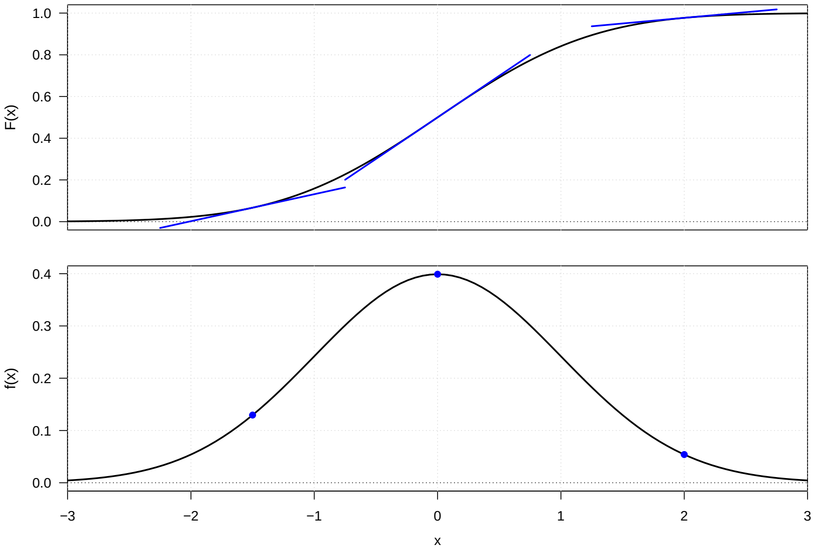
12 - Nonparametric Density Estimation

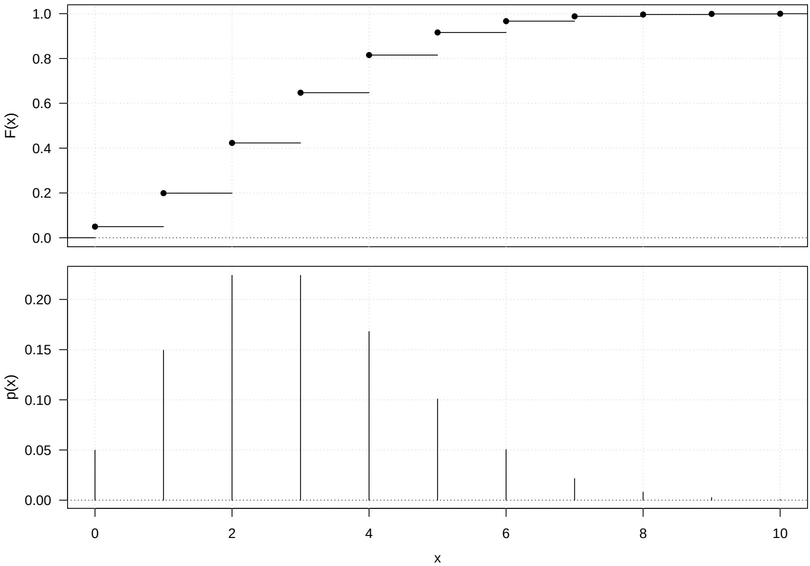
ST 697 | Fall 2017
University of Alabama

Density Review

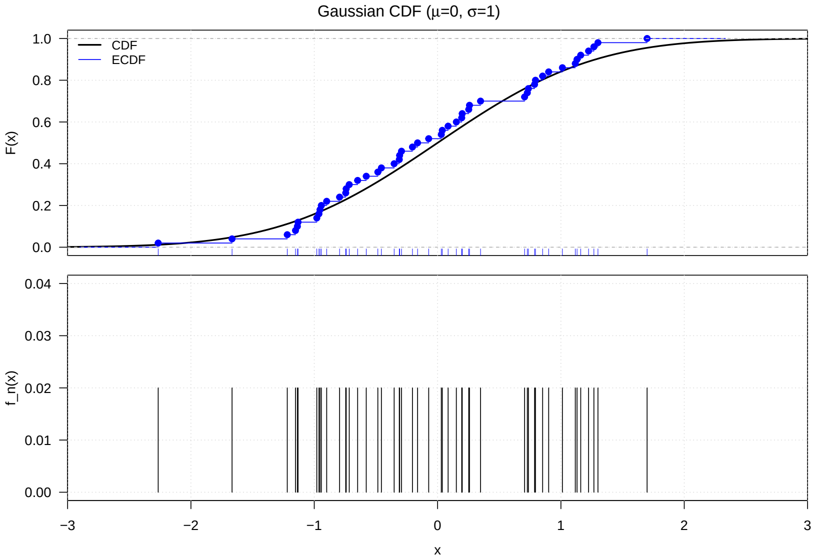
Continuous Random Variables



Discrete Random Variables



Empirical CDF and PDF



ECDF of $n = 50$ from $X \sim N(0, 1)$

Review of Parametric Density Estimation

1. Choose parametric distribution family
2. Estimate parameters
    - Method of Moments
    - Maximum Likelihood
    - Bayesian (also choose prior)

Density Histograms

Density Histograms

Histograms estimate the density as a piecewise constant function.

