03 - Kernel Density Estimation

SYS 4582/6018 | Spring 2019

03-kde.pdf

Contents

1 Non-Parametric Density Estimation Intro

1.1 Required R Packages

We will be using the R packages of:

- `tidyverse` for data manipulation and visualization
- `ks` for kernel density estimation
- `mixtools` for mixture modeling

```
library(tidyverse)     # install.packages("tidyverse")
library(ks)            # install.packages("ks")
library(mixtools)      # install.packages("mixtools")
```

2 Density Histograms

2.1 Example: Old Faithful

The old faithful geyser in Yellowstone National Park is one of the most regular geysers in the park. The waiting time between eruptions is between 35 and 120 mins.

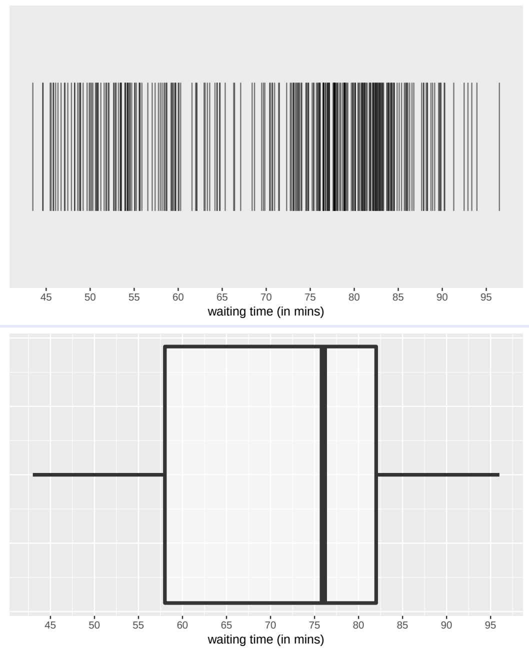
Live Streaming Webcam with eruption predictions

Because the nearby Yellowstone Lodge is nice and warm in the winter, and serves good ice cream in the summer, you may be distracted from stepping outside to watch the eruption. Let's see if we can determine the best time to go out and watch.

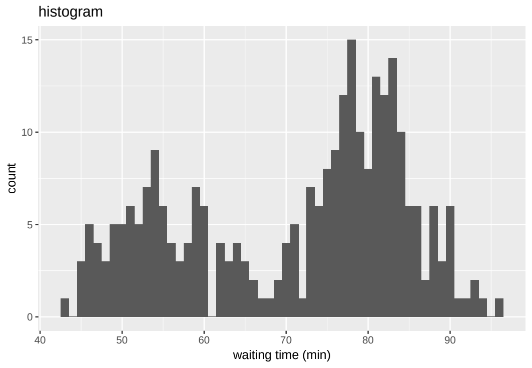
Your Turn #1 : Old Faithful

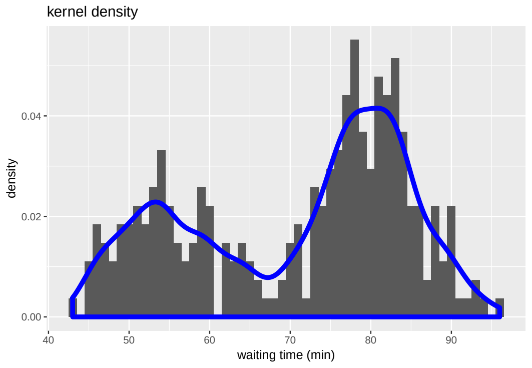
The data, summary statistics, and plots below represent a sample of *waiting times*, the time (in min) between Old Faithful eruptions.

```
#-- Load the Old Faithful data
wait = datasets::faithful$waiting

#-- Calculate summary stats
length(wait)             # sample si
#> [1] 272
summary(wait)            # six numbe
#>    Min. 1st Qu.  Median    Mean
#>    43.0    58.0    76.0    70.9
mean(wait)               # mean
#> [1] 70.9
sd(wait)
#> [1] 13.59
median(wait)
#> [1] 76
quantile(wait, probs=c(.25,.50,.75)
#> 25% 50% 75%
#>  58  76  82
```



```
#-- Put data into a data.frame/tibble for use with ggplot
wait.df = tibble(wait)

#-- Make a ggplot object
pp = ggplot(wait.df, aes(x=wait)) + xlab("waiting time (min)")

#-- Histogram
pp + geom_histogram(binwidth = 1) + ggtitle("histogram")

#-- overlay kernel density plot
pp + geom_histogram(binwidth = 1, aes(y=stat(density))) +  # *density* histogram
  geom_density(bw=2, size=2, color="blue") + ggtitle("kernel density")
```





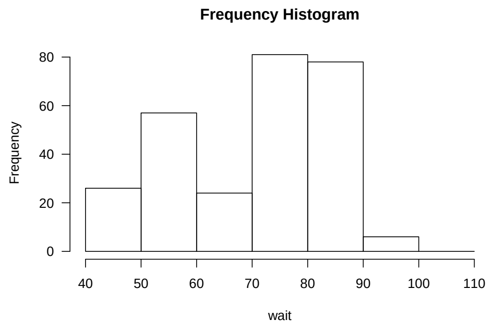
1. What can you say about the shape of the distribution?
2. Would a Gaussian (i.e., Normal) Distribution be a good choice for modeling the distribution of these data?
3. What would you recommend?

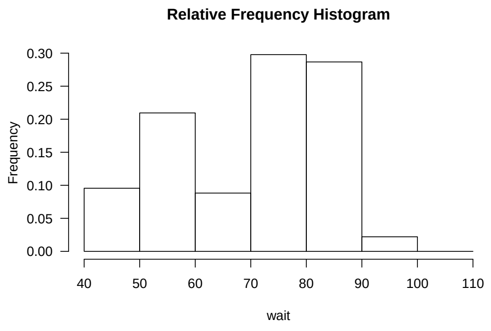
2.2 Histograms

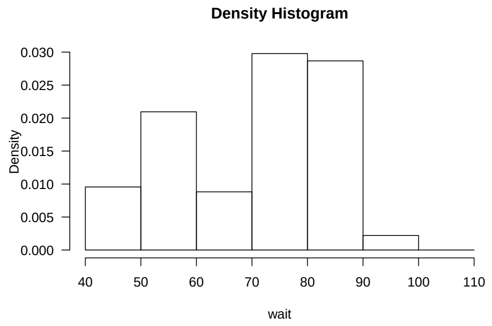
- A histogram is a visual representation of a *piece-wise constant* function.
- There are three primary types of histograms: **frequency**, **relative frequency**, and **density**.

```
#-- Histogram settings
bw = 10                         # binwidth parameter
bks = seq(40, 110, by=bw)       # create a sequence of numbers

#-- Frequency Histogram
hist(wait, breaks=bks, las=1, main="Frequency Histogram")

#-- Relative Frequency Histogram
h.rf = hist(wait, breaks=bks, plot=FALSE)
h.rf$counts = h.rf$counts/sum(h.rf$counts)    # make relative frequency
plot(h.rf, las=1, main="Relative Frequency Histogram")
```

```
#-- Density Histogram
hist(wait, freq=FALSE, breaks=bks, las=1, main="Density Histogram")
```







