Variable-Resolution Bivariate Plots

Chisheng Huang *
Institute of Statistical Science
Academia Sinica
John Alan McDonald ⋆
Werner Stuetzle ⋆
Department of Statistics
University of Washington
Seattle, WA 98195

January 24, 1997

Abstract

Scatterplots are the method of choice for displaying the distribution of points in two dimensions. They are used to discover patterns such as holes, outliers, modes, and association between the two variables. A common problem is overstriking, the overlap on the plotting surface of glyphs representing individual observations. Overstriking can create a misleading impression of the data distribution. The variable resolution bivariate plots (*Varebi plots*) proposed in this paper deal with the problem of overstriking by mixing display of a density estimate and display of individual observations. The idea is to determine the display format by analyzing the actual amount of overstriking on the screen. Thus, the display format will depend on the sample size, the distribution of the observations, the size and shape of individual icons, and the size of the window. It may change automatically when the window is resized. Varebi plots reveal detail wherever possible, and show the overall trend when displaying detail is not feasible.

*Supported by DOE grant DE-FG06-85-ER25006 and NSF grant DMS-9114027

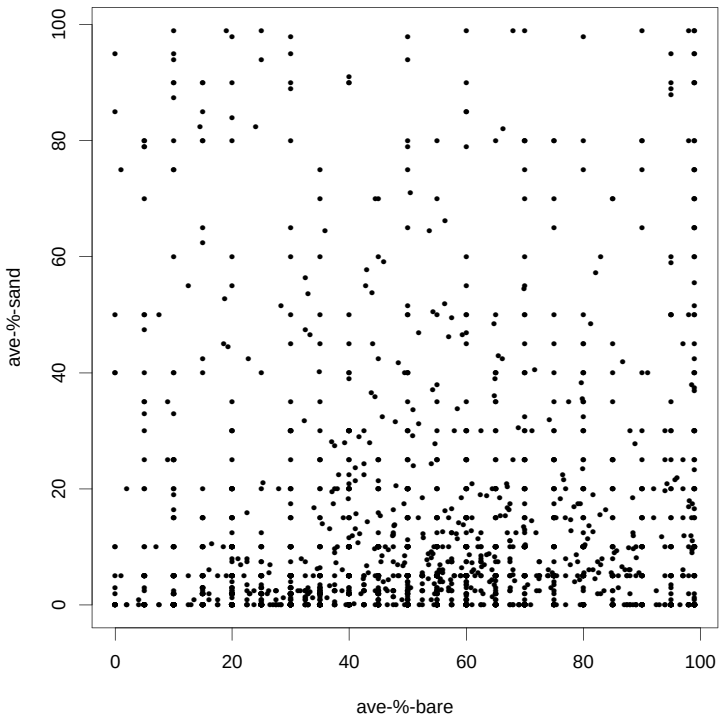


Figure 1: Where is the mode?

1 Introduction and Motivation

Scatterplots are the method of choice for visualizing the distribution of points in two dimensions. They are used to discover patterns such as holes (areas with few data points), outliers, modes, association between the two variables, etc.

A common problem with scatterplots is overstriking, the overlap on the plotting surface of glyphs representing individual observations. Overstriking can create a misleading impression of the data distribution. As an example, consider Fig. 1. This scatterplot was produced during the analysis of data on a colony of Magellanic penguins in Punta Tombo, Argentina. The observations are penguin nest sites. The variables are ground composition at a site (percentage of sand) on the vertical axis and vegetation coverage (percentage of bare ground) on the horizontal axis. The *displayed* point pattern suggests the mode to be within the rectangle defined by $x = (30, 80)$ and $y = (0, 15)$. However, Fig. 2, obtained by binning the points into a 20×20 grid and encoding the counts into gray levels, reveals that the visual impression is misleading and that there is a mode in the lower right corner of the plot.

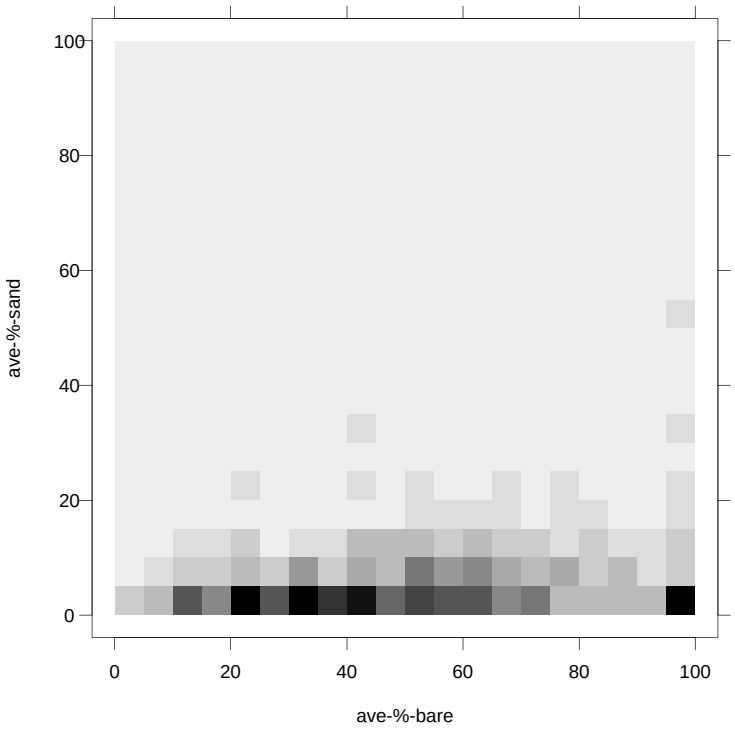


Figure 2: Same data as Figure 1; 2d histogram drawn by encoding bin counts in gray scale.

Overstriking is obvious if the data set is so large or the plotting surface so small that we just see a solid blob of ink. (We use the term "ink" although most plots today are drawn on a computer screen). However, as our example illustrates, it can be a problem even with small data sets (hundreds or thousands of observations), where it is difficult to notice and therefore more insidious.

A way of coping with the problem of overstriking is to abandon the idea of drawing individual points and instead draw *agglomerative glyphs* representing collections of points. We can, for example, bin the drawing area into rectangular or hexagonal bins and compute a two-dimensional histogram (Carr et al, 1987). The histogram can be drawn as a perspective plot, or we can encode the counts in gray scale (as in Fig. 2) or glyph size (Fig. 3).