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

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

- $\mathrm{bin}_j = [t_j, t_{j+1})$
- $t_1 < t_2 < \ldots < t_J$ 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
- $\mathrm{bin}_j \cap \mathrm{bin}_k = \emptyset$

Some slight adjustments need to be made for multivariate

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

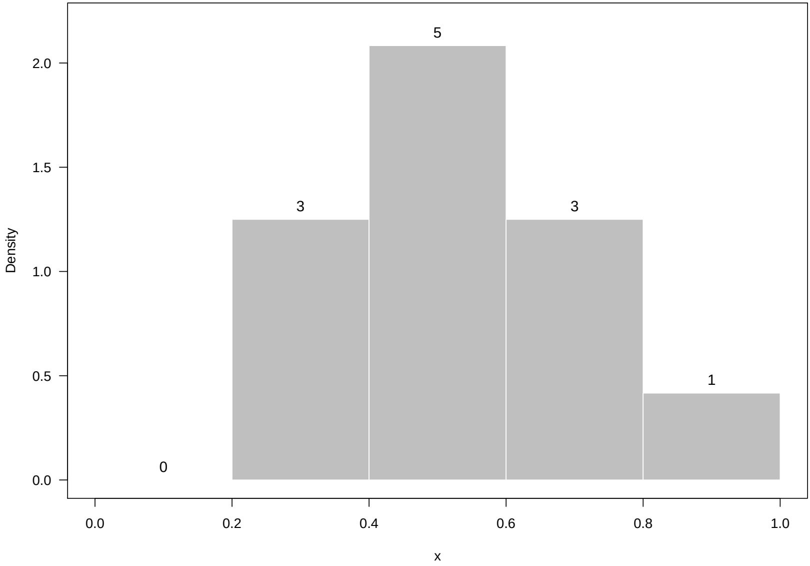
$$\begin{aligned}\hat{\theta}_j &= \left(\frac{n}{n+A}\right)\hat{p}_j + \left(\frac{A}{n+A}\right)u_j \\ &= \pi\hat{p}_j + (1-\pi)u_j\end{aligned}$$

where

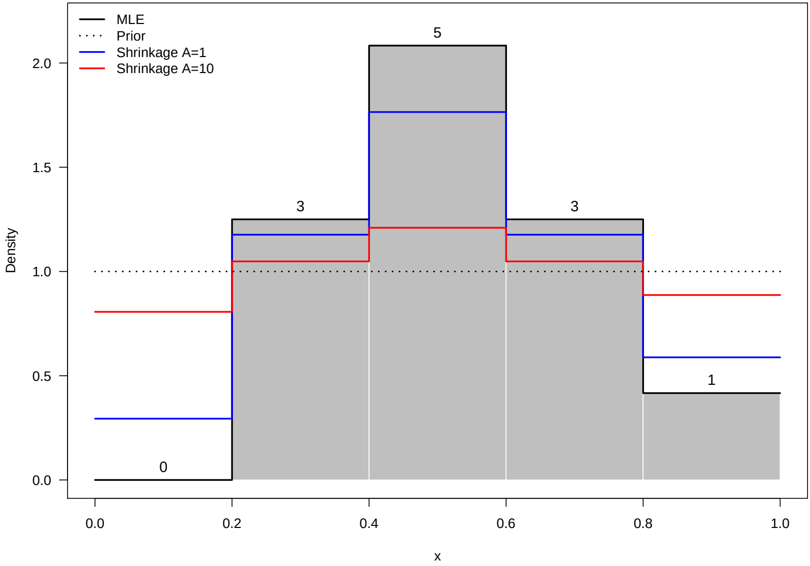
- A (number of *pseudo* observations in bin j)
- $u_j = h_j / \sum_k h_k$ (*uniform* prior)
- $0 \leq \pi \leq 1$ (alternative representation)

