Cluster Analysis and other unsupervised learning methods

Density estimation

Werner Stuetzle
Department of Statistics
University of Washington

March 26, 2017

```
## Preamble

stat593.dir <- "/Users/wxs/Dropbox/Unsupervised-learning-spring-2016"
dir.sep <- "/"

data.dir <- paste(stat593.dir, "Data", sep = dir.sep)
tc.dir <- paste(data.dir, "Test-collection", sep = dir.sep)

echo <- F
## quartz()

source(paste(stat593.dir, "Code", "gsl-functions-5-28-2010.R",
             sep = dir.sep), echo = echo)

source(paste(data.dir, "generate-artificial-data-functions-3-19-2014.R",
             sep = dir.sep), echo = echo)

library(MASS)
library(cluster)
library(colorspace)
library(mvtnorm)

options(expressions = 10000)

opts_chunk$set(fig.width=5, fig.height=5)
```

Generate data for illustrations

Make samples from a univariate 3 component Gaussian mixture.

```
nobs <- 100
nrep <- 100
ngrid <- 500

mixture.par <- NULL
Sigma = matrix(1, nrow = 1, ncol = 1)
mixture.par[[1]] <- list(p = 0.4, mu = 0, Sigma = Sigma)  ## (pi, mean, var)
mixture.par[[2]] <- list(p = 0.4, mu = 4, Sigma = Sigma)
mixture.par[[3]] <- list(p = 0.2, mu = 10, Sigma = Sigma)

set.seed(1)
samples <- matrix(0, nrow = nobs, ncol = nrep)
for (i in 1:nrep) {
  samples[, i] <- rgmm(nobs, mixture.par, multinomial = T)$X
}

x <- samples[, 1]
ngrid <- 500
```

The density estimation problem

Given:
$\mathbf{x}_1, \ldots, \mathbf{x}_n \in R^m$ assumed to be iid observations of a random vector $\mathbf{X}$ with unknown density $p(\mathbf{x})$

Goal: Estimate p

Will first look at univariate case ($m = 1$)

The histogram estimate

Choose grid origin g and bin width h

Define bin $B_i = [g + (i-1)h, g + ih)$ for $i = \ldots -2, -1, 0, 1, 2, \ldots$.
Define bin count $n_i = \sum_j I(x_j \in B_i)$
The histogram estimate $\hat{p}_{hist}$ is piecewise constant over bins:

$$\hat{p}_{hist}(x) = \sum_i c_i \, I(x \in B_i)$$

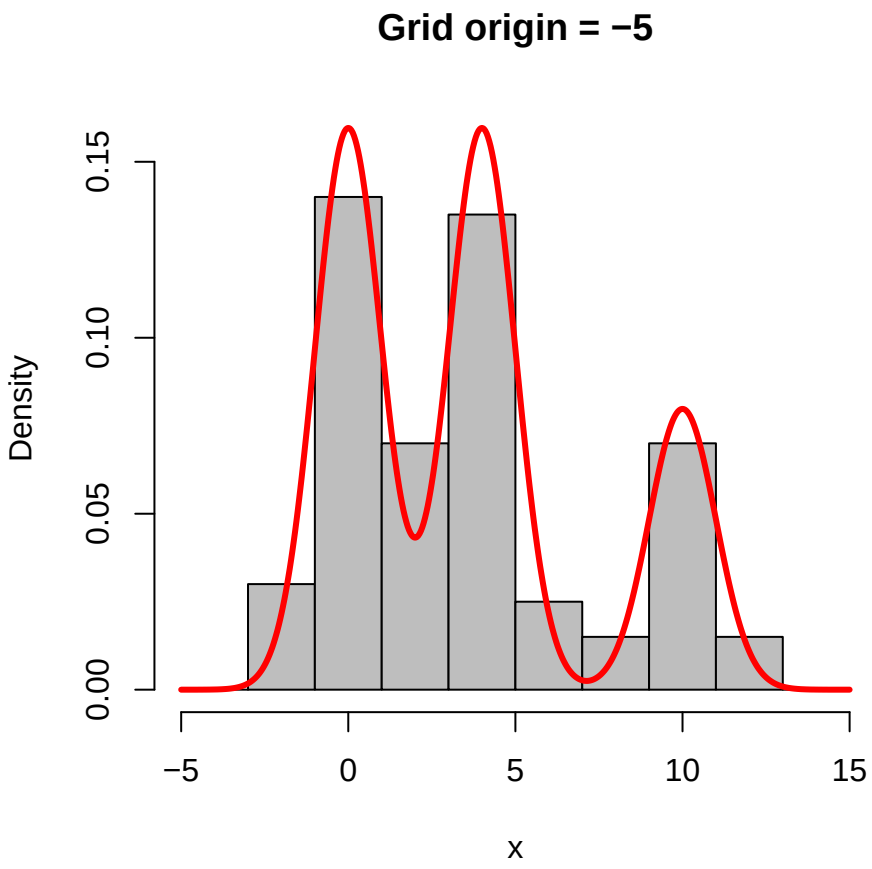
The constants c_i are chosen to make the expected bin counts equal to the observed bin counts: $n\,h\,c_i = n_i$ or $c_i = n_i/(nh)$.

Note: The estimate $\hat{p}_{hist}$ depends critically on the bin width h. It also depends on the grid origin g.

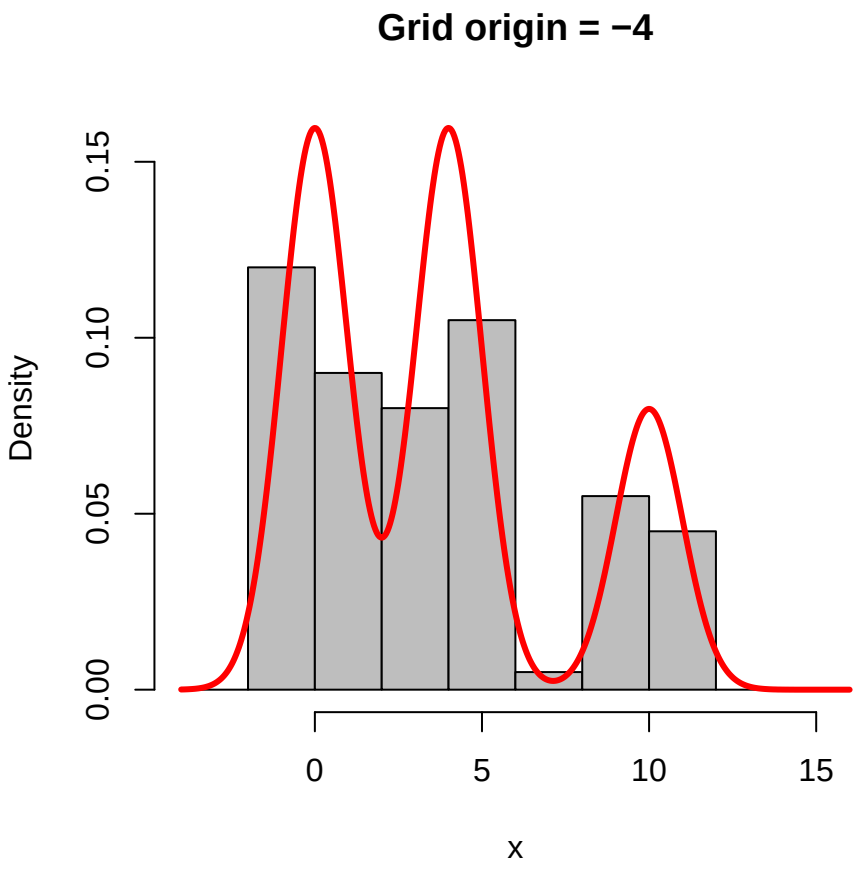
Illustrate dependence on grid origin

Draw histograms of 3 component mixture sample with bin width $h = 2$ for grid origins $g = -5$ and $g = -4$.

```
##-------------------------------------------------------------------
## Draw histogram and true density for bin width = 2, grid origin = -5

nbin <- 10
lower <- -5
upper <- 15

grid <- seq(from = lower, to = upper, length = ngrid)
Grid <- matrix(grid, ncol = 1)
ptrue <- dgmm(Grid, mixture.par)

breaks <- seq(from = lower, to = upper, length = nbin + 1)
hist.out <- hist(x, breaks = breaks, right = F, freq = F,
                 ylim = c(0, max(ptrue)), col = "grey", main = "")
title("Grid origin = -5")
lines(grid, ptrue, col = "red", lwd = 3)
```




```r
##-------------------------------------------------------------------
## Now the same for grid origin = -4

nbin <- 10
lower <- -4
upper <- 16

grid <- seq(from = lower, to = upper, length = ngrid)
Grid <- matrix(grid, ncol = 1)
ptrue <- dgmm(Grid, mixture.par)

breaks <- seq(from = lower, to = upper, length = nbin + 1)
hist.out <- hist(x, breaks = breaks, right = F, freq = F,
                 ylim = c(0, max(ptrue)), col = "grey", main = "")
lines(grid, ptrue, col = "red", lwd = 3)
title("Grid origin = -4")
```