Besides force of habit and inertia there are at least two other arguments against routinely replacing scatterplots by two-dimensional histograms:

- The discretization inherent in a histogram smears out fine structure. As an illustration,

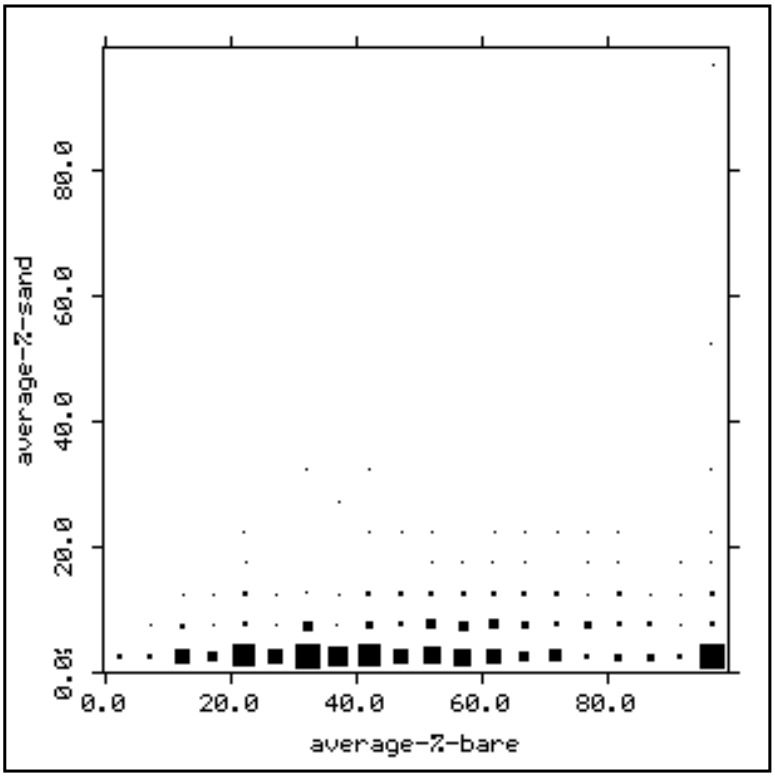


Figure 3: Same data as Figure 1; 2d histogram drawn by encoding bin counts in rectangle size.

Fig. 4 (a) shows a snapshot of 200 rotating 3D points whose coordinates were generated by the infamous RANDU (Knuth, 1981) random number generator. See Tierney (1990) for the particular implementation that we used. Fig. 4 (b), (c), and (d) show histograms of the same 200 points, with different numbers of bins. None of the histograms reveals RANDU's lack of randomness as clearly as the scatterplot.

- Encoding counts into grey level or glyph size requires specification of a mapping. If we want to judge on how many data points a feature of the histogram, like an apparent mode, is based, we need to mentally invert this mapping. This process is not immediate, and we want to avoid it whenever possible.

In this paper we present two suggestions for dealing with the problem of overstriking in scatterplots:

1. We mix individual and agglomerative glyphs in the same plot.

2. We choose between drawing individual and agglomerative glyphs by analyzing the actual amount of overstriking on the screen. The display format thus will depend on the sample size, the distribution of points, the size and shape of the individual glyphs, and the size of the drawing area.

Adapting the type of display to the size of the drawing area is particularly helpful and effective when plots are displayed in windows on a screen, where they can be shrunk to free up space and expanded again for closer inspection.

As an illustration, figures 5 and 6 show the same data set as figures 1 and 2 for different sizes of the drawing area. Notice that the areas around $(65, 16)$ and $(100, 16)$ in Fig. 5 displays an agglomerative glyph and switch to displaying point glyphs in the bigger Fig. 6.

The rest of this paper is organized as follows: In section 2 we describe in detail how Varebi plots are drawn. Section 3 contains additional examples. A discussion concludes the paper.