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

Regular Histogram



Histogram with Shrinkage



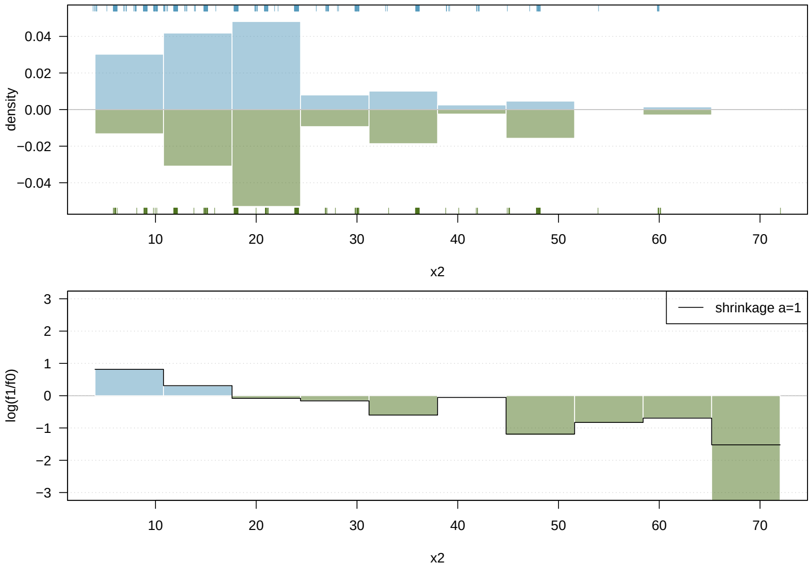
German Credit Data

http://archive.ics.uci.edu/ml/machine-learning-databases/statlog/german/german.data

```
url = "http://archive.ics.uci.edu/ml/machine-learni
data = read.delim(url,sep=" ",header=FALSE)

germanCredit = data[,-21]
Y = data[,21]
G = ifelse(Y==1, "good", "bad")
good = (G == "good")
```

Histogram Density Ratio: German Credit Data



Binning

The bins of a histogram can be of fixed width $h_j = h$ or variable width.

- Smaller bins have less bias, but more variance
- Shrinkage lowers variance, but increases bias
- Percentile binning set the bin width according to the number of observations

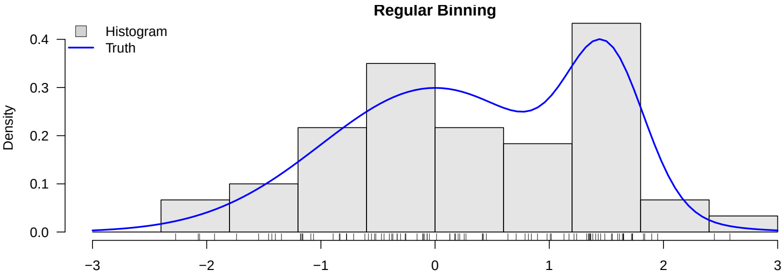
Creating J bins $\{t_j : j = 1, 2, \ldots, J + 1\}$:

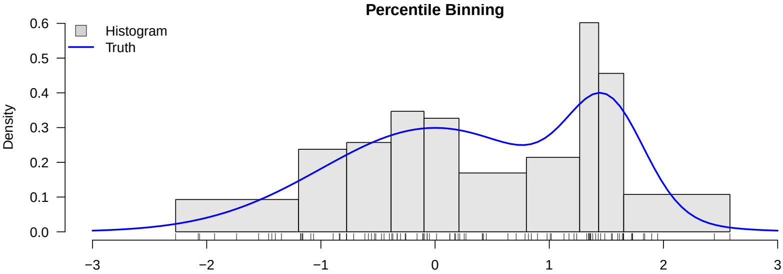
$$= \{t_1, h, J\} \qquad \text{(equal bin width of width } h\text{)}$$

$$= \{t_1, t_{J+1}, J\} \qquad \text{(equal bin width } h = (t_{J+1} - t_1)/J\text{)}$$