2.3 Density Histograms

- A *density histogram* is a special type of histogram that has the property of being a proper pdf: non-negative and integrate to 1.
  - $f(x) \geq 0 \quad \forall x$ and $\int f(x)dx = 1$

Histograms estimate the density as a piecewise constant function.

$$\hat{f}(x) = \sum_{j=1}^{J} b_j(x)\,\hat{\theta}_j$$

where $b_j(x) = \mathbb{1}(x \in \text{bin}_j)/h_j$ and

- $\text{bin}_j = [t_j, t_{j+1})$
- $t_1 < t_2 < \ldots < t_J$ are the break points for the bins
- $h_j = [t_j, t_{j+1}) = t_{j+1} - t_j$ is the **bin width** of bin j
  - bin widths do *not* have to be equal
- $\text{bin}_j \cap \text{bin}_k = \emptyset$

2.3.1 Estimating Density Histograms

- Observe data $D = \{X_1, X_2, \ldots, X_N\}$
- Denote n_j as the number of observations in bin j
- $\hat{\theta}_j = \hat{p}_j = n_j/N$ is the usual (MLE) estimate
- Shrinkage Estimator / Laplace Smoothing

$$\begin{aligned}\hat{\theta}_j &= \left(\frac{N}{N+A}\right)\hat{p}_j + \left(\frac{A}{N+A}\right)u_j \\ &= \pi\hat{p}_j + (1-\pi)u_j\end{aligned}$$

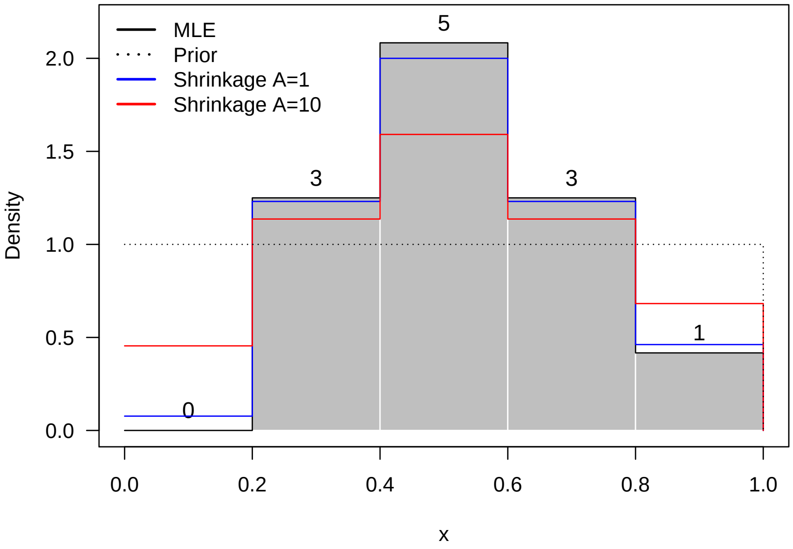
- A (number of *pseudo* observations)

- u_j (prior probability of a *pseudo* observation falling in bin j)
  - $u_j = h_j / \sum_k h_k$ (*uniform* prior)
- $0 \leq \pi \leq 1$ (alternative representation)

Your Turn #2

What are the constraints on $\{\theta_j\}$ that ensure $\hat{f}$ is a proper density?

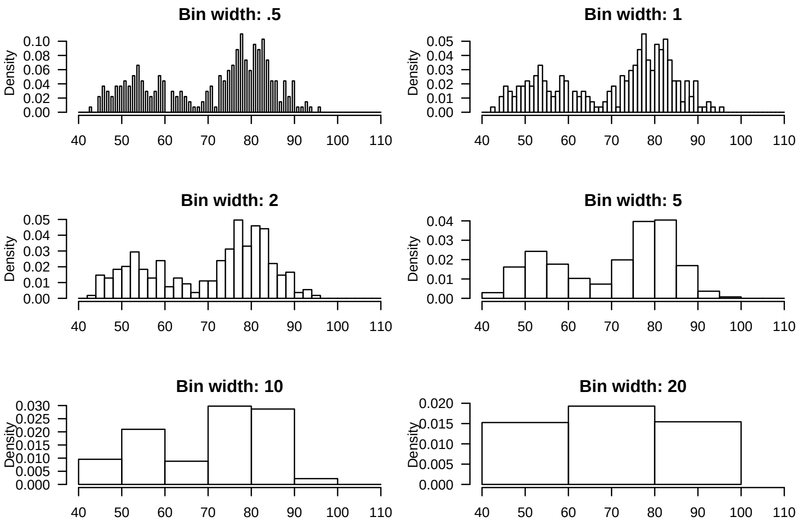
2.3.2 Example



2.4 Local Properties of Histograms

- Histograms can be thought of as a *local* method of density estimation.
  - Points are local to each other if they fall in the same bin
    - Local is determined by bin breaks
- But this has some issues:
  - Some observations “closer" to observations in a neighboring bin
  - Estimate is not smooth (but many true density can often be assumed smooth)
    - **Bin shifts can have a big influence on the resulting density**
- Do parametric approaches estimate a density locally?