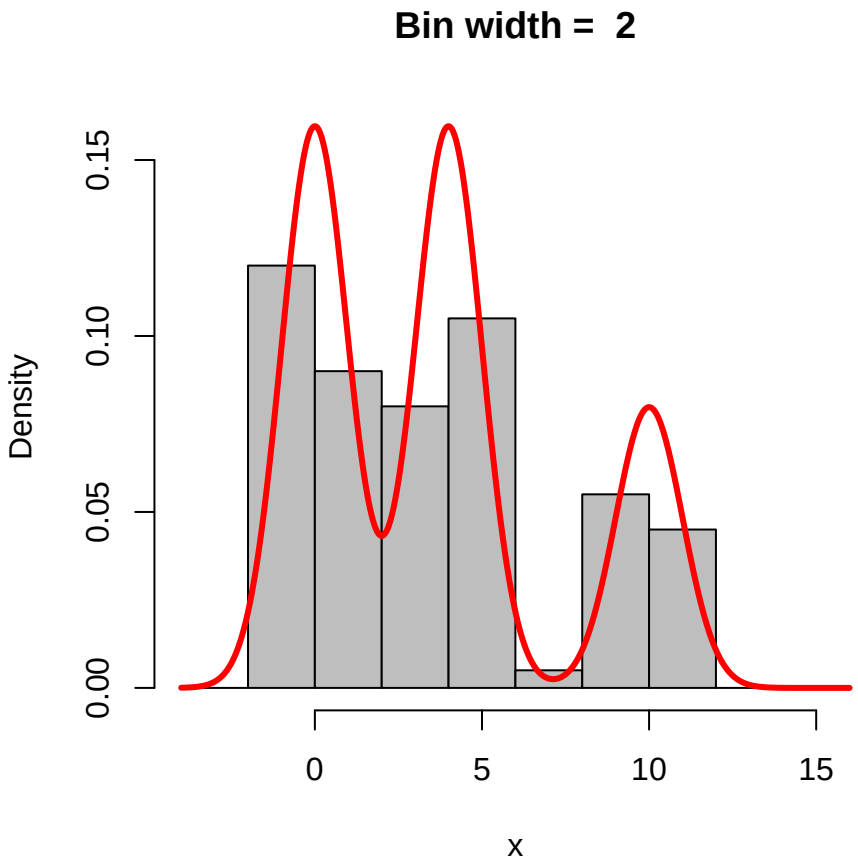


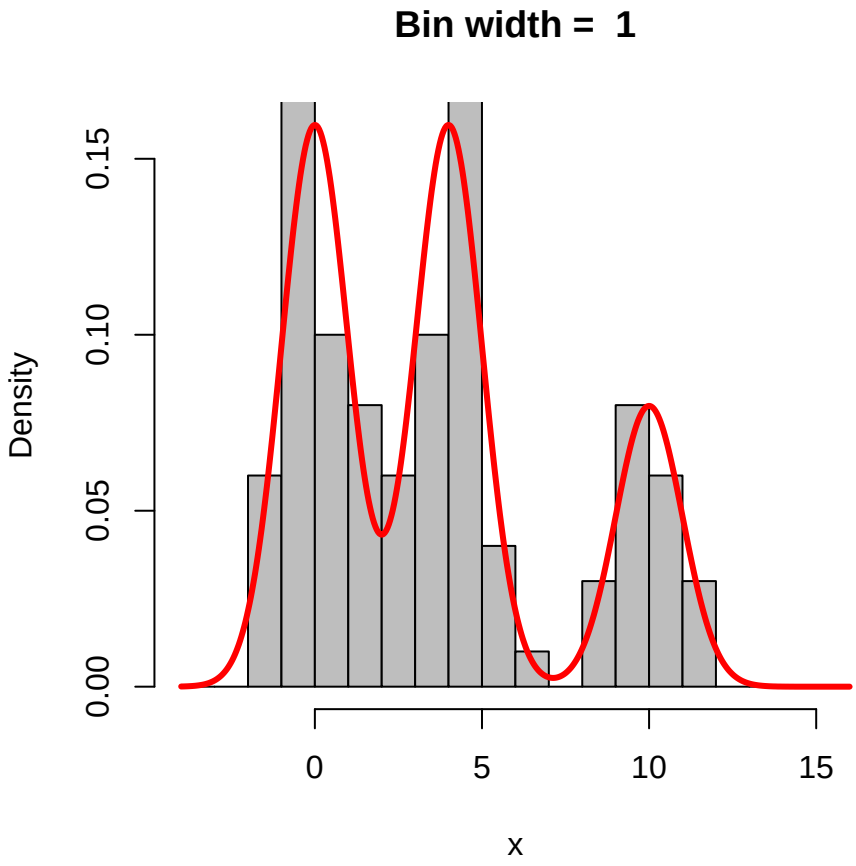
Illustrate dependence on bin width

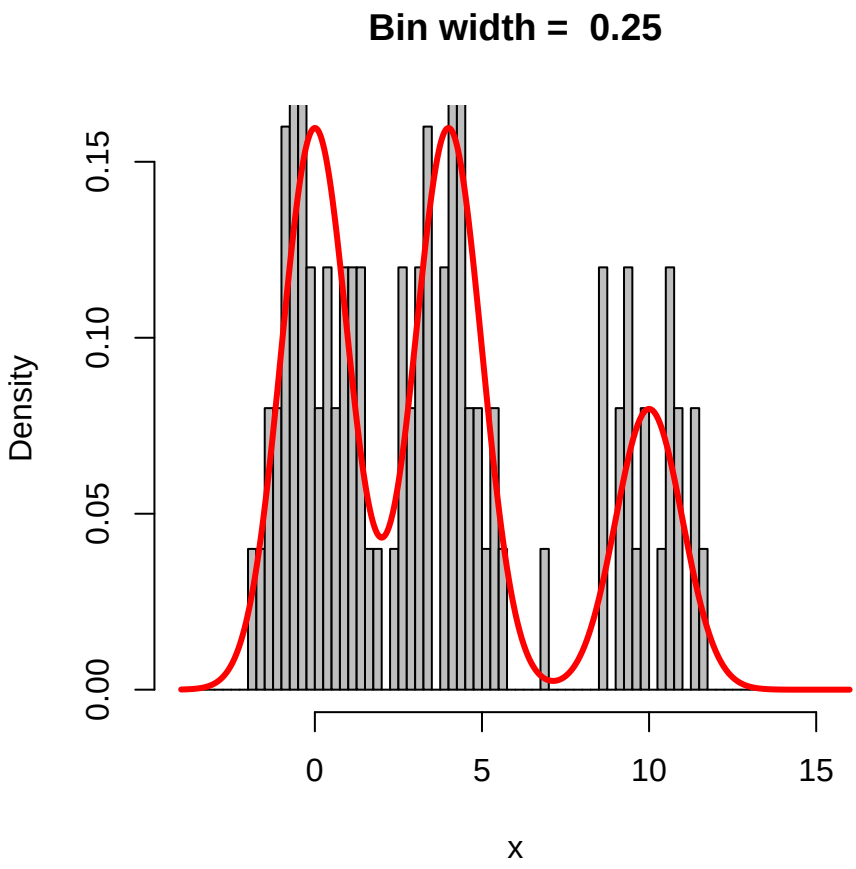
Draw histograms for $g = -4$ and bin widths $h = 2, 1, 0.25$.

```
lower <- -4
upper <- 16

grid <- seq(from = lower, to = upper, length = ngrid)
Grid <- matrix(grid, ncol = 1)
ptrue <- dgmm(Grid, mixture.par)

for (h in c(2, 1, 0.25)) {
  nbin <- (upper - lower) / h
  breaks <- seq(from = lower, to = upper, length = nbin + 1)
  hist.out <- hist(x, breaks = breaks, right = F, freq = F,
                   ylim = c(0, max(ptrue)), col = "grey", main = "")
```

```
  title(paste("Bin width = ", h))
  lines(grid, ptrue, col = "red", lwd = 3)
}
```




Bin width = 1
Density
0.00
0.05
0.10
0.15
0
5
10
15
x



Message: Small bin width results in noisy estimates.

The average shifted histogram estimate

A clever idea to reduce dependence on grid origin:

- Pick grid origin g_1, bin width h, and spacing $s = h/k$ for some k (for example $k = 10$).
- Define $g_i = g_1 + (i-1)s, i = 1, \ldots, k$ and let $\hat{p}^i_{hist}$ be the the histogram estimate for grid origin g_i and bin width h.
- Define $\hat{p}_{ash}(x) = \textbf{mean}\,(\hat{p}^i_{hist}(x))$.

Kernel density estimates

Idea: Instead of fixing a grid and estimating $p(x)$ from the empirical probability content of the bin containing x, center the bin at x:

$$\hat{p}(x) = 1/(nh) \sum_i I(x - h/2 \leq x_i \leq x + h/2).$$

Why does this make sense?

$$\sum_i I(x - h/2 \leq x_i \leq x + h/2) \sim \text{Binomial}(n, q)$$

with $q = \int_{x-h/2}^{x+h/2} p(u)\,\mathrm{d}u$. Therefore

$$\begin{aligned} \mathbf{E}\,\hat{p}(x) &= q/h \\ \mathbf{var}\,\hat{p}(x) &= 1/(nh^2)\, q(1-q) \end{aligned}$$

Small $h \Rightarrow p$ approximately constant over the the bin $\Rightarrow q \approx hp(x) \Rightarrow \mathbf{E}\,\hat{p}(x) \approx p(x)$.
Large $n \Rightarrow \mathbf{var}\,\hat{p}(x)$ small.
Asymptotically (for $n \to \infty$) choice of $h(n) \to 0$ and $nh(n) \to \infty$ will result in consistent estimate.
More detailed analysis later.

Alternative view

Define

$$\begin{aligned} K(x) &= I(-1/2 \leq x \leq 1/2) \\ K(x,h) &= 1/h\; K(x/h) \qquad \text{“Parzen kernel with bandwidth h”} \end{aligned}$$

Easy to see: $K(x,h)$ is symmetric around 0 and $\int K(x,h)dx = 1$.
We can rewrite the kernel estimate:

$$\hat{p}(x) = 1/(nh) \sum_i K(\frac{x - x_i}{h}) = 1/n \sum_i K(x - x_i, h)$$

Interpretation:

- Center (a copy of) the kernel at each of the observations x_i
- Average the values of the kernels at x.

Smooth kernels

Disadvantages of Parzen kernel:

- Density estimate is a step function (non-smooth)
- All observations in the window get the same weight, idependent of their distance form the query point x.

