CHAPTER 7

Enhancing Scatterplots with Smoothed Densities

Abstract

Scatterplots of microarray data generally contain a very large number of dots, making it difficult to get a good impression of their distribution in dense areas. We present a fast an simple algorithm for two-dimensional histogram smoothing to visually enhance scatterplots. Functions for Matlab and R are available from the authors.

7.1 Introduction

The scatterplot is a simple but effective tool in microarray analysis. It is one of the best ways to visualize expressions of two arrays (or of two dye colours on one array). Still the scatterplot leaves much to be desired. Because of the large number of dots, up to ten thousand or more, large parts of the picture can become completely black. Then it is hard to get a good impression of the distribution of the spots. Figure 7.1 shows an example. When the plotting symbols are large, as in the left panel, the center of the graph gets completely filled with ink. As the right panel shows, it helps to use very small symbols, but then isolated dots can easily be missed.

A solution is to move from plotting of the individual dots to a presentation of their empirical distribution. An obvious choice is the two-dimensional histogram. Unfortunately, either one has to use rather wide bins, or to accept a rather choppy histogram. Figure 7.2 shows examples.

This is a pre-copy-editing, author-produced version of an article accepted for publication in *Bioinformatics* following peer review. The definitive publisher-authenticated version: P. H. C. Eilers and J. J. Goeman (2004). Enhancing scatterplots with smoothed densities. *Bioinformatics 20*(5), 623–628 is available online at: `http://dx.doi.org/10.1093/bioinformatics/btg454`.

We can achieve large improvements if we use a histogram with narrow bins and additional smoothing, as is shown in Figure 7.3, which is based on histograms with 200 bins in both directions. In this paper we present an algorithm for fast and effective smoothing of two-dimensional histograms. Speed is important, because in everyday work many scatterplots are made on a computer screen to help in exploratory data analysis.

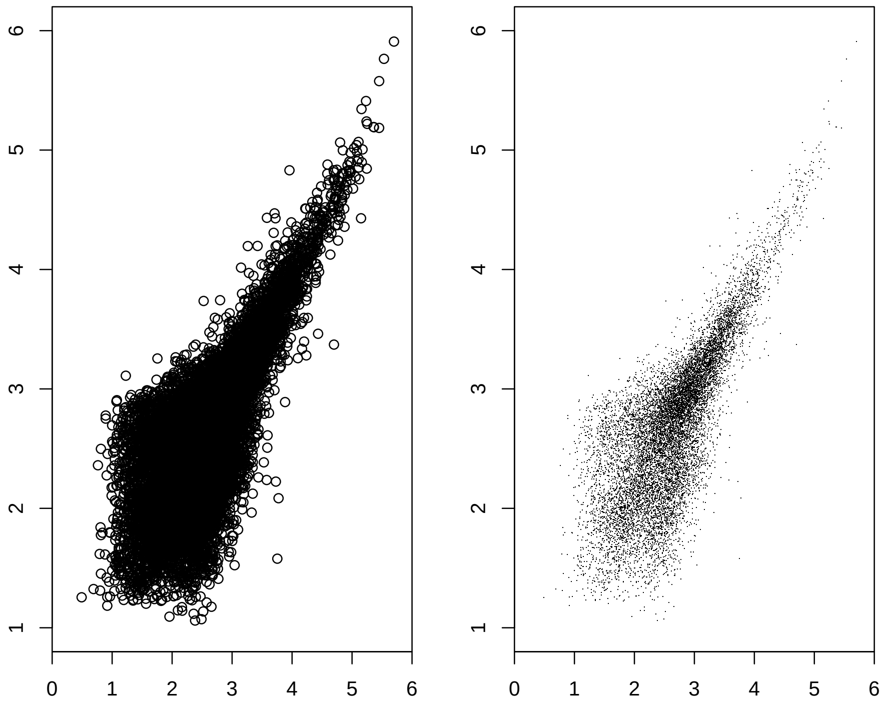

FIGURE 7.1: *Two scatterplots of log-expressions of a pair of microarrays. Left: large symbols, right: small symbols.*

7.2 Algorithm

The two-dimensional histogram is a natural generalization of the well-known histogram. The x-y domain is cut into rectangles and the number of observations in each rectangle is counted. As Figure 7.2 shows, a graphical display of this raw histogram is not a great succes. We can make it more informative (and