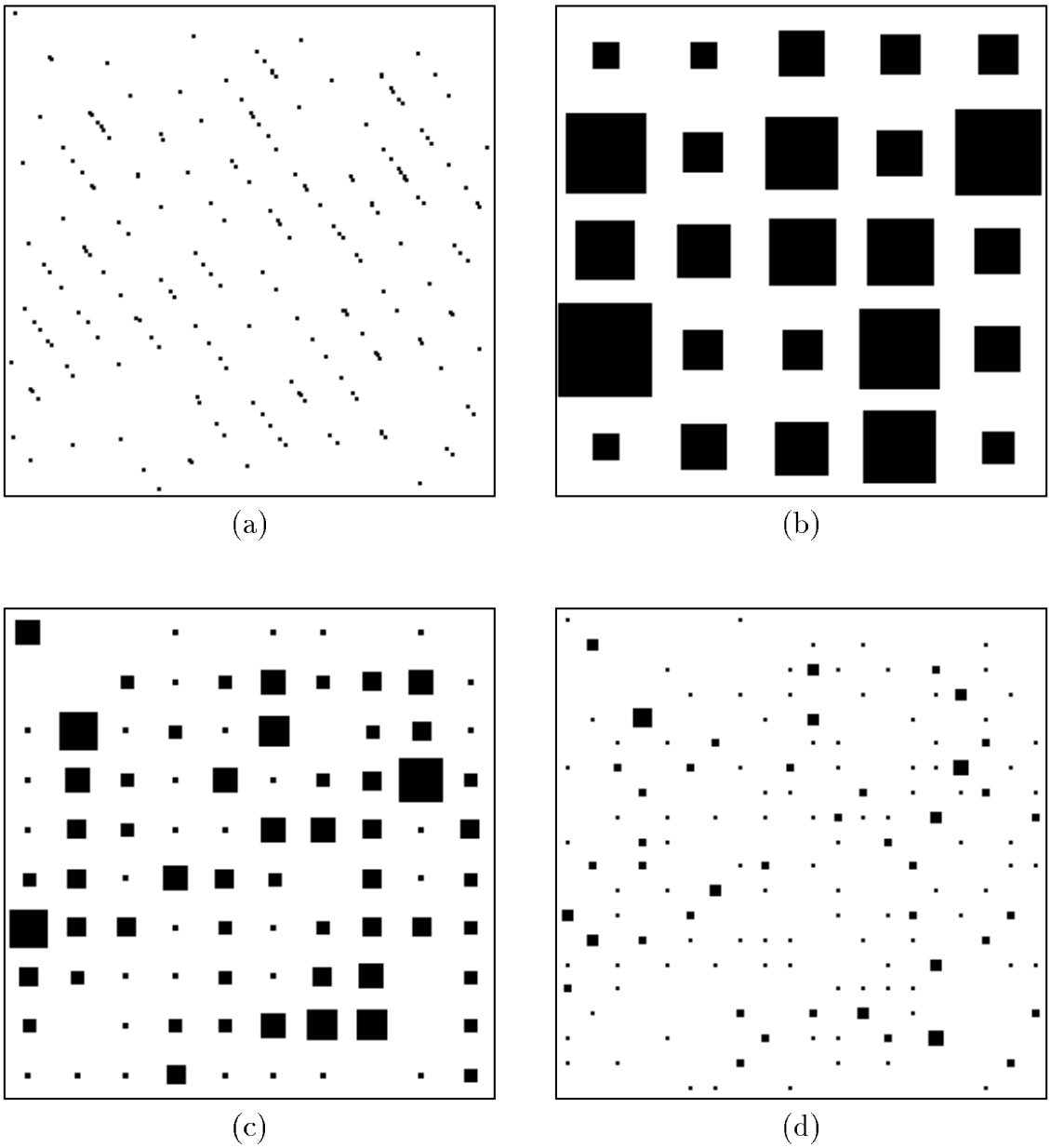


Figure 4: 200 points generated by RANDU. Scatterplot (a) and 2d histograms with 5×5 (b), 10×10 (c) and 20×20 (d) bins.

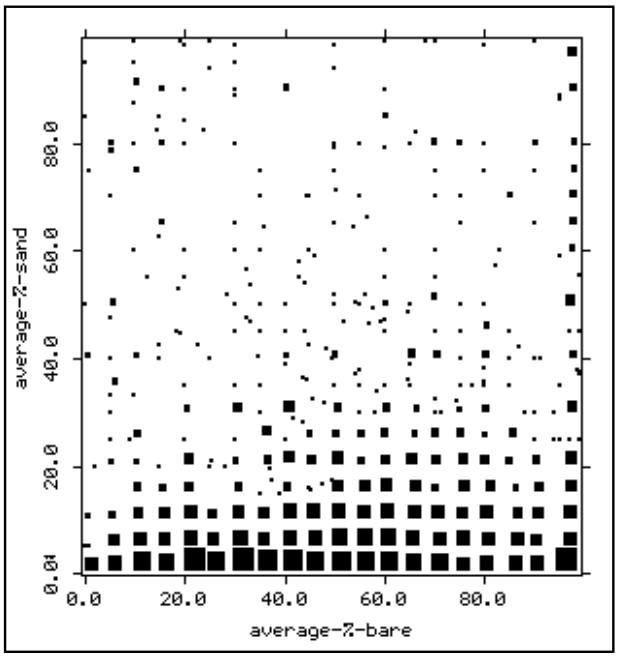


Figure 5: Same data as Figure 1; plot size on a computer screen is 313×334 pixels.

2 Description of Varebi plots

We will first discuss Varebi plots on black and white displays and then describe a variation designed for gray level displays. The gray level version can convey information better on the data distribution when the drawing space is very small.

2.1 Varebi plots on black and white displays

Drawing a Varebi plot involves a sequence of steps:

1. Binning

We bin the data points into a regular grid with a default resolution of 20×20. This results in bins with a side length on the order of 15 pixels for a 300×300 window, a size that we frequently observed ourselves using. A bin size of 15×15 pixels is large enough to allow drawing of agglomerative glyphs in at least five visually distinguishable sizes.

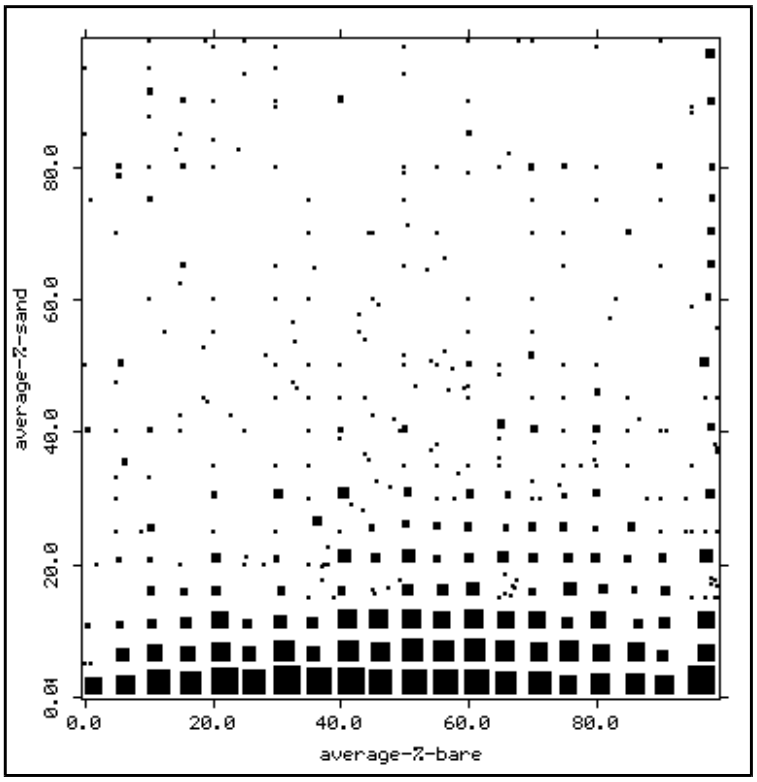


Figure 6: Same data as Figure 1; plot size on a computer screen is 384×397 pixels.

Choosing a fixed number of bins, independent of the plot size on the screen, has the advantage that the plot changes in predictable ways when the window is reshaped. Enlarging the window may change some bins from showing agglomerative glyphs to showing individual observations, but never vice-versa. The analogous statement is true for reduction of the window size.

2. Application of a transfer function

The basic tenet of binary Varebi plots is that the amount of ink actually deposited in a bin should be a non-decreasing function of the number of observations in the bin (up to some tolerance, as described in (3) below). Let s be the bin area on the screen (in pixels), n_{max} the largest bin count, and τ the size of an individual point glyph (in pixels). (On many displays single pixel dots are hard to see, requiring $\tau > 1$.) A non-decreasing function $T : [1 : n_{max}] \to [1 : s]$ assigning an amount of ink to each bin count is called a *transfer function.*

For small bin counts T has to be linear with slope τ, because we want to draw individual observations in bins without overstriking. However, if $\tau \times n_{max} > s$, the simple choice $T_l(n) = \tau \times n$ is not feasible. In this case we have to "blunt" T, i.e. flatten it out for counts above some cutoff n_{crit}. There are many possible ways of doing so: any non-decreasing function T will do as long as $T(n) = \tau \times n$ for $0 \leq n \leq n_{crit} \leq \lfloor s/\tau \rfloor$, and $T(n_{max}) = s$. We chose a simple one, shown in Fig. 7. If we have to use the blunted transfer function T_b, we draw agglomerative glyphs in all bins with bin count $n_{ij} > n_{crit}$, in order to avoid violating the monotonicity condition.

