LOCI: Fast Outlier Detection Using the Local Correlation Integral

Spiros Papadimitriou Hiroyuki Kitagawa[1]
Phillip B. Gibbons Christos Faloutsos[2]

November 2002
CMU-CS-02-188

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

[1]Research conducted while visiting CMU under the support of MEXT, Japan.

[2]This material is based upon work supported by the National Science Foundation under Grants No. IIS-9817496, IIS-9988876, IIS-0083148, IIS-0113089, IIS-0209107 IIS-0205224 and by the Defense Advanced Research Projects Agency under Contract No. N66001-00-1-8936. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation, DARPA, or other funding parties.

Keywords: outlier, correlation integral, box counting

Abstract

Outlier detection is an integral part of data mining and has attracted much attention recently [BKNS00, JTH01, KNT00]. In this paper, we propose a new method for evaluating outlier-ness, which we call the *Local Correlation Integral* (*LOCI*). As with the best previous methods, LOCI is highly effective for detecting outliers and groups of outliers (*a.k.a.* micro-clusters). In addition, it offers the following advantages and novelties: *(a)* It provides an automatic, data-dictated cut-off to determine whether a point is an outlier—in contrast, previous methods force users to pick cut-offs, without any hints as to what cut-off value is best for a given dataset. *(b)* It can provide a LOCI plot for each point; this plot summarizes a wealth of information about the data in the vicinity of the point, determining clusters, micro-clusters, their diameters and their inter-cluster distances. None of the existing outlier-detection methods can match this feature, because they output only a single number for each point: its outlier-ness score. *(c)* Our LOCI method can be computed as quickly as the best previous methods. *(d)* Moreover, LOCI leads to a practically linear approximate method, *aLOCI* (for *approximate LOCI*), which provides fast highly-accurate outlier detection. To the best of our knowledge, this is the first work to use approximate computations to speed up outlier detection.

Experiments on synthetic and real world data sets show that LOCI and aLOCI can automatically detect outliers and micro-clusters, without user-required cut-offs, and that they quickly spot both expected and unexpected outliers.

1 Introduction

Due to advances in information technology, larger and larger amounts of data are collected in databases. To make the most out of this data, efficient and effective analysis methods are needed that can extract non-trivial, valid, and useful information. Considerable research has been done toward improving knowledge discovery in databases (KDD) in order to meet these demands.

KDD covers a variety of techniques to extract knowledge from large data sets. In several problem domains (e.g., surveillance and auditing, stock market analysis, health monitoring systems, to mention a few), the problem of detecting rare events, deviant objects, and exceptions is very important. Methods for finding such outliers in large data sets are drawing increasing attention [AY01, AAR96, BL94, BKNS00, JKM99, JKN98, KN97, KN98, KN99, KNT00]. The salient approaches to outlier detection can be classified as either *distribution-based* [BL94], *depth-based* [JKN98], *clustering* [JMF99], *distance-based* [KN97, KN98, KN99, KNT00], or *density-based* [BKNS00] (see Section 2).

In this paper we propose a new method (LOCI—LOcal Correlation Integral method) for finding outliers in large, multidimensional data sets. The main contributions of our work can be summarized as follows:

- We introduce the *multi-granularity deviation factor* (MDEF), which can cope with local density variations in the feature space and detect both isolated outliers as well as outlying clusters. Our definition is simpler and more intuitive than previous attempts to capture similar concepts [BKNS00]. This is important, because the users who interpret the findings of an outlier detection tool and make decisions based on them are likely to be domain experts, not KDD experts.
- We propose a novel (statistically intuitive) method that selects a point as an outlier if its MDEF value deviates significantly (more than three standard deviations) from the local averages. We also show how to quickly estimate the average and standard deviation of MDEF values in a neighborhood. Our method is particularly appealing, because it provides an automatic, data-dictated cut-off for determining outliers, by taking into account the distribution of distances between pairs of objects.
- We present several outlier detection schemes and algorithms using MDEF. Our LOCI algorithm, using an exact computation of MDEF values, is at least as fast as the best previous methods.
- We show how MDEF lends itself to a much faster, approximate algorithm (aLOCI) that still yields high-quality results. In particular, because the MDEF is associated with the *correlation integral* [BF95, TTPF01], it is an aggregate measure. We show how approximation methods such as *box counting* can be used to reduce the computational cost to only $O(kN)$, i.e., linear both with respect to the data set size N and the number of dimensions k. Previous methods are considerably slower, because for each point, they must iterate over every member of a local neighborhood or cluster; aLOCI does not.
- We extend the usual notion of an "outlier-ness" score to a more informative *LOCI plot*. Our method computes a LOCI plot for each point; this plot summarizes a wealth of information about the points in its vicinity, determining clusters, micro-clusters, their diameters and their inter-cluster distances. Such plots can be displayed to the user, as desired. For example, returning the LOCI plots for the set of detected outliers enables users to drill down on outlier points for further understanding. None of the existing outlier-detection methods can match this feature, because they restrict themselves to a single number as an outlier-ness score.

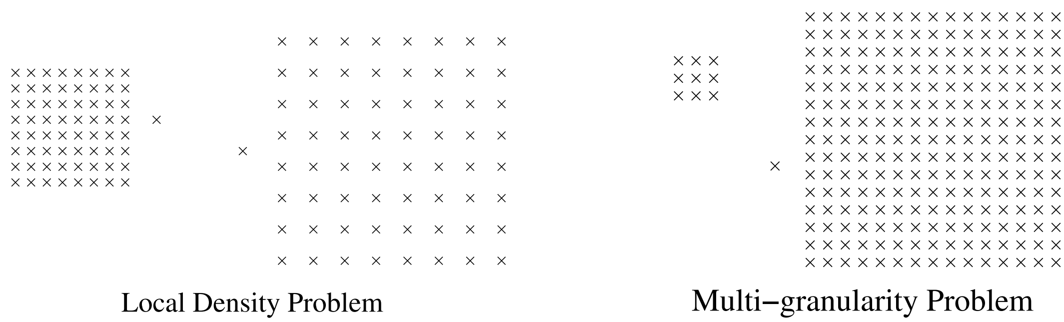


Figure 1: (a) Local density problem, and (b) multi-granularity problem

- We present extensive experimental results using both real world and synthetic data sets to verify the effectiveness of the LOCI method. We show that, in practice, the algorithm scales linearly with data size and with dimensionality. We demonstrate the time-quality trade-off by comparing results from the exact and approximate algorithms. The approximate algorithm can, in most cases, detect all outstanding outliers very efficiently.

To the best of our knowledge, this is the first work to use approximate computations to speed up outlier detection. Using fast approximate calculations of the aggregates computed by an outlier detection algorithm (such as the number of neighbors within a given distance) makes a lot of sense for large databases. Considerable effort has been invested toward finding good measures of distance. However, very often it is quite difficult, if not impossible, to precisely quantify the notion of "closeness". Furthermore, as the data dimensionality increases, it becomes more difficult to come up with such measures. Thus, there is already an inherent fuzziness in the concept of an outlier and any outlier score is more of an informative indicator than a precise measure.

This paper is organized as follows. In Section 2 we give a brief overview of related work on outlier detection. Section 3 introduces the LOCI method and describes some basic observations and properties. Section 4 describes our LOCI algorithm, while Section 5 describes our aLOCI algorithm. Section 6 presents our experimental results, and we conclude in Section 7.

2 Related work

The existing approaches to outlier detection can be classified into the following five categories.

Distribution-based approach. Methods in this category are typically found in statistics textbooks. They deploy some standard distribution model (e.g., Normal) and flag as outliers those objects which deviate from the model [BL94, Haw80, RL87]. However, most distribution models typically apply *directly* to the feature space and are univariate (i.e., have very few degrees of freedom). Thus, they are unsuitable even for moderately high-dimensional data sets. Furthermore, for arbitrary data sets without any prior knowledge of the distribution of points, we have to perform expensive tests to determine which model fits the data best, if any!

Depth-based approach. This is based on computational geometry and computes different layers of k-d convex hulls [JKN98]. Objects in the outer layer are detected as outliers. However, it is well-known that these algorithms suffer from the dimensionality curse and cannot cope with large k.

Clustering approach. Many clustering algorithms detect outliers as by-products [JMF99]. However, since the main objective is clustering, they are not optimized for outlier detection. Furthermore, in most cases, the outlier detection criteria are implicit and cannot easily be inferred from the clustering procedures. An intriguing clustering algorithm using the fractal dimension has been suggested by [BC00]; however it has not been demonstrated on real datasets.

The above three approaches for outlier detection are not appropriate for high-dimensional, large, arbitrary data sets. However, this is often the case with KDD in large databases. The following two approaches have been proposed and are attracting more attention.

Distance-based approach. This was originally proposed by E.M. Knorr and R.T. Ng [KN97, KN98, KN99, KNT00]. An object in a data set P is a *distance-based outlier* if at least a fraction β of the objects in P are further than r from it. This outlier definition is based on a single, global criterion determined by the parameters r and β. This can lead to problems when the data set has both dense and sparse regions [BKNS00] (see Figure 1(a); either the left outlier is missed or every object in the sparse cluster is also flagged as an outlier).

Density-based approach. This was proposed by M. Breunig, et al. [BKNS00]. It relies on the *local outlier factor* (*LOF*) of each object, which depends on the local density of its neighborhood. The neighborhood is defined by the distance to the $MinPts$-th nearest neighbor. In typical use, objects with a high LOF are flagged as outliers. W. Jin, et al. [JTH01] proposed an algorithm to efficiently discover top-n outliers using clusters, for a particular value of $MinPts$.

LOF does not suffer from the local density problem. However, selecting $MinPts$ is non-trivial. In order to detect outlying clusters, $MinPts$ has to be as large as the size of these clusters (see Figure 1(b); if we use a "shortsighted" definition of a neighborhood—i.e., too few neighbors—then we may miss small outlying clusters), and computation cost is directly related to $MinPts$. Furthermore, the method exhibits some unexpected sensitivity on the choice of $MinPts$. For example, suppose we have only two clusters, one with 20 objects and the other with 21 objects. For $MinPts = 20$, *all* objects in the smaller cluster have large LOF values, and this affects LOF values over any range that includes $MinPts = 20$.

