Real-time outlier detection for large datasets by RT-DetMCD

Chemometrics and Intelligent Laboratory Systems 199 (2020) 103957

Contents lists available at ScienceDirect

Chemometrics and Intelligent Laboratory Systems

journal homepage: www.elsevier.com/locate/chemometrics

Real-time outlier detection for large datasets by RT-DetMCD

Bart De Ketelaere [a], Mia Hubert [b], Jakob Raymaekers [b], Peter J. Rousseeuw [b,*], Iwein Vranckx [b]

[a] KU Leuven, Division of Mechatronics, Biostatistics and Sensors, Kasteelpark Arenberg 30, BE-3001, Leuven, Belgium
[b] KU Leuven, Section of Statistics and Data Science, Celestijnenlaan 200B, BE-3001, Leuven, Belgium[1]

ARTICLE INFO

Keywords:
Anomaly detection
Minimum covariance determinant
Parallel computing
Robust aggregation
Robust estimation

ABSTRACT

Modern industrial machines can generate gigabytes of data in seconds, frequently pushing the boundaries of available computing power. Together with the time criticality of industrial processing this presents a challenging problem for any data analytics procedure. We focus on the deterministic minimum covariance determinant method (DetMCD), which detects outliers by fitting a robust covariance matrix. We construct a much faster version of DetMCD by replacing its initial estimators by two new methods and incorporating update-based concentration steps. The computation time is reduced further by parallel computing, with a novel robust aggregation method to combine the results from the threads. The speed and accuracy of the proposed real-time DetMCD method (RT-DetMCD) are illustrated by simulation and a real industrial application to food sorting.

1. Introduction

Modern industries are data-rich environments where information from multiple sensors is captured at a high sampling frequency. Processing such data has to cope with typical challenges such as the presence of outliers. While classical statistical estimators can be highly affected by outliers, their robust counterparts can cope with a significant fraction of contamination. There is a vast literature about robust statistical techniques (e.g. Refs. [1–4]). Although substantial research has already gone into constructing fast robust algorithms, more work is needed to be able to handle real-time multivariate situations with many thousands of observations per second, as required by some industrial processes.

For this task we will focus on the Minimum Covariance Determinant (MCD) approach [2,5,6] which provides highly robust estimators for multivariate location and covariance matrices. Its first practical algorithm was FastMCD [7]. More recently the DetMCD algorithm [8] was constructed, which is deterministic unlike the random sampling component of FastMCD. Although DetMCD is significantly faster it is still prohibitive for the huge sample sizes envisaged here. For routine use in real-time industrial environments we need to speed it up further, which motivated this research.

A recent review paper [9] discussed the perspectives of robust methods for industrial process management when outliers are present. It highlighted several paths that can be explored. One of these is the evolution from a centralized analysis of large datasets towards parallel computing, whereby multiple threads work in parallel on data subsets after which the results are combined for the final result. Our work on DetMCD will indeed incorporate parallel computing.

The remainder of the paper is organized as follows. In Section 2 we describe the DetMCD estimator and its main properties. Section 3 proposes an improved serial version which incorporates various new techniques and is substantially faster. Section 4 constructs a parallelized version, which speeds up computation even more. The simulation in Section 5 confirms the robustness, speed and accuracy of the proposed method. Section 6 analyzes a real industrial dataset, and Section 7 concludes.

2. The minimum covariance determinant approach

Our goal is to detect outliers in a multivariate dataset with n observations and p variables. We denote the data by $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)^T$ where each observation $\boldsymbol{x}_i = (x_{i1}, x_{i2}, \ldots, x_{ip})^T$ is a p-dimensional column vector. Here we assume that p is moderate, say no more than 40, otherwise a dimension reduction technique such as robust PCA [10] can be used. The sample size n should be higher than p and is allowed to be huge, even up to several millions. We assume that the inliers roughly follow a multivariate Gaussian distribution $N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ with center $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$, possibly after transforming some skewed variables.