It remains to discuss the choice of n_{crit}. Clearly, n_{crit} should increase when the size of the plot and therefore the bin area s increases — we want to use additional screen space to improve the resolution of the plot, i.e. draw more individual glyphs. We set $n_{crit} = \lfloor s^2/(\tau^2 n_{max}) \rfloor$. The motivation for this choice is that, as $s \to \tau \times n_{max}$, the blunted transfer function T_b approaches T_l. This makes for a smooth transition of display format when the drawing space

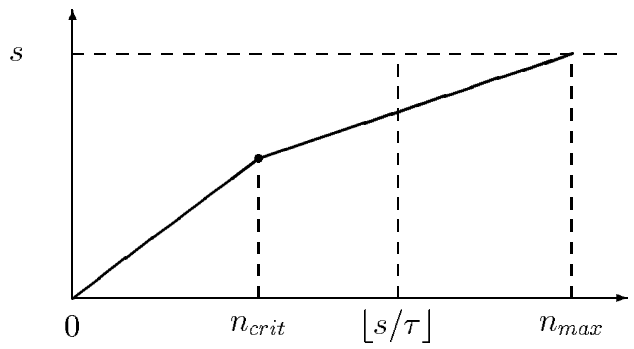


Figure 7: Blunted transfer function

increases to the point where we can switch to the linear transfer function.

3. Calculating the overplotting index and drawing

The final step in producing a Varebi plot is to decide, for each bin, whether to draw individual glyphs or an agglomerative glyph. Let a_{ij} be the amount of ink (number of black pixels) in the corresponding bin on the screen if we simply drew the glyphs without paying any attention to overplotting. We measure the amount of overplotting in a bin by the overplotting index

$$o_{ij} = \frac{\tau \times n_{ij}}{a_{ij}}.$$

If we can use the linear transfer function T_l, we draw agglomerative glyphs for all bins for which the overplotting index o_{ij} is greater than some threshold o_{crit}. If we have to use the blunted transfer function T_b, we draw agglomerative glyphs for all bins with $n_{ij} > n_{crit}$ or $o_{ij} > o_{crit}$. In the examples, $o_{crit} = 1.35$. An agglomerative glyph is drawn as closely as possible to the center of the mass within each bin while keeping the glyph completely inside the bin. This has the additional benefit of breaking the artificial regularity imposed by the grid (Carr et al, 1987).

2.2 Varebi plots on gray level displays

Varebi plots will loose their ability to convey an accurate image of the data distribution when the size of the plotting area gets so small that there are only a few possible values

for the size of an agglomerative glyph. If the display can show gray levels we can, however, represent bin count by gray scale instead of glyph size.

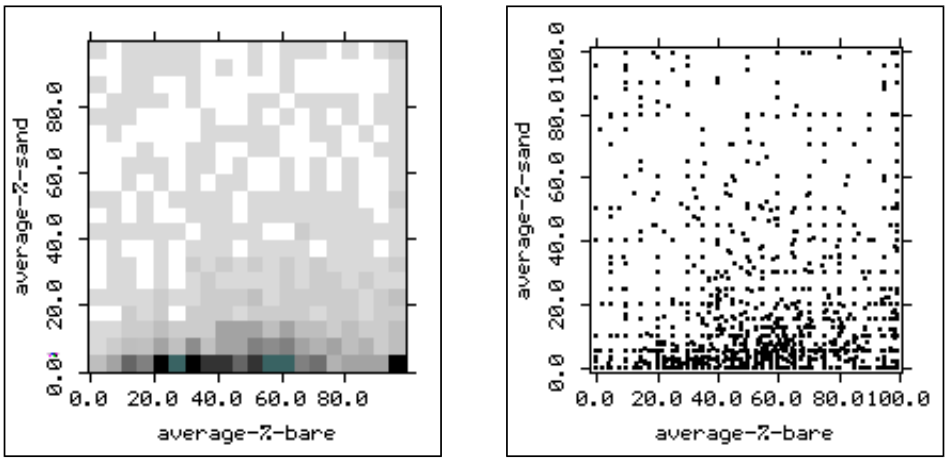
Let n_{min} be the smallest bin count among all the bins with significant overstriking, i.e. with $o_{ij} > o_{crit}$. Bins with $n_{ij} \geq n_{min}$ are drawn in gray. The gray level is determined by a nondecreasing function $\mathcal{D} : [n_{min}, n_{max}] \rightarrow [d_1, d_2]$, where d_1 is lightest and d_2 is darkest. All bins with $n_{ij} < n_{min}$ display individual observations. Therefore, gray level bins always have a higher count than bins showing individual observations. This is necessary, because it is impossible to visually establish an ordering between amount of (black) ink in an area, and gray level.

In the examples we switch to encoding counts by gray level instead of glyph size whenever either bin width or bin height is less than 10 pixels. We do not use gray scale encoding otherwise because we want as many bins as possible to display point glyphs, because perception of gray level is affected by the surrounding area (Foley et al, 1990), and because comparing gray levels is not easy, especially for objects positioned far apart.

2.3 Hexagonal versus rectangular bins

The Varebi plots were implemented using rectangular bins, rather than hexagonal bins as advocated by Carr (1987). Hexagonal bins give slightly better density estimates and result in more eye pleasing displays because they de-emphasize horizontal and vertical directions. We chose rectangular bins for reasons of speed and ease of implementation.

Rectangular bins also have an advantage when Varebi plots are used in conjunction with scatterplot brushing, or when we wish to encode an additional categorical variable in color. In this situation it is not clear how to divide a hexagonal glyph into colored parts, whereas we can convert rectangles into divided color bars.



(a) (b)

Figure 8: Same data as Figure 1; Varebi plot (a) and scatterplot (b) of size 205×212 pixels.