In contrast, LOCI automatically flags outliers, based on probabilistic reasoning. Also, MDEF is not so sensitive to the choice of parameters, as in the above 20-21 clusters example. Finally, LOCI is well-suited for fast, one pass, $O(kN)$ approximate calculation. Although some algorithms exist for approximate nearest neighbor search [AMN$^+$98, Ber93, GIM99], it seems unlikely that these can be used to achieve $O(kN)$ time with LOF. Our method uses an aggregate measure (the proposed local correlation integral) that relies strictly on counts. Because it can be estimated (with box-counting) *without* iterating over every point in a set, it can easily cope with multiple granularities, without an impact on speed.

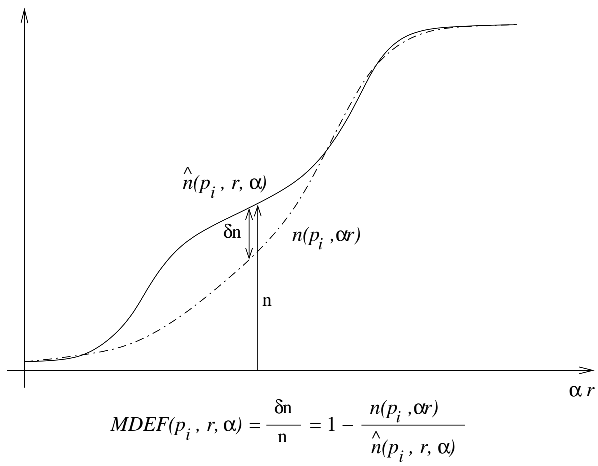


Figure 2: Estimation of MDEF from the local correlation integral and neighbor count functions. The dashed curve is the number of αr-neighbors of p_i and the solid curve is the average number of αr-neighbors over the r-neighborhood (i.e., sampling neighborhood) of p_i.

3 Proposed method

One can argue that, intuitively, an object is an "outlier" if it is in some way "significantly different" from its "neighbors." Two basic questions that arise naturally are:

- What constitutes a "neighborhood?"
- How do we determine "difference" and whether it is "significant?"

Inevitably, we have to make certain choices. Ideally, these should lead to a definition that satisfies the following, partially conflicting criteria:

- It is intuitive and easy to understand: Those who interpret the results are experts in their domain and not on outlier detection.
- It is widely applicable and provides reasonable flexibility: Not everyone has the same idea of what constitutes an outlier and not all data sets conform to the same, specific rules (if any).
- It should lend itself to fast computation: This is obviously important with today's ever-growing collections of data.

3.1 Multi-granularity deviation factor (MDEF)

In this section, we introduce the multi-granularity deviation factor (MDEF), which satisfies the properties listed above. Let the r-neighborhood of an object p_i be the set of objects within distance r of p_i.

Intuitively, the MDEF at radius r for a point p_i is the relative deviation of its local neighborhood density from the average local neighborhood density in its r-neighborhood. Thus, an object whose neighborhood density matches the average local neighborhood density will have an MDEF of 0. In contrast, outliers will have MDEFs far from 0.

Symbol	Definition
$\mathbb{P}$ p_i	Set of objects $\mathbb{P} = \{p_1, \ldots, p_i, \ldots, p_N\}$.
N	Data set size ($\lvert\mathbb{P}\rvert \equiv N$).
k	Dimension of data set, i.e., when $\mathbb{P}$ is a vector space, $p_i = (p_i^1, p_i^2, \ldots, p_i^k)$.
$d(p_i, p_j)$	Distance between p_i and p_j.
$R_\mathbb{P}$	Point set radius, i.e., $R_\mathbb{P} \equiv \max_{p_i, p_j \in \mathbb{P}} d(p_i, p_j)$.
$NN(p_i, m)$	The m-th nearest neighbor of object p_i ($NN(p_i, 0) \equiv p_i$).
$\mathcal{N}(p_i, r)$	The set of r-neighbors of p_i, i.e., $$\mathcal{N}(p_i, r) \equiv \{p \in \mathbb{P} \mid d(p, p_i) \leq r\}$$ Note that the neighborhood contain p_i itself, thus the counts can never be zero.
$n(p_i, r)$	The number of r-neighbors of p_i, i.e., $n(p_i, r) \equiv \lvert\mathcal{N}(p_i, r)\rvert$.
$\hat{n}(p_i, r, \alpha)$	Average of $n(p, \alpha r)$ over the set of r-neighbors of p_i, i.e., $$\hat{n}(p_i, r, \alpha) \equiv \frac{\sum_{p \in \mathcal{N}(p_i, r)} n(p, \alpha r)}{n(p_i, r)}$$
$\sigma_{\hat{n}}(p_i, r, \alpha)$	Standard deviation of $n(p, \alpha r)$ over the set of r-neighbors, i.e., $$\sigma_{\hat{n}}(p_i, r, \alpha) \equiv \sqrt{\frac{\sum_{p \in \mathcal{N}(p_i, r)} (n(p, \alpha r) - \hat{n}(p_i, r, \alpha))^2}{n(p_i, r)}}$$ When clear from the context ($\hat{n}$), we use just $\sigma_{\hat{n}}$.
$MDEF(p_i, r, \alpha)$	Multi-granularity deviation factor for point p_i at radius (or scale) r.
$\sigma_{MDEF}(p_i, r, \alpha)$	Normalized deviation (thus, directly comparable to $MDEF$).
k_σ	Determines what is *significant* deviation, i.e., points are flagged as outliers iff $$MDEF(p_i, r, \alpha) > k_\sigma \sigma_{MDEF}(p_i, r, \alpha)$$ We fix this value to $k_\sigma = 3$ (see Lemma 1).
$\mathcal{C}(p_i, r, \alpha)$	Set of cells on some grid, with cell side $2\alpha r$, each fully contained within $\mathcal{L}_\infty$-distance r from object p_i.
C_i	Cell in some grid.
c_i	The object count within the corresponding cell C_i.
$S_q(p_i, r, \alpha)$	Sum of box counts to the q-th power, i.e., $$S_q(p_i, r, \alpha) \equiv \sum_{C_i \in \mathcal{C}(p_i, r, \alpha)} c_i^q$$

Table 1: Symbols and definitions.

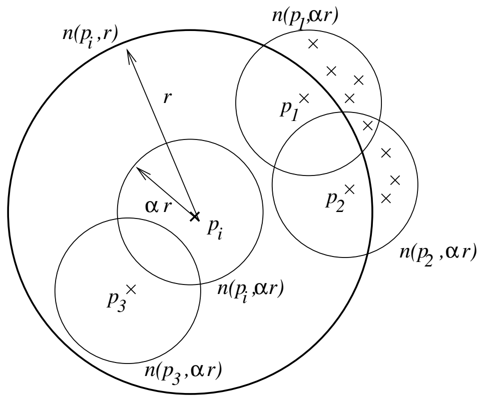


Figure 3: Definitions for n and $\hat{n}$—for instance $n(p_i, r) = 4$, $n(p_r, \alpha r) = 1$, $n(p_1, \alpha r) = 6$ and $\hat{n}(p_i, r, \alpha) = (1 + 6 + 5 + 1)/4 = 3.25$.

To be more precise, we define the following terms (Table 1 describes all symbols and basic definitions). Let $n(p_i, \alpha r)$ be the number of objects in the αr-neighborhood of p_i. Let $\hat{n}(p_i, r, \alpha)$ be the average, over all objects p in the r-neighborhood of p_i, of $n(p, \alpha r)$ (see Figure 3). The use of two radii serves to decouple the neighbor size radius αr from the radius r over which we are averaging. We denote as the *local correlation integral* the function $\hat{n}(p_i, \alpha, r)$ over all r.

Definition 1 (MDEF). *For any p_i, r and α we define the* multi-granularity deviation factor *(MDEF) at radius (or scale) r as:*

$$MDEF(p_i, r, \alpha) = \frac{\hat{n}(p_i, r, \alpha) - n(p_i, \alpha r)}{\hat{n}(p_i, \alpha, r)} \tag{1}$$

$$= 1 - \frac{n(p_i, \alpha r)}{\hat{n}(p_i, \alpha, r)} \tag{2}$$

See Figure 2. Note that the r-neighborhood for an object p_i always contains p_i. This implies that $\hat{n}(p_i, \alpha, r) > 0$ and so the above quantity is always defined.

For faster computation of MDEF, we will sometimes *estimate* both $n(p_i, \alpha r)$ and $\hat{n}(p_i, r, \alpha)$. This leads to the following definitions:

Definition 2 (Counting and sampling neighborhood). *The* counting neighborhood *(or αr-neighborhood) is the neighborhood of radius αr, over which each $n(p, \alpha r)$ is estimated. The* sampling neighborhood *(or r-neighborhood) is the neighborhood of radius r, over which we collect samples of $n(p, \alpha r)$ in order to estimate $\hat{n}(p_i, r, \alpha)$.*

In Figure 3, for example, the large circle bounds the sampling neighborhood for p_i, while the smaller circles bound counting neighborhoods for various p (see also Figure 2).

The main outlier detection scheme we propose relies on the standard deviation of the αr-neighbor count over the sampling neighborhood of p_i. We thus define the following quantity

$$\sigma_{MDEF}(p_i, r, \alpha) = \frac{\sigma_{\hat{n}}(p_i, r, \alpha)}{\hat{n}(p_i, r, \alpha)} \tag{3}$$

which is the normalized standard deviation $\sigma_{\hat{n}}(p_i, r, \alpha)$ of $n(p, \alpha r)$ for $p \in \mathcal{N}(p_i, r)$ (in Section 5 we present a fast, approximate algorithm for estimating σ_{MDEF}).

The main reason we use an *extended* neighborhood ($\alpha < 1$) for sampling is to enable fast, approximate computation of MDEF as explained in Section 5. Besides this, $\alpha < 1$ is desirable in its own right to deal with certain singularities in the object distribution (we do not discuss this due to space considerations).

Advantages of our definitions. Among several alternatives for an outlier score (such as $\max(\hat{n}/n, n/\hat{n})$, to give one example), our choice allows us to use probabilistic arguments for flagging outliers. This is a very important point and is exemplified by Lemma 1 in Section 3.2. The above definitions and concepts make minimal assumptions. The only general requirement is that a distance is defined. Arbitrary distance functions are allowed, which may incorporate domain-specific, expert knowledge, if desired. Furthermore, the standard deviation scheme assumes that pairwise distances *at a sufficiently small scale* are drawn from a single distribution, which is reasonable.