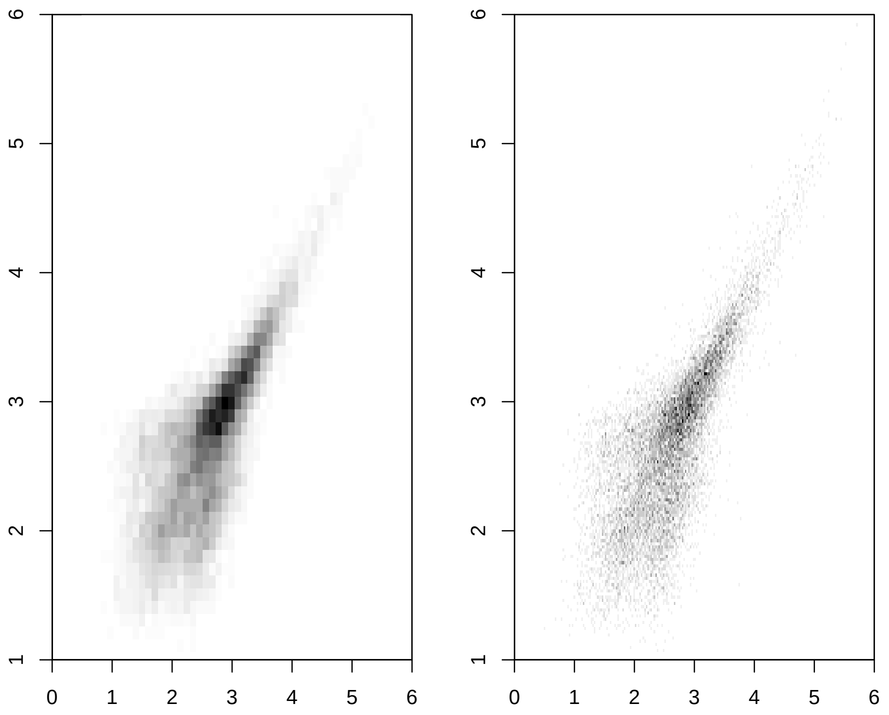

FIGURE 7.2: *Two-dimensional histograms, derived from the scatterplots in Figure 1. Left: 50 by 50 bins, right: 200 by 200 bins.*

attractive) with a simple smoothing algorithm.

Let H be the matrix of counts resulting from a two-dimensional histogram. Consider smoothing of one column of H, we will call it the vector y, to get a vector z. The distance from z to y can be measured as the sum of squares of the residuals $y - z$:

$$S = \sum_i (y_i - z_i)^2 = |y - z|^2.$$

The roughness of z can be measured by first computing differences,

$$\Delta z_i = z_i - z_{i-1},$$

and then summing their squares:

$$R = \sum_i (\Delta z_i)^2 = |D_1 z|^2.$$

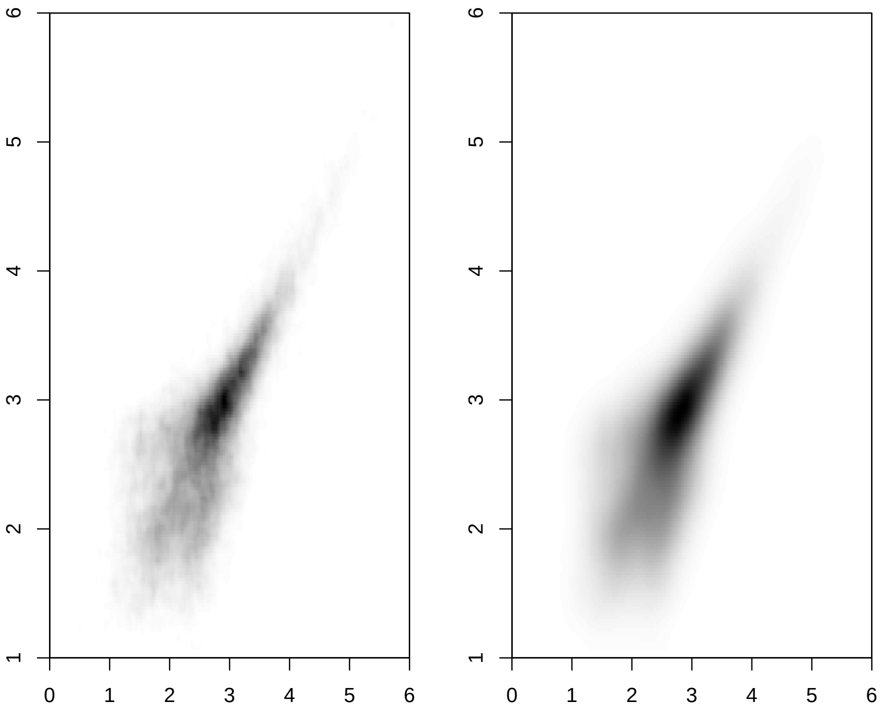

FIGURE 7.3: *Smoothed two-dimensional histograms, derived from the scatterplots in Figure 1, using 200 by 200 bins. Left:* $\lambda = 1$, *right:* $\lambda = 10$.

Here D_1 is a first-order difference matrix such that $D_1 z = \Delta z$:

$$D_1 = \begin{pmatrix} -1 & 1 & & \emptyset \\ & \ddots & \ddots & \\ \emptyset & & -1 & 1 \end{pmatrix}.$$

We combine S and R in one penalized least squares function Q:

$$Q = S + \lambda R = |y - z|^2 + \lambda |D_1 z|^2, \tag{7.1}$$

and compute the vector $\hat{z}$ that minimizes Q. By changing λ we can balance our preference between fit to the data y (the first term) and roughness of z (the second term). The higher λ, the more the roughness of z will be penalized, leading to a smoother result, at the cost of the fit to y getting worse. The minimizer of

Q is the solution to the following linear system of equations:

$$(I + \lambda D_1' D_1)\hat{z} = y, \tag{7.2}$$

where I is the identity matrix. For moderate lengths of y, say 200 or less, it can be solved very quickly on modern computers.

A refinement uses second order differences:

$$\Delta^2 z_i = \Delta(\Delta z_i) = (z_i - z_{i-1}) - (z_{i-1} - z_{i-2}) = z_i - 2z_{i-1} + z_{i-2}.$$

The only change to the system (7.2) is that D_1 is changed to a second order difference matrix

$$D_2 = \begin{pmatrix} 1 & -2 & 1 & & \emptyset \\ & \ddots & \ddots & \ddots & \\ \emptyset & & 1 & -2 & 1 \end{pmatrix}.$$

and λ changed to λ^2.

Figure 7.4 shows one column of H and the effect of smoothing with different values of λ, using first or second order differences. The latter choice gives a somewhat smoother result and follows the peaks better. However, there is a slight problem: we can get negative values if we use second order differences, especially with strong smoothing. The explanation is shown in Figure 7.5, where the impulse response of the smoothers is displayed. Imagine a degenerate histogram with zeroes in all cells but one, which contains a one. Smoothing this impulse shows what happens to one count. Any histogram can be interpreted as a sum of many of these impulses, with different positions of the single count. Because the smoother is linear, the smoothed histogram is the sum of the corresponding smoothed impulses. With first-order differences the impulse response has the shape of decaying exponentials in both directions and it cannot become negative. With second-order differences, each branch of the impulse response consists of two exponentials, in a combination that leads to a negative minimum.

For visual display, negative values of the smoothed histogram are not really a problem. But it is inelegant and it can be harmful when results are used for further computations that expect non-negative probabilities. A solution is to use both a first and second-order penalty (Eilers, 1994). We use the penalty $\lambda^2|D_2 z|^2 + \alpha\lambda|D_1 z|^2$ and search for (a "pleasant" number) α that keeps the impulse response from becoming non-positive; $\alpha = 2$ is a round number that works well. The penalized least squares function becomes.