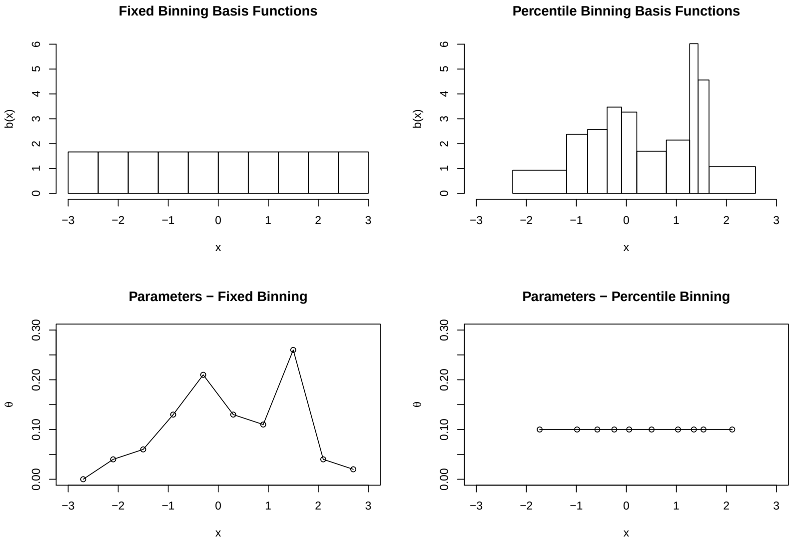
$$= \{t_j : t_j = F_n^{-1}(j/J)\} \qquad \text{(percentile binning)}$$

Binning Example





Binning Example: Basis and Parameters



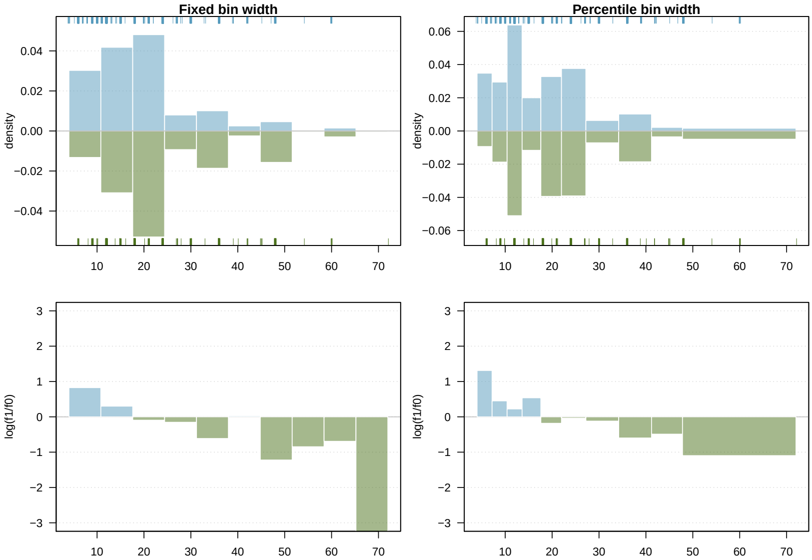
Optimal Bin Width

The optimal bin widths for density estimation may *not* be the same as the optimal bin widths for finding the log density ratio (like we need for naive Bayes or anomaly detection).

- If we are interested in the log density ratio, should we use the same binning?
  - What if unbalanced classes?