Idea: Use smooth kernels, for example

$$\begin{aligned} K(x) &= \frac{1}{\sqrt{2\pi}} e^{\frac{x^2}{2}} && \text{Gaussian kernel} \\ K(x) &= I(|x| \leq 1)\, \frac{15}{16}(1-x^2)^2 && \text{Biweight kernel} \end{aligned}$$

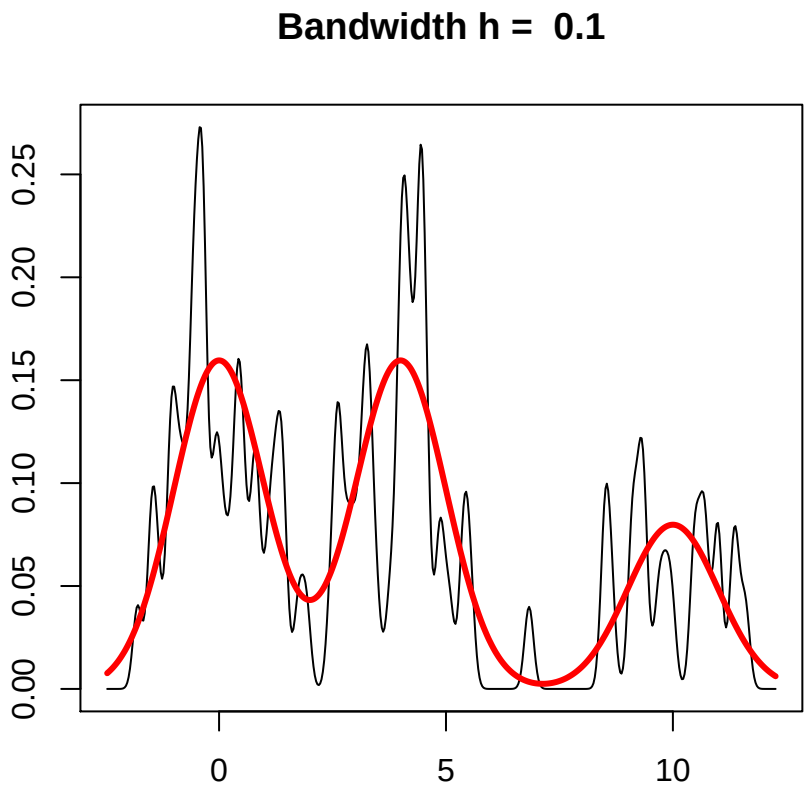
Note: Kernels should satisfy $K(x) \geq 0$, $K(x) = K(-x)$, and $\int K(x)\,\mathrm{d}x = 1$.

Influence of bandwidth - Illustration

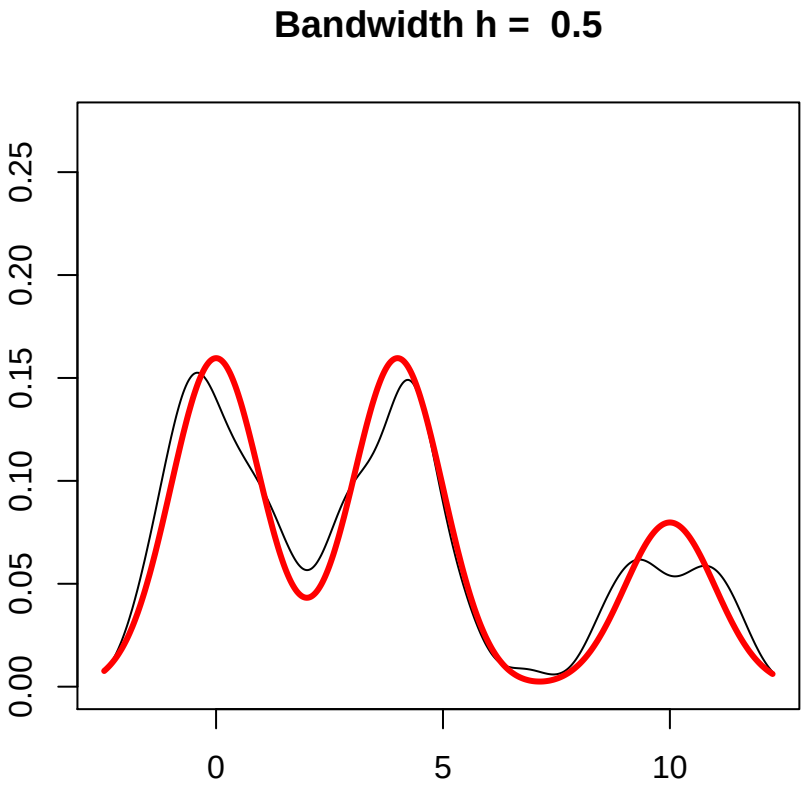
Apply Gaussian kernel density estimate with different bandwidths to Gaussian mixture data. Red curve = true density, black curve = kernel estimate.

```
r <- max(x) - min(x)
lower <- min(x) - 0.05 * r
upper <- max(x) + 0.05 * r
grid <- seq(from = lower, to = upper, length = ngrid)
Grid <- as.matrix(grid, ncol = 1)

h <- c(0.1, 0.5, 1)
nh <- length(h)

all.curves <- matrix(0, nrow = nh + 1, ncol = ngrid)
for (i in 1:nh) {
  all.curves[i, ] <- gaussian.kernel.density.estimate(x, grid, h[i])
}
all.curves[nh + 1,] <- dgmm(Grid, mixture.par)

ylim <- c(0, max(all.curves))
```

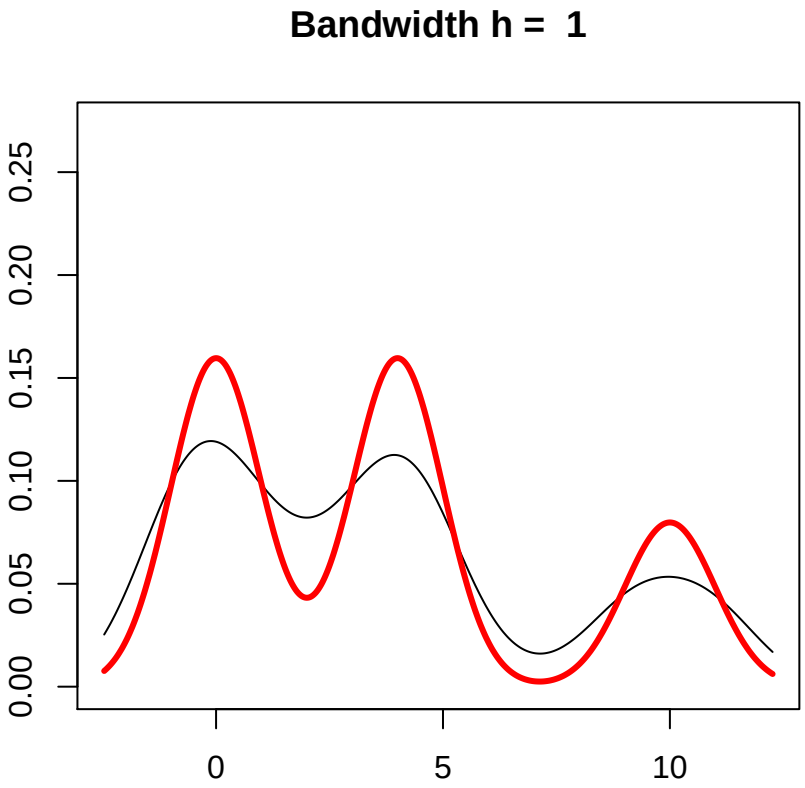
```
plot(grid, all.curves[1,], type = "l", ylim = ylim, lwd = 1, xlab = "",
     ylab = "")
lines(grid, all.curves[4,], col = "red", lwd = 3)
title(paste("Bandwidth h = ", h[1]))
```




```
plot(grid, all.curves[2,], type = "l", ylim = ylim, lwd = 1, xlab = "",
     ylab = "")
lines(grid, all.curves[4,], col = "red", lwd = 3)
title(paste("Bandwidth h = ", h[2]))
```




```
plot(grid, all.curves[3,], type = "l", ylim = ylim, lwd = 1, xlab = "",
     ylab = "")
lines(grid, all.curves[4,], col = "red", lwd = 3)
title(paste("Bandwidth h = ", h[3]))
```



Message: Small bandwidth $\Rightarrow$ noisy estimate. Large bandwidth $\Rightarrow$ smooth estimate, but erodes peaks and fills up valleys.

Influence of bandwidth - Simulation experiment

Define $\hat{p}(x, h, \mathcal{X})$ = value at x of Gaussian kernel density estimate with bandwidth h applied to "training sample" $\mathcal{X}$.

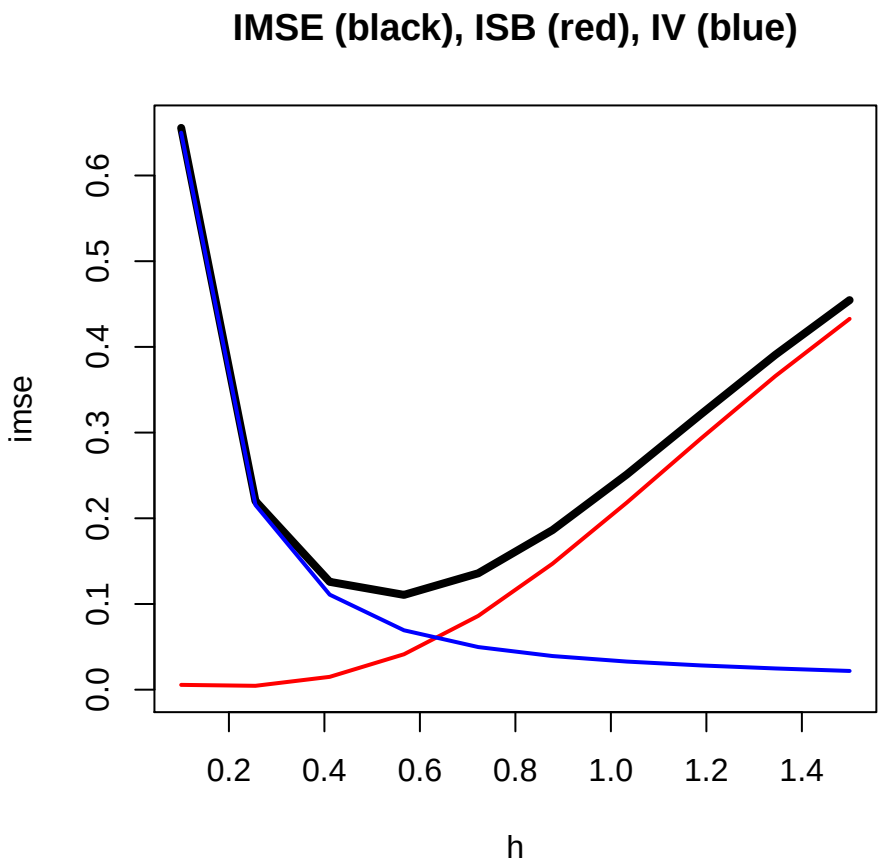
Define

$$\begin{aligned} \text{bias}(x,h) &= \mathbf{E}\,\hat{p}(x,h,\mathcal{X}) - p(x) \\ \text{var}(x,h) &= \mathbf{var}\,\hat{p}(x,h,\mathcal{X}) \\ \text{ese}(x,h) &= \mathbf{E}\,(\hat{p}(x,h,\mathcal{X})) - p(x))^2 \\ &= \text{bias}^2(x,h) + \text{var}(x,h) \end{aligned}$$

Expectations are taken over training samples.