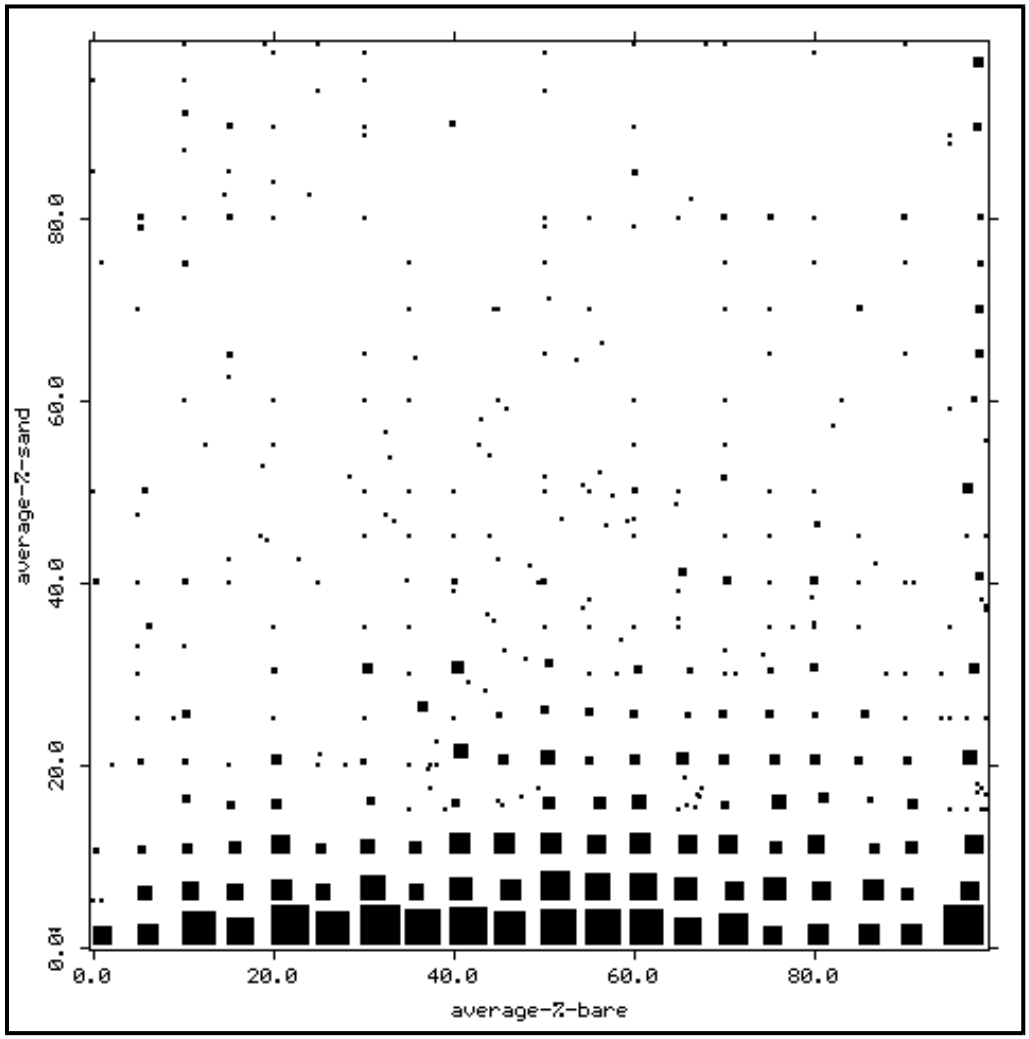
3 Varebi Plot Examples

We present three sets of examples:

- Figures 8a, 9a and Figures 8b, 9b show Varebi plots and conventional scatterplots, respectively, of the same data displayed in Figure 1, for different window sizes. Overstriking in this example is quite serious, so that even a large plot like the one shown in Figure 9, which would take up a quarter of a 1000×1000 screen, displays many agglomerative glyphs.

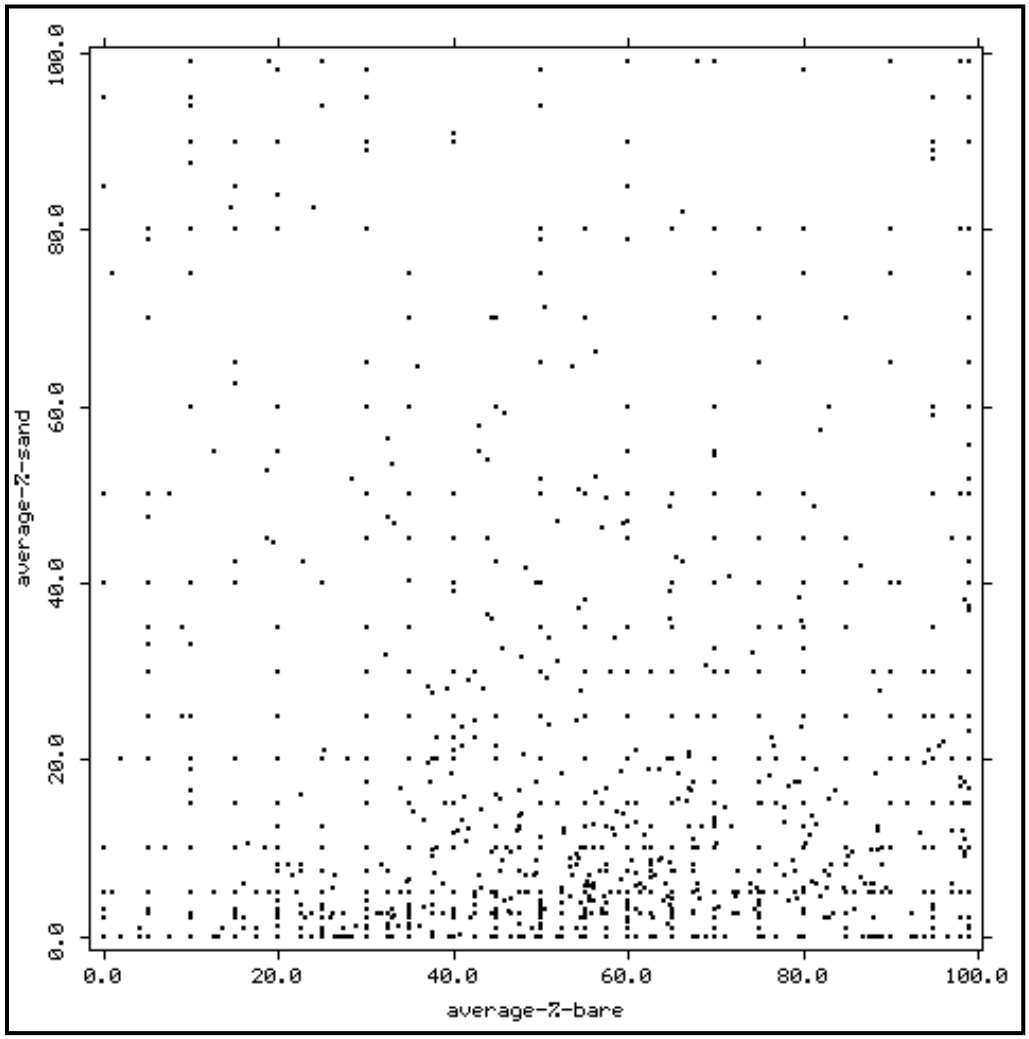
- Figures 10 – 12 show RANDU samples of sizes ranging from 400 to 6,400. Varebi plots are on the left; two-dimensional histograms on the right. Note that, in contrast to the bivariate histograms, all Varebi plots clearly reveal the lack of randomness.