$$Q = |y - z|^2 + \lambda^2|D_2 z|^2 + 2\lambda|D_1 z|^2. \tag{7.3}$$

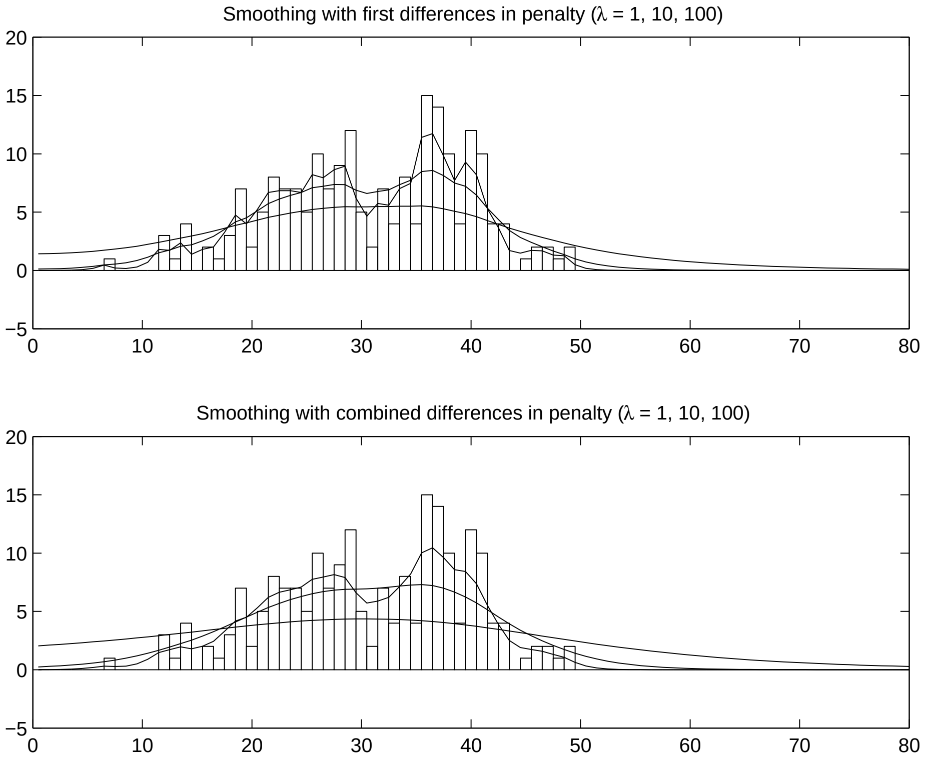

FIGURE 7.4: *Smoothing of a one-dimensional histogram. Top: using first-order differences, bottom: using second-order differences.*

The impulse response of this smoother is also shown in Figure 7.5. The peak is rounded like that of the second-order smoother and the tails are like that of the first-order smoother.

The idea of using differences in a penalty goes back at least to Whittaker (1923). Extensions and fast algorithms for one-dimensional smoothing have been presented elsewhere (Eilers, 1994, 2003). An attractive property of this smoother is that it respects boundaries. This is unlike a kernel density smoother, which computes a weighted local mean and implicitly assumes zero counts past the boundaries of a histogram. This can do little harm on densities with tails that gradually slope down. But it is when a density has its peak at, or near, zero. The peak will be rounded too much by a kernel smoother. This type of density frequently occurs with when one studies squares of absolute values of data, to get an impression of variance or standard deviation.

