ISyE 6404 – Nonparametric Statistics

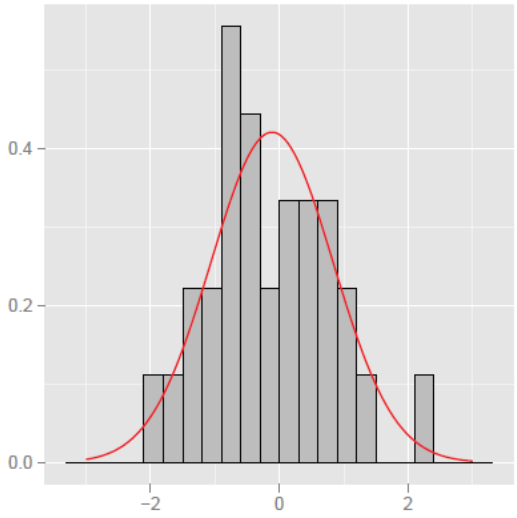
Lecture #6 – Density Estimation and Curve-Fitting

(Textbook page 205 and 243, respectively)

11.1 Histogram

The histogram provides a quick picture of the underlying density by weighting fixed intervals according to their relative frequency in the data.

In R, the function `hist()` in the R's base graphic system will create a histogram using the input vector `x`. Also advanced graphic system such as `ggplot2` package produces the same result using `geom_histogram()` function. Figure 11.2 shows (a) the empirical density function where vertical bars represent Dirac's point masses at the observations, and (b) a histogram for a set of 30 generated $\mathcal{N}(0,1)$ random variables. Obviously, by aggregating observations within the disjoint intervals, we get a better, *smoother* visual construction of the frequency distribution of the sample.



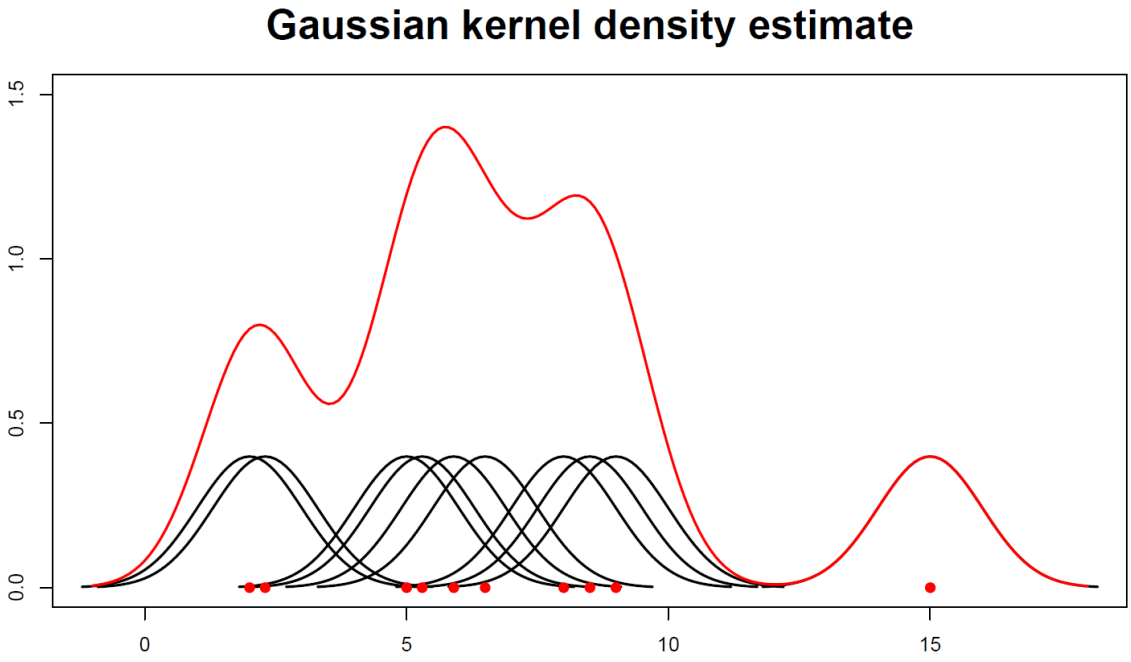
11.2 Kernel and Bandwidth

In this chapter, we focus on the kernel density estimator that more fairly spreads out the probability mass of each observation, not arbitrarily in a fixed interval, but

smoothly around the observation, typically in a symmetric way. With a sample $X_1, \ldots, X_n$, we write the density estimator

$$\hat{f}(x) = \frac{1}{nh_n} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h_n}\right), \tag{11.1}$$

for $X_i = x_i$, $i = 1, \ldots, n$. The *kernel function* K represents how the probability mass is assigned, so for the histogram, it is just a constant in any particular interval. The smoothing function h_n is a positive sequence of bandwidths analogous to the bin width in a histogram.