- Figures 13 – 15 show 65,536 pairs of measurements from a 256×256 MRI (Magnetic Resonance Imaging) image. The MRI data set has an enormous peak (38,455 points) at $(0, 0)$, corresponding to the background of the image. The second highest peak corresponds to 70 points. The scatterplot in Figure 13 fails to reveal the peak at the



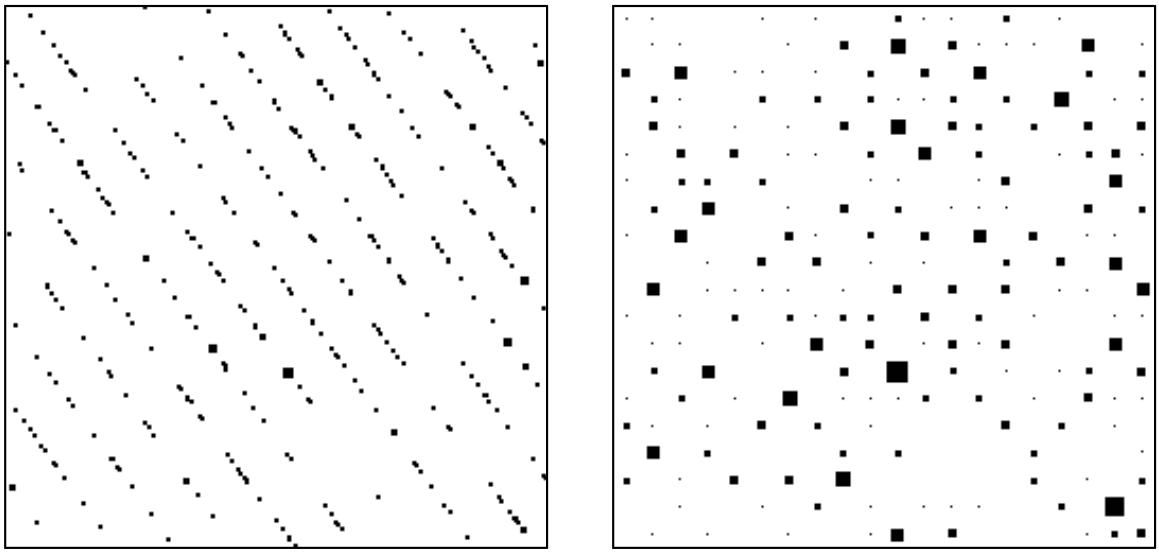
(a)

Figure 9: Same data as Figure 1; Varebi plot of size 480×485 pixels.



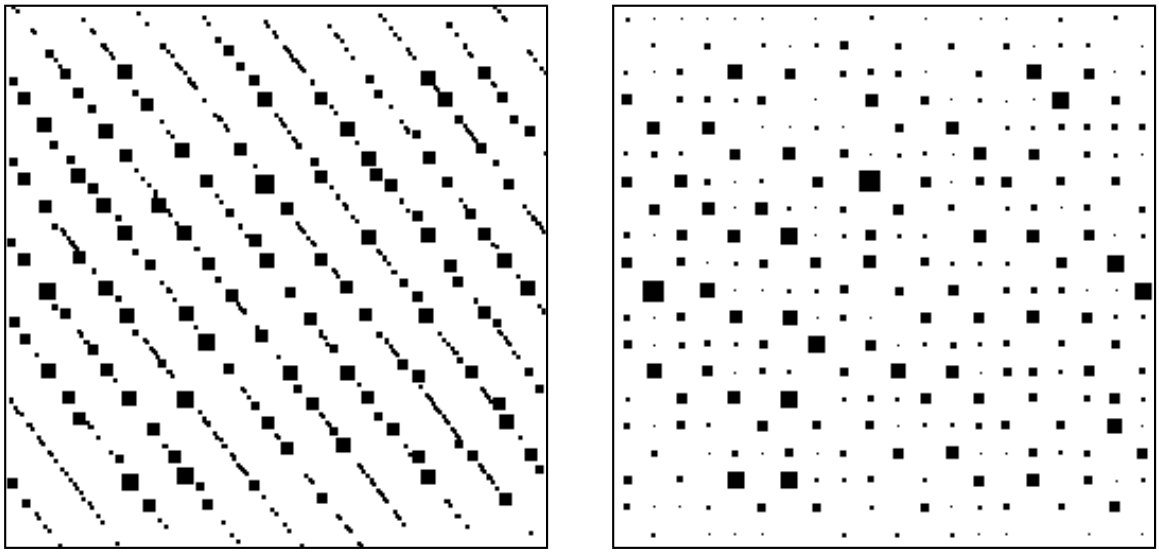
(b)

Figure 9: Same data as Figure 1; scatterplot of size 480×485 pixels.



