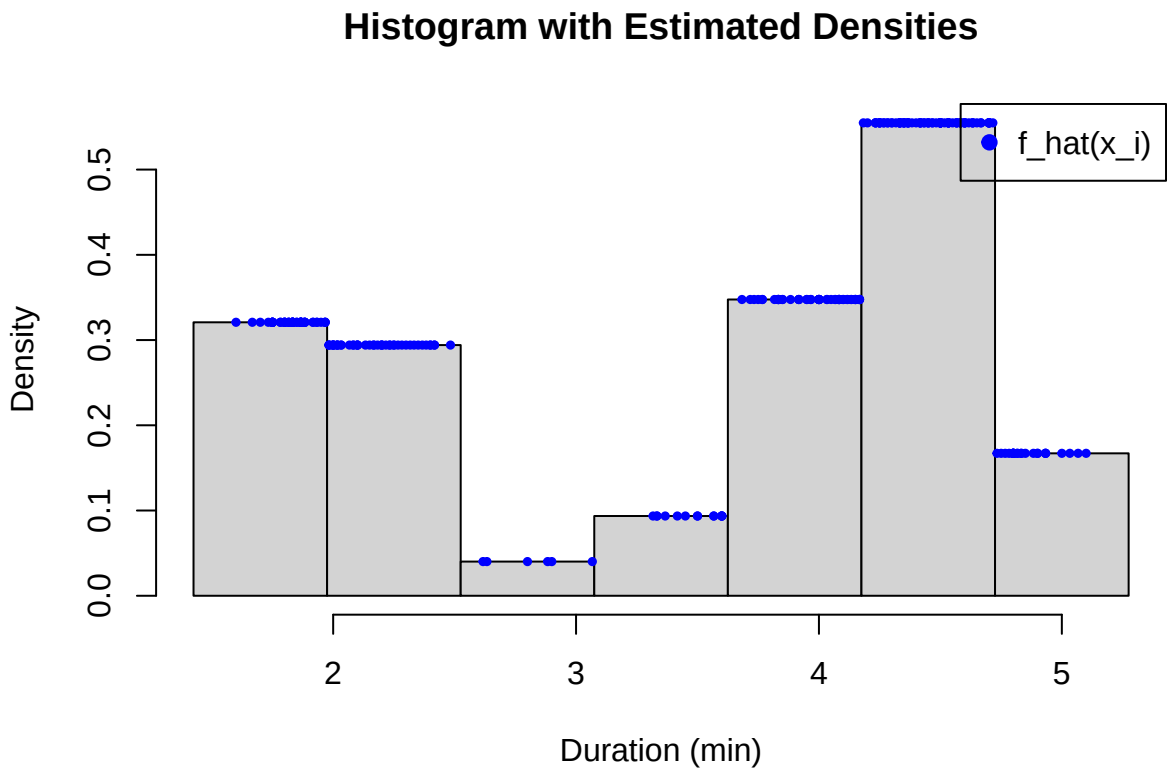
The following sentence converts this histogram into a function that can be evaluated at any point of $\mathbb{R}$, or at a vector of real numbers:

```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.

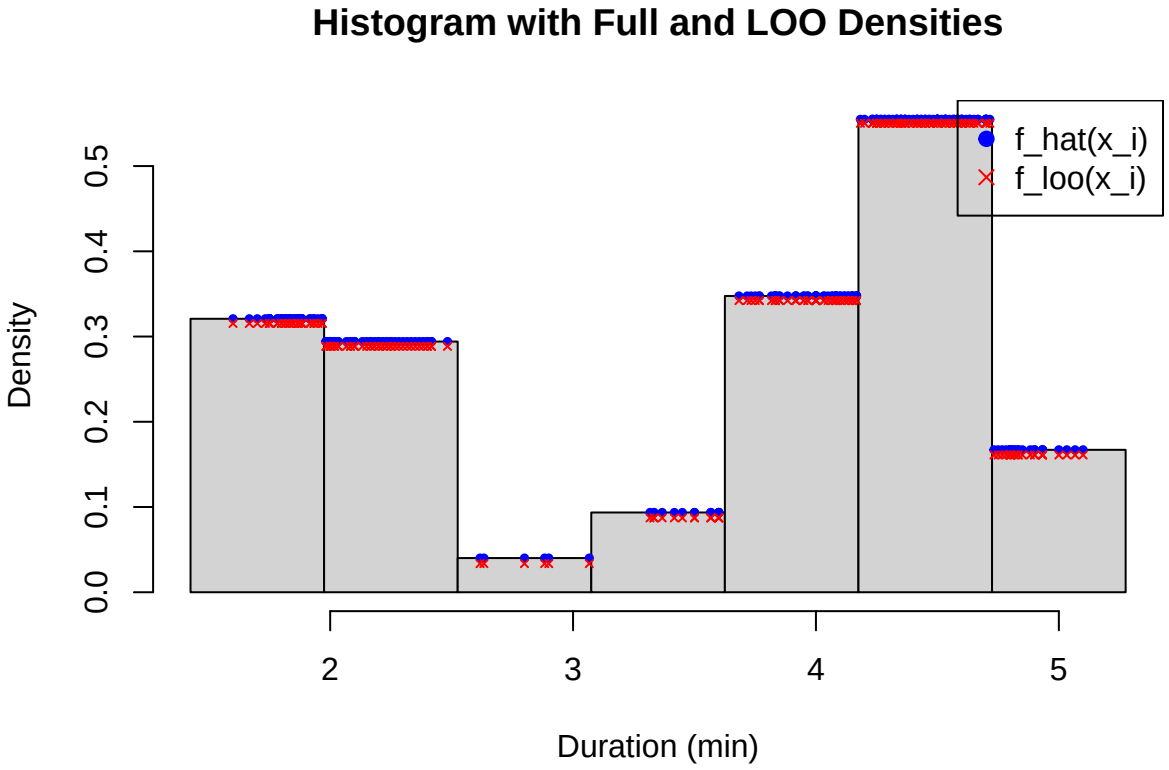
```r
# Evaluate the histogram estimator at each data point
f_hat <- hx_f(x)