Will now explore dependence of the integrated expected squared error $\text{iese}(h) = \int_x \text{ese}(x,h)dx$ and its compenents $\text{ibias}^2(h)$ and $\text{ivar}(h)$ on the bandwidth h through a simulation experiment.

```
nh <- 10
h <- seq(from = 0.1, to = 1.5, length = nh)
r <- max(samples) - min(samples)
grid <- seq(from = min(samples) - 0.1 * r, to = max(samples) + 0.1 * r,
            length = ngrid)
ptrue <- dgmm(matrix(grid, ncol = 1), mixture.par)
phat <- matrix(0, nrow = ngrid, ncol = nrep)

isb <- rep(0, nh)
iv <- rep(0, nh)

for (i in 1:nh) {
  for (j in 1:nrep) {
    phat[, j] <- gaussian.kernel.density.estimate(samples[, j], grid, h[i])
    b <- apply(phat, 1, mean) - ptrue
    v <- apply(phat, 1, var)
    isb[i] <- sum(b^2)
    iv[i] <- sum(v)
  }
  cat("* ")
}

## * * * * * * * * * *

imse <- isb + iv
plot(h, imse, type = "l", col = "black", ylim = c(0, max(imse)), lwd = 4)
lines(h, isb, col = "red", lwd = 2)
lines(h, iv, col = "blue", lwd = 2)
title("IMSE (black), ISB (red), IV (blue)")
```



Message: Result agrees with intuition:

- Integrated variance IV (blue curve) is decreasing in h.
- Integrated square bias ISB (red curve) is increasing in h.
- There is a sweet spot ($h \approx 0.6$) where the integrated mean squared error IMSE (black curve) is minimized.

Bandwidth selection by least squares cross-validation (LSCV)

Choose bandwidth h to minimize (an estimate of) the integrated squared error

$$\begin{aligned} \text{ise}(h) &= \int (\hat{p}(x,h) - p(x))^2 \, \mathrm{d}x \\ &= \int \hat{p}^2(x,h) \, \mathrm{d}x \qquad \text{can be computed} \\ &\quad -2 \int \hat{p}(x,h) p(x) \, \mathrm{d}x \qquad \text{can be estimated (see below)} \\ &\quad + \int p^2(x) \, \mathrm{d}x \qquad \text{does not depend on } h \end{aligned}$$

Supress the dependence of $\hat{p}$ on the training sample.

Don't know $p(x) \Rightarrow$ can't compute

$$\int \hat{p}(x,h)p(x)\,\mathrm{d}x = \mathbf{E}\,\hat{p}(x,h)$$

but we do have sample $x_1, \ldots, x_n$ from $p(x)$.
Naive estimate:

$$\mathbf{E}\,\hat{p}(x,h) \approx \frac{1}{n}\sum \hat{p}(x_i,h)$$

leading to

$$\begin{aligned} \hat{h}_{naive} &= \arg\min_h \Big(\int \hat{p}^2(x,h)\,\mathrm{d}x - \frac{2}{n}\sum \hat{p}(x_i,h)\Big) \\ &= \arg\min_h (\phi_{naive}(h)) \end{aligned}$$

However, for $h \to 0$, $2/n \sum \hat{p}(x_i,h) \to \infty$ faster than $\int \hat{p}^2(x,h)\,\mathrm{d}x \Rightarrow \hat{h}_{naive} = 0$.

Aside: Calculation for Parzen kernel

For h small enough x_i is the only observation in $B_i = [x_i - \frac{h}{2}, x_i + \frac{h}{2}] \Rightarrow$

$$\begin{aligned} \hat{p}(x_i,h) &= \frac{1}{nh} \\ \frac{1}{n}\sum \hat{p}(x_i,h) &= \frac{1}{nh} \end{aligned}$$

and

$$\begin{aligned} \hat{p}^2(x_i,h) &= \frac{1}{n^2h^2}\sum_{i,j} I(x \in B_i)I(x \in B_j) = \frac{1}{n^2h^2}\sum_i I(x \in B_i) \qquad \Rightarrow \\ \int \hat{p}^2(x,h) &= \frac{1}{n^2h^2}\sum \int I(x \in B_i)dx = \frac{1}{nh} \qquad \Rightarrow \\ \phi_{naive} &= \frac{1}{nh} - \frac{2}{nh} = -\frac{1}{nh} \end{aligned}$$

So $\phi_{naive}(h)$ is minimized for $h = 0$.

Problem: $\sum \hat{p}(x_i,h)$ measures how well the density estimate fits the observations (similar to the log likelihood).
However, we use the same data for calculating the estimate and evaluating the estimate $\Rightarrow$ optimistic.

Need to use cross-validation. Define

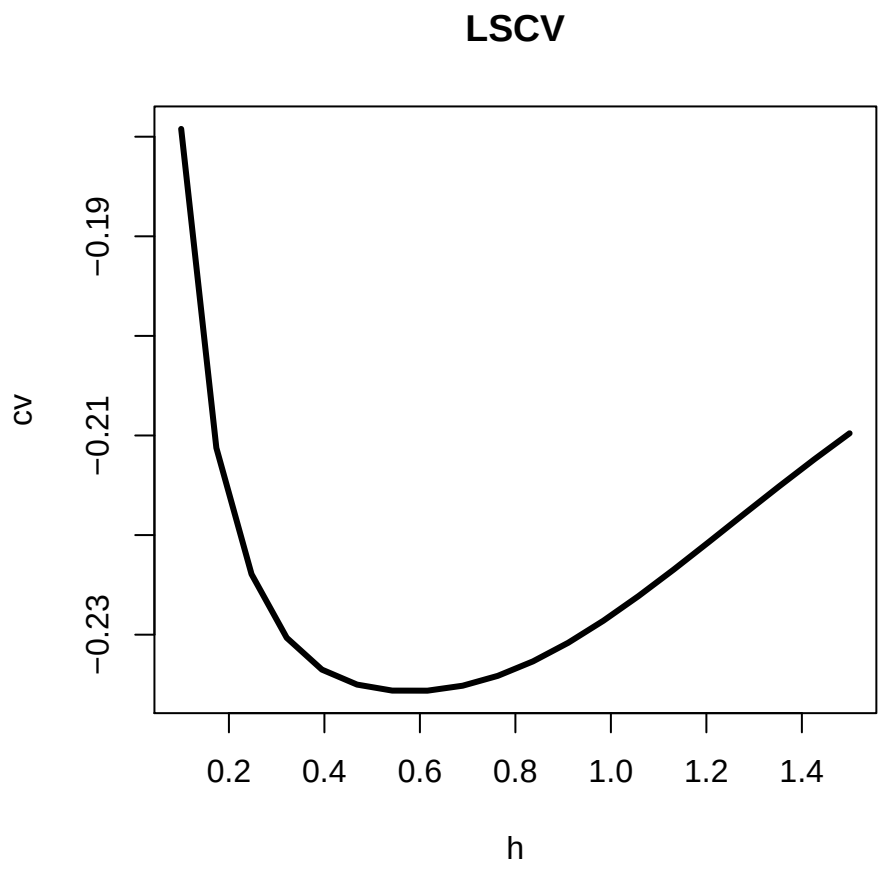
$$\begin{aligned}\hat{p}^{-i}(x_i, h) &= \frac{1}{(n-1)h}\sum_{j\neq i} K(\frac{x_i - x_j}{h}) \\ \hat{h}_{LSCV} &= \arg\min(\int \hat{p}^2(x,h)dx - \frac{2}{n}\sum \hat{p}^{-i}(x_i,h) \\ &= \arg\min(\phi_{LSCV}(h))\end{aligned}$$