2.4.1 Old Faithful: Sensitivity to bin width



2.4.2 Histogram Neighborhood

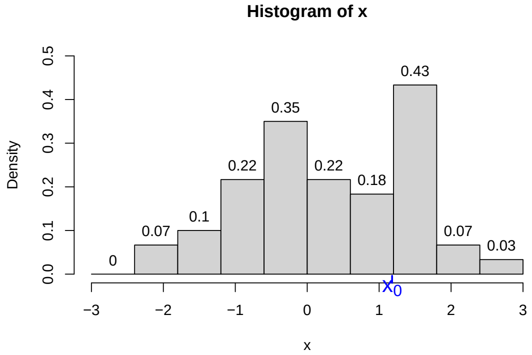
Consider estimating the density at a location x_0

- For a regular histogram (with bin width h), the MLE density is

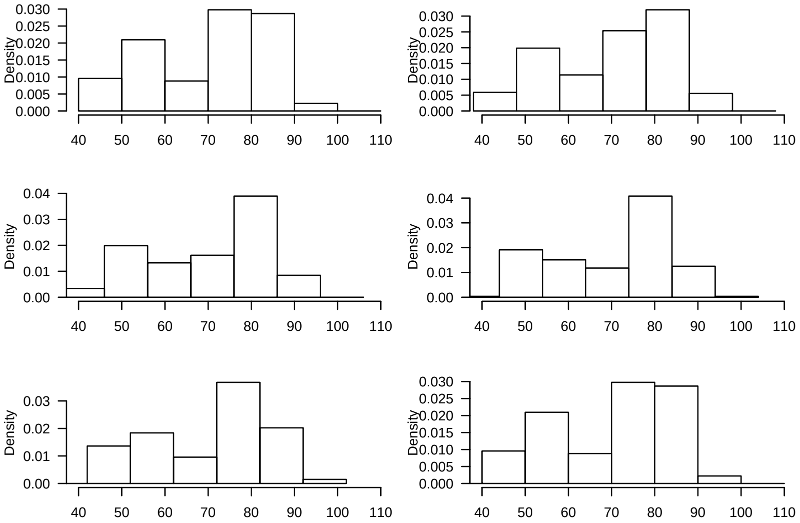
$$\hat{f}(x_0) = \frac{n_j}{nh} \qquad \text{for } x_0 \in \text{bin}_j$$

  which is a function of the number of observations in bin j.

- But how do you feel if x_0 is close to the boundary?



2.4.3 Old Faithful: Sensitivity to bin shifts



3 Kernel Density Estimation (KDE)

Kernel Density Estimation is an improvement on density histograms:

- Removes the bin anchor point parameter
- Allows more flexible definition of "neighborhood"

3.1 Local Density Estimation - Moving Window

Consider again estimating the density at a location x_0

- Regular Histogram (with midpoints m_j and bin width h)

$$\hat{f}(x_0) = \frac{1}{n}\sum_{i=1}^{n} \frac{\mathbb{1}\left(|x_i - m_j| \leq \frac{h}{2}\right)}{h} \qquad \text{for } x_0 \in B_j$$

- Consider a moving window approach

$$\hat{f}(x_0) = \frac{1}{n}\sum_{i=1}^{n} \frac{\mathbb{1}\left(|x_i - x_0| \leq \frac{h}{2}\right)}{h}$$

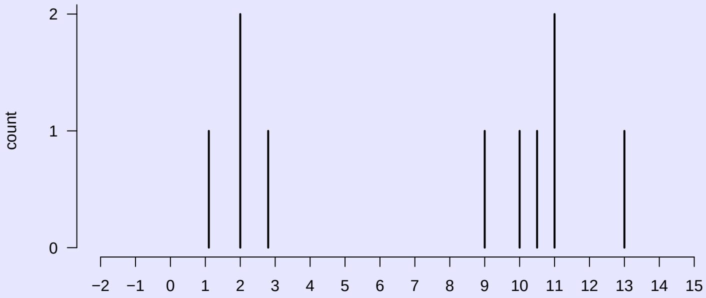
  This gives a more pleasing definition of local by centering a bin at x_0.

- Equivalently this estimates the derivative of ECDF

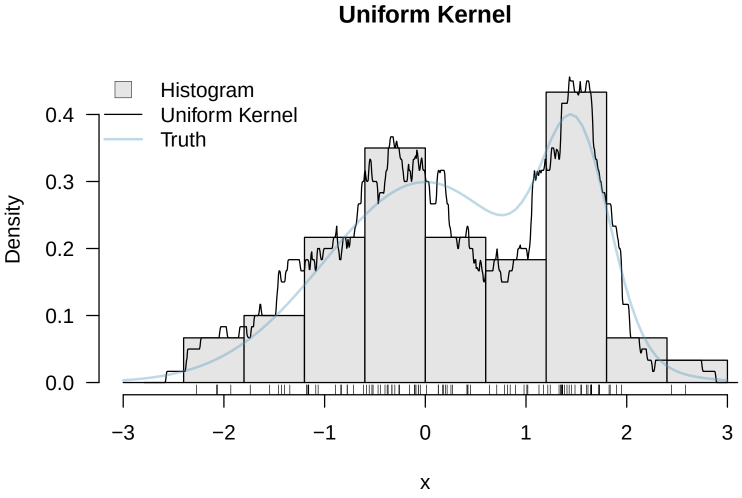
$$\hat{f}(x_0) = \frac{F_n(x_0 + h/2) - F_n(x_0 - h/2)}{h}$$

Your Turn #3

Consider a dataset $D = \{1.1, 2, 2, 2.8, 9, 10, 10.5, 11, 11, 13\}$. Use a *moving window* (*uniform kernel*) estimator to estimate the density at locations $x_0 = \{0, 6, 10\}$ using $h = 4$.

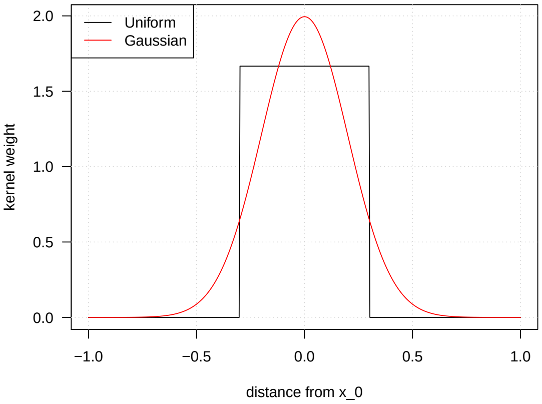


3.1.1 Uniform Kernel

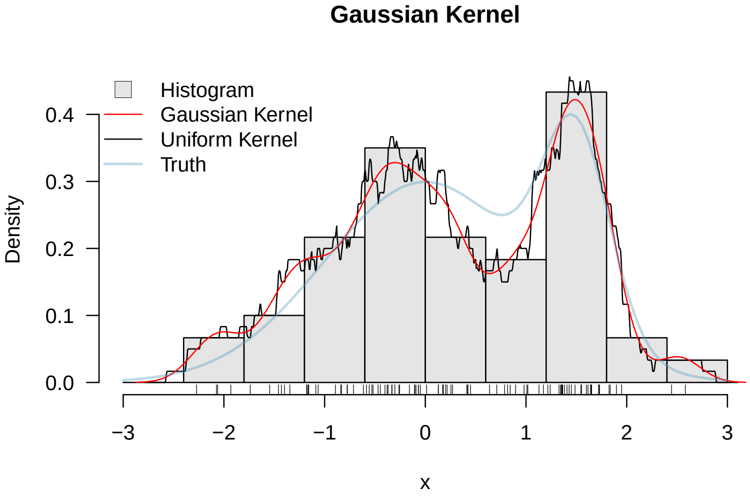


3.1.2 Kernels

- The moving window approach looks better than a histogram with the same bin width, but it is still not smooth
- Instead of giving every observation in the window the same weight, we can assign a weight according to its distance from x_0