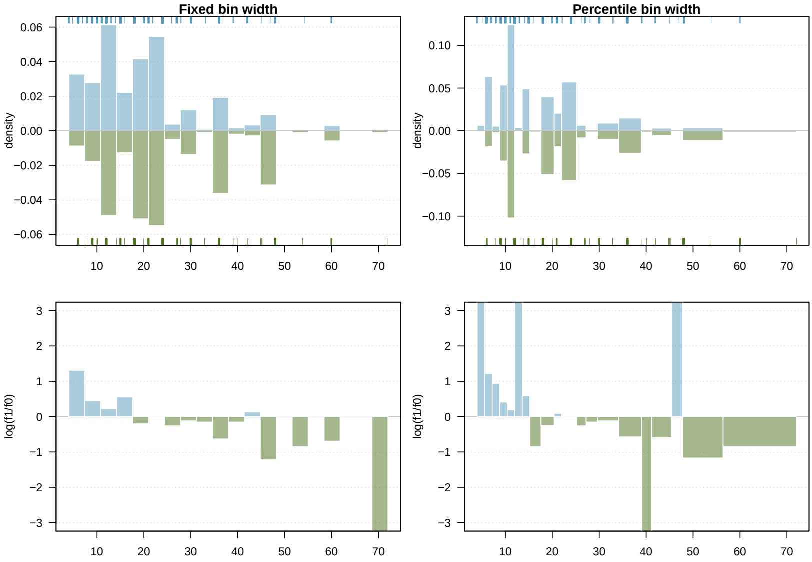
Histogram Bin Width: German Credit Data

$J = 10$ Bins



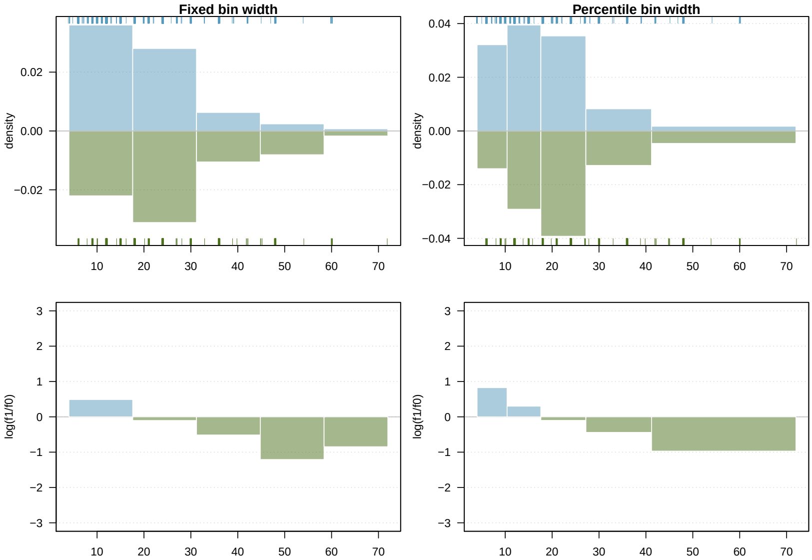
Histogram Bin Width: German Credit Data

$J = 20$ Bins



Histogram Bin Width: German Credit Data

$J = 5$ Bins



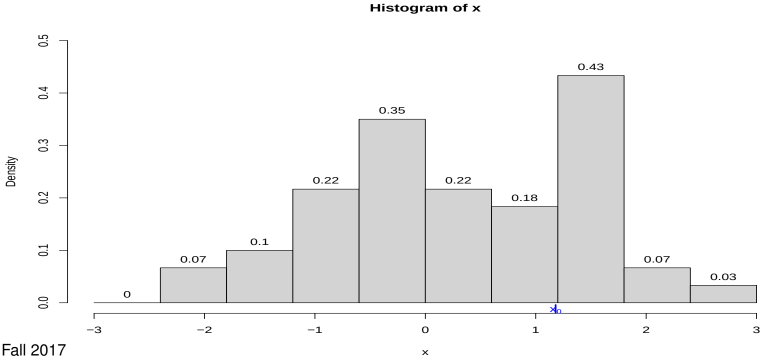
Review of Histograms

- We have considered regular and percentile histograms for nonparametric density estimation
- 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 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?

Histogram Neighborhood

Consider estimating the density at a location x_0

- For a regular histogram (with bin width h), the MLE density is
$$\hat{f}(x_0) = \frac{n_j}{nh} \qquad \text{for } x_0 \in \mathrm{bin}_j$$
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?



Buffalo Snowfall (1910-1973): Bin width

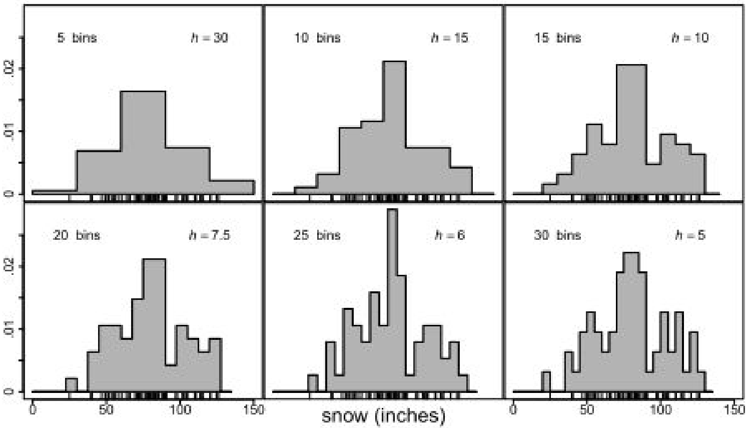


Figure 4.8: Histograms of the Buffalo snowfall data with bin origin $t_0 = 0$, and bin widths of 30, 15, 10, 7.5, 6, and 5 inches over the interval $(0, 150)$.