* Corresponding author.
E-mail address: peter.rousseeuw@kuleuven.be (P.J. Rousseeuw).
[1] wis.kuleuven.be/statdatascience/robust

https://doi.org/10.1016/j.chemolab.2020.103957
Received 12 October 2019; Received in revised form 24 January 2020; Accepted 25 January 2020
Available online 30 January 2020

2.1. The MCD estimator

Robust statistical methods aim to model the inlying cases and then flag outliers as those observations that deviate too much from that model. Here we will focus on the Minimum Covariance Determinant (MCD) estimator [6]. Given a user-specified tuning constant h, where $\lfloor (n+p+1)/2 \rfloor \leq h < n$, the raw MCD estimator is $(\widehat{\boldsymbol{\mu}}_{raw}, \widehat{\boldsymbol{\Sigma}}_{raw})$ where the location estimate $\widehat{\boldsymbol{\mu}}_{raw}$ is the mean of the h observations whose sample covariance matrix has the smallest determinant. Intuitively these h observations are the most concentrated, since the determinant of a covariance matrix corresponds to the volume of its tolerance ellipsoid. The scatter matrix estimate $\widehat{\boldsymbol{\Sigma}}_{raw}$ is that covariance matrix multiplied by the consistency factor $c(\alpha)$ of [11] that depends on $\alpha = h/n$ and compensates for the fact that only h out of n observations are included.

The indices i of these h observations form a set H, called an h-subset. The raw MCD estimates are then given by

$$\widehat{\boldsymbol{\mu}}_{raw} = \frac{1}{h} \sum_{i \text{ in } H} \boldsymbol{x}_i \,, \tag{1}$$

$$\widehat{\boldsymbol{\Sigma}}_{raw} = \frac{c(\alpha)}{h-1} \sum_{i \text{ in } H} (\boldsymbol{x}_i - \widehat{\boldsymbol{\mu}}_{raw})(\boldsymbol{x}_i - \widehat{\boldsymbol{\mu}}_{raw})^T. \tag{2}$$

Note that the MCD is only defined when $h > p$, otherwise the covariance matrix of any h-subset is singular, so we want $n > 2p$. In practice it is however recommended that n be much larger, in order to obtain a more accurate result.

The raw MCD estimator is highly robust as it can withstand up to $n - h$ outliers. The breakdown value of an estimator is the proportion of outliers that can be resisted. The breakdown value of the MCD is $1 - \alpha$. Choosing $\alpha = 0.5$ yields an estimator with a maximal breakdown value of 50% but a rather low statistical efficiency, whereas taking $\alpha = 0.75$ yields a more efficient estimator with lower 25% breakdown value.

To increase the efficiency we carry out a reweighting step. For this we first measure how much each data point $\boldsymbol{x}_i$ deviates from the raw MCD fit, by computing the *robust distances* $\mathrm{RD}_i = d(\boldsymbol{x}_i, \widehat{\boldsymbol{\mu}}_{raw}, \widehat{\boldsymbol{\Sigma}}_{raw})$ where the statistical distance d is defined as

$$d(\boldsymbol{x}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sqrt{(\boldsymbol{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x}-\boldsymbol{\mu})} \,.$$

The reweighted MCD estimates $(\widehat{\boldsymbol{\mu}}_{rew}, \widehat{\boldsymbol{\Sigma}}_{rew})$ are then computed as the mean and covariance matrix of the observations $\boldsymbol{x}_i$ whose RD_i do not exceed the cut-off value $c_p = \sqrt{\chi^2_{p,0.975}}$ (where χ^2_p is the chi-squared distribution with p degrees of freedom). Then outliers are flagged as those cases whose final robust distance $\mathrm{RD}_i = d(\boldsymbol{x}_i, \widehat{\boldsymbol{\mu}}_{rew}, \widehat{\boldsymbol{\Sigma}}_{rew})$ exceeds c_p. Note that a higher cutoff such as $\sqrt{\chi^2_{p,0.99}}$ could be chosen, but in this paper the 0.975 quantile was used throughout to be able to detect outliers that are relatively close to the majority. This was important in the application on food sorting in Section 6, where letting pass some foreign material creates bigger problems (such as regulatory) than discarding a small fraction of potentially clean food.

Note that the reweighted MCD inherits the breakdown value of the raw MCD, so setting $\alpha = 0.5$ yields a reweighted estimator with a breakdown value of 50%.

When any nonsingular affine transformation is applied to the data (such as a rotation, a reflection or rescaling) the MCD estimator transforms along with it. This is called affine equivariance. Therefore the robust distances RD_i remain invariant under such a transformation.