For the fast approximation algorithms, we make the following additional assumptions (the exact algorithms do not depend on these):

- Objects belong to a k-dimensional vector space, i.e., $p_i = (p_i^1, p_i^2, \ldots, p_i^k)$. This assumption holds in most situations. However, if the objects belong to an arbitrary metric space, then it is possible to embed them into a vector space. There are several techniques for this [CNBYM01] which use the $\mathcal{L}_\infty$ norm on the embedding vector space[1].

- We use the $\mathcal{L}_\infty$ norm, which is defined as $\|p_i - p_j\|_\infty \equiv \max_{1 \le m \le k} |p_i^m - p_j^m|$. This is not a restrictive hypothesis, since it is well-known that, in practice, there are no clear advantages of one particular norm over another [FLM77, GIM99].

3.2 LOCI outlier detection

In this section, we describe and justify our main outlier detection scheme. It should be noted that, among all alternatives in the problem space LOCI can be easily adapted to match several choices. It computes the necessary summaries in one pass and the rest is a matter of interpretation.

In particular, given the above definition of MDEF, we still have to make a number of decisions. In particular, we need to answer the following questions:

- **Sampling neighborhood:** Which points constitute the sampling neighborhood of p_i, or, in other words, which points do we average over to compute $\hat{n}$ (and, in turn, MDEF) for a p_i in question?

- **Scale:** Regardless of the choice of neighborhood, over what range of distances do we compare n and $\hat{n}$?

- **Flagging:** After computing the MDEF values (over a certain range of distances), how do we use them to choose the outliers?

[1] Given objects π_i in a metric space $\mathbb{M}$ with distance function $\delta(\pi_i, \pi_j)$, one typical approach is to choose k landmarks $\{\Pi_1, \ldots, \Pi_k\} \subseteq \mathbb{M}$ and map each object π_i to a vector with components $p_i^j = \delta(\pi_i, \Pi_j)$.

LOCI outlier detection method. The proposed LOCI outlier detection method answers the above questions as follows. Advantages and features of LOCI are due to these design choices combined with inherent properties of MDEF.

- **Large sampling neighborhood:** For each point and counting radius, the sampling neighborhood is selected to be large enough to contain enough samples. We choose $\alpha = 1/2$ in all exact computations, and we typically use $\alpha = 1/16$ in aLOCI (introduced in Section 5) for robustness (particularly in the estimation of σ_{MDEF}).

- **Full-scale:** The MDEF values are examined for a wide range of sampling radii. In other word, the maximum sampling radius is $r_{max} \approx \alpha^{-1} R_{\mathbb{P}}$ (which corresponds to maximum counting radius of $R_{\mathbb{P}}$). The minimum sampling radius r_{min} is determined based on the number of objects in the sampling neighborhood. We always use a smallest sampling neighborhood with $\hat{n}_{min} = 20$ neighbors; in practice, this is small enough but not too small to introduce statistical errors in MDEF and σ_{MDEF} values.

- **Standard deviation-based flagging**: A point is flagged as an outlier, if for *any* $r \in [r_{min}, r_{max}]$ its MDEF is *sufficiently* large, i.e.,

  $$MDEF(p_i, r, \alpha) > k_\sigma \sigma_{MDEF}(p_i, r, \alpha)$$

  In all our experiments, we use $k_\sigma = 3$ (see Lemma 1).

The standard deviation-based flagging is one of the main features of the LOCI method. It replaces any "magic cut-offs" with probabilistic reasoning based on σ_{MDEF}. It takes into account *distribution* of *pairwise distances* and compares each object to those in its sampling neighborhood. Note that, even if the global distribution of distances varies significantly (e.g., because it is a mixture of very different distributions), the use of the *local* deviation successfully solves this problem. In fact, in many *real* data sets, the distribution of pairwise distances follows a specific distribution over all or most scales [TTPF01, BF95]. Thus, this approach works well for many real data sets. The user may alter the minimum neighborhood size r_{min} and k_σ if so desired, but in practice this is unnecessary.

Lemma 1 (Deviation probability bounds). *For any distribution of pairwise distances, and for any randomly selected p_i, we have*

$$Pr\left\{MDEF(p_i, r, \alpha) > k_\sigma \sigma_{MDEF}(p_i, r, \alpha)\right\} \leq \frac{1}{k_\sigma^2}$$

Proof. From Chebyshev's inequality it follows that

$$\begin{aligned}
&\Pr\left\{MDEF(p_i, r, \alpha) > k_\sigma \sigma_{MDEF}(p_i, r, \alpha)\right\} \\
&\leq \Pr\left\{|MDEF(p_i, r, \alpha)| > k_\sigma \sigma_{MDEF}(p_i, r, \alpha)\right\} \\
&\leq \sigma^2_{MDEF}(p_i, r, \alpha)/(k_\sigma \sigma_{MDEF}(p_i, r, \alpha))^2 = 1/k_\sigma^2 \quad \blacksquare
\end{aligned}$$

$\square$

This is a relatively loose bound, but it holds regardless of the distribution. For known distributions, the actual bounds are tighter; for instance, if the neighborhood sizes follow a normal distribution and $k_\sigma = 3$, much less than 1% of the points should deviate by that much (as opposed to $\approx 10\%$ suggested by the above bound).

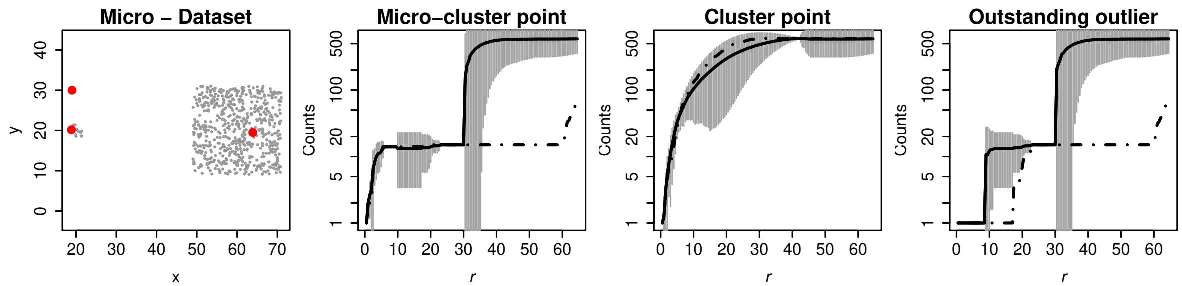


Figure 4: LOCI plots from an actual dataset—see also Section 6.

3.3 Alternative interpretations

As mentioned in Section 3.2, we have a range of design choices for outlier detection schemes. Different answers give rise to different outlier detection schemes and provide the user with alternative views. We should emphasize that, if the user want, LOCI can be adapted to *any* desirable interpretation, without any re-computation. Our fast algorithms estimate all the necessary quantities with a single pass over the data and build the appropriate "summaries," no matter how they are later interpreted.

Sampling neighborhood: Small vs. large. The choice depends on whether we are interested in the deviation of p_i from a small (highly local) or a relatively large neighborhood. Since LOCI employs standard deviation-based flagging, a sampling neighborhood large enough to get a sufficiently large sample is desirable. However, when the distance distribution varies widely (which rarely happens, except at *very* large radii) or if the user chooses non-deviation based scheme (which, although possible, is not recommended) this is an option.

Scale: Single vs. range and distance-based vs. population-based. Regardless of sampling neighborhood, users could choose to examine MDEF and σ_{MDEF} at either a single radius (which is very close to the distance-based approach [KN99]) or a limited range of radii (same for all the points). Alternatively, they may implicitly specify the radius (or radii) by neighborhood size (effectively varying the radius at each p_i, depending on density). Either approach might make sense.

Flagging: Thresholding vs. ranking vs. standard deviation-based. Use of the standard deviation is our main contribution and the recommended approach. However, we can easily match previous methods either by "hard thresholding" (*if* we have prior knowledge about what to expect of distances and densities) or "ranking" (if we want to catch a few "suspects" blindly and, probably, "interrogate" them manually later).

3.4 LOCI plot

In this section we introduce the *LOCI plot*. This is a powerful tool, no matter what outlier detection scheme is employed. It can be constructed instantly from the computed "summaries" for any point p_i the user desires and it gives a wealth of information about the vicinity of p_i: why it is an outlier with regard to its vicinity, as well as information about nearby clusters and micro-clusters, their diameters and inter-cluster distances.

Definition 3 (LOCI plot). *For any object p_i, the plot of $n(p_i, \alpha r)$ and $\hat{n}(p_i, r, \alpha)$ with $\hat{n}(p_i, r, \alpha) \pm 3\sigma_{\hat{n}}(p_i, r, \alpha)$, versus r (for a range of radii of interest), is called its* LOCI plot.

We give detailed examples from actual datasets in Section 6. Here we briefly introduce the main features (see also Figure 4). The solid line shows $\hat{n}$ and the dashed line is n is all plots.

- Consider the point in the micro-cluster (at $x = 18, y = 20$). The n value looks similar up to the distance (roughly 30) we encounter the large cluster. Earlier, the increase in deviation (in the range of $\approx$ 10–20) indicates the presence of a (small) cluster. Half the width (since $\alpha = 1/2$, and the deviation here is affected by the counting radius) of this range (about $10/2 = 5$) is the radius of this cluster.

- A similar increase in deviation happens at radius 30, along with an increase in $\hat{n}$. Also, note that n shows a similar jump at $\alpha^{-1} \times 30 = 60$ (this time it is the sampling radius that matters). Thus, $\approx$ 30 is the distance to the next (larger) cluster.

- In the cluster point (at $x = 64, y = 19$) we see from the middle LOCI plot that the two counts ($\hat{n}$ and $\sigma_{\hat{n}}$) are similar, as expected. The increase in deviation, however, provides the information described above for the first increase (here the counting radius matters again, so we should multiply the distances by α).

- The general magnitude of the deviation always indicates how "fuzzy" (i.e., spread-out and inconsistent) a cluster is.