Scott (1992) Multivariate Density Estimation, Chapter 4

Buffalo Snowfall (1910-1973): Sensitivity to bin shifts

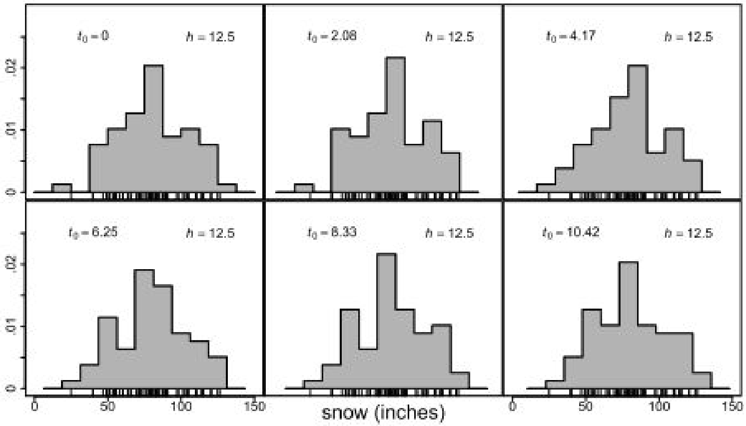


Figure 4.9: Six shifted histograms of the Buffalo snowfall data. All have a bin width of 12.5 inches, but different bin origins $t_0 = h/m$, $m = 1, \ldots, 6$.

Scott (1992) Multivariate Density Estimation, Chapter 4

Kernel Density Estimation

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)

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

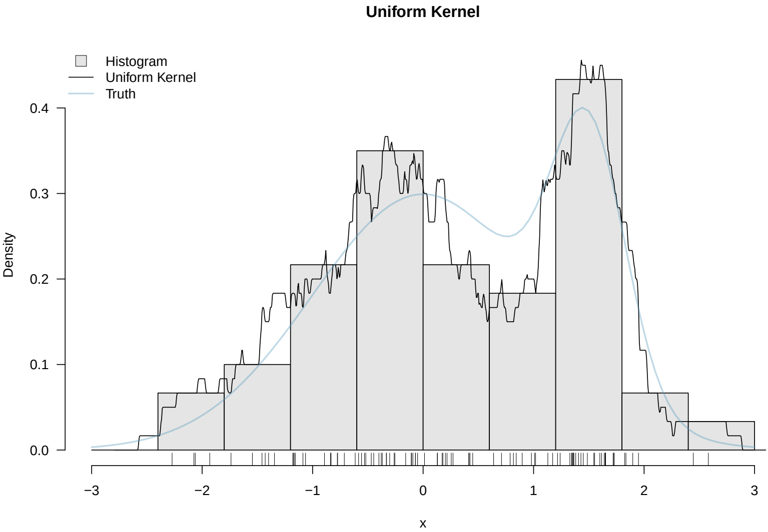
- Consider a moving window approach

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

  This gives a more pleasing definition of local by centering a bin at x_0.
- Equivalently this estimates the derivative of ECDF

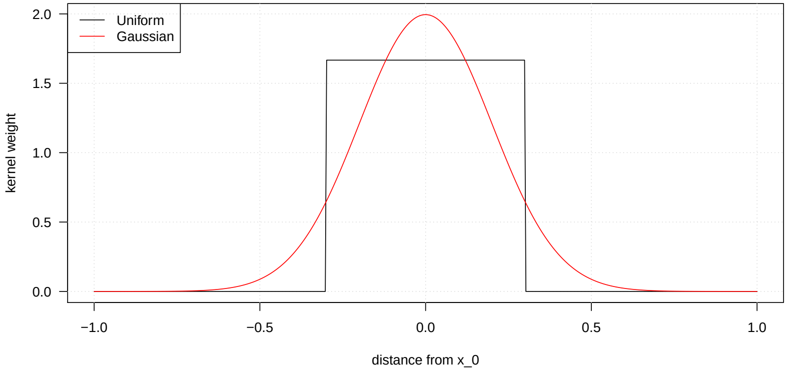
$$\hat{f}(x_0) = \frac{F_n(x_0 + h/2) - F_n(x_0 - h/2)}{h}$$