3.1.3 Gaussian Kernel



3.1.4 Kernel Density

- More generally, the weights $K_h(u) = h^{-1}K(\frac{u}{h})$ are called kernel functions
- Thus, a kernel density estimator is of the form

$$\hat{f}(x_0) = \frac{1}{n}\sum_{i=1}^{n} K_h(x_i - x_0)$$

where the smoothing parameter h is called the bandwidth and controls how fast the weights decay as a function from x_0

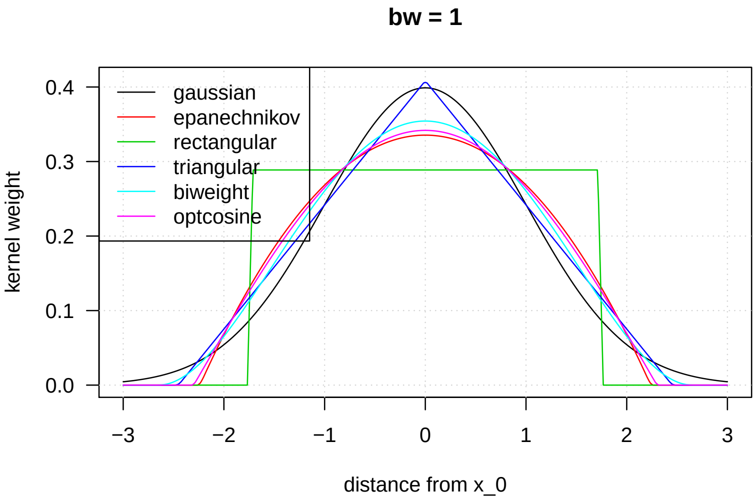
3.2 Kernel Properties

A kernel is usually considered to be a symmetric probability density function:

- $K_h(u) \geq 0$ (non-negative)
- $\int K_h(u)\, du = 1$ (integrates to one)
- $K_h(u) = K_h(-u)$ (symmetric about 0)
- Notice that if the kernel has compact support, so does the resulting density estimate
- The Gaussian kernel is the most popular, but has infinite support
  - The is good when the true density has infinite support
  - However, this requires more computation

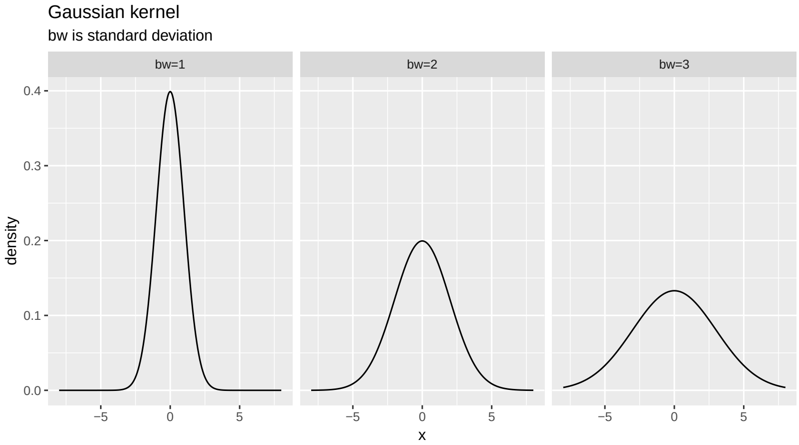
  - But easier to calculate properties (e.g., bandwidth selection)

3.2.1 Popular Kernels



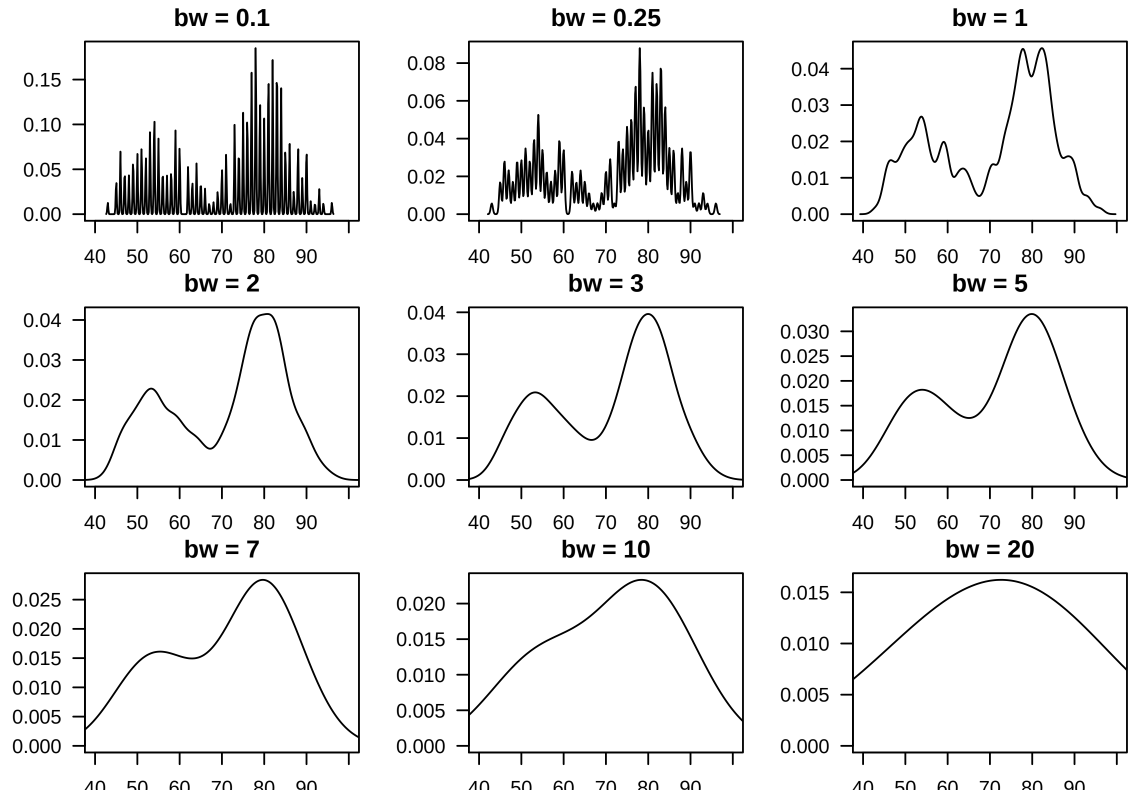
- We will use *Gaussian* kernel unless otherwise stated.
- For a Gaussian/Normal kernel:

$$
\begin{aligned}
K_h(u) &= \frac{1}{h\sqrt{2\pi}} \exp\left(-\frac{u^2}{2h^2}\right) \\
&= h^{-1}K(u/h)(\text{where } K(\cdot) \text{ is standard normal pdf})
\end{aligned}
$$



3.3 Bandwidth

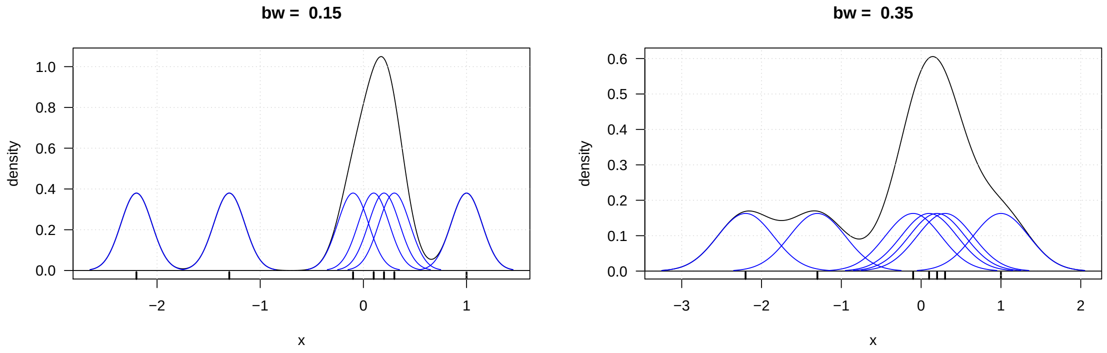
- The bandwidth parameter, h controls the amount of smoothing
- What happens when $h \uparrow \infty$?
- What happens when $h \downarrow 0$?
- **The choice of bandwidth is much more important than the choice of kernel**
- Bandwidth is usually defined as the standard deviation of the kernel
  - but not always, so check the implementation.



3.4 Another Perspective

- We have described KDE as taking a local density around location x_0
- An alternative perspective is to view KDE as an n component *mixture model* with mixture weights of $1/n$

$$\begin{aligned}\hat{f}(x) &= \frac{1}{n}\sum_{i=1}^{n} K_h(x_i - x) \\ &= \sum_{j=1}^{n} \frac{1}{n} f_i(x) \qquad\qquad (f_i(x) = K_h(x_i - x))\end{aligned}$$



3.5 Bandwidth Selection

- The best bandwidth is the one that is best for your problem
  - The best bandwidth for density estimation may not be best for classification
  - Choosing a bandwidth that is visually appealing may be suitable

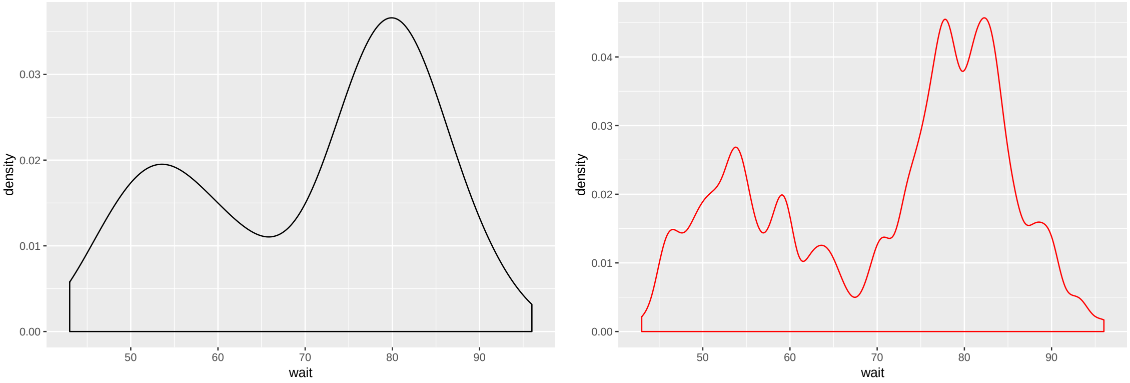
- For density estimation, we want a "Goldilocks" bandwidth; one that is not *too wiggly* nor *too smooth*.
- Or to state it plainly, we want to find the bandwidth that gives a density estimate closest to the true density. But,
    a. We don't know the true density
    b. There are many ways to define "close"
- For this class, we will not go into the details other than to say most bandwidth selection methods are based on *plug-in* or *cross-validation*.

3.5.1 Bandwidth Selection in R

There are a few KDE functions in R.

1. `density()` in base R is used for ggplot's `geom_density()`

```
ggplot() + geom_density(aes(x=wait))   # runs bw.nrd0()
ggplot() + geom_density(aes(x=wait), bw=1, color="red")
```

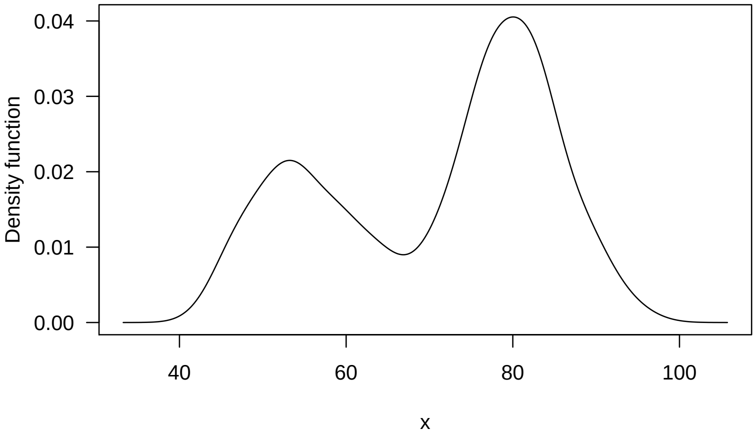


See the R help: `?bw.nrd0` to get a description of several bandwidth selection methods.

```
bw.nrd0(wait)     # get default bandwidth
#> [1] 3.988
c(bw.nrd0 = bw.nrd0(wait), bw.nrd=bw.nrd(wait), bw.bcv=bw.bcv(wait),
  bw.SJ = bw.SJ(wait), bw.ucv=bw.ucv(wait))
#> bw.nrd0  bw.nrd  bw.bcv   bw.SJ  bw.ucv
#>   3.988   4.696   2.598   2.504   2.658
```