- For the outstanding outlier point (at $x = 18, y = 30$), we see the deviation increase along with the pair of jumps in $\hat{n}$ and n (the distance between the jumps determined by α) *twice*, as we would expect: the first time when we encounter the micro-cluster and the second time when we encounter the large cluster.

```
// Pre-processing
Foreach p_i ∈ ℙ:
    Perform a range-search
      for N_i = {p ∈ ℙ | d(p_i, p) ≤ r_max}
    From N_i, construct a sorted list D_i
      of the critical and α-critical distances of p_i
// Post-processing
Foreach p_i ∈ ℙ:
    For each radii r ∈ D_i (ascending):
        Update n(p_i, αr) and n̂(p_i, r, α)
        From n and n̂, compute
          MDEF(p_i, r, α) and σ_MDEF(p_i, r, α)
        If MDEF(p_i, r, α) > 3σ_MDEF(p_i, r, α),
          flag p_i
```

Figure 5: The exact LOCI algorithm.

4 The LOCI algorithm

In this section, we describe our algorithm for detecting outliers using our LOCI method. This algorithm computes exact MDEF and σ_{MDEF} values for all objects, and then reports an outlier whenever MDEF is more than three times larger than σ_{MDEF} for the same radius. Thus the key to a fast algorithm is an efficient computation of MDEF and σ_{MDEF} values.

We can considerably reduce the computation time for MDEF and σ_{MDEF} values by exploiting the following properties:

Observation 1. *For each object p_i and each α, $n(p_i, r)$, $\hat{n}(p_i, r, \alpha)$, and thus $MDEF(p_i, r, \alpha)$ and $\sigma_{MDEF}(p_i, r, \alpha)$ are all piecewise constant functions of r. In particular, $n(p_i, r)$ and $n(p, \alpha r)$ for all p in the r-neighborhood of p_i can change only when the increase of r causes a new point to be added to either the r-neighborhood of p_i or the αr-neighborhood of any of the p.*

This leads to the following definition, where N is the number of objects and $NN(p_i, m)$ is the m-th nearest neighbor of p_i.

Definition 4 (Critical Distance). *For $1 \leq m \leq N$, we call $d(NN(p_i, m), p_i)$ a* critical distance *of p_i and $d(NN(p_i, m), p_i)/\alpha$ an α-*critical distance *of p_i.*

By observation 1, we need only consider radii that are critical or α-critical. Figure 5 shows our LOCI algorithm. In a pre-processing pass, we determine the critical and α-critical distances D_i for each object p_i. Then considering each object p_i in turn, and considering increasing radius r from D_i, we maintain $n(p_i, \alpha r)$, $\hat{n}(p_i, r, \alpha)$, $MDEF(p_i, r, \alpha)$, and $\sigma_{MDEF}(p_i, r, \alpha)$. We flag p_i as an outlier if $MDEF(p_i, r, \alpha) > 3\sigma_{MDEF}(p_i, r, \alpha)$ for some r.

The worst-case complexity of this algorithm is $O(N \times (time\_of\_rmax\_range\_search + n_{ub}^2))$, where $n_{ub} = \max\{n(p_i, r_{max}) \mid p_i \in \mathbb{P}\}$. Alternatively, if we specify the range of scales indirectly by numbers of neighbors n_{min} and n_{max} instead of explicit r_{min} and r_{max}, then $r_{min} = d(NN(p_i, n_{min}), p_i)$ and $r_{max} = d(NN(p_i, n_{max}), p_i)$. The complexity of this alternative is $O(N \times (time\_of\_Rmax\_range\_search + n_{max}^2)$, where $R_{max} = \max\{d(NN(p_i, n_{max}), p_i) \mid p_i \in \mathbb{P}\}$. Thus, the complexity of our LOCI algorithm is roughly comparable to that of the best previous density-based approach [BKNS00].

5 The aLOCI algorithm

In this section we present our fast, approximate LOCI algorithm (aLOCI). Although algorithms exist for approximate range queries and nearest neighbor search [AMN+98, Ber93, GIM99], applying them directly to previous outlier detection algorithms (or the LOCI algorithm; see Figure 5) would not eliminate the high cost of iterating over each object in the (sampling) neighborhood of each p_i. Yet with previous approaches, *failing* to iterate over each such object means the approach cannot effectively overcome the multi-granularity problem (Figure 1(b)). In contrast, our MDEF-based approach is well-suited to fast approximations that avoid these costly iterations, yet are able to overcome the multi-granularity problem. This is because our approach essentially requires only counts at various scales.

5.1 Definitions and observations

Our aLOCI algorithm is based on a series of observations and techniques outlined in this section.

To quickly estimate the average number of αr-neighbors over all points in an r-neighborhood of an object $p_i \in \mathbb{P}$ (from now on, we assume $\mathcal{L}_\infty$ distances), we can use the following approach. Consider a grid of cells with side $2\alpha r$ over the set $\mathbb{P}$. Perform a *box count* of the grid: For each cell C_j in the grid, compute the count, c_j, of the number of objects in the cell. Each object in C_j has c_j neighbors in the cell (counting itself), so the total number of neighbors over all objects in C_j is c_j^2. Denote by $\mathcal{C}(p_i, r, \alpha)$ the set of all cells in the grid such that the entire cell is within distance r of p_i. We use $\mathcal{C}(p_i, r, \alpha)$ as an approximation for the r-neighborhood of p_i. Summing over the entire r-neighborhood, we get $S_2(p_i, r, \alpha)$, where $S_q(p_i, r, \alpha) \equiv \sum_{C_j \in \mathcal{C}(p_i, r, \alpha)} c_j^q$. The total number of objects is simply the sum of all box counts, i.e., $S_1(p_i, r, \alpha)$.

Lemma 2 (Approximate average neighbor count). *Let $\alpha = 2^{-l}$ for some positive integer l. The average neighbor count over p_i's sampling neighborhood is approximately:*

$$\hat{n}(p_i, r, \alpha) = \frac{S_2(p_i, r, \alpha)}{S_1(p_i, r, \alpha)}$$

Proof. Follows from the above observations; for details, see [Sch88]. $\square$

However, we need to obtain information at several scales. We can efficiently store cell counts in a k-dimensional quad-tree: The first grid consists of a single cell, namely the bounding box of $\mathbb{P}$. We then recursively subdivide each cell of side $2\alpha r$ into 2^k subcells, each with radius αr, until we reach the scale we desire (specified either in terms of its side length or cell count). We keep only pointers to the non-empty child subcells in a hash table (typically, for large dimensions k, most of the 2^k children are empty, so this saves considerable space over using an array). For our purposes, we only need to store the c_j values (one number per non-empty cell), and not the objects themselves.

The recursive subdivision of cells dictates the choice[2] of $\alpha = 2^{-l}$ for some positive integer l, since we essentially discretize the range of radii at powers of two.

In addition to approximating $\hat{n}$, our method requires an estimation of $\sigma_{\hat{n}}$. The key to our fast approximation of $\sigma_{\hat{n}}$ is captured in the following lemma:

Lemma 3 (Approximate std. deviation of neighbor count). *Let $\alpha = 2^{-l}$ for some positive integer l. The standard deviation of the neighbor count is approximately:*

$$\sigma_{\hat{n}}(p_i, r, \alpha) = \sqrt{\frac{S_3(p_i, r, \alpha)}{S_1(p_i, r, \alpha)} - \frac{S_2^2(p_i, r, \alpha)}{S_1^2(p_i, r, \alpha)}}$$

Proof. Following the same reasoning as in Lemma 2, the deviation for each object within each cell C_j is $c_j - \hat{n}(p_i, r, \alpha) \approx c_j - S_2(p_i, r, \alpha)/S_1(p_i, r, \alpha)$. Thus, the sum of squared differences for all objects within the cell is $c_j \left(c_j - S_2(p_i, r, \alpha)/S_1(p_i, r, \alpha)\right)^2$. Summing over all cells and dividing by the count of objects $S_1(p_i, r, \alpha)$ gives $\frac{1}{S_1} \sum_j \left(c_j^3 - \frac{2c_j^2 S_2}{S_1} + \frac{c_j S_2^2}{S_1^2} \right) = \frac{S_3}{S_1} - \frac{2S_2^2}{S_1^2} + \frac{S_2^2}{S_1^2}$, which leads to the above result. $\square$

[2] In principle, we can choose any integer power $\alpha = c^{-l}$ by subdividing each cell into c^k subcells. However, this makes no difference in practice.

// Initialization
Select set of shifts $S = \{s_0, s_1, \ldots, s_g\}$, where $s_0 = 0$
$l_\alpha = -\lg(\alpha)$
Foreach $s_i \in S$:
  Initialize quadtree $Q(s_i)$
// Pre-processing stage
Foreach $p_i \in \mathbb{P}$:
  Foreach $s_i \in S$:
    Insert p_i in $Q(s_i)$
// Post-processing stage
Foreach $p_i \in \mathbb{P}$:
  Foreach level l:
    Select cell C_i in $Q(s_a)$ with side
      $d_i = R_{\mathbb{P}}/2^l$ and center closest to p_i
    Select cell C_j in $Q(s_b)$ with side
      $d_j = R_{\mathbb{P}}/2^{l-l_\alpha}$ and center closest to center of C_i
    Estimate $MDEF(p_i, \frac{d_j}{2}, \alpha)$ and $\sigma_{MDEF}(p_i, \frac{d_j}{2}, \alpha)$
    If $MDEF(p_i, \frac{d_j}{2}, \alpha) > 3\sigma_{MDEF}(p_i, \frac{d_j}{2}, \alpha)$, flag p_i

Figure 6: The approximate aLOCI algorithm.

From the above discussion, we see that box counting within quad trees can be used to quickly estimate the MDEF values and σ_{MDEF} values needed for our LOCI approach. However, in practice, there are several important issues that need to be resolved to achieve accurate results, which we address next.