Evaluate goodness of fit at x_i without using observation x_i

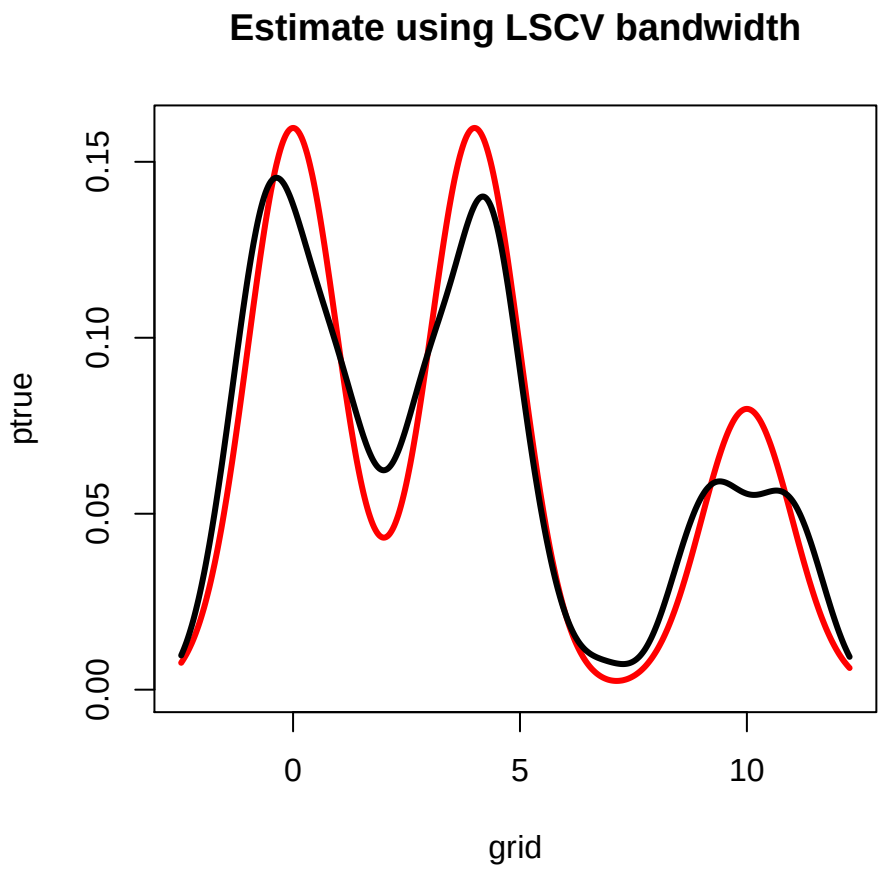
---

LSCV - Illustration

```
nh <- 20
h <- seq(from = 0.1, to = 1.5, length = 20)
cv <- gaussian.kernel.least.squares.cv(x, h)
plot(h, cv, type = "l", lwd = 3)
title("LSCV")
```



```r
hopt <- cv.search(x)$opt.smopar
hopt

## [1] 0.5803022

ngrid <- 500
r <- max(x) - min(x)
lower <- min(x) - 0.05 * r
upper <- max(x) + 0.05 * r
grid <- seq(from = lower, to = upper, length = ngrid)
Grid <- matrix(grid, ncol = 1)
ptrue <- dgmm(Grid, mixture.par)

phat <- gaussian.kernel.density.estimate(x, grid, hopt)
ymax <- max(c(ptrue, phat))

plot(grid, ptrue, type = "l", lwd = 3, ylim = c(0, ymax), col = "red")
lines(grid, phat, lwd = 3)
title("Estimate using LSCV bandwidth")
```

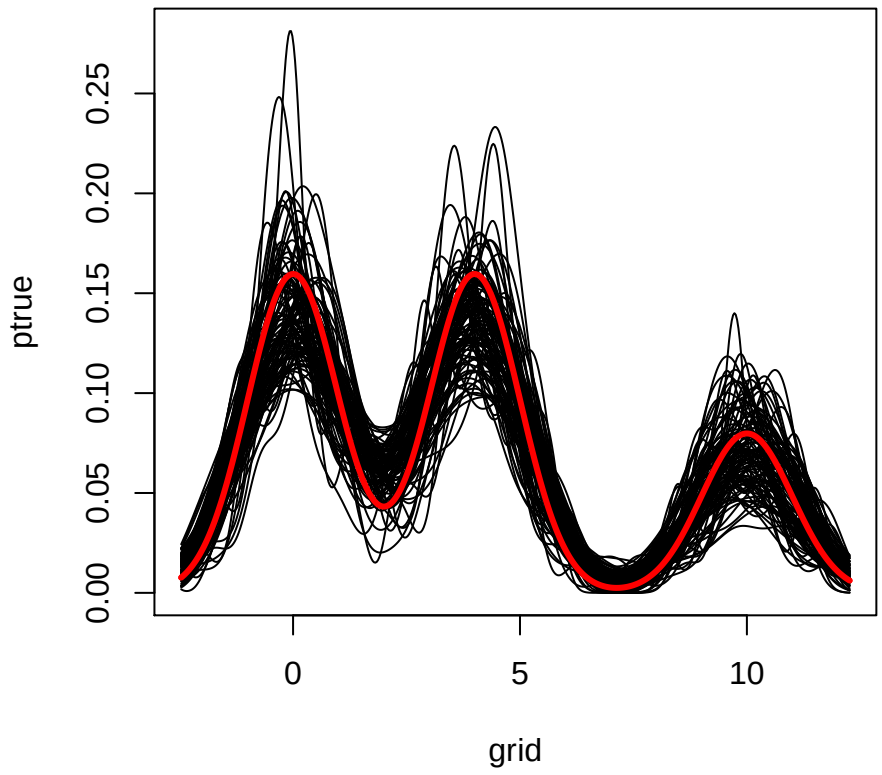


LSCV - Simulation experiment

Run LSCV on 100 samples from our mixture density.

```
cv.search.out <- vector("list", nrep)
hopt <- rep(0, nrep)
for (i in 1:nrep) {
  x <- samples[, i]
  cv.search.out[[i]] <- cv.search(x)
  hopt[i] <- cv.search.out[[i]]$opt.smopar
  cat(" ", i)
}

##   1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23

## Count number of successes
```

```r
ngood <- sum(!is.na(hopt))

phat <- matrix(0, nrow = ngrid, ncol = ngood)
ngood <- 0
for (i in 1:nrep) {
  if (is.na(hopt[i])) next
  ngood <- ngood + 1
  phat[, ngood] <- gaussian.kernel.density.estimate(samples[, i], grid, hopt[i])
}

ymax <- max(c(phat, ptrue))

nshow <- ngood
plot(grid, ptrue, type = "l", col = "red", lwd = 3, ylim = c(0, ymax))
for (i in 1:nshow) {
  lines(grid, phat[, i])
}
lines(grid, ptrue, col = "red", lwd = 3)
```



Multivariate Kernel density estimation

Given:
$\mathbf{x}_1, \ldots, \mathbf{x}_n \in R^m$ assumed to be iid observations of a random vector $\mathbf{X}$ with unknown density $p(\mathbf{x})$

Goal: Estimate p

Straight-forward generalization of univariate case

Choose kernel $K(\mathbf{x})$ with

$$\begin{aligned} K(\mathbf{x}) &\geq 0 \\ K(\mathbf{x}) &= K(-\mathbf{x}) \\ \int K(\mathbf{x})\,\mathrm{d}\mathbf{x} &= 1 \end{aligned}$$

Define

$$\begin{aligned} K(\mathbf{x}, h) &= \frac{1}{h^m} K(\frac{\mathbf{x}}{h}) \\ \hat{p}(\mathbf{x}, h, \mathcal{X}) &= \frac{1}{n} K((\mathbf{x} - \mathbf{x}_i), h) = \frac{1}{nh^m} K(\frac{\mathbf{x} - \mathbf{x}_i}{h}) \end{aligned}$$

Choices for kernel:

Product kernels:

- Choose univariate kernel, for example Parzen kernel $K(x) = I(-1/2 \leq x \leq 1/2)$
- Define multivariate kernel $K(\mathbf{x}) = \prod_i K(x_i)$

Spherical kernels:

- Choose weight function $w(r)$, for example $w(r) = e^{-\frac{r^2}{2}}$
- Define $K(\mathbf{x}) = cw(\|\mathbf{x}\|)$, where $c = 1/\int w(\|x\|)\,\mathrm{d}\mathbf{x}$.

Note: The only spherical product kernel is the standard Gaussian kernel.

Standardizing and sphering

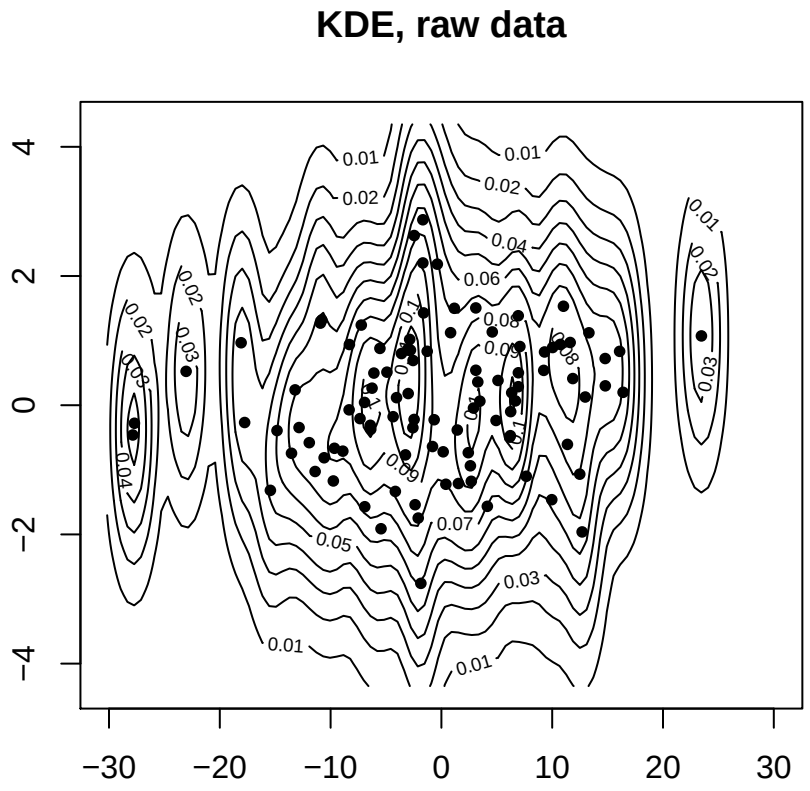
Using a spherical kernel or a product kernel does not give good results if the variables have widely different spreads or are highly correlated.

```
##=====================================================================
## Make data
nobs <- 100
m <- 2
sd1 <- 10
sd2 <- 1
X <- matrix(rnorm(nobs*m), nrow = nobs)
X[, 1] <- sd1 * X[, 1]
X[, 2] <- sd2 * X[, 2]

Sigma <- var(X)
mu <- apply(X, 2, mean)

##---------------------------------------------------------------------
## Estimate optimal bandwidth by LSCV

cvs <- cv.search(X)
h <- cvs$opt.smopar
phat <- make.gaussian.kernel.density.estimate(X, h)

##---------------------------------------------------------------------
## Evaluate estimate over regular grid
## Choose grid limits such that that density estimate is essentially
## 0 outside the box.

ngrid <- 100
sd1.conv <- sqrt(sd1^2 + h^2)
sd2.conv <- sqrt(sd2^2 + h^2)
x1.lim <- c(-3 * sd1.conv, 3 * sd1.conv)
x2.lim <- c(-3 * sd2.conv, 3 * sd2.conv)
x1.grid <- seq(from = x1.lim[1], to = x1.lim[2], length = ngrid)
x2.grid <- seq(from = x2.lim[1], to = x2.lim[2], length = ngrid)
Grid <- matrix(0, nrow = ngrid^2, ncol = 2)
Grid[, 1] <- rep(x1.grid, rep(ngrid, ngrid))
Grid[, 2] <- rep(x2.grid, ngrid)
phat.grid <- phat(Grid)

##---------------------------------------------------------------------
## Just to be sure: verify that density estimate has unit mass

x1.spacing <- x1.grid[2] - x1.grid[1]
x2.spacing <- x2.grid[2] - x2.grid[1]
cell.volume <- x1.spacing * x2.spacing
mass <- sum(phat.grid) * cell.volume
```

```
mass
```