# Add the points to the plot
plot(hx, freq = FALSE, main = "Histogram with Estimated Densities", xlab = "Duration (min)")
points(x, f_hat, col = "blue", pch = 19, cex = 0.5)
```

```r
legend("topright", "f_hat(x_i)", col = "blue", pch = 19)
```



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

# 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 = "Duration (min)")
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))
```



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)
```

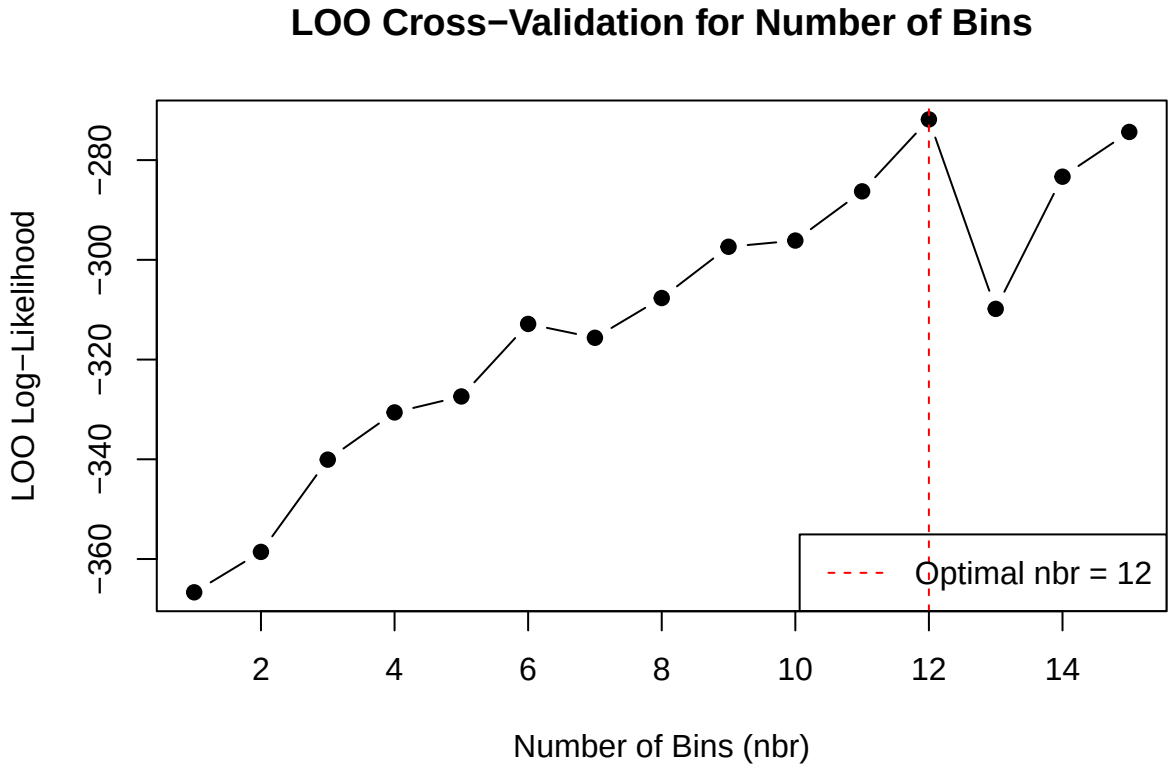
```
## Leave-one-out log-likelihood for nbr=7: -315.6349
```

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.

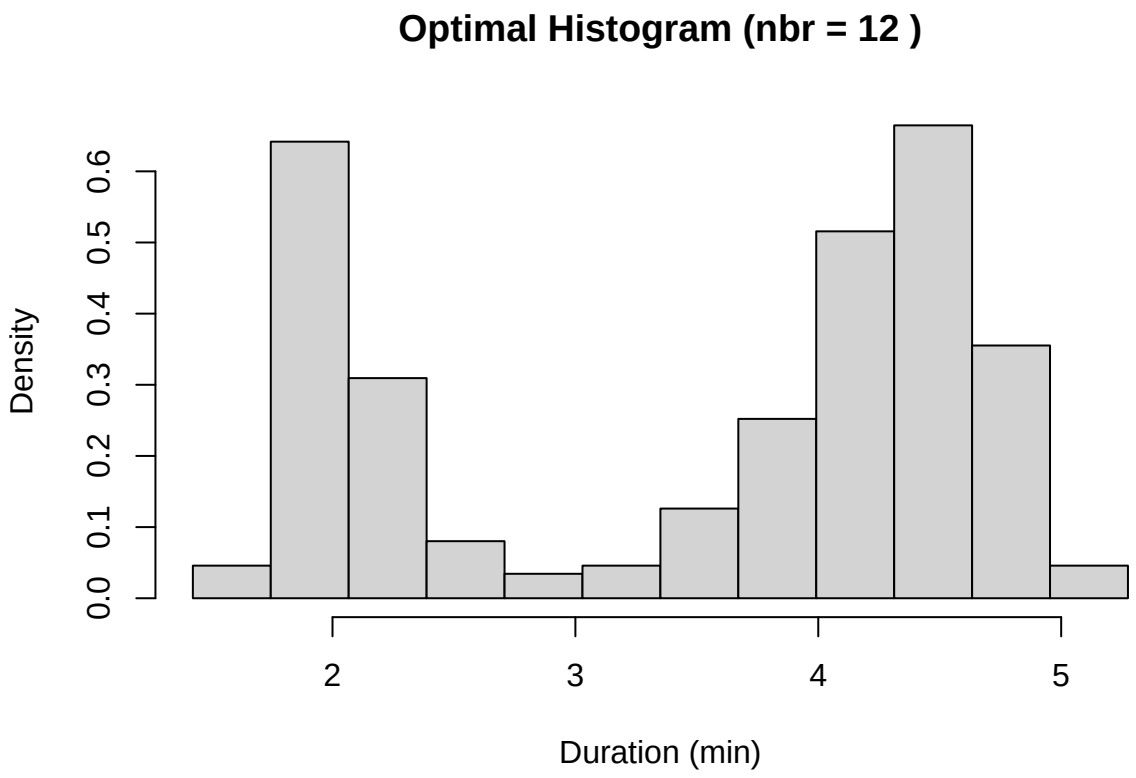
```
n <- length(x)
nbr_values <- 1:15
looCV_log_lik_nbr <- numeric(length(nbr_values))