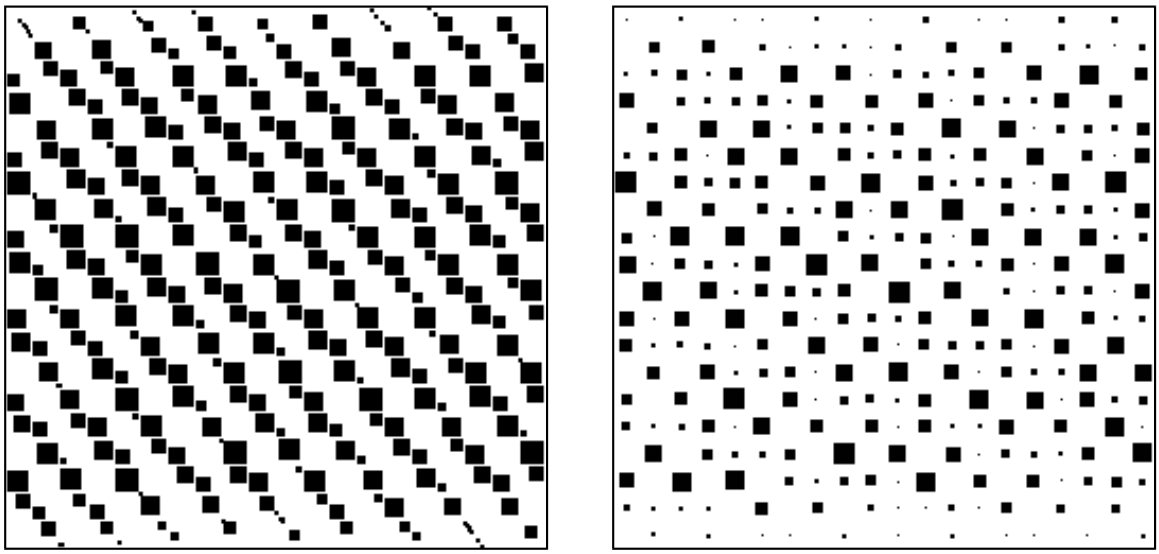
(a) (b)

Figure 10: 400 points generated by RANDU; Varebi plot (a) and 2d histogram (b).



(a) (b)

Figure 11: 1600 points generated by RANDU; Varebi plot (a) and 2d histogram (b).



(a) (b)

Figure 12: 6400 points generated by RANDU; Varebi plot (a) and 2d histogram (b).

origin. The Varebi plot in Figure 14 shows the enormous peak, but has not much resolution anywhere else. This is hard to avoid because the dynamic range is so large. Figure 15 shows a Varebi plot with the background pixels removed. This plot gives a much clearer picture of the rest of the data.

4 Discussion

There are a number of existing techniques attempting to deal with the problem of overstriking in scatterplots. They tend to fall into one of two categories: those that display individual data points, and those that display a density estimate.

Examples for techniques in the first category are *jittering* (Chambers et al, 1983) and use of unfilled circles as plotting symbols (Cleveland, 1985).

Jittering was originally proposed to alleviate overstriking in plots of a euclidean variable against a categorical variable. In scatterplots, points obscured by overplotting could be offset by a small random displacement. (Of course, it would be necessary to remind viewers

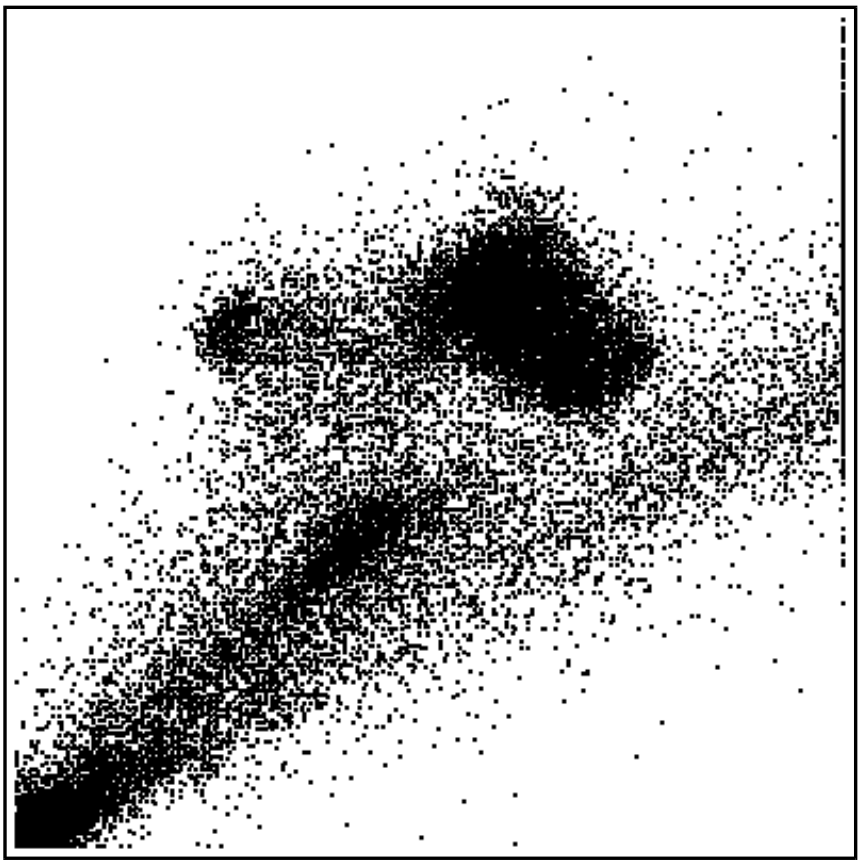


Figure 13: Scatterplot of 65,536 points from a 256×256 MRI image.