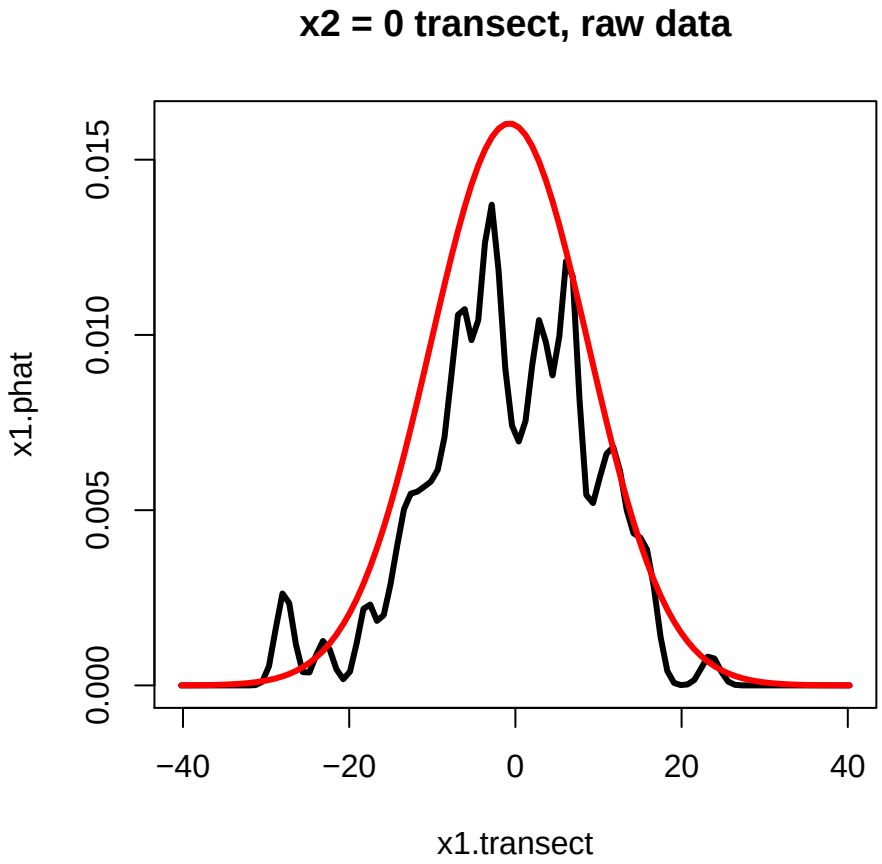
```
## [1] 0.996935
```

```
##------------------------------------------------------------------
## Draw contour plot for density estimate

z <- matrix(phat.grid, nrow = ngrid, ncol = ngrid, byrow = T)
contour(x = x1.grid, y = x2.grid, sqrt(z))
points(X[,1:2], pch = 20)
title("KDE, raw data")
```

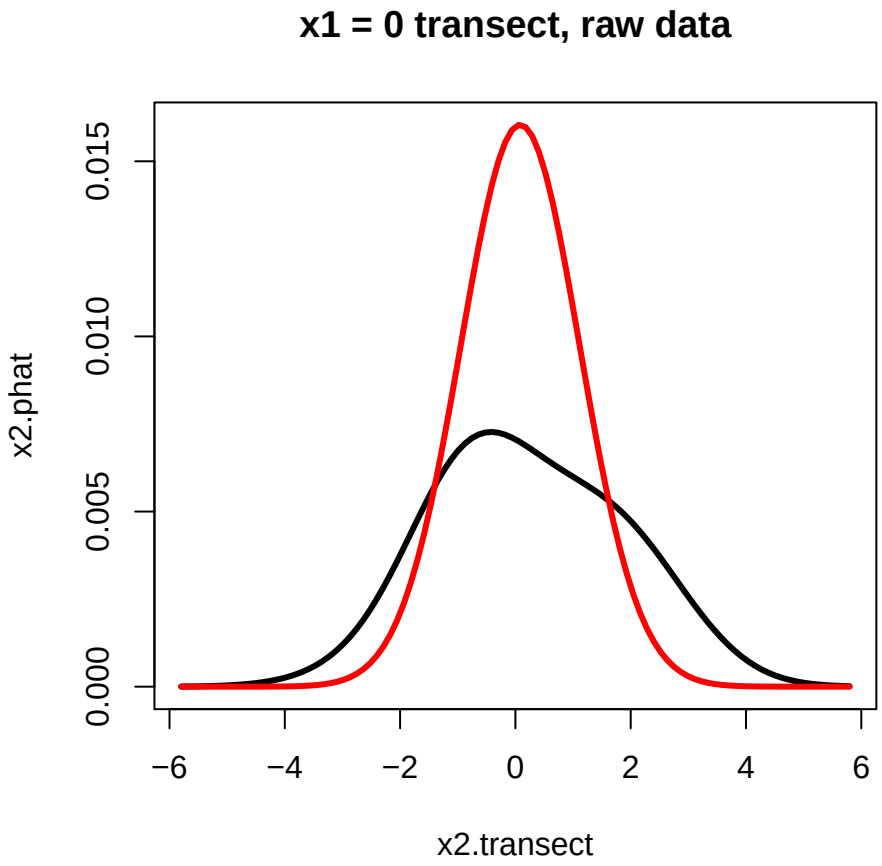



```
##------------------------------------------------------------------
## Plot x1 transect at x2 = 0

x1.transect <- seq(from = - 4*sd1.conv, to = 4*sd1.conv, length = ngrid)
x1.p <- dmvnorm(cbind(x1.transect, rep(0, ngrid)), mu, Sigma)
x1.phat <- phat(cbind(x1.transect, rep(0, ngrid)))
```

```
ymax <- max(c(x1.p, x1.phat))
plot(x1.transect, x1.phat, type = "l", lwd = 3, ylim = c(0, ymax),
     main = "")
lines(x1.transect, x1.p, type = "l", col = "red", lwd = 3)
title("x2 = 0 transect, raw data")
```




```
## Plot x2 transect at x1 = 0

x2.transect <- seq(from = - 4*sd2.conv, to = 4*sd2.conv, length = ngrid)
## X2.p <- dnorm(X2.transect, sd = sd) * dnorm(0, sd = 1)
x2.p <- dmvnorm(cbind(rep(0, ngrid), x2.transect), mu, Sigma)
x2.phat <- phat(cbind(rep(0, ngrid), x2.transect))
ymax <- max(c(x2.p, x2.phat))
plot(x2.transect, x2.phat, type = "l", lwd = 3, ylim = c(0, ymax),
     main = "")
lines(x2.transect, x2.p, type = "l", col = "red", lwd = 3)
title("x1 = 0 transect, raw data")
```



Idea: Replace bandwidth h by $m \times m$ bandwidth matrix H. Define

$$\begin{aligned} K(\mathbf{x}, H) &= \det(H^{-1}) K(H^{-1}\mathbf{x}) \\ \hat{p}(\mathbf{x}, H, \mathcal{X}) &= \frac{1}{n} \sum K(\mathbf{x} - \mathbf{x}_i, H) \end{aligned}$$

Common choics for H:

1. $H = hI$: appropriate if variables have similar spreads. For standard Gaussian kernel K the isopleths of $K(\mathbf{x}, H)$ are spheres.

2. $H = h(\text{diag}\Sigma)^{1/2}$ where Σ = covariance matrix of sample: Stretch the kernel along axes for which the spread of the data is large. For standard Gaussian kernel K the isopleths of $K(\mathbf{x}, H)$ are axis parallel ellipsoids.

3. $H = hR^T$ where $R^T R$ is the Choleski decomposition of Σ: Stretch kernel along directions for which the spread of the data is large. For standard Gaussian kernel K the isopleths of $K(\mathbf{x}, H)$ are ellipsoids whose axes are the principal component directions of the data.

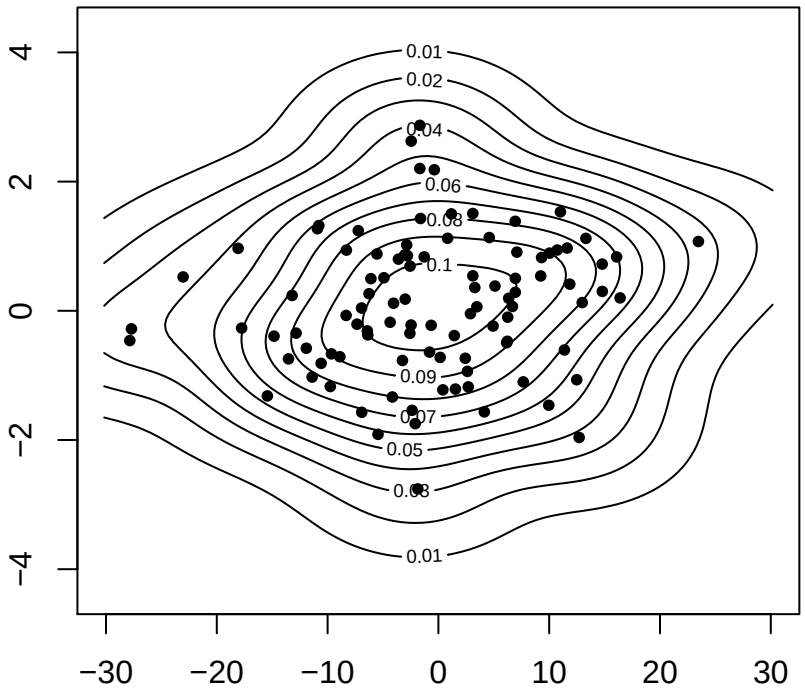
The overall scale parameter h can be estimated by LSCV.