Discretization. A quad-tree decomposition of the feature space inherently implies that we can sample the actual averages and deviations at radii that are proportional to powers of two (or, in general, c^l multiples of r_{min}, for some integers c and l). In essence, we discretize all quantities involved by sampling them at intervals of size 2^l. However, perhaps surprisingly, this discretization does not have a significant impact on our ability to detect outliers. Consider a relatively isolated object p_i and a distant cloud of objects. Recall that we compute MDEF values for an object starting with the smallest radius for which its sampling neighborhood has $n_{min} = 20$ objects, in order to make the (exact) LOCI algorithm more robust and self-adapting to the local density. Similarly, for the aLOCI algorithm, we start with the smallest discretized radius for which its sampling neighborhood has at least 20 neighbors. Considering our point p_i, observe that at large enough radius, both its sampling and counting neighborhoods will contain many objects from the cloud, and these points will have similar neighborhood counts to p_i, resulting in an MDEF near zero (i.e., no outlier detection). However, at some previous scale, the sampling neighborhood will contain part of the cloud but the counting neighborhood will not, resulting in an MDEF near one, as desired for outlier detection. Note that, in order for this to work, it is crucial that (a) we use an $\alpha \leq 2^{-l}$, and (b) we perform n_{min} neighborhood thresholding based on the sampling neighborhood and not the counting neighborhood.

Locality. Ideally, we would like to have the quad-tree grids contain each object of the dataset at the exact center of cells. This is not possible, unless we construct one quad-tree per object, which is ridiculously expensive. However, a single grid may provide a close enough approximation for many objects in the data set. Furthermore, outstanding outliers are typically detected no matter what the grid positioning is: the further an object is from its neighbors, the more "leeway" we have to be off-center (by up to at least half the distance to its closest neighbor!).

In order to further improve accuracy for less obvious outliers, we utilize several grids. In practice, the number of grids g does not depend on the feature space dimension k, but rather on the distribution of objects (or, the *intrinsic* dimensionality [CNBYM01, BF95] of the data set, which is typically much smaller than k). Thus, in practice, we can achieve good results with a small number of grids.

To summarize, the user may select g depending on the desired accuracy vs. speed. Outstanding outliers are typically caught regardless of grid alignment. Performance on less obvious outliers can be significantly improved using a small number $g - 1$ of extra grids.

Next we have to answer two related questions: how should we pick grid alignments and, given the alignments, how should we select the appropriate grid for each point?

Grid alignments. Each grid is constructed by shifting the quad-tree bounding box by s (a k-dimensional vector)[3]. At each grid level l (corresponding to cell diameter $d_l = R_{\mathbb{P}}/2^l$), the shift effectively "wraps around," i.e., each cell is effectively shifted by $s \mod d_l$, where $\mod$ is applied element-wise and should be interpreted loosely (as the fractional part of the division). Therefore, with a few shifts (each portion of significant digits essentially affecting different levels), we can achieve good results throughout all levels. In particular, we recommend using shifts obtained by selecting each coordinate uniformly at random from its domain.

Grid selection. For any object p_i in question, which cells and from which grids do we select to (approximately) cover the counting and sampling neighborhoods? For the counting neighborhood of p_i, we select a cell C_i (at the appropriate level l) that contains p_i as close as possible to its center; this can be done in $O(kg)$ time.

For the sampling neighborhood, a naive choice might be to search all cells in the *same* grid that are adjacent to C_i. However, the number of such cells is $O(2^k)$, which leads to prohibitively high computational cost for high dimensional data. Unfortunately, if we insist on this choice, this cost cannot be avoided; we will either have to pay it when building the quad-tree or when searching it.

Instead, we select a cell C_j of diameter d_l/α (where $d_l = R_{\mathbb{P}}/2^l$) in some grid (possibly a different one), such that the center of C_j lies as close as possible to the center of C_i. The reason we pick C_j based on its distance from the center of C_i and *not* from p_i is that we want the maximum possible volume overlap of C_i and C_j. Put differently, we have already picked an approximation for the counting neighborhood of p_i (however good or bad) and next we want the best approximation of the sampling neighborhood, *given* the choice of C_i. If we used the distance from p_i we might end up with the latter approximation being "incompatible" with the former. Thus, this choice is the one that gives the best results. The final step is to estimate MDEF and σ_{MDEF}, by performing a box-count on the sub-cells of C_j.

Deviation estimation. A final important detail has to do with successfully estimating σ_{MDEF}. In certain situations (typically, in either very small or very large scales), many of the sub-cells of C_j may

[3]Conceptually, this is equivalent to shifting the entire data set by $-s$

Dataset	Description
`Dens`	Two 200-point clusters of different densities and one outstanding outlier.
`Micro`	A micro-cluster with 9 points, a large, 600-point cluster (same density) and one outstanding outlier.
`Sclust`	A Gaussian cluster with 500 points.
`Multimix`	A 250-point Gaussian cluster, two uniform clusters (200 and 400 points), three outstanding outliers and 3 points along a line from the sparse uniform cluster.
`NBA`	Games, points per game, rebounds per game, assists per game (1991–92 season).
`NYWomen`	Marathon runner data, 2229 women from the NYC marathon: average pace (in minutes per mile) for each stretch (6.2, 6.9, 6.9 and 6.2 miles)

Table 2: Description of synthetic and real data sets.

be empty. If we do a straight box-count on these, we may under-estimate the deviation and erroneously flag objects as outliers.

This problem is essentially solved by giving more weight to the counting neighborhood of p_i: in the set of box counts used for $S_q(p_i, r, \alpha)$, we also include c_i w times ($w = 2$ works well in all the datasets we have tried), besides the counts for the sub-cells of C_j.

Lemma 4 (Deviation smoothing). *If we add a new value a to set of N values with average m and variance s^2, then the following hold about the new average μ and variance σ^2:*

$$\sigma^2 > s^2 \Leftrightarrow \frac{|a-m|}{s} > \frac{N+w}{N} \quad \textit{and} \quad \lim_{N\to\infty} \frac{\sigma^2}{s^2} = 1$$

where w is the weight of a (i.e., it is counted w times).

Proof. From the definitions for mean and standard deviation, we have

$$\mu = \frac{w}{N+w}a + \frac{N}{N+w}m, \quad \sigma^2 = \frac{w}{N+w}(a-\mu)^2 + \frac{N}{N+w}s^2$$
$$\text{and} \quad (a-\mu)^2 = \left(\frac{N}{N+w}\right)^2 (a-m)^2$$

Therefore $\frac{\sigma^2}{s^2} = \frac{N^2}{(N+w)^3}\left(\frac{a-m}{s}\right)^2 + \frac{N}{N+w}$. The results follow from this relation. □

From Lemma 4, if the number of non-empty sub-cells is large, a small w weighting has small effect. For outstanding outliers (i.e., large $|a-m|/s$), this weighting does not affect the the estimate of σ_{MDEF} significantly. Thus, we may only err on the conservative side for a few outliers, while avoiding several "false alarms" due to underestimation of σ_{MDEF}.

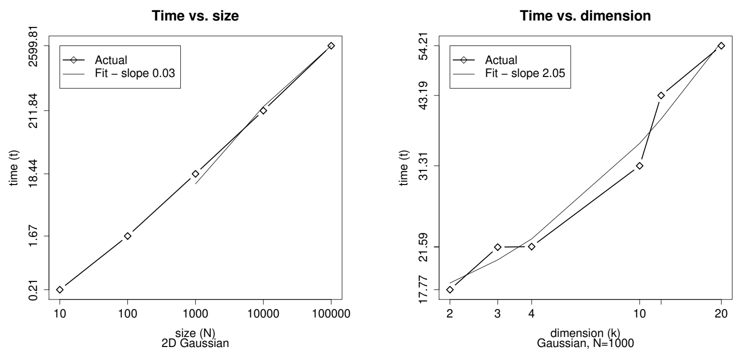


Figure 7: Time versus data set size and dimension (log-log scales).

5.2 The approximation algorithm

The aLOCI algorithm, based on the discussion in the previous section, is illustrated in Figure 6. The quad-tree construction stage takes time $O(NLkg)$, where L is the total number of levels (or scales), i.e., $O(\lg(r_{max}/r_{min}))$. The scoring and flagging stage takes an additional $O(NL(kg+2^k)$ time (recall that α is a constant). As noted above, the number of grids g depends on the intrinsic dimensionality of $\mathbb{P}$. We found $10 \leq g \leq 30$ sufficient in all our experiments. Similarly, L can be viewed as fixed for most data sets. Finally, the 2^k term is a pessimistic bound because of the sparseness in the box counts. As shown in Section 6, in practice the algorithm scales linearly with data size and with dimensionality. Moreover, even in the worst case, it is asymptotically significantly faster than the best previous density-based approach.

6 Experimental evaluation

In this section we discuss results from applying our method to both synthetic and real datasets (described in Table 2). We also briefly discuss actual performance measurements (wall-clock times).

6.1 Complexity and performance

Our prototype system is implemented in Python, with Numerical Python for fast matrix manipulation and certain critical components (quad-trees and distance matrix computation) implemented in C as

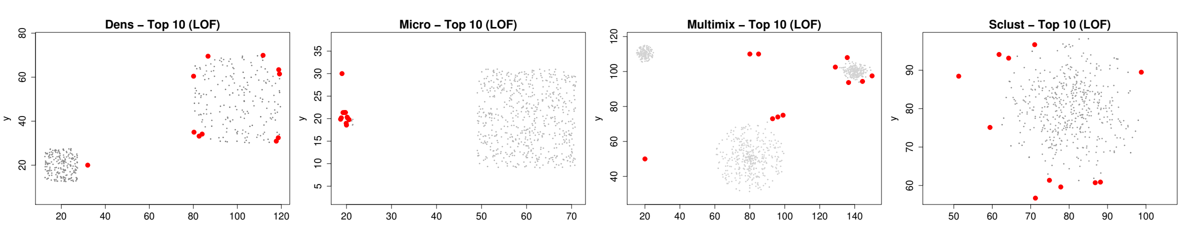


Figure 8: Synthetic data: LOF ($MinPts = 10$ to 30, top 10).