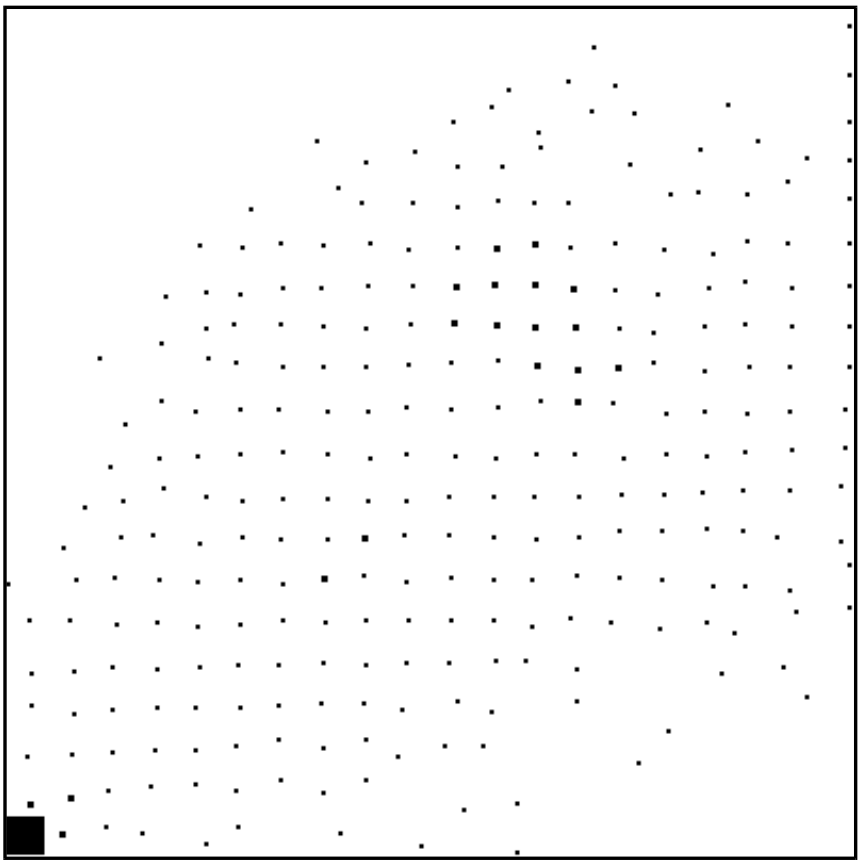


Figure 14: Varebi plot of 65,536 points from a 256×256 MRI image

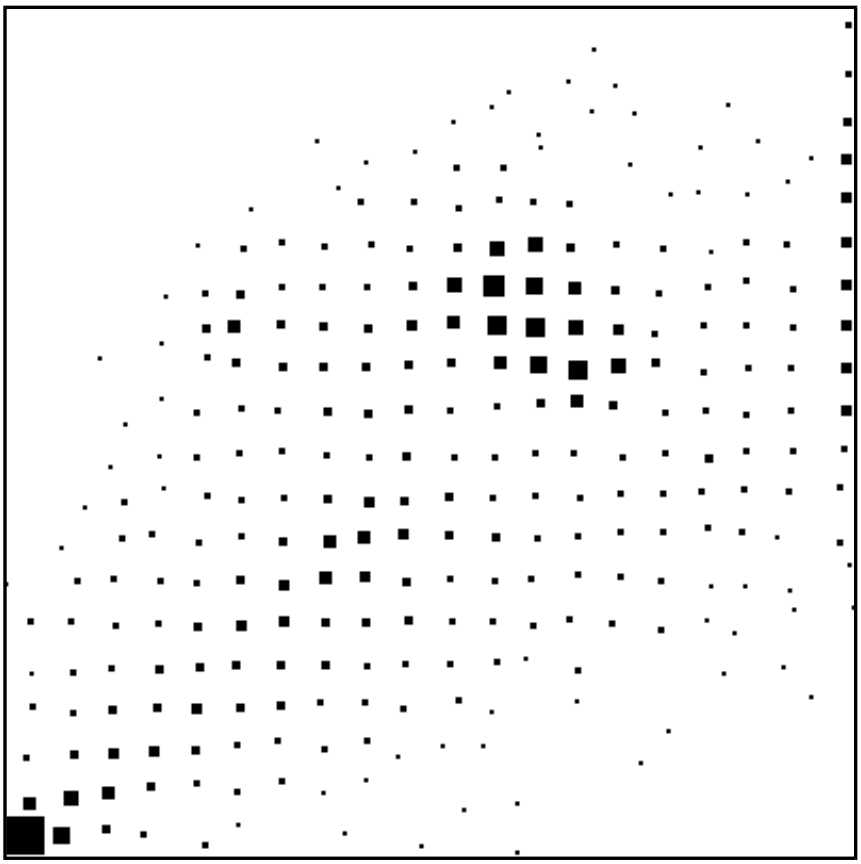


Figure 15: Varebi plot of 27,081 non-background points from a 256×256 MRI image

that jittering was employed in producing the plot).

Using unfilled circles as plotting symbols helps as long as there are not too many exact ties. The intersection of unfilled circles gives a geometric shape distinctly different from a circle. Intersections of axis-parallel rectangles, on the other hand, again are rectangles, which makes it hard to differentiate observations.

Those techniques clearly break down if the density of points in the drawing area gets too large.

Techniques displaying a density estimate typically use a two-dimensional histogram with rectangular or hexagonal bins. The histogram can be displayed as a three-dimensional perspective plot or a contour plot. Alternatively, bin counts can be encoded into size or grey level of glyphs representing the bins. Cleveland and McGill (1984) propose representing bins by sunflowers. The number of petals in a sunflower encodes the bin count. If the bin count is 1, the sunflower degenerates to a point. Sunflowers can also be used with unbinned data when exact overplotting occurs.

These techniques result in an unnecessary loss of detail in regions of the drawing area with low point density, which could be the entire area if it is large, or if the sample size is small.

Carr et al (1987) propose a technique using both individual and agglomerative glyphs. They compute a two-dimensional histogram density estimate with hexagonal bins, encode bin count into the size of hexagons drawn into bins with four or more observations, and draw individual glyphs in bins with three or fewer observations. This technique falls in between the two categories.

The Varebi plots proposed and illustrated in this paper mix agglomerative and individual glyphs, display of a density estimate and display of individual observations. The novel idea is to determine the display format by analyzing the actual amount of overstriking on the screen. Thus, the display format will depend on the sample size, the distribution of the

observations, the size and shape of the individual glyphs, and the size of the window. It may automatically change when the window is resized. Varebi plots do not suffer from the shortcomings of techniques using a fixed display format, and they can be used on binary displays. They reveal detail wherever possible, and show the overall trend when displaying detail is not feasible.

References

[1] D. B. Carr, R. J. Littlefield, W. L. Nicholson, and J. S. Littlefield. Scatterplot matrix techniques for large n. *JASA*, 82:424–436, 1987.

[2] John M. Chambers, William S. Cleveland, B. Kleiner, and P. A. Tukey. *Graphical Methods for Data Analysis*. Wadsworth, Monterey, California, 1983.

[3] William S. Cleveland. *The Elements of Graphing Data*. Wadsworth Advanced Books and Software, Monterey, California, 1985.

[4] William S. Cleveland and Robert McGill. The many faces of a scatterplot. *JASA*, 79:807–822, 1984.

[5] James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes. *Computer Graphics: Principles and Practice*. Addison-Wesley Publishing Company, Inc, 1990.

[6] Donald E. Knuth. *Seminumerical Algorithms*, volume 2 of *The Art of Computer Programming*. Addison-Wesley Publishing Company, second edition, 1981.

[7] Luke Tierney. *Lisp-Stat*. John Wiley and Sons, 1990.