The kernel function K has five important properties –

$$
\begin{array}{lrll}
1. & K(x) & \geq 0 & \forall x \\
2. & K(x) & = K(-x) & \text{for } x > 0 \\
3. & \int K(u)du & = 1 & \\
4. & \int uK(u)du & = 0 & \\
5. & \int u^2 K(u)du & = \sigma_K^2 < \infty. &
\end{array}
$$

Figure 11.4 shows four basic kernel functions:

1. Normal (or Gaussian) kernel $K(x) = \phi(x)$,
2. Triangular kernel $K(x) = c^{-2}(c - |x|)\,\mathbf{1}(-c < x < c),\ c > 0.$
3. Epanechnickov kernel (described below).
4. Box kernel, $K(x) = \mathbf{1}(-c < x < c)/(2c),\ c > 0.$

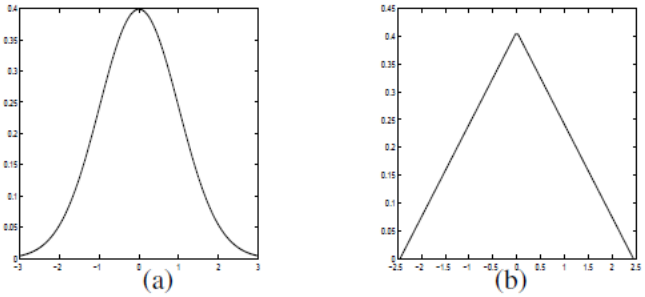


Figure 11.4: (a) Normal and (b) Triangular Kernel Functions

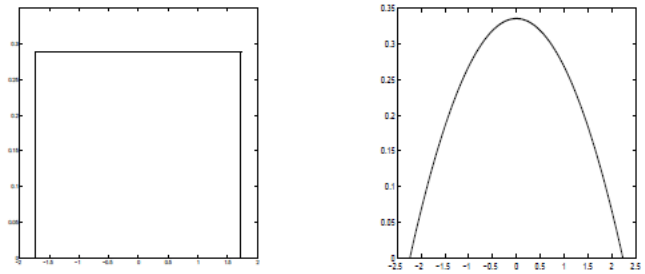


Figure 11.4: (c) Box and (b) Epanechnickov Kernel Functions

While K controls the shape, h_n controls the spread of the kernel. The accuracy of a density estimator can be evaluated using the mean integrated squared error, defined as

$$\begin{aligned} \text{MISE} &= \mathbb{E}\left(\int (f(x) - \hat{f}(x))^2 dx\right) \\ &= \int \text{Bias}^2(\hat{f}(x))dx + \int \mathbb{V}\text{ar}(\hat{f}(x))dx. \end{aligned} \tag{11.2}$$

- Perfomance of kernel is measured by MISE (mean integrated squared error) or AMISE (asymptotic MISE).
- Epanechnikov kernel minimizes AMISE and is therefore optimal.
- Kernel efficiency is measured in comparison to Epanechnikov kernel.

Kernel	Efficiency
Epanechnikov	1.000
Biweight	0.994
Triangular	0.986
Normal	0.951
Uniform	0.930

$\Rightarrow$ Choice of kernel is not as important as choice of bandwidth.

To find a density estimator that minimizes the MISE under the five mentioned constraints, we also will assume that $f(x)$ is continuous (and twice differentiable), $h_n \to 0$ and $nh_n \to \infty$ as $n \to \infty$. Under these conditions it can be shown that

$$\begin{aligned} \text{Bias}(\hat{f}(x)) &= \frac{\sigma_K^2}{2} f''(x) + O(h_n^4) \text{ and} \\ \mathbb{V}\text{ar}\left(\hat{f}(x)\right) &= \frac{f(x)R(K)}{nh_n} + O(n^{-1}), \end{aligned} \tag{11.3}$$

where $R(g) = \int g(u)^2 du$.

We determine (and minimize) the MISE by our choice of h_n. From the equations in (11.3), we see that there is a tradeoff. Choosing h_n to reduce bias will increase the variance, and vice versa. The choice of bandwidth is important in the construction of $\hat{f}(x)$. If h is chosen to be small, the subtle nuances in the main part of the density will be highlighted, but the tail of the distribution will be unseemly bumpy. If h is chosen large, the tails of the distribution are better handled, but we fail to see important characteristics in the middle quartiles of the data.