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))
```

```r
  # 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)
```



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

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

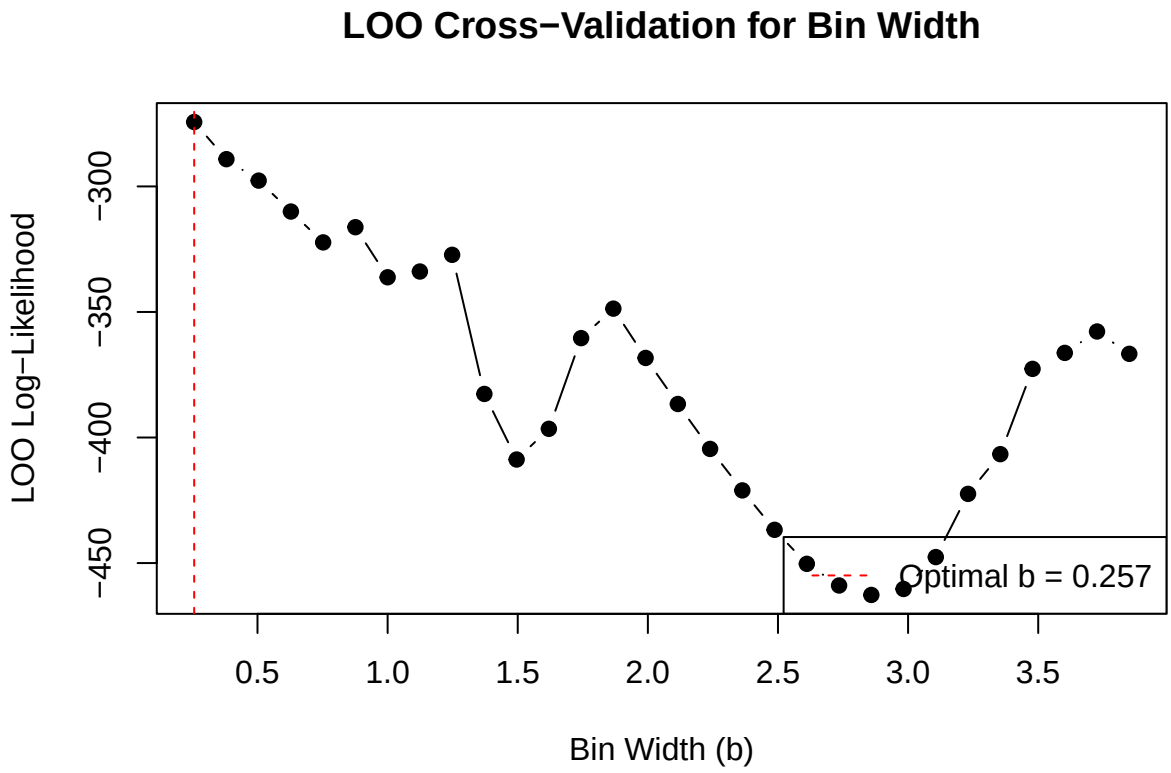


Choosing `b` by looCV

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.

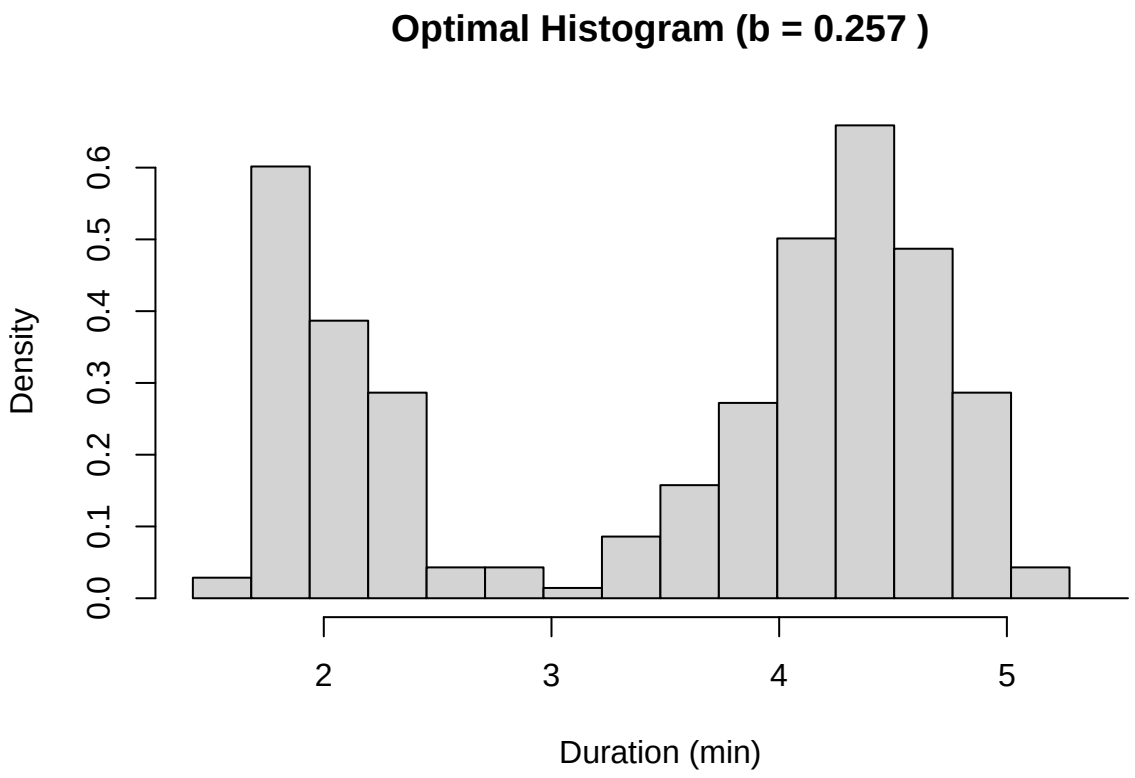
```
n <- length(x)
b_values <- seq((Z - A) / 15, (Z - A) / 1, length = 30)
looCV_log_lik_b <- numeric(length(b_values))