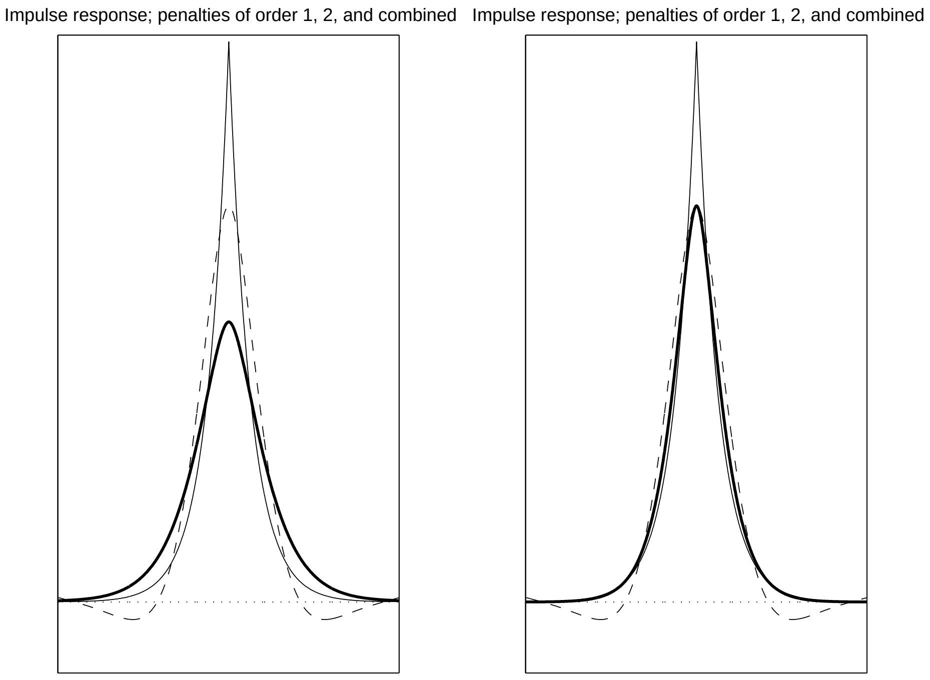

FIGURE 7.5: *Impulse response of the smoother (arbitrary scales). Peaked curve: first-order differences; rounded broken-line curve with negative lobe: second-order differences; thick-line curve: combined differences. Left panel: equal values of λ; right panel: $\lambda/2$ for combined differences.*

Once we have a good smoother for vectors, it is trivial to apply it to all columns of a matrix. The standard algorithms in Matlab, R or S-plus for the solution of linear equations accept a matrix as the right-hand side and return the solution as a matrix. Thus we can write the smoothing of a matrix Y, to get Z as a simple modification of (7.2):

$$(I + 2\lambda D_1' D_1 + \lambda^2 D_2' D_2)\hat{Z} = Y. \tag{7.4}$$

This is the basis for fast smoothing of a two-dimensional histogram H: first smooth the columns of H to get, say, G and then smooth the columns of G' (which are the rows of G) to get F', the transpose of the desired result. It is easily checked that the result is invariant to the order of the smoothing operations: smoothing the rows of H before the columns leads to the same result.

Only a few lines of Matlab are needed to apply the smoother to a histogram given in a matrix `H`: `F = expsm(expsm(H, lambda)', lambda)';`, where `expsm` is a function defined as

```
function Z = expsm(Y, lambda)
m = size(Y, 1);
E = eye(m);
D1 = diff(E);
D2 = diff(D1);
P = lambda ^ 2 * D2' * D2 + 2 * lambda *  D1' * D1;
Z = (E + P) \ Y;
```

An implementation in S-plus or R would look very similar. Note that Matlab can speed up the computations (about 3 times) by exploiting the sparseness of the system of equations in a very simple way, using the sparse identity matrix `speye` instead of the full `eye`.

7.3 Implementation

Figure 7.6 shows the application to four different displays of one pair of arrays with 12625 expressions. The NW panel shows a scatterplot derived from 12625 log-expressions (base 10). The NE panel shows mean and difference. The SW panel displays mean and absolute value of the difference. This is an example of a skew density with a peak near the origin (for each column of the histogram). The logarithms of the absolute differences (plus a shift of 0.01 to accommodate zeroes) are shown in the SE panel. In each graph a random selection of 1000 data points is also plotted.

The choice of λ partly is a matter of taste. The user should play with it to get a visual appearance to his/her taste. It also depends on the number of bins and the number of data pairs. Our personal experience is that λs in the range from 1 to 100 work well with 100 or 200 bins per dimension and approximately 10^4 data pairs.

The colour scale is also a matter of taste. Using white for zero values and a dark colour for the maximum seems attractive (and saves expensive ink or toner). We advice to take the colour for the maximum not too dark, so that black symbols for the data points will be clearly visible.

7.4 Discussion

We have presented an algorithm for visual enhancement of a scatterplot, using a smoothed histogram. The algorithm is fast: computing and plotting Figure 7.6 takes less than a second, using Matlab 6.5 on a 1000 MHz Pentium III PC. So it can be used in a routine way when exploring scatterplots and one can nearly instantaneously see the effects of changing the amount of smoothing. Even