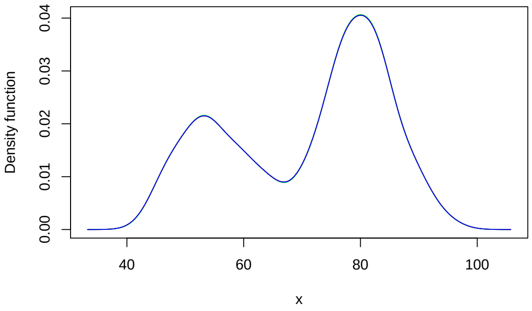
2. I recommend using the function `kde()` in the `ks` package.
    - It allows multivariate kernel estimation and bandwidth selection

```
library(ks)
f.kde = kde(wait)
f.kde$h                 # The bandwidth parameter is denoted by h
#> [1] 2.636
plot(f.kde, las=1)      # plots the "kde" object
```



- There are several bandwidth selection methods
    - `hpi`: plug-in
    - `hlscv`: least squares cross-validation
    - `hscv`: smoothed cross-validation
    - `hucv`: unbiased cross-validation
    - Details

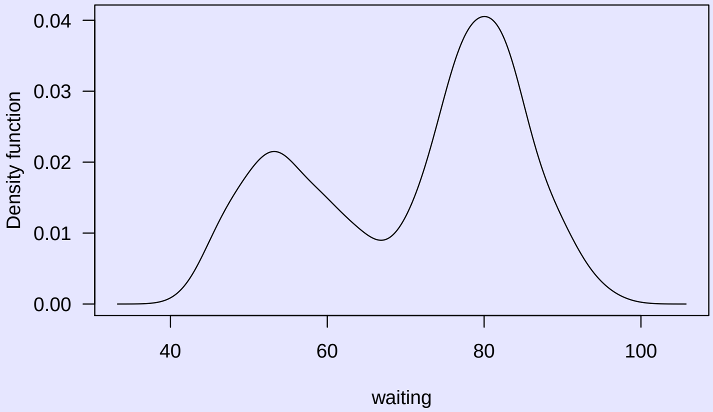
```
h1 = hpi(wait)
h2 = hlscv(wait)
#> Warning in hlscv(wait): Data contain duplicated values: LSCV is not well-behaved
#> in this case
h3 = hscv(wait)
h4 = hucv(wait)
#> Warning in hlscv(...): Data contain duplicated values: LSCV is not well-behaved
#> in this case
c(hpi=h1, hlscv=h2, hscv=h3, hucv=h4)
#>   hpi hlscv  hscv  hucv
#> 2.636 2.627 2.581 2.627

plot(kde(wait, h=h1))
plot(kde(wait, h=h2), add=TRUE, col="red")
plot(kde(wait, h=h3), add=TRUE, col="green")
plot(kde(wait, h=h4), add=TRUE, col="blue")
```



4 Multivariate Kernel Density Estimation

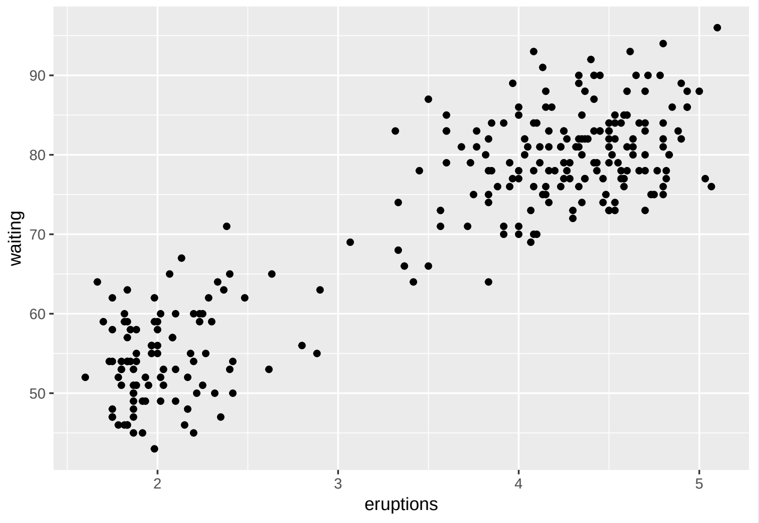
Your Turn #4 : The Return to Old Faithful

```
#-- Univariate density
f.wait = kde(wait)
plot(f.wait, las=1, xlab="waiting")
```



Did I forget to mention what we have additional information about old faithful eruptions? It turns out that we also have information on the *duration of the previous eruption*.

```
#-- Load the Old Faithful data
X = datasets::faithful

#-- Scatterplot
ggplot(X) + geom_point(aes(eruptions, waiting))
```



```
# plot(X, las=1) # base R scatterplot
```

1. What patterns do you see?
2. Think about how to estimate this *bivariate* density.
3. Would a 2D Gaussian be appropriate?

There are three primary approach to multivariate (d dimensional) KDE:

1. Multivariate kernels

- (e.g., $K(u) = N(\mathbf{0}, \Sigma)$)

$$\hat{f}(x) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}n}\sum_{i=1}^{n}\exp\left(-\frac{1}{2}(x-x_i)^{\mathsf{T}}\Sigma^{-1}(x-x_i)\right)$$

- Let $\Sigma = h^2 A$ where $|A| = 1$, thus $|\Sigma| = h^{2d}$

$$\hat{f}(x) = \frac{1}{(2\pi)^{d/2}h^d n}\sum_{i=1}^{n}\exp\left(-\frac{1}{2}(x-x_i)^{\mathsf{T}}A^{-1}(x-x_i)\right)$$

2. Product Kernels ($A = I_d$)

$$\hat{f}(x) = \frac{1}{n}\sum_{i=1}^{n}\left(\prod_{j=1}^{d}K_{h_j}(x_j - x_{ij})\right)$$