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
```

```r
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 = "Duration (min)")
```



Mixture of Normals Example

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

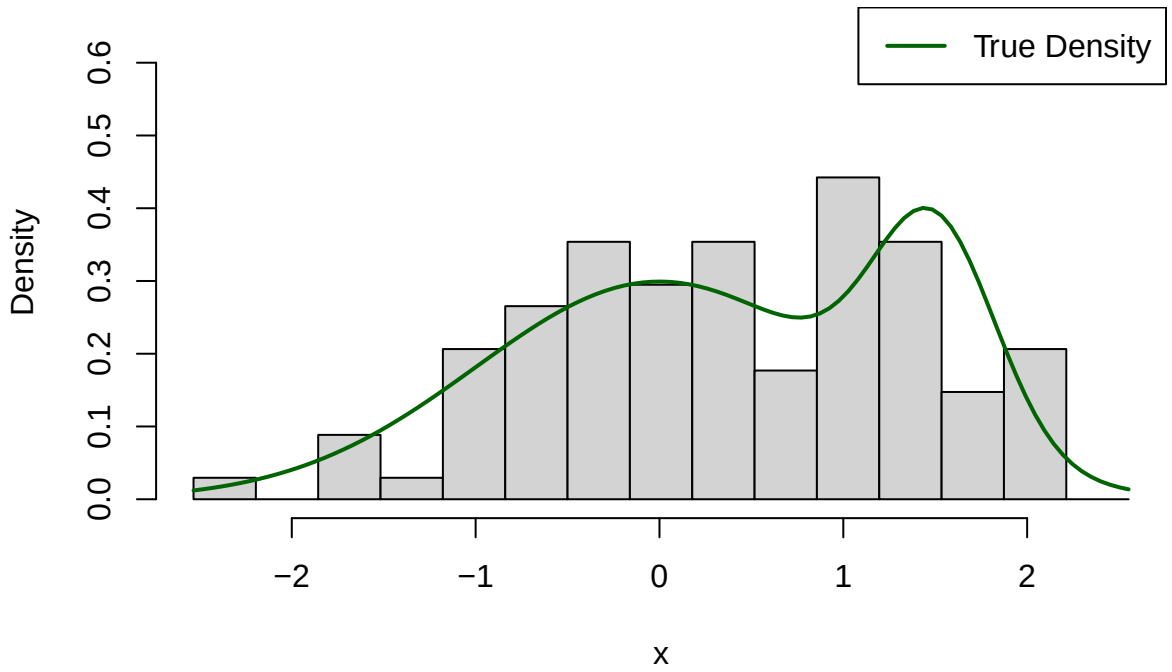
$$f(x) = (3/4)N(x; m = 0, s = 1) + (1/4)N(x; m = 3/2, s = 1/3)$$

```r
# Function to simulate data from the mixture
sim.mixt <- function(n) {
  p <- 3/4
  m1 <- 0; s1 <- 1
  m2 <- 3/2; s2 <- 1/3

  # Generate component indicators (0 for component 1, 1 for component 2)
  z <- rbinom(n, 1, 1 - p)

  # Generate data
  x <- numeric(n)
  x[z == 0] <- rnorm(sum(z == 0), mean = m1, sd = s1)
  x[z == 1] <- rnorm(sum(z == 1), mean = m2, sd = s2)

  return(x)
}

# Function to calculate the true density of the mixture
graph.mixt <- function(x_grid) {
  p <- 3/4
  m1 <- 0; s1 <- 1
```

```
  m2 <- 3/2; s2 <- 1/3

  density <- p * dnorm(x_grid, mean = m1, sd = s1) + (1 - p) * dnorm(x_grid, mean = m2, sd = s2)
  return(density)
}

# Generate n=100 data from the mixture
set.seed(123) # for reproducibility
x_mix <- sim.mixt(100)
```

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.