By substituting in the bias and variance in the formula for (11.2), we minimize MISE with

$$h_n^* = \left(\frac{R(K)}{\sigma_K^4 R(f')} \right)^{1/5} n^{-1/5}.$$

At this point, we can still choose $K(x)$ and insert a "representative" density for $f(x)$ to solve for the bandwidth. Epanechnickov (1969) showed that, upon substituting in $f(x) = \phi(x)$, the kernel that minimizes MISE is

$$K_E(x) = \begin{cases} \frac{3}{4}(1 - x^2) & |x| \leq 1 \\ 0 & |x| > 1. \end{cases}$$

The resulting bandwidth becomes $h_n^* \approx 1.06 \hat{\sigma} n^{-1/5}$, where $\hat{\sigma}$ is the sample standard deviation. This choice relies on the approximation of σ for $f(x)$. Alternative approaches, including cross-validation, lead to slightly different answers.

See https://en.wikipedia.org/wiki/Kernel_density_estimation for Aymptotic MISE (AMISE)'s definition. Note that

$$\text{MISE}\,(h) = \text{AMISE}(h) + o(1/(\text{nh}) + \text{h}^4).$$

Bandwidth selection

– Simple versus high-tech selection rules.

– Objective function: MISE/AMISE.

– R-function `density` offers several selection rules.

Comparison of bandwidth selectors

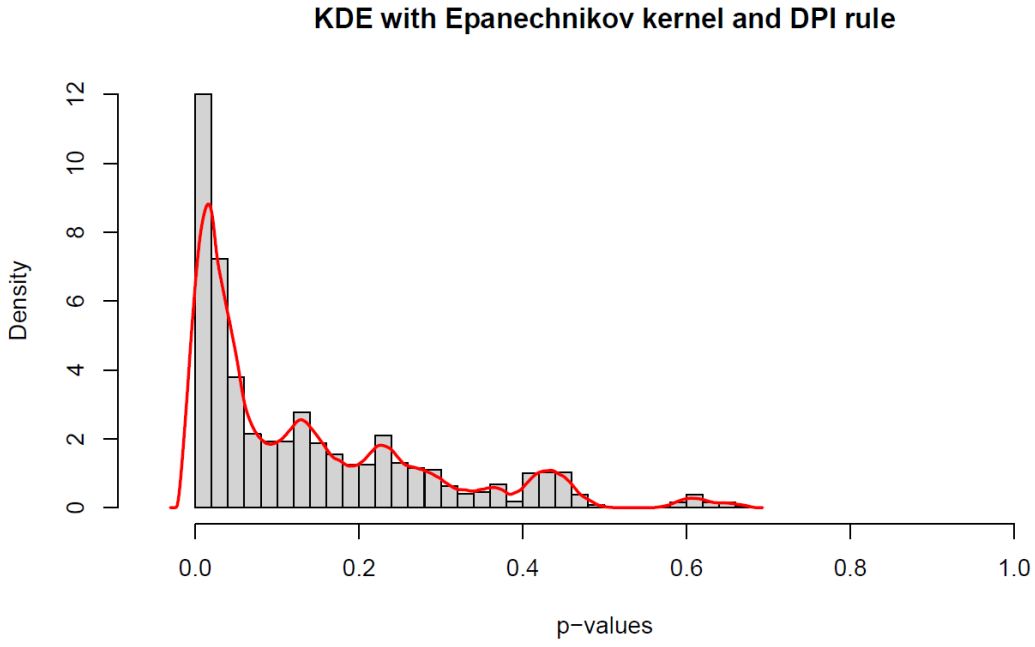
– Simulation results depend on selected true densities.

– Selectors with pilot bandwidths perform quite well but rely on asymptotics $\Rightarrow$ less accurate for densities with "sharp features" (e.g. multiple modes).

– UCV has high variance but does not depend on asymptotics.

– BCV performs bad in several simulations.

– Authors' recommendation: DPI or STE better than UCV or BCV.



```
bw.SJ(method=c("ste", "dpi"))
```

- The AMISE optimization involves the estimation of density functionals like integrated squared density derivatives.
- `dpi`: Direct plug-in rule. Estimates the needed functionals by KDE. Problem: Choice of pilot bandwidth.
- `ste`: Solve-the-equation rule. The pilot bandwidth depends on h.

density

From stats v3.5.1by R-core R-core@R-project.org
99.99th

Percentile

Kernel Density Estimation

The (S3) generic function `density` computes kernel density estimates. Its default method does so with the given kernel and bandwidth for univariate observations.