Note: A different way of thinking about H:

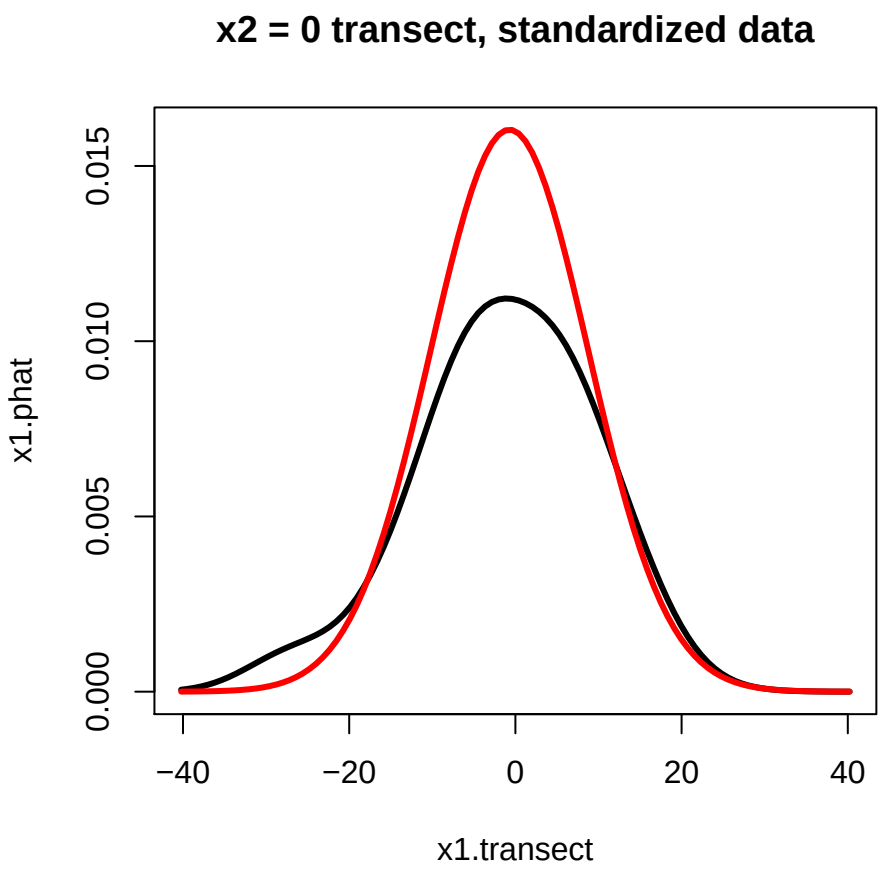
- Transform the data so that the transformed data are homoscedastic (option (2), standardization) or have unit sample covariance (option (3), sphering)
- Estimate density in the transformed data with a spherical kernel
- Carry the estimate back to the original space

Illustration: Standardization can improve the performance of a kernel density estimate if the data are heteroscedastic.

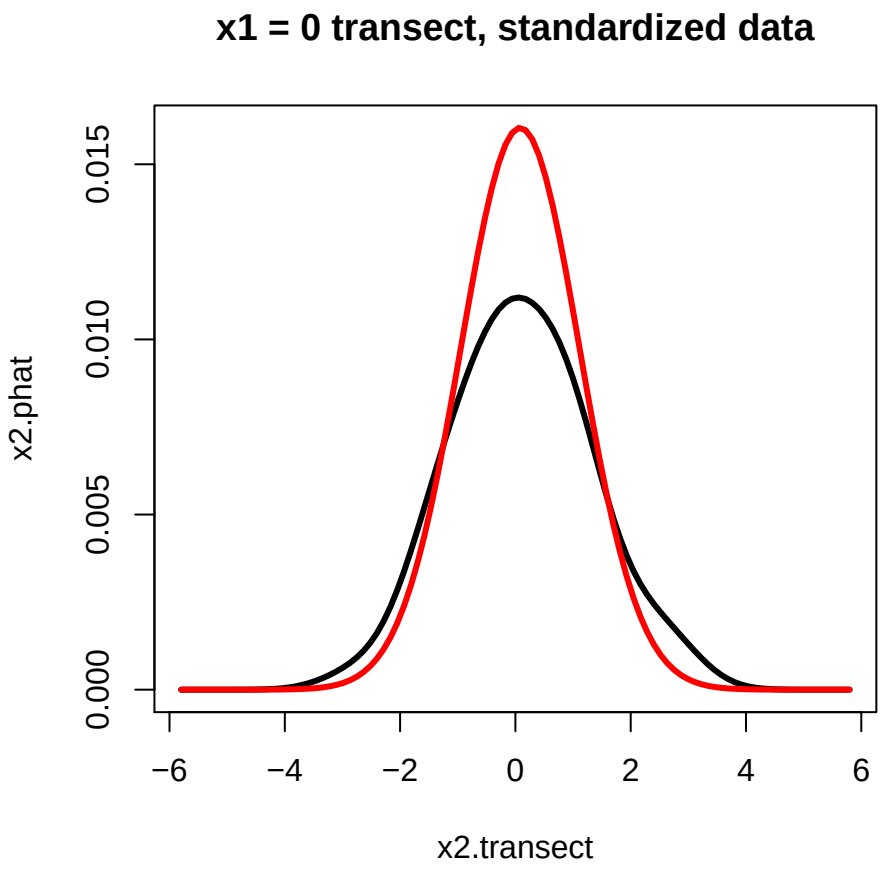
```
##====================================================================
## Do the same experiment as above but with standardization

Hinv <- diag(1/sqrt(diag(Sigma)))
det.Hinv <- Hinv[1, 1] * Hinv[2, 2]
X.stand <- X %*% Hinv

cvs <- cv.search(X.stand)
h <- cvs$opt.smopar
phat <- make.gaussian.kernel.density.estimate(X.stand, h)

##--------------------------------------------------------------------
## Just to be sure: verify that density estimate has unit mass
phat.grid <- phat(Grid %*% Hinv) * det.Hinv
mass <- sum(phat.grid) * cell.volume
mass

## [1] 0.9916454

##--------------------------------------------------------------------
## Draw contour plot for density estimate

z <- matrix(phat.grid, nrow = ngrid, ncol = ngrid, byrow = T)
contour(x = x1.grid, y = x2.grid, sqrt(z))
points(X[,1:2], pch = 20)
title("KDE, standardized data")
```




```
##-------------------------------------------------------------------
## Plot x1 transect at x2 = 0

x1.transect <- seq(from = - 4*sd1.conv, to = 4*sd1.conv, length = ngrid)
x1.p <- dmvnorm(cbind(x1.transect, rep(0, ngrid)), mu, Sigma)
x1.phat <- phat(cbind(x1.transect, rep(0, ngrid)) %*% Hinv) * det.Hinv
ymax <- max(c(x1.p, x1.phat))
plot(x1.transect, x1.phat, type = "l", lwd = 3, ylim = c(0, ymax))
lines(x1.transect, x1.p, type = "l", col = "red", lwd = 3)
title("x2 = 0 transect, standardized data")
```




```
## Plot x2 transect at x1 = 0

x2.transect <- seq(from = - 4*sd2.conv, to = 4*sd2.conv, length = ngrid)
x2.p <- dmvnorm(cbind(rep(0, ngrid), x2.transect), mu, Sigma)
x2.phat <- phat(cbind(rep(0, ngrid), x2.transect) %*% Hinv) * det.Hinv
ymax <- max(c(x2.p, x2.phat))
plot(x2.transect, x2.phat, type = "l", lwd = 3, ylim = c(0, ymax))
lines(x2.transect, x2.p, type = "l", col = "red", lwd = 3)
title("x1 = 0 transect, standardized data")
```



Remarks

- One could in principle estimate the elements of H by cross-validation. High-dimensional optimization problem. Will increase variance of density estimate. May be practical for large samples in low dimensions
- There are situation where H should depend on $\mathbf{x}$. Hard to do; may be practical for large samples in low dimensions.

Asymptotics for multivariate KDE

Quantities of interest:

$$\begin{aligned} \mathbf{ese}\,(\mathbf{x}) &= \mathbf{E}\left((\hat{p}(\mathbf{x}, H, \mathcal{X}) - p(\mathbf{x}))^2\right) \quad \text{expected squared error at } \mathbf{x} \\ \mathbf{iese} &= \int_{\mathbf{X}} \mathbf{ese}\,(\mathbf{x})\,\mathrm{d}\mathbf{x} \quad \text{integrated expected squared error} \end{aligned}$$

Expectation is over training samples.
To simplify notation, will write $\hat{p}(\mathbf{x})$ instead of $\hat{p}(\mathbf{x}, H, \mathcal{X})$.

$\mathbf{ese}\,(\mathbf{x})$ depends on true density p, query point $\mathbf{x}$, kernel K, sample size n, and bandwidth matrix H.
We would hope $\mathbf{ese}\,(\mathbf{x})$ and $\mathbf{iese} \to 0$ for proper choice of $H = H(n)$.

Expected squared error = variance + squared bias:

$$\begin{aligned}
\mathbf{ese}\,(\mathbf{x}) &= \mathbf{E}\,(\hat{p}(\mathbf{x}) - p(\mathbf{x}))^2 \\
&= \mathbf{E}\,(\hat{p}(\mathbf{x}) - \mathbf{E}\,(\hat{p}(\mathbf{x})))^2 + (\mathbf{E}\,(\hat{p}(\mathbf{x})) - p(\mathbf{x}))^2 \\
&= \mathbf{var}\,(\hat{p}(\mathbf{x})) + \mathbf{bias}^2(\hat{p}(\mathbf{x}))
\end{aligned}$$

For simplicity consider uniform spherical kernel with radius h:

$$K(\mathbf{x}, h) = \frac{1}{h^m v} I(\mathbf{x} \in S_h)$$

where v = volume of unit sphere in R^m and S_h = sphere of radius h centered at $\mathbf{0}$.
Wlog assume query point $\mathbf{x} = \mathbf{0}$.

Will first calculate the bias term:

$$\begin{aligned}
\hat{p}(\mathbf{0}) &= \frac{1}{nvh^m} \sum_i \mathrm{I}(\mathbf{x}_i \in S_h) \qquad \text{and therefore} \\
\mathbf{E}\,(\hat{p}(\mathbf{0})) &= \frac{1}{vh^m} \int_{S_h} p(\mathbf{x})\,\mathrm{d}\mathbf{x} = \bar{p}_h(\mathbf{0})
\end{aligned}$$

Not surprising: $\mathbf{E}\,(\hat{p}(\mathbf{0}))$ = average of p over S_h.