```
n_mix <- length(x_mix)
A_mix <- min(x_mix) - 0.05 * diff(range(x_mix))
Z_mix <- max(x_mix) + 0.05 * diff(range(x_mix))

b_values_mix <- seq((Z_mix - A_mix) / 20, (Z_mix - A_mix) / 2, length = 50)
looCV_log_lik_b_mix <- numeric(length(b_values_mix))

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

  hx <- hist(x_mix, breaks = seq(A_mix, Z_mix + current_b, by = current_b), plot = FALSE)
  hx_f <- stepfun(hx$breaks, c(0, hx$density, 0))

  f_hat <- hx_f(x_mix)
  f_loo <- (n_mix / (n_mix - 1)) * f_hat - 1 / ((n_mix - 1) * current_b)

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

# Find optimal b and plot
optimal_b_mix <- b_values_mix[which.max(looCV_log_lik_b_mix)]

hx_optimal_mix <- hist(x_mix, breaks = seq(A_mix, Z_mix + optimal_b_mix, by = optimal_b_mix), plot = FA
plot(hx_optimal_mix, freq = FALSE,
     main = paste("Optimal Histogram for Mixture Data (b_looCV =", round(optimal_b_mix, 3), ")"),
     xlab = "x", ylim = c(0, 0.65))

# Overlay true density
curve(graph.mixt, add = TRUE, col = "darkgreen", lwd = 2)
legend("topright", "True Density", col = "darkgreen", lwd = 2)
```



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

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

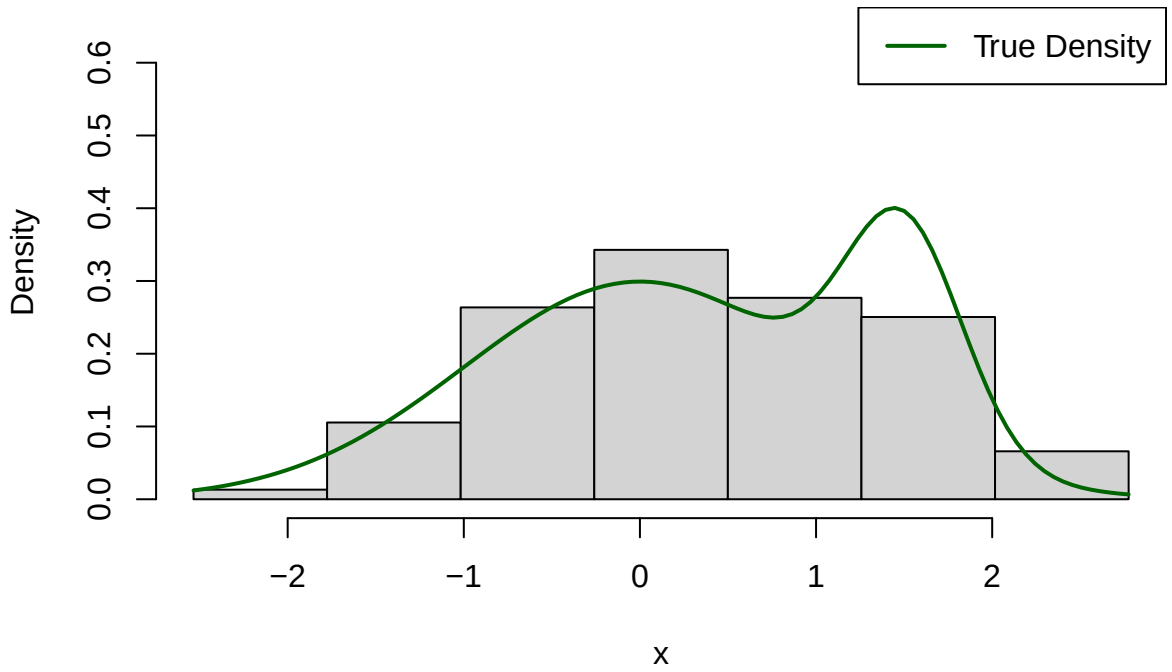
```
b_scott <- 3.49 * sd(x_mix) * (n_mix)^(-1/3)