Keywords

distribution, smooth

Usage

```
density(x, …)
# S3 method for default
density(x, bw = "nrd0", adjust = 1,
        kernel = c("gaussian", "epanechnikov", "rectangular",
                   "triangular", "biweight",
                   "cosine", "optcosine"),
        weights = NULL, window = kernel, width,
        give.Rkern = FALSE,
        n = 512, from, to, cut = 3, na.rm = FALSE, …)
```

Arguments

x

the data from which the estimate is to be computed. For the default method a numeric vector: long vectors are not supported.

bw

the smoothing bandwidth to be used. The kernels are scaled such that this is the standard deviation of the smoothing kernel. (Note this differs from the reference books cited below, and from S-PLUS.)

`bw` can also be a character string giving a rule to choose the bandwidth. See `bw.nrd`. The default, `"nrd0"`, has remained the default for historical and compatibility reasons, rather than as a general recommendation, where e.g., `"SJ"` would rather fit, see also Venables and Ripley (2002).

The specified (or computed) value of `bw` is multiplied by `adjust`.

adjust

the bandwidth used is actually `adjust*bw`. This makes it easy to specify values like 'half the default' bandwidth.

kernel, window

a character string giving the smoothing kernel to be used. This must partially match one of `"gaussian"`, `"rectangular"`, `"triangular"`, `"epanechnikov"`, `"biweight"`, `"cosine"` or `"optcosine"`, with default `"gaussian"`, and may be abbreviated to a unique prefix (single letter).

`"cosine"` is smoother than `"optcosine"`, which is the usual 'cosine' kernel in the literature and almost MSE-efficient. However, `"cosine"` is the version used by S.

weights

numeric vector of non-negative observation weights, hence of same length as `x`. The default `NULL` is equivalent to `weights = rep(1/nx, nx)` where `nx`is the length of (the finite entries of) `x[]`.

```
# NOT RUN {

require(graphics)


plot(density(c(-20, rep(0,98), 20)), xlim = c(-4, 4))  # IQR = 0


# The Old Faithful geyser data

d <- density(faithful$eruptions, bw = "sj")

d

plot(d)
```

Adaptive kernels were derived to alleviate this problem. If we use a more general smoothing function tied to the density at x_j, we could generalize the density estimator as

$$\hat{f}(x) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{h_{n,i}}K\left(\frac{x-x_i}{h_{n,i}}\right). \tag{11.4}$$

This is an advanced topic in density estimation, and we will not further pursue learning more about optimal estimators based on adaptive kernels here. We will also leave out details about estimator limit properties, and instead point out that if h_n is a decreasing function of n, under some mild regularity conditions, $|\hat{f}(x) - f(x)| \xrightarrow{P} 0$. For details and more advanced topics in density estimation, see Silverman (1986) and Efromovich (1999).

The (univariate) density estimator from R, called

`density(data)`

is illustrated in Figure 11.5 using a sample of seven observations. The default estimate is based on a gaussian kernel; to use another kernel, just enter 'rectangular', 'triangular', or 'epanechnikov' (see code below). Figure 11.5 shows how the normal kernel compares to the (a) rectangular, (2) triangle and (c) epanechnikov kernels. Figure 11.6 shows the density estimator using the same data based on the normal kernel, but using five different bandwidth selectors. Note the optimal bandwidth (0.5689) for the default selector(`bw.nrd0`) can be found by looking the result in the command line.

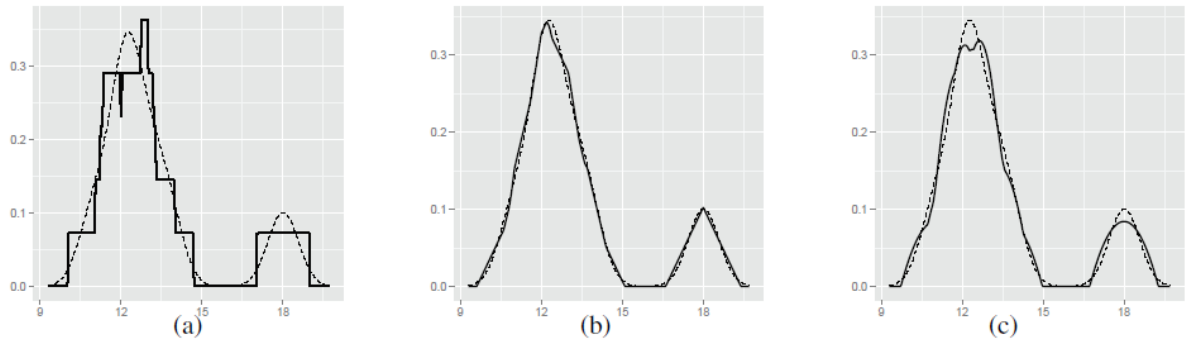


Figure 11.5 Density estimation for sample of size n=7 using various kernels: (all) Gaussian, (a) Rectangular, (b) Triangular, (c) Epanechnikov.