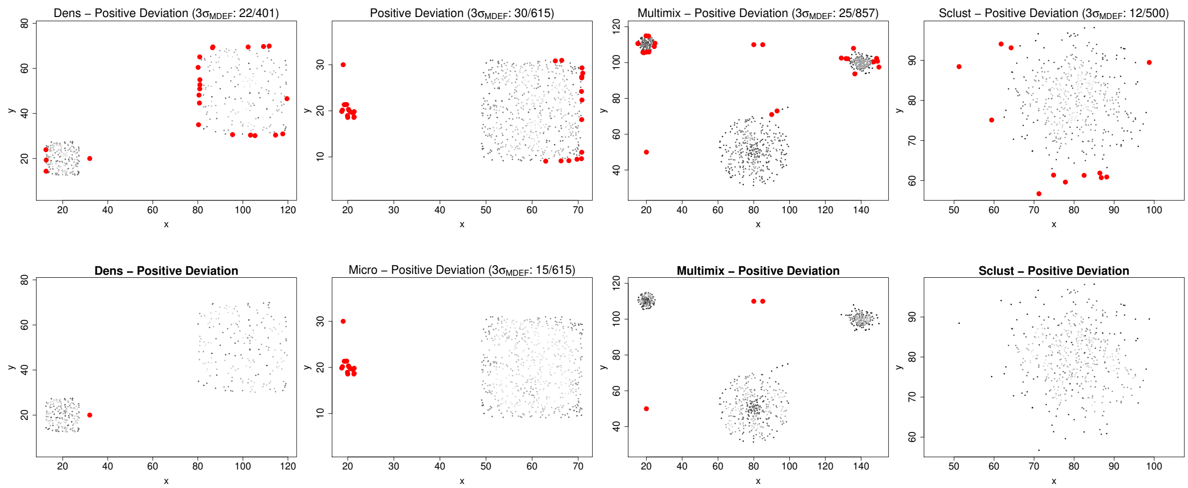


Figure 9: Synthetic, LOCI. Top row: $\hat{n} = 20$ to full radius, $\alpha = 0.5$. Bottom row: $\hat{n} = 20$ to 40 except `micro` where $\hat{n} = 200$ to 230, $\alpha = 0.5$.

language extensions (achieving a $5\times$ to $15\times$ speedup). We are currently re-implementing the system in C and preliminary results show at least a $10\times$ overall speedup. Figure 7 shows the wall clock times on a synthetic dataset, versus data set size and dimension. All experiments were run on a PII 350MHz with 384Mb RAM. The graphs clearly show that aLOCI scales linearly with dataset size as well as dimension, as expected. In should be noted that the dataset chosen (a multi-dimensional Gaussian cluster) is actually much denser throughout than a real dataset would be. Thus, the time vs. dimension results are on the conservative side ($l_\alpha = 4$, or $\alpha = 1/16$ in our experiments).

6.2 Synthetic data

We illustrate the intuition behind LOCI using a variety of synthetic datasets, demonstrate that LOCI and aLOCI provide sound and useful results and we discuss how to interpret LOCI plots "in action." The results from LOF are shown in Figure 8. LOF is the current state of the art in outlier detection. However, it provides no hints about how high an outlier score is high enough. A typical use of selecting a range of interest and examining the top-N scores will either erroneously flag some points (N too large) or fail to capture others (N too small). LOCI provides an automatic way of determining outliers within the range of interest and captures outliers correctly.

Figure 9 shows the results from LOCI on the entire range of scales, from 20 to $R_{\mathbb{P}}$ on the top row. On the bottom row, we show the outliers at a subset of that range (20 to 40 neighbors around each point). The latter is much faster to compute, even exactly, and still detects the most significant outliers. Finally, Figure 10 shows the aLOCI results. However, LOCI does not stop there and can provide information about *why* each point is an outlier and about its vicinity (see Figure 12 and Figure 11).

`Dens` dataset. LOCI captures the outstanding outlier. By examining the LOCI plots we can get much more information. In the leftmost column of Figure 11 it is clear that the outstanding outlier is indeed significantly different from its neighbors. Furthermore, the radius where the deviation first increases (≈ 5) and the associated jumps in counts correspond to the distance ($\approx 5/2$) to the first cluster. The deviation increase (without change in counts) in the range of 50–80 corresponds to the

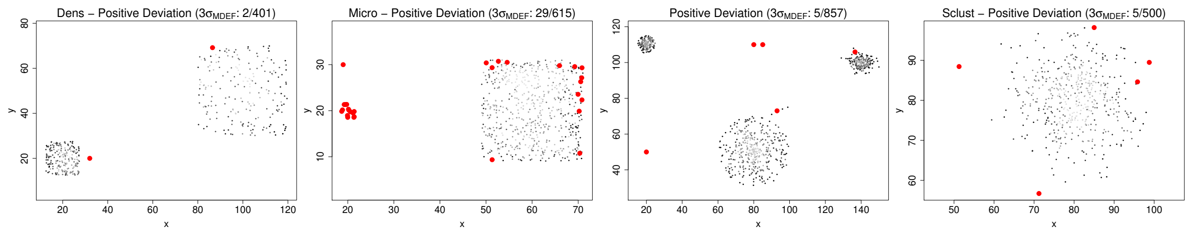


Figure 10: Synthetic: aLOCI (10 grids, 5 levels, $l_\alpha = 4$, except `micro`, where $l_\alpha = 3$).

diameter (≈ 30) of the second cluster.

The second column in Figure 11 shows a point in the micro-cluster, which behaves very similarly to those in its sampling neighborhood. Once again, the deviation increases correspond to the diameters of the two clusters.

Finally, the two rightmost columns of Figure 11 show the LOCI plots for two points in the large cluster, one of them on its fringe. From the rightmost column it is clear that the fringe point is tagged as an outlier at a large radius and by a small margin. Also, the width of the radius range with increased deviation corresponds to the radius of the large cluster.

"Drill-down." It is important to note that the aLOCI plots (bottom row) already provide much of the information contained in the LOCI plots (top row), such as the scale (or radius range) at which each point is an outlier. *If* users desire detailed information about a particular range of radii, they can select a few points flagged by aLOCI and obtain the LOCI plots. Such a "drill-down" operation is common in decision support systems. Thanks to the accuracy of aLOCI, the user can immediately focus on just a few points. Exact computation of the LOCI plots for a handful of points is fast (in the worst case—i.e., full range of radii—it is $O(kN)$ with a very small hidden constant; typical response time is about one to two minutes on real datasets).

`Micro` dataset. In the `micro` dataset, LOCI automatically captures *all* 14 points in the micro-cluster, as well as the outstanding outlier. At a wider range of radii, some points on the fringe of the large cluster are also flagged. The LOCI and aLOCI plots are in Figure 4 and Figure 12, respectively (see Section 3.4 for discussion).

`Sclust` and `Multimix` datasets. We discuss these briefly, due to space constraints (LOCI plots are similar to those already discussed, or combinations thereof). In the `sclust` dataset, as expected, for small radii we do not detect any outliers, whereas for large radii we capture some large deviants. Finally, in the `multimix` dataset, LOCI captures the isolated outliers, some of the "suspicious" ones along the line extending from the bottom uniform cluster and large deviants from the Gaussian cluster.

6.3 Real data

In this section we demonstrate how the above rules apply in a real dataset (see Table 2). In the previous section we discussed the shortcomings of other methods that provide a single number as an "outlier-ness" score. Due to space constraints, we only show LOCI and aLOCI results and discuss the LOCI plots from one real dataset (more results are in the full version of the paper).

`NBA` dataset. Results from LOCI and aLOCI are shown in Figure 13 (for comparison, see Table 3). Figure 14 shows the LOCI plots. The overall deviation indicates that the points form a large, "fuzzy" cluster, throughout all scales. Stockton is clearly an outlier, since he is far different from all other players, with respect to *any* statistic. Jordan is an interesting case; although he is the top-scorer, there are several other players whose overall performance is close (in fact, Jordan does not stand out with respect to any of the other statistics). Corbin is one of the players which aLOCI misses. In Figure 13 he does not really stand out. In fact, his situation is similar to that of the fringe points in the `Dens` dataset!

`NYWomen` dataset. Results from LOCI are shown in Figure 15 (aLOCI provides similar results, ommited for space). This dataset also forms a large cluster, but the top-right section of the cluster is much less dense than the part containing the vast majority of the runners. Although it may initially seem surprising, upon closer examination, the situation here is very similar to the `Micro` dataset! There are two outstanding outliers (extremely slow runners), a sparser but significant "micro-cluster" of slow/recreational runners, then the vast majority of "average" runners which slowly merges with an equally tight (but smaller) group of high-performers. Another important observation is that the fraction of points flagged by both LOCI and aLOCI (about 5%) is well within our expected bounds. The LOCI plots are shown in Figure 16 and can be interpreted much like those for the `Micro` dataset.

7 Conclusions

In summary, the main contributions of LOCI are:

- Like the state of the art, it can detect outliers and groups of outliers (or, micro-clusters). It also includes several of the previous methods (or slight variants thereof) as a "special case."
- Going beyond any previous method, it proposes an automatic, data-dictated cut-off to determine whether a point is an outlier—in contrast, previous methods let the users decide, providing them with no hints as to what cut-off is suitable for each dataset.

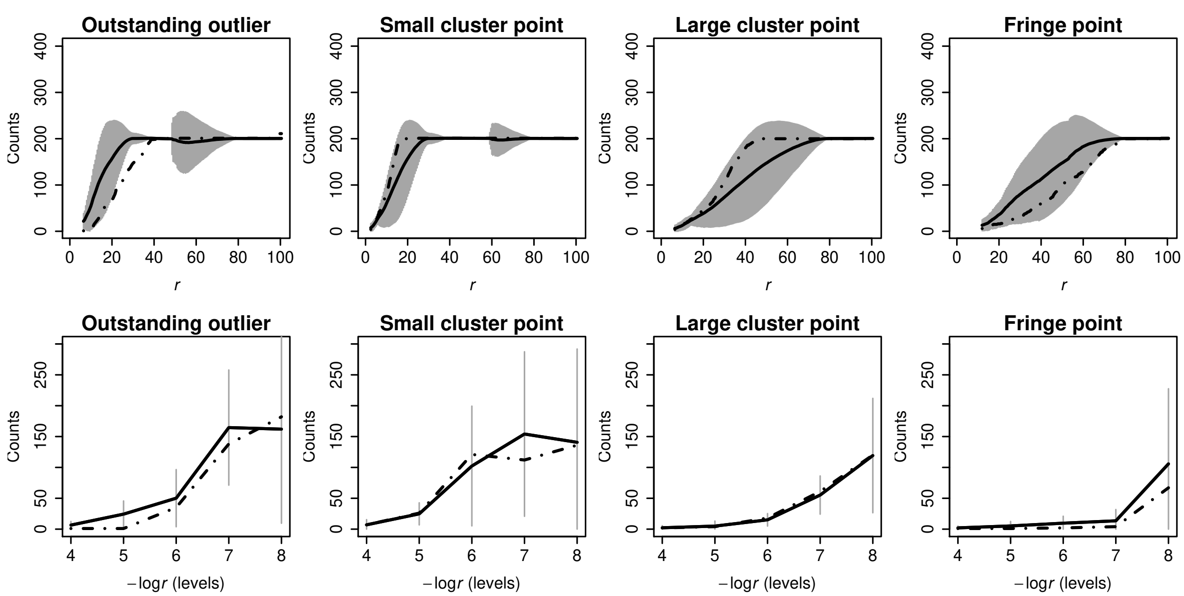


Figure 11: `Dens`, LOCI plots.