3. Independence

$$\begin{aligned}\hat{f}(x) &= \prod_{j=1}^{d}\hat{f}_j(x)\\ &= \prod_{j=1}^{d}\left(\frac{1}{n}\sum_{i=1}^{n}K_{h_j}(x_j - x_{ij})\right)\end{aligned}$$

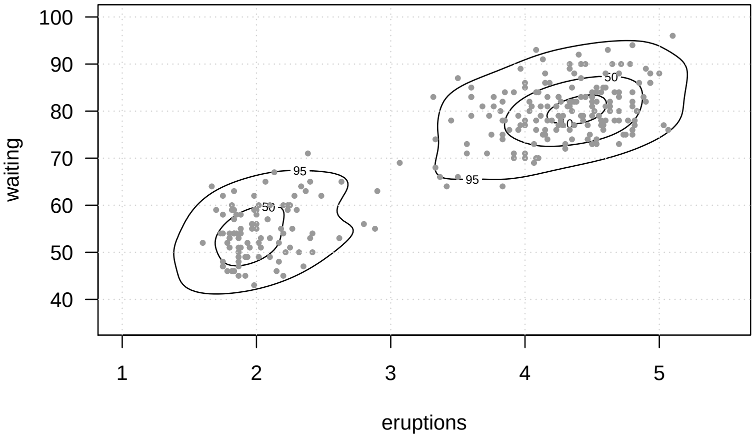
4.1 Multivariate KDE with kde

```
head(X)     # first 6 rows of the full old faithful data
#>   eruptions waiting
#> 1     3.600      79
#> 2     1.800      54
#> 3     3.333      74
#> 4     2.283      62
#> 5     4.533      85
#> 6     2.883      55
```

```
(H1 = Hscv(X))      # smoothed cross-validation bw estimator
```

```
#>         [,1]   [,2]
#> [1,] 0.0601  0.511
#> [2,] 0.5110 12.408
```

```
f1 = kde(X, H=H1)   # use H for multivariate data
plot(f1,
     cont = c(10, 50, 95),                        # set contour levels
     # display = "filled.contour",                # use filled contour
     las=1, xlim = c(1.0, 5.5), ylim=c(35, 100))  # set asthetics
points(X, pch=19, cex=.5, col='grey60')           # add points
grid()                                            # add grid lines
```



Above is the *unconstrained* Gaussian kernel.

- The Kernel is a MV Normal, $K(\mathbf{X}; H)$, with variance-covariance matrix

$$\Sigma = H = \begin{bmatrix} H_{11} & H_{12} \\ H_{12} & H_{22} \end{bmatrix}$$

- The diagonal terms correspond to the **variances** in each dimension
  - Note: take square root to compare with univariate bandwidth
- The off-diagonal term corresponds to the *correlation*: $H_{12} = \rho h_1 h_2$, where $h_i = \sqrt{H_{ii}}$

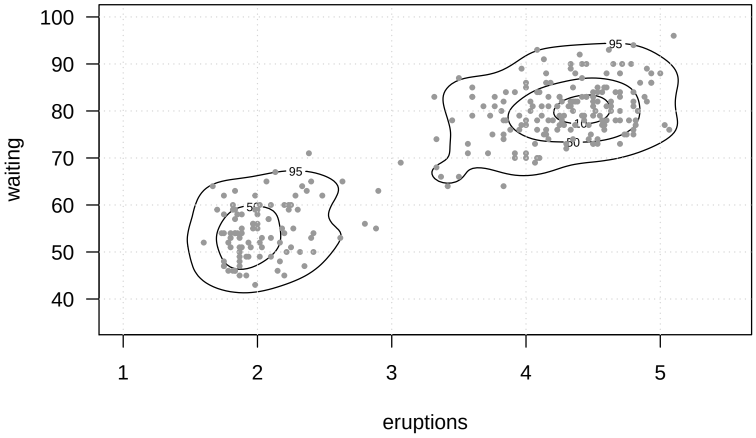
4.1.1 Product Kernel

If the off-diagonal/correlation is zero, then kernel reduces to the product of two univariate kernels:

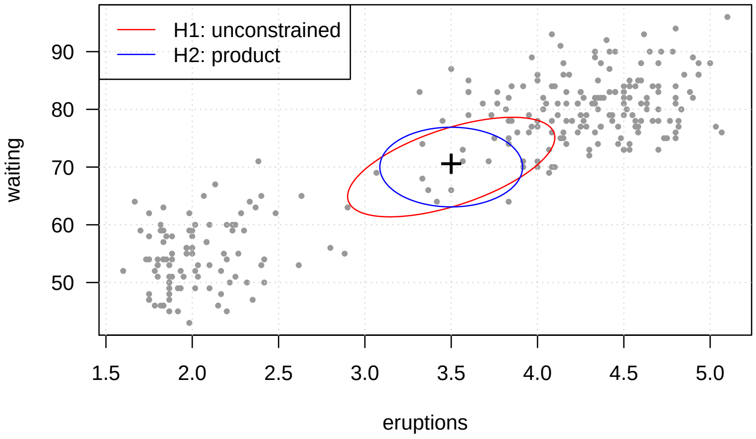
$$K((x_1, x_2); H) = K_1(x_1; h_1) K_2(x_2; h_2)$$

where $h_1 = \sqrt{H_{11}}$.

```
(H2 = Hscv.diag(X))                                # product kernel
#>          [,1]   [,2]
#> [1,] 0.0285 0.000
#> [2,] 0.0000 7.949
f2 = kde(X, H=H2)

plot(f2,
     cont = c(10, 50, 95),                         # set contour levels
     las=1, xlim = c(1.0, 5.5), ylim=c(35, 100))   # set asthetics
points(X, pch=19, cex=.5, col='grey60')            # add points
grid()
```



We can plot the kernels (using `mixtools::ellipse()` to see their shape. Here is the kernel at location (3.5, 70).



- The vignette [ks: Kernel density estimation for bivariate data] https://cran.r-project.org/web/packages/ks/vignettes/kde.pdf has some more information on using `ks::kde()` for bivariate data.