cat("Optimal bin width from looCV:", optimal_b_mix, "\n")
```

```
## Optimal bin width from looCV: 0.3391106
```

```
cat("Bin width from Scott's rule:", b_scott, "\n")
```

```
## Bin width from Scott's rule: 0.7584686
```

```
# Plot histogram using Scott's rule
hx_scott <- hist(x_mix, breaks = seq(A_mix, Z_mix + b_scott, by = b_scott), plot = FALSE)
plot(hx_scott, freq = FALSE,
     main = paste("Histogram using Scott's Rule (b_Scott =", round(b_scott, 3), ")"),
     xlab = "x", ylim = c(0, 0.65))
curve(graph.mixt, add = TRUE, col = "darkgreen", lwd = 2)
legend("topright", "True Density", col = "darkgreen", lwd = 2)
```



Scott's rule, being based on the assumption of normality, gives a wider bin width which oversmooths the bimodal structure of the data. The looCV method selects a smaller bin width that is better able to capture the two modes of the true density.

Kernel density estimator

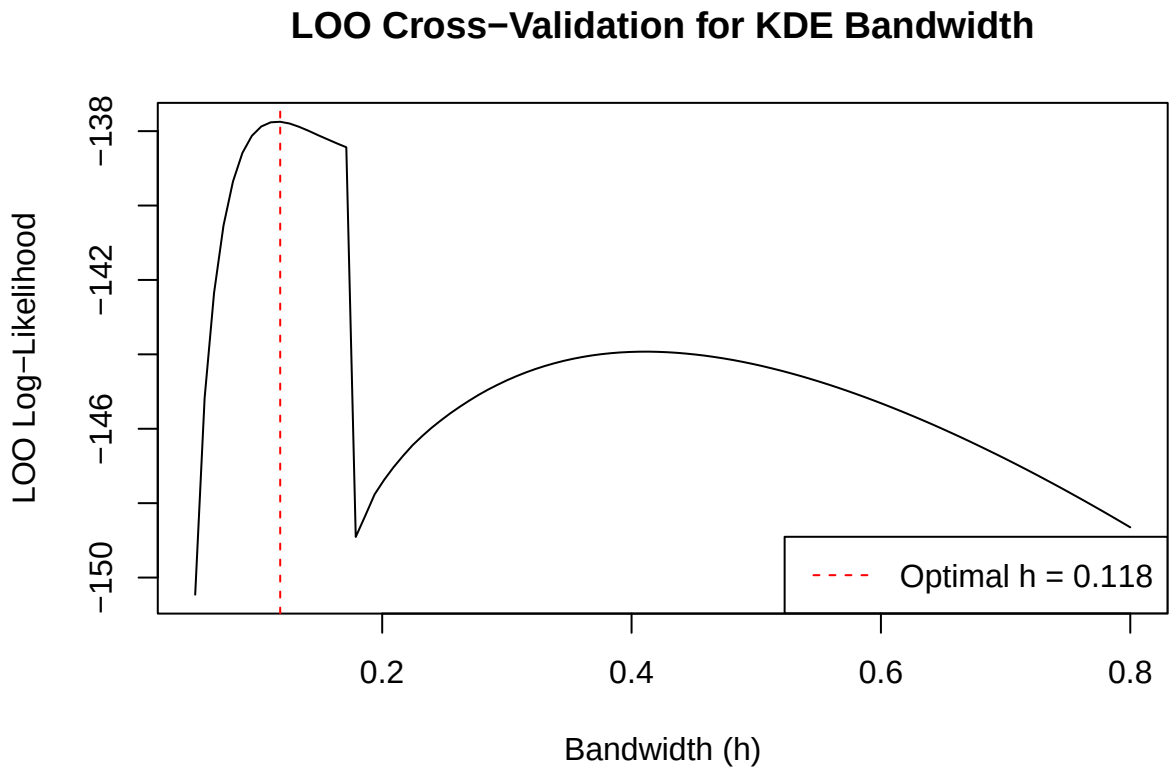
Consider the vector `x` of data you have generated before from the mixture of two normals. Use the relationship

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

to select the value of `h` maximizing the leave-one-out log-likelihood function, and plot the corresponding kernel density estimator. We will use the default Gaussian kernel, for which $K(0) = 1/\sqrt{2\pi}$.

```r
# Use the mixture data x_mix from the previous section
n_mix <- length(x_mix)

# Value of the standard normal kernel at 0
K0 <- dnorm(0)

# Range of bandwidths to test
h_values <- seq(0.05, 0.8, length.out = 100)
looCV_log_lik_h <- numeric(length(h_values))

for (i in seq_along(h_values)) {
  current_h <- h_values[i]