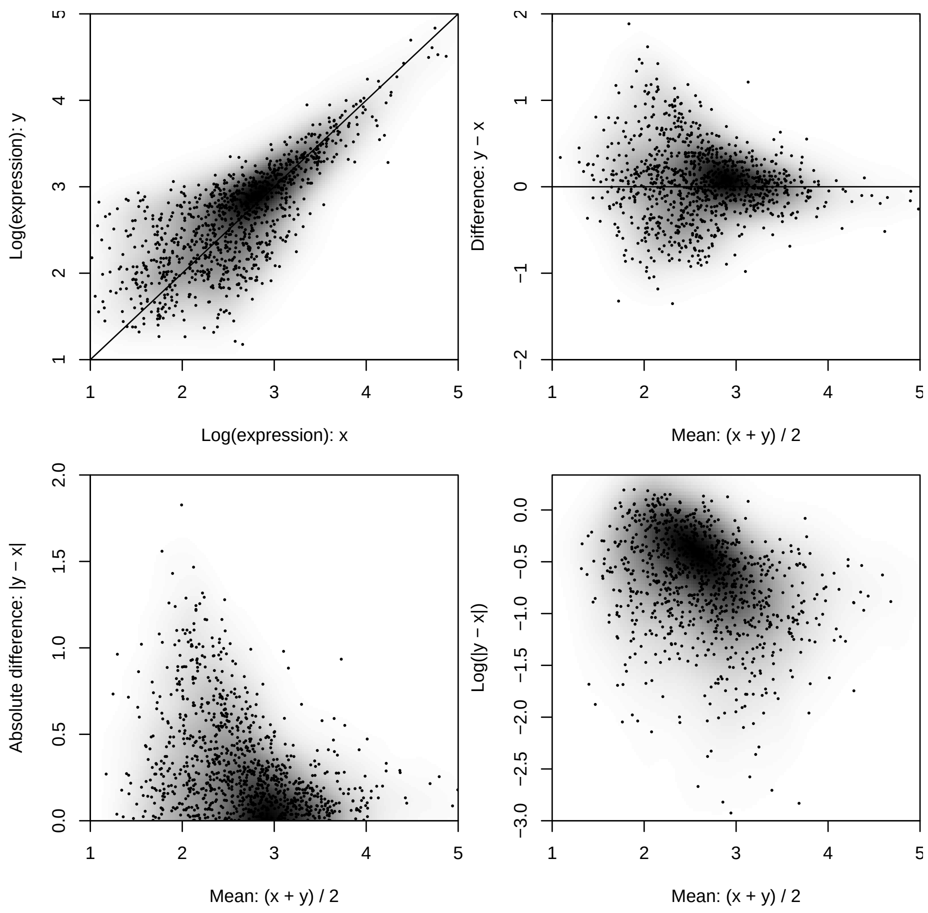

FIGURE 7.6: *Four different displays of a pair of microarrays, using histogram smoothing and plotting of 1000 (of 12625) data points.*

one million data points are handled in less than 10 seconds. R is several times slower; the bottleneck there is the computation of the histogram.

We investigated the performance of the R function `kde2d()` for kernel estimation of two-dimensional densities. With 2000 data points or less it has the same performance as the our algorithm. With 10^4 data points it is over five times slower and above 3×10^4 data points too much memory is needed (on a 256 Mb PC). Either swapping to the hard disk slows done the process, or the computations stop with an error message.

In our experience it is useful to plot part of the dots, to give a good impression of the spread of the raw data. Their number should not be too small, to be representative enough, but also not too large, to not fill the graph with too much ink. A subject for further research is to use the estimated density to determine the probability of plotting a point.

There exist good algorithms for density smoothing, like kernels and local likelihood and they will produce results that look much like ours. Our method is also not very original: penalized likelihood has been used before. But the algorithm presented here has a number of specific advantages:

1. It does not use any special smoothing libraries, but only a few lines of straightforward linear algebra computations, which are easily implemented in high-level languages like Matlab, S-plus or R.
2. It works directly on the two-dimensional histogram matrix, avoiding translation to triples (row, column, count) that other algorithms demand.
3. It respects domain boundaries, which is important when smoothing densities of very skewly distributed data, like variance estimates.
4. It is fast.
5. It can handle extremely large (10^6 or more) numbers of data points.

The four displays of the "scatterquad" in Figure 7.6 help to better understand systematic and random differences between two microarrays. Further refinements seem possible. One could use a smoothing algorithm to estimate and display trends in the upper panels and use trend-corrected differences for the displays of spread in the lower panels and possibly add trends to these plots as well. We will not pursue this issue here further, as it would carry us too far away from the main theme of the paper.

Our approach to visualization of scatterplots is in essence a simplification of density smoothing in two dimensions. Because visual display is the only goal, a refined algorithm is unnecessary. Simonoff (1996) discusses kernel estimation, while Loader (1999) presents algorithms and software for local likelihood.