Let $\mathbf{g}(\mathbf{0})$ and $A(\mathbf{0})$ denote gradient and Hessian of p at $\mathbf{x} = \mathbf{0}$.
By expanding p in a Taylor series around $\mathbf{0}$:

$$\begin{aligned}
p(\mathbf{x}) &\approx p(\mathbf{0}) + \mathbf{g}(\mathbf{0}) \cdot \mathbf{x} + \frac{1}{2}\mathbf{x}^T A(\mathbf{0})\mathbf{x} \qquad \text{we obtain} \\
\bar{p}_h(\mathbf{0}) &\approx p(\mathbf{0}) + \frac{1}{2vh^m} \int_{S_h} \mathbf{x}^t A(\mathbf{0})\mathbf{x}\,\mathrm{d}\mathbf{x}
\end{aligned}$$

The linear term drops out for symmetry reasons.
Change of variables: $\mathbf{y} = \mathbf{x}/h$ gives

$$\begin{aligned}
\int_{S_h} \mathbf{x}^t A(\mathbf{0})\mathbf{x}\,\mathrm{d}\mathbf{x} &= \int_{S_1} h^2 \mathbf{y}^T A(\mathbf{0})\mathbf{y}\; h^m\,\mathrm{d}\mathbf{y} \qquad \text{and} \\
\bar{p}_h(\mathbf{0}) &\approx p(\mathbf{0}) + h^2 \frac{1}{2v} \int_{S_1} \mathbf{y}^T A(\mathbf{0})\mathbf{y}\,\mathrm{d}\mathbf{y}
\end{aligned}$$

So $\mathbf{bias}\,(\hat{p}(\mathbf{0})) = \bar{p}_h(\mathbf{0}) - p(\mathbf{0}) = O(h^2)$. To get a consistent estimate we need $h(n) \to 0$ as $n \to \infty$.

Next look at variance term:

$$\begin{aligned}
\mathbf{var}\,(\hat{p}(\mathbf{0})) &= \frac{1}{nv^2h^{2m}}\,\mathbf{var}\,\mathrm{I}(\mathbf{x}_i \in S_h) \\
\mathbf{var}\,(\mathrm{I}(\mathbf{x}_i \in S_h)) &= P(S_h) - P^2(S_h) \Rightarrow \\
\mathbf{var}\,(\hat{p}(\mathbf{0})) &= \frac{1}{nvh^m}\bar{p}_h(\mathbf{0}) - \frac{1}{n}\bar{p}_h(\mathbf{0})^2
\end{aligned}$$

$h(n) \to 0 \Rightarrow$ second term is of smaller order $\Rightarrow$

$$\begin{aligned}
\mathbf{var}\,(\hat{p}(\mathbf{0})) &\approx \frac{1}{nvh^m}(p(\mathbf{0}) + (\bar{p}_h(\mathbf{0}) - p(\mathbf{0})) \\
&= \frac{p(\mathbf{0})}{nvh^m} + \text{terms of smaller order}
\end{aligned}$$

To get a consistent estimate we need $nh(n) \to \infty$.

Putting things together:

$$\begin{aligned}
\mathbf{ese}\,(\mathbf{0}) &= c^2h^4 + \frac{p(\mathbf{0})}{nvh^m} \qquad \text{where} \\
c &= \frac{1}{2v}\int_{S_1} \mathbf{y}^T A\mathbf{y}\,\mathrm{d}\mathbf{y}
\end{aligned}$$

Diferentiating wrt h shows that the optimal choice is $h(n) \sim n^{-\frac{1}{m+4}}$, resulting in $\mathbf{ese}\,(\mathbf{0}) \sim n^{-\frac{4}{m+4}}$.

Remarks:

- Result not immediately useful because we don't know the Hessian of p.
- Convergence rate is slower than $1/n$ rate we encounter in typical parameter estimation problems. Slower rate = price for not having to make prior assumptions about the shape of p.
- Growth rate of $h(n)$ amazingly slow. Theory suggests that for or $m = 10$ doubling the sample size calls for a 5% increase in bandwidth.

Nearest neighbor density estimates

Given: $\mathbf{x}_1, \ldots, \mathbf{x}_n \in R^m$ i.i.d. $\sim p(\mathbf{x})$

Goal: Estimate p

$S(\mathbf{x}, r)$: Sphere around query point $\mathbf{x}$ with radius r

k : Number of observations in $S(\mathbf{x}, r)$
V_m: Volume of unit sphere in R^m

$$\begin{aligned} \mathbf{E}(k) &= n \int_{S(\mathbf{x},r)} p(\mathbf{y})\, \mathrm{d}\mathbf{y} \\ &\approx n\, r^m\, V_m\, p(\mathbf{x}) \end{aligned}$$

Replace $\mathbf{E}(k)$ by its observed value $\Rightarrow$

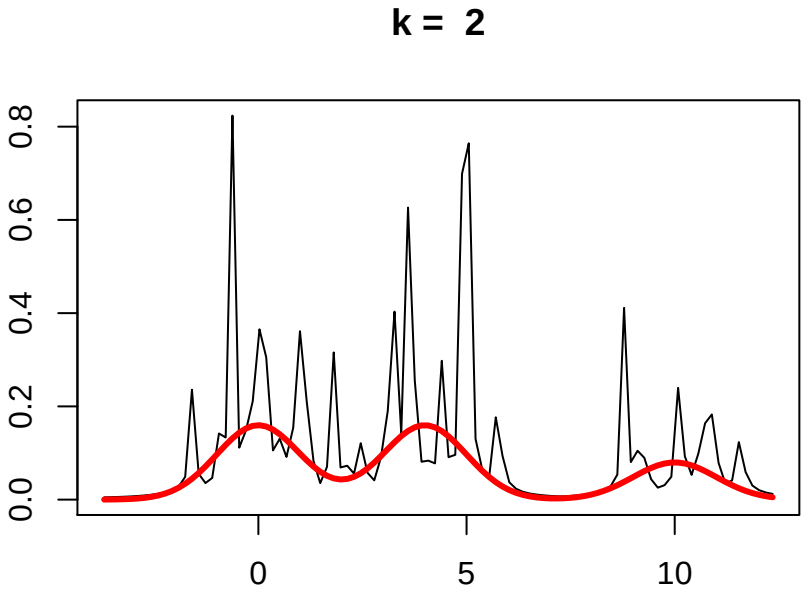
$$\hat{p}(\mathbf{x}) = \frac{k}{n\, r^m\, V_m}$$

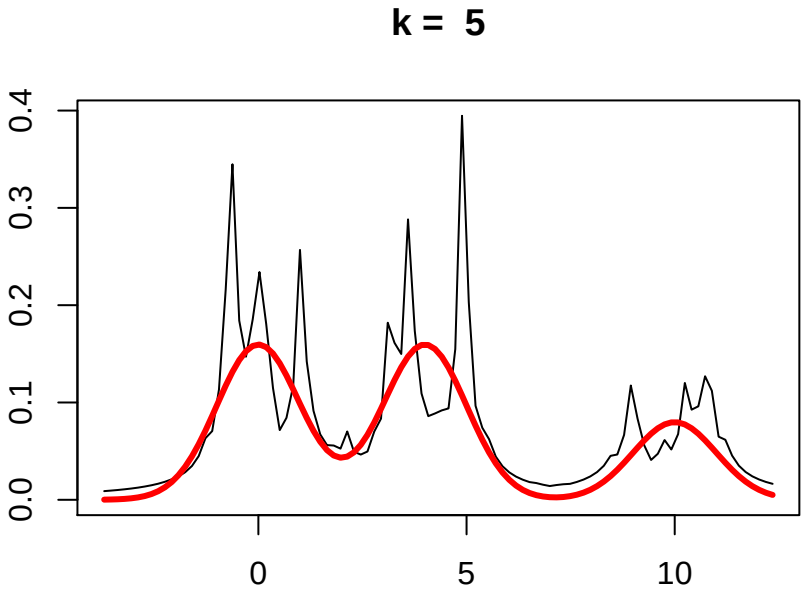
Kernel estimate: Fix r and observe k

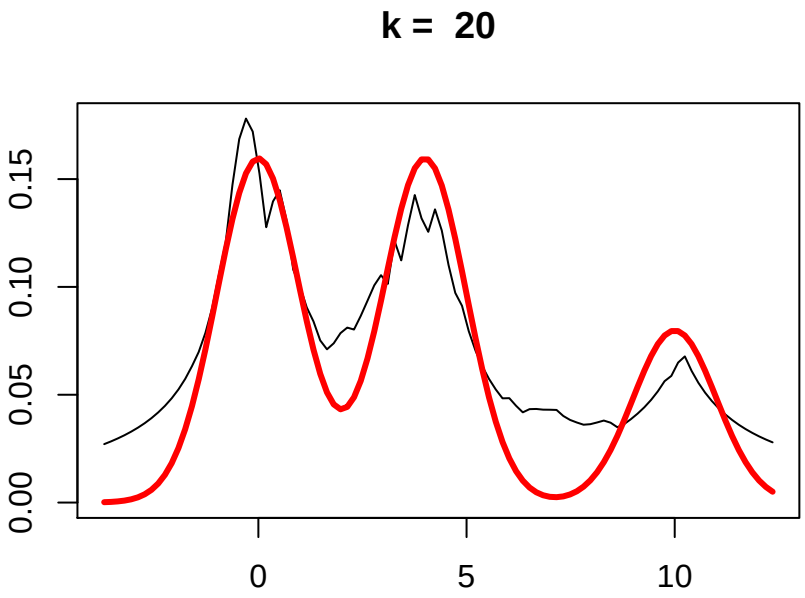
Nearest neighbor estimate: Fix k and observe r.
r = radius of smallest sphere around $\mathbf{x}$ containing k observations.

Influence of k - Illustration

Apply knn density estimate with different values of k to Gaussian mixture data. Red curve = true density, black curve = knn estimate.







Disadvantages of knn density estimates

- Heavy tails
- Cannot be normalized

- Spiky
- No automatic method to choose k

However, there are applications (classification) where it works surprisingly well.