  # Get kernel density estimate on a grid
  kx <- density(x_mix, bw = current_h, kernel = "gaussian")
```

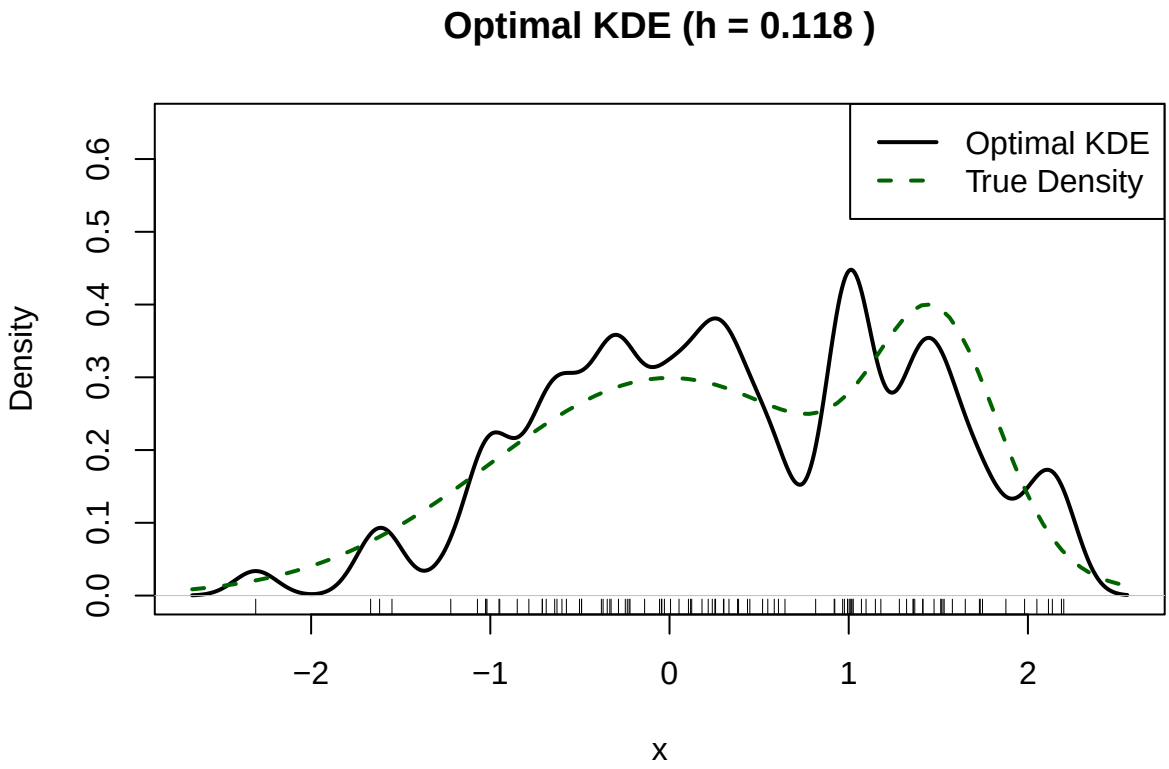
```r
  # Create a function to evaluate the KDE at any point
  kx_f <- approxfun(x = kx$x, y = kx$y, method = 'linear', rule = 2)

  # Evaluate KDE at the observed data points
  f_hat <- kx_f(x_mix)

  # Calculate leave-one-out estimates
  f_loo <- (n_mix / (n_mix - 1)) * (f_hat - K0 / (n_mix * current_h))

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

# Plot log-likelihood vs bandwidth
plot(h_values, looCV_log_lik_h, type = "l",
     xlab = "Bandwidth (h)", ylab = "LOO Log-Likelihood",
     main = "LOO Cross-Validation for KDE Bandwidth")

# Find the optimal h
optimal_h <- h_values[which.max(looCV_log_lik_h)]
abline(v = optimal_h, col = "red", lty = 2)
legend("bottomright", legend = paste("Optimal h =", round(optimal_h, 3)), col = "red", lty = 2)
```



Plot the corresponding kernel density estimator using the optimal bandwidth `h`.

```r
# Compute the final KDE with the optimal bandwidth
kde_optimal <- density(x_mix, bw = optimal_h, kernel = "gaussian")

# Plot the KDE
```

```r
plot(kde_optimal, main = paste("Optimal KDE (h =", round(optimal_h, 3), ")"),
     xlab = "x", lwd = 2, ylim = c(0, 0.65))

# Overlay the true density
curve(graph.mixt, add = TRUE, col = "darkgreen", lwd = 2, lty = 2)

# Add the data points as a rug
rug(x_mix)

legend("topright", c("Optimal KDE", "True Density"),
       col = c("black", "darkgreen"), lwd = 2, lty = c(1, 2))
```



The kernel density estimator with the bandwidth selected by leave-one-out cross-validation provides a very good approximation of the true bimodal density function.