Uniform Kernel

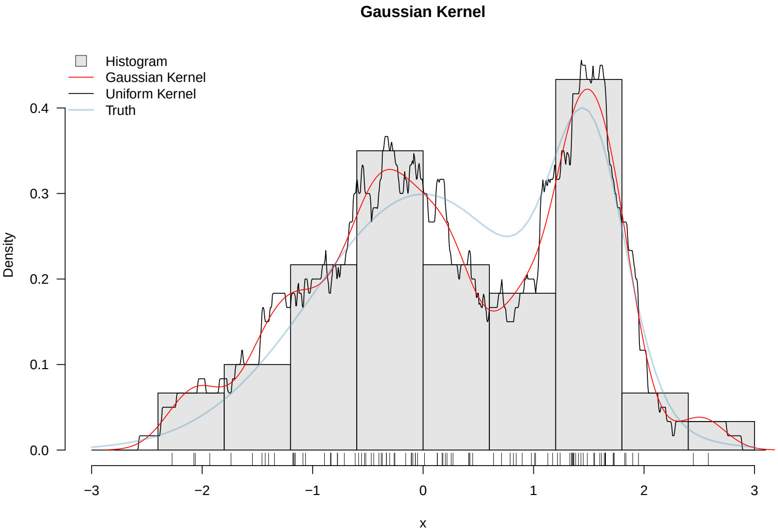


Kernel Density Estimation

- 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



Gaussian Kernel



Kernel Density Estimation

- 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

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

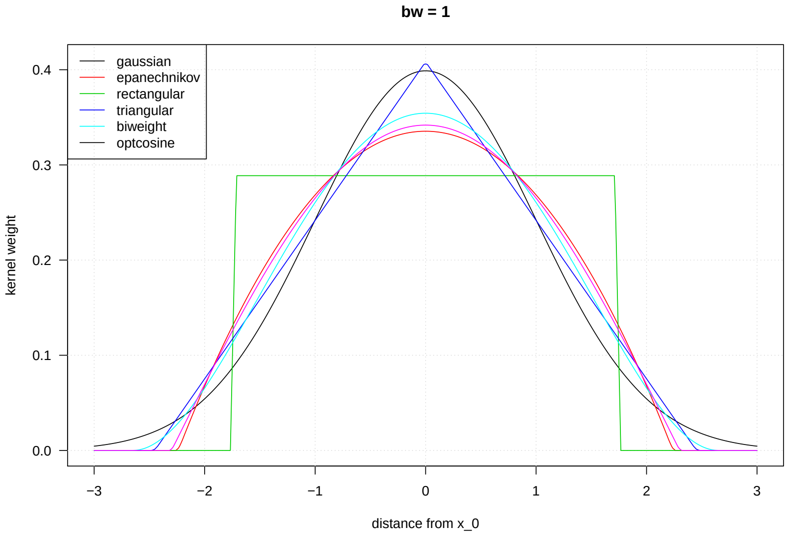
  where the smoothing parameter h is called the bandwidth and controls how fast the weights decay as a function from x_0
- In R, `density()` function uses a bandwidth (`bw` argument) that is the standard deviation of the kernel

Kernel Properties

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

- $K_h(u) \geq 0$ (non-negative)
- $\int K_h(u)\,\mathrm{d}u = 1$ (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)

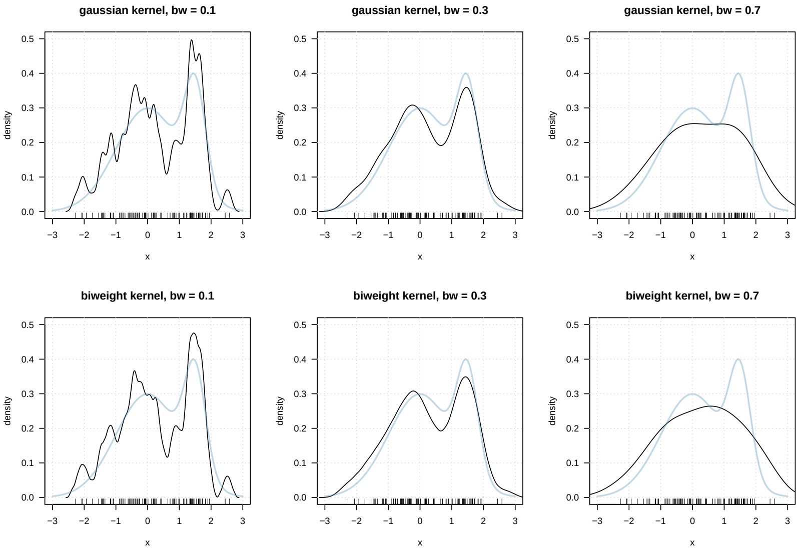
Popular Kernels



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
- There is no standard representation of the bandwidth parameter h
  - In R, `density()` uses two arguments to set the bandwidth: `bw` is the standard deviation of the kernel, and `width` is the length of the support of the kernel
    - The two are very different so check carefully what an author is using in their definition of bandwidth