LOCI		aLOCI		LOCI		aLOCI	
#	**Player**	**#**	**Player**	**#**	**Player**	**#**	**Player**
1	Stockton J. (UTA)	1	Stockton J (UTA)	8	Corbin T. (MIN)		
2	Johnson K. (PHO)	2	Johnson K (PHO)	9	Malone K. (UTA)		
3	Hardaway T. (GSW)	3	Hardaway T (GSW)	10	Rodman D. (DET)		
4	Bogues M. (CHA)			11	Willis K. (ATL)	6	Willis K (ATL)
5	Jordan M. (CHI)	4	Jordan M (CHI)	12	Scott D. (ORL)		
6	Shaw B. (BOS)			13	Thomas C.A. (SAC)		
7	Wilkins D. (ATL)	5	Wilkins D (ATL)				

Table 3: `NBA` outliers with LOCI and aLOCI. All aLOCI outliers are shown in this table; see also Figure 13.

- Our method successfully deals with both local density and multiple granularity.
- Instead of just an "outlier-ness" score, it provides a whole plot for each point that gives a wealth of information.
- Our exact LOCI method can be computed as quickly as previous methods.
- Moreover, LOCI leads to a very fast, practically linear approximate algorithm, *aLOCI*, which gives accurate results. To the best of our knowledge, this is the first time approximation techniques have been proposed for outlier detection.
- Extensive experiments on synthetic and real data show that LOCI and aLOCI can automatically detect outliers and micro-clusters, without user-required cut-offs, and that they quickly spot outliers, expected and unexpected.

References

[AAR96] A. Arning, R. Agrawal, and P. Raghavan. A linear method for deviation detection in large database. In *Proc. KDD*, pages 164–169, 1996.

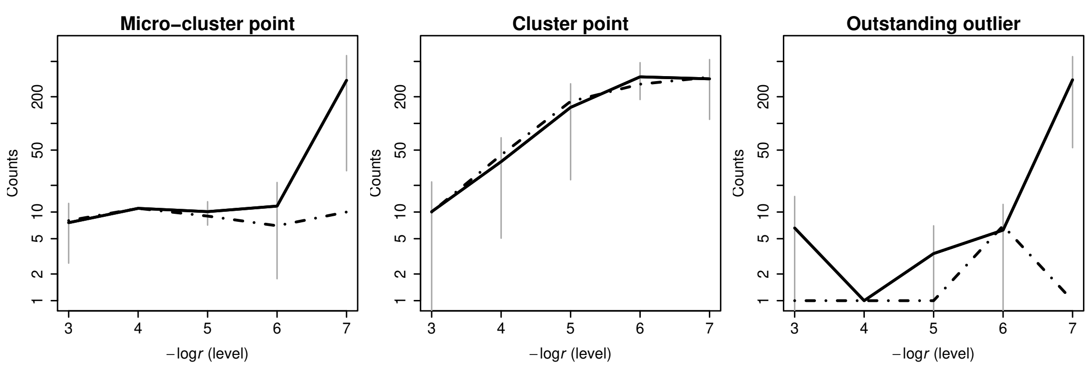
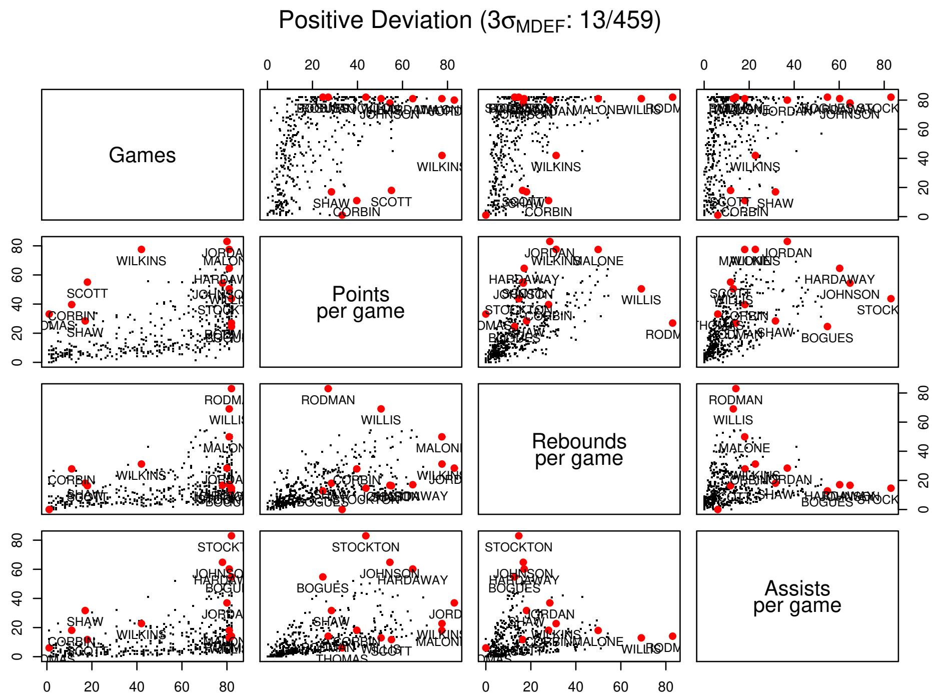


Figure 12: `Micro`, LOCI plots—see Figure 4 for corresponding exact plots.



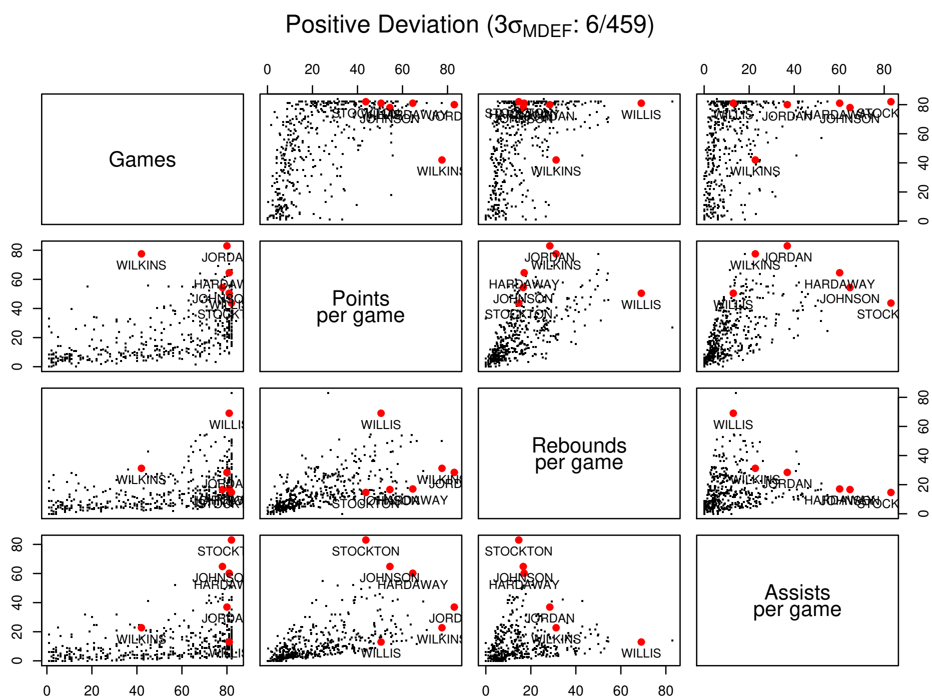


Figure 13: NBA results, LOCI ($\hat{n} = 20$ to full radius) and aLOCI (bottom; 5 levels, $l_\alpha = 4$, 18 grids).

[AMN+98] S. Arya, D.M. Mount, N.S. Netanyahu, R. Silverman, and A.Y. Wu. An optimal algorithm for approximate nearest neighbor searching in fixed dimensions. *JACM*, 45(6):891–923, 1998.

[AY01] C.C. Aggarwal and P.S. Yu. Outlier detection for high dimensional data. In *Proc. SIGMOD*, 2001.

[BC00] Daniel Barbará and Ping Chen. Using the fractal dimension to cluster datasets. In *Proc. KDD*, pages 260–264, 2000.

[Ber93] M. Bern. Approximate closest-point queries in high dimension. *Information Processing Letters*, 45:95–99, 1993.

[BF95] A. Belussi and C. Faloutsos. Estimating the selectivity of spatial queries using the 'correlation' fractal dimension. In *Proc. VLDB*, pages 299–310, 1995.

[BKNS00] M.M. Breunig, H.P. Kriegel, R.T. Ng, and J. Sander. Lof: Identifying density-based local outliers. In *Proc. SIGMOD Conf.*, pages 93–104, 2000.

[BL94] V. Barnett and T. Lewis. *Outliers in Statistical Data*. John Wiley, 1994.

[CNBYM01] E. Chávez, G. Navarro, R. Baeza-Yates, and J.L. Marroquín. Searching in metric spaces. *ACM Comp. Surveys*, 33(3):273–321, 2001.

[FLM77] T. Fiegel, J. Lindenstrauss, and V.D. Millman. The dimensions of almost spherical sections of convex bodies. *Acta Math.*, 139(1-2):53–94, 1977.

[GIM99] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *Proc. VLDB 1999*, pages 518–528, 1999.