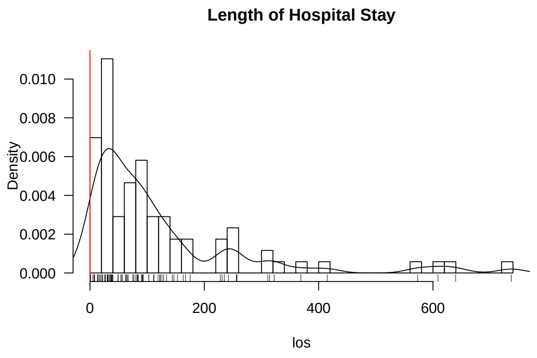
5 Extra Details

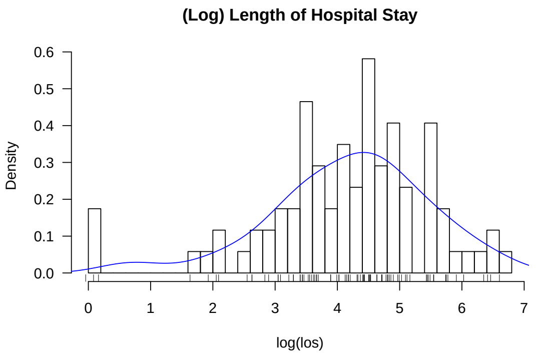
5.1 Edge Effects

Sometimes there are known boundaries in the data (e.g., the amount of rainfall cannot be negative). Here are some options:

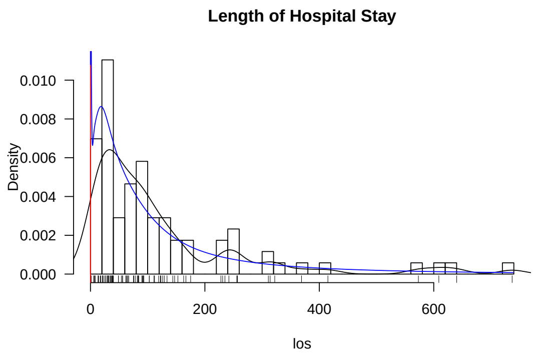
1. Do nothing - as long as not many events are near the boundary and the bandwidth is small, this may not be too problematic. However, it will lead to an increased bias around the boundaries.
2. Transform the data (e.g., $x' = \log(x)$), estimate the density is the transformed space, then transform back
3. Use an edge correction technique

5.1.1 Log-Transformations





- Let $Y = \ln(X)$ be the transformed RV.
- $F_X(x) = \Pr(X \leq x) = \Pr(e^Y \leq x) = \Pr(Y \leq \ln(x)) = F_Y(\ln(x))$
- $f_X(x) = \frac{d}{dx} F_X(x) = \frac{d}{dx} F_Y(\ln(x)) = \frac{1}{x} f_Y(\ln(x))$
- The argument `positive=TRUE` in `ks::kde()` will perform this transformation.



5.1.2 Edge Correction

The simplest approach requires a modification of the kernels near the boundary. Let $\mathcal{S} = [a, b]$.

- Recall that $\int_a^b K_h(x_i - x)dx$ should be 1 for every i.
- But near a boundary $\int_a^b K_h(x_i - x)dx \neq 1$
- Denote $w_h(x_i) = \int_a^b K_h(x_i - x)dx$
- The resulting edge corrected KDE equation becomes

$$\hat{f}(x) = \frac{1}{n} \sum_{i=1}^{n} w_h(x_i)^{-1} K_h(x_i - x)$$

Another approach corrects the kernel for each particular x

- Denote $w_h(x) = \int_a^b K_h(u - x)$ụ
- The resulting edge corrected KDE equation becomes

$$\hat{f}(x) = \frac{1}{n} \sum_{i=1}^{n} w_h(x)^{-1} K_h(x_i - x)$$

- This approach is not guaranteed to integrate to 1, but for some problems this is not a major concern

5.2 Adaptive Kernels

Up to this point, we have considered fixed bandwidths. But what if we let the bandwidth vary? There are two main approaches:

- Balloon Estimator

$$\begin{aligned}\hat{f}(x) &= \frac{1}{n}\sum_{i=1}^{n} K_{h(x)}(x_i - x) \\ &= \frac{1}{nh(x)}\sum_{i=1}^{n} K\left(\frac{x_i - x}{h(x)}\right)\end{aligned}$$

- Sample Point Estimator

$$\begin{aligned}\hat{f}(x) &= \frac{1}{n}\sum_{i=1}^{n} K_{h(x_i)}(x_i - x) \\ &= \frac{1}{n}\sum_{i=1}^{n} h(x_i)^{-1} K\left(\frac{x_i - x}{h(x_i)}\right)\end{aligned}$$

5.3 k-Nearest Neighbor Density Approach

Like what we discussed for percentile binning, we can estimate the density from the size of the window containing the k nearest observations.

$$\hat{f}_k(x) = \frac{k}{nV_k(x)}$$

where $V_k(x)$ is the volume of a neighborhood that contains the k-nearest neighbors.

- This is an adaptive version of the moving window (uniform kernel) approach
- Probably won't integrate to 1
- It is also possible to use the k-NN distance as a way to select an adaptive bandwidth $h(x)$.

5.4 Mixture Models

Mixture models offer a flexible compromise between kernel density and parametric methods.

- A mixture model is a mixture of densities

$$f(x) = \sum_{j=1}^{p} \pi_j g_j(x|\xi_j)$$

  where
  - $0 \leq \pi_j \leq 1$ and $\sum_{j=1}^{p} \pi_j$ are the mixing proportions
  - $g_j(x|\xi_j)$ are the component densities
- This idea is behind model-based clustering (ST 640), radial basis functions (ESL 6.7), etc.
- Usually the parameters θ_j and weights π_j have to be estimated (EM algorithm shows up here)

5.5 Kernels, Mixtures, and Splines

All of these methods can be written:

$$f(x) = \sum_{j=1}^{J} b_j(x)\theta_j$$

- For KDE:
  - $J = n, b_j(x) = K_h(x - x_j), \theta_j = 1/n$ (bw h estimated)
- For Mixture Models:
  - $b_j(x) = g_j(x|\xi_j), \theta_j = \pi_j$ (J, ξ_j, π_j estimated)
- B-splines
  - $b_j(x)$ is a B-spline, (θ_j, and maybe J and knots, estimated)
  - Note: For density estimation, $\log f(x) = \sum_{j=1}^{J} b_j(x)\theta_j$ may be easier to estimate

5.6 Other Issues

- One major problem with KDE and kernel regression (and k-NN) is that all of training data must be stored in memory.
  - For large data, this is unreasonable
  - Binning (i.e., histograms) are used to reduce data storage and improve computation
- Multi-dimensional kernels are not very good for high dimensions (unless simplified by using product kernels)
- But temporal kernels good for adaptive procedures (e.g., only remembers most recent observations)
  - Think about what EWMA is doing