Bandwidth Effects



Another Perspectives

- 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 $1/n$

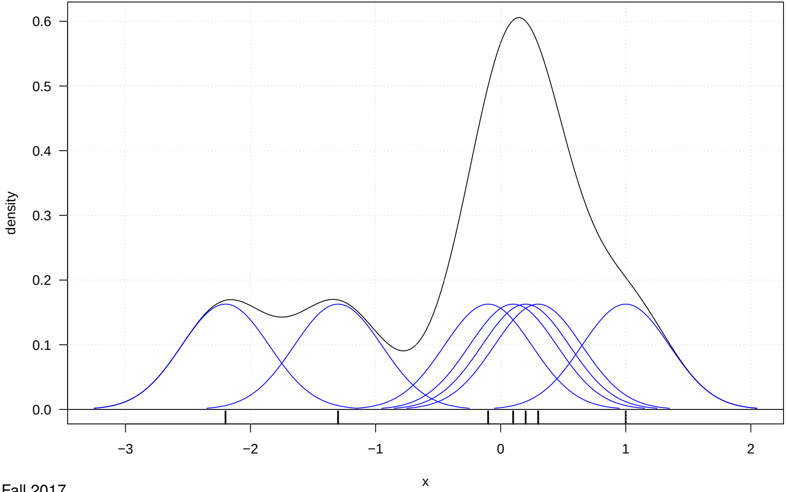
$$\begin{aligned}\hat{f}(x) &= \frac{1}{n}\sum_{i=1}^{n} K_h(x_i - x) \\ &= \sum_{j=1}^{n} \frac{1}{n}\, f_i(x) \qquad\qquad (f_i(x) = K_h(x_i - x))\end{aligned}$$

- Or in a basis function representation

$$\hat{f}(x) = \sum_{j=1}^{n} \theta_j \, b_i(x) \qquad \left(\theta_j = \frac{1}{n},\ b_i(x) = K_h(x_i - x)\right)$$

Component Kernels

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

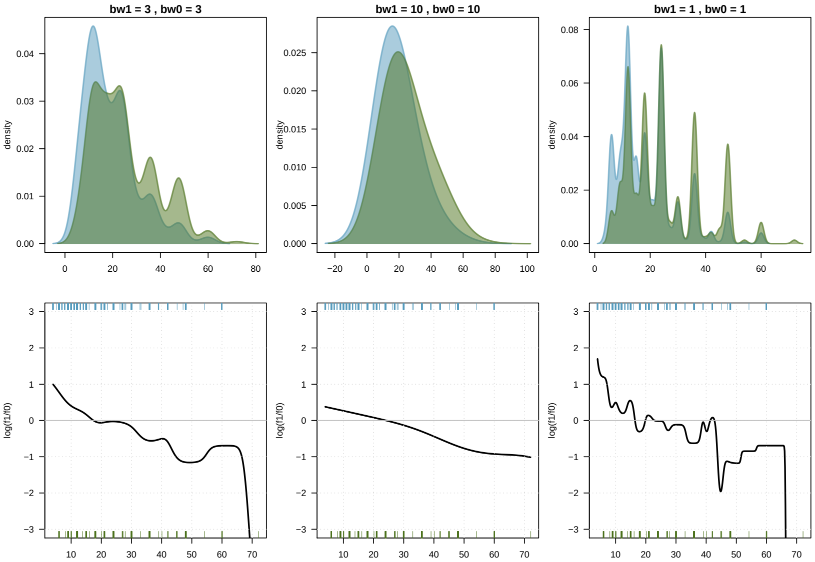


Kernel Based Density Ratio

Using kernel density estimation (KDE), the density ratio useful for naive Bayes, etc. becomes

$$\frac{f_1(x)}{f_0(x)} \hat{=} \frac{\hat{f}_1(x)}{\hat{f}_0(x)} = \frac{\frac{1}{n_1}\sum_{i=1}^{n_1} K_{h_1}(x_i - x)}{\frac{1}{n_0}\sum_{j=1}^{n_0} K_{h_0}(x_j - x)}$$

KDE: German Credit Data



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

Edge Correction (1)

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

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

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

Edge Correction (2)

Another approach corrects the kernel for each particular x

- Denote $w_h(x) = \int_a^b K_h(u - x)\,\mathrm{d}u$
- 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

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

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

\end{frame}

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 $h(x)$.

Multivariate Density Estimation

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 ($A = I_d$)

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

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)

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

Kernel Regression

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
- 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 remember most recent observations)
  - Think about what EWMA is doing

Smoothing for Categorical Data