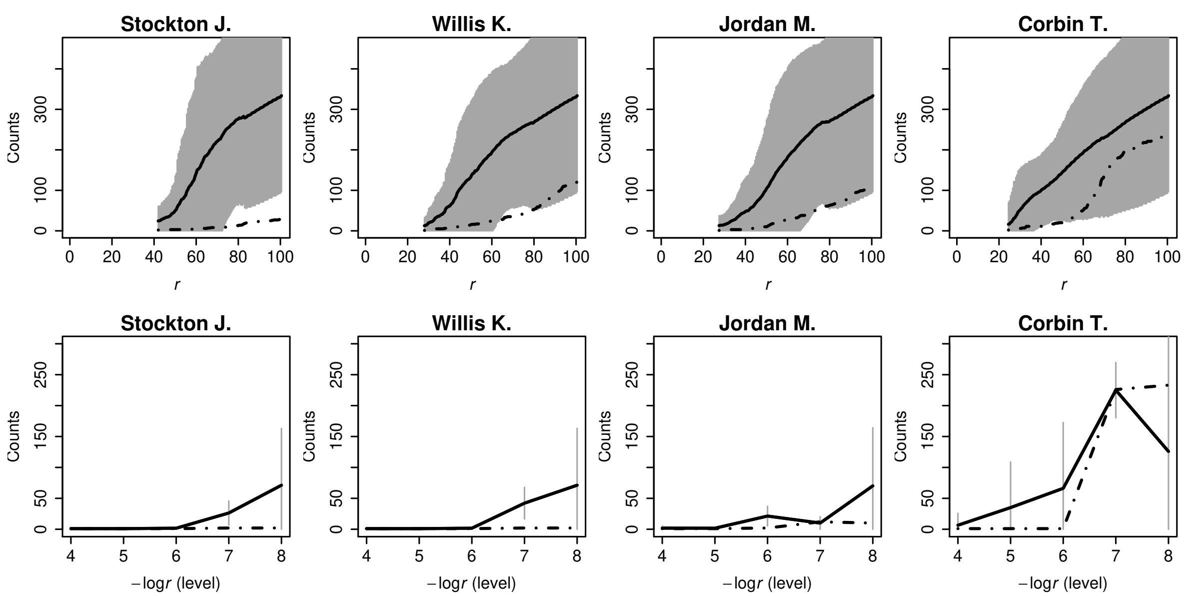


Figure 14: NBA, LOCI plots.

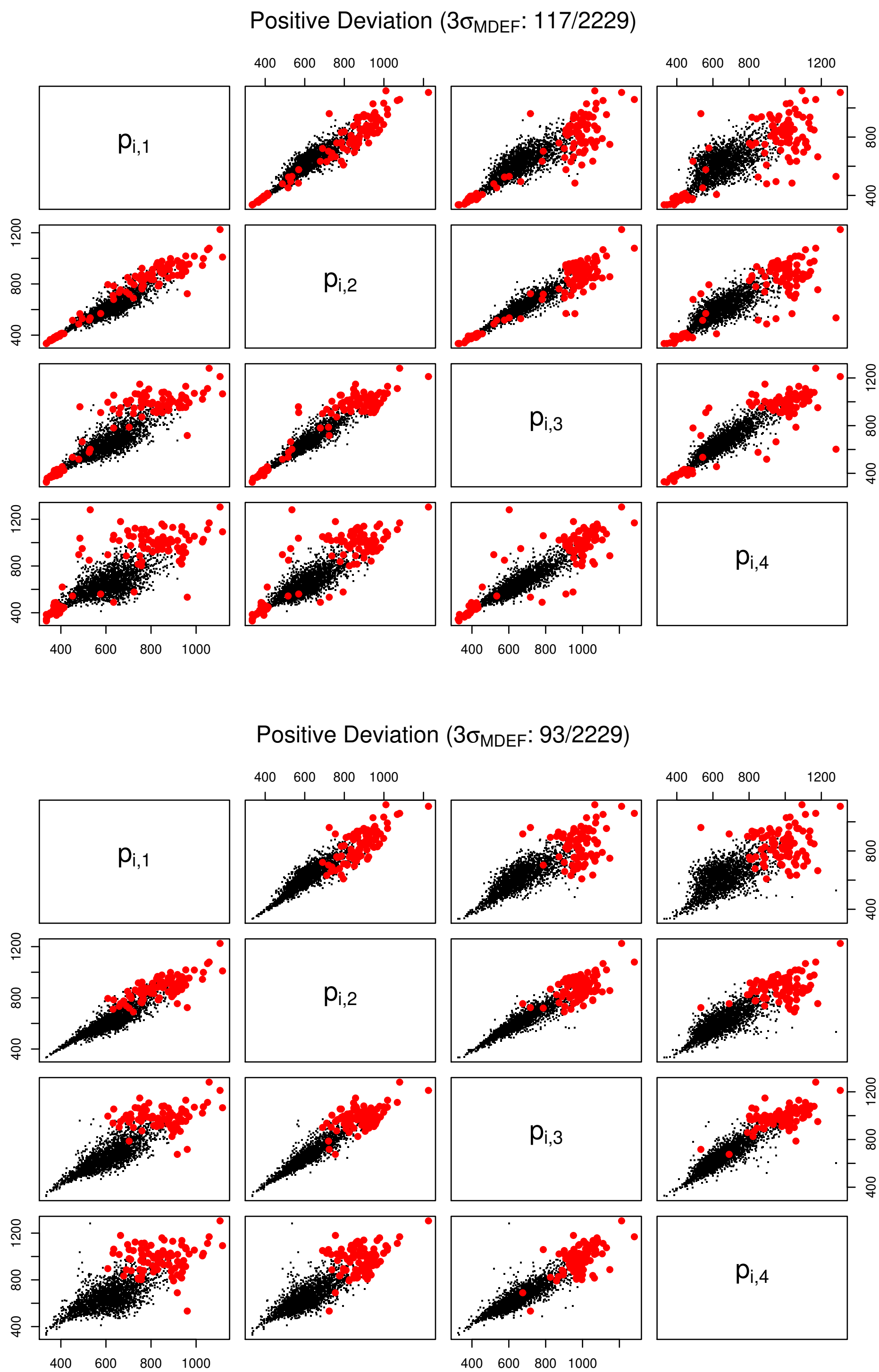


Figure 15: NYWomen, results, LOCI ($\hat{n} = 20$ to full radius) and aLOCI (bottom; 6 levels, $l_\alpha = 3$, 18 grids).

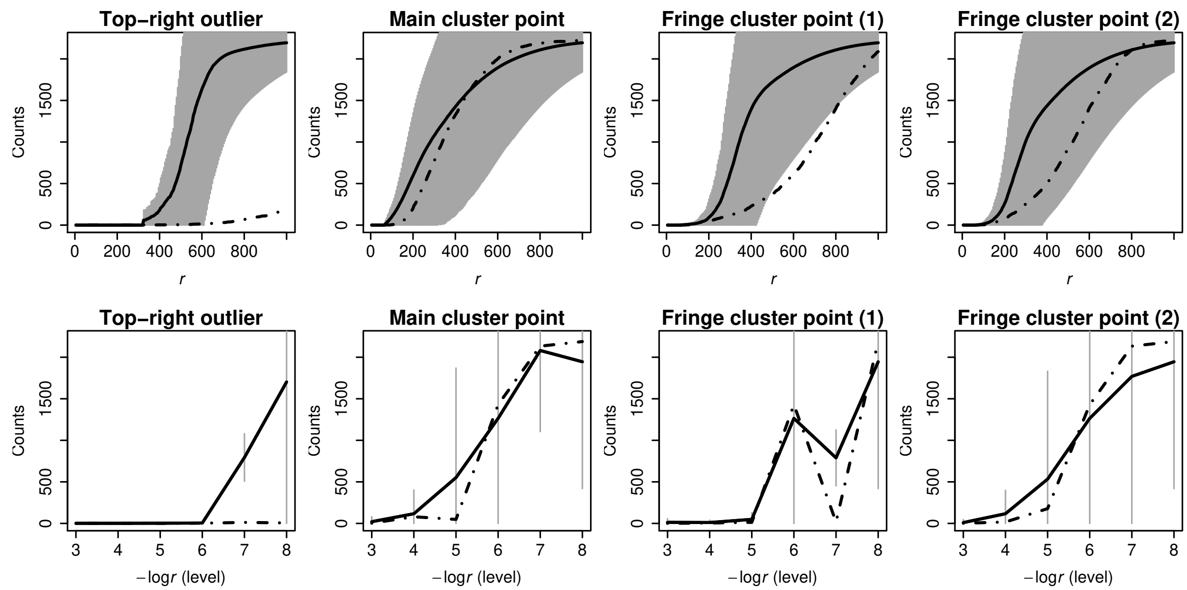


Figure 16: `NYWomen`, LOCI plots.

[Haw80] D.M. Hawkins. *Identification of Outliers*. Chapman and Hall, 1980.

[JKM99] H.V. Jagadish, N. Koudas, and S. Muthukrishnan. Mining deviants in a time series database. In *Proc. VLDB*, pages 102–113, 1999.

[JKN98] T. Johnson, I. Kwok, and R.T. Ng. Fast computation of 2-dimensional depth contours. In *Proc. KDD*, pages 224–228, 1998.

[JMF99] A.K. Jain, M.N. Murty, and P.J. Flynn. Data clustering: A review. *ACM Comp. Surveys*, 31(3):264–323, 1999.

[JTH01] W. Jin, A.K.H. Tung, and J. Ha. Mining top-n local outliers in large databases. In *Proc. KDD*, pages 293–298, 2001.

[KN97] E.M. Knorr and R.T. Ng. A unified notion of outliers: Properties and computation. In *Proc. KDD*, pages 219–222, 1997.

[KN98] E. M. Knorr and R.T. Ng. Algorithms for mining distance-based outliers in large datasets. In *Proc. VLDB 1998*, pages 392–403, 1998.

[KN99] E.M. Knorr and R.T. Ng. Finding intentional knowledge of distance-based outliers. In *Proc. VLDB*, pages 211–222, 1999.

[KNT00] E.M. Knorr, R.T. Ng, and V. Tucakov. Distance-based outliers: Algorithms and applications. *VLDB Journal*, 8:237–253, 2000.

[RL87] P.J. Rousseeuw and A.M. Leroy. *Robust Regression and Outlier Detection*. John Wiley and Sons, 1987.

[Sch88] H.G. Schuster. *Deterministic Chaos*. VCH Publisher, 1988.

[TTPF01] A. Traina, C. Traina, S. Papadimitriou, and C. Faloutsos. Tri-plots: Scalable tools for multidimensional data mining. In *Proc. KDD*, pages 184–193, 2001.