The exact raw MCD is very hard to compute, as it requires the evaluation of all $\binom{n}{h}$ subsets of size h which is infeasible for increasing n. The FastMCD algorithm of [7] approximates the MCD in an efficient, robust and affine equivariant way. A major component of FastMCD is the so-called *concentration step* (C-step), which works as follows. Given initial estimates $\widehat{\boldsymbol{\mu}}_{old}$ for the center and $\widehat{\boldsymbol{\Sigma}}_{old}$ for the scatter matrix, we do:

1. Compute the distances of all n observations as

$$d_{old}(i) = d(\boldsymbol{x}_i, \widehat{\boldsymbol{\mu}}_{old}, \widehat{\boldsymbol{\Sigma}}_{old}). \tag{3}$$

2. Sort these distances, yielding a permutation π for which

$$d_{old}(\pi(1)) \leq d_{old}(\pi(2)) \leq \ldots \leq d_{old}(\pi(n)).$$

3. Define the h-subset H_{new} as

$$H_{new} = \{\pi(1), \pi(2), \ldots, \pi(h)\}.$$

4. Compute the new estimates based on H_{new} :

$$\widehat{\boldsymbol{\mu}}_{new} = \frac{1}{h} \sum_{i \text{ in } H_{new}} \boldsymbol{x}_i \,, \tag{4}$$

$$\widehat{\boldsymbol{\Sigma}}_{new} = \frac{1}{h-1} \sum_{i \text{ in } H_{new}} (\boldsymbol{x}_i - \widehat{\boldsymbol{\mu}}_{new})(\boldsymbol{x}_i - \widehat{\boldsymbol{\mu}}_{new})^T. \tag{5}$$

Proposition 1 in Ref. [7] showed that $\det(\widehat{\boldsymbol{\Sigma}}_{new}) \leq \det(\widehat{\boldsymbol{\Sigma}}_{old})$, with equality if and only if $\widehat{\boldsymbol{\Sigma}}_{new} = \widehat{\boldsymbol{\Sigma}}_{old}$. When C-steps are applied iteratively, the sequence of determinants must therefore converge.

FastMCD starts by drawing a random $(p+1)$-subset from the data. Next, its mean and covariance matrix serve as $\widehat{\boldsymbol{\mu}}_{old}$ and $\widehat{\boldsymbol{\Sigma}}_{old}$ in a C-step. The algorithm draws many such $(p+1)$-subsets, applies several C-steps to each, and keeps the solution with the overall lowest determinant.

The computational cost of FastMCD obviously depends on n and p, but also on the number of random $(p+1)$-subsets. The default number of initial subsets is 500, but [8] illustrates that this is insufficient at high contamination levels when p exceeds 10, independent of the sample size n. In those situations a substantially larger number of initial subsets would be required, thereby increasing the computational cost significantly.

2.2. The DetMCD algorithm

As an alternative the DetMCD algorithm [8] was constructed. It is fully deterministic as it does not use random subsets. It is more robust than FastMCD, and needs less computation time. The only price to pay is the loss of affine equivariance. DetMCD is only location and scale equivariant, but simulations in Ref. [8] showed that it is very close to affine equivariant. The main steps of DetMCD are summarized below, and its flowchart is depicted in Fig. 1. For all details we refer to Ref. [8].

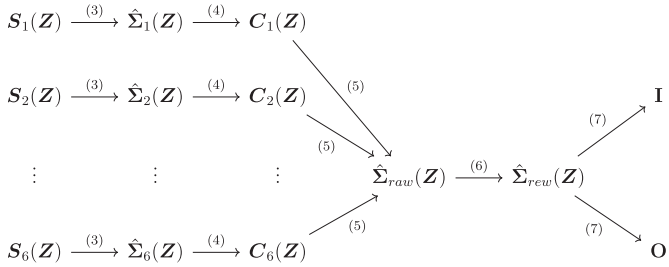

Fig. 1. The DetMCD algorithm. From left to right: six scatter matrices $\boldsymbol{S}_k$ from step 2 are refined (step 3) to $\widehat{\boldsymbol{\Sigma}}_k(\boldsymbol{Z})$, followed by C-steps until convergence (step 4). The matrix $\widehat{\boldsymbol{\Sigma}}_{raw}(\boldsymbol{Z})$ is the $\boldsymbol{C}_k(\boldsymbol{Z})$ with the lowest determinant (step 5). Step 6 creates the reweighted estimate $\widehat{\boldsymbol{\Sigma}}_{rew}(\boldsymbol{Z})$ which is then used to flag outliers (step 7).

1. Each variable of the dataset X is standardized by subtracting its median and dividing by a robust scale estimate, yielding the standardized dataset Z.
2. Six initial estimates $S_k(Z)$, $k = 1, \ldots, 6$ of the scatter of Z are constructed. These initial estimators are fully deterministic and each of them is resistant to certain types of outliers.
3. As the eigenvalues of $S_k(Z)$ might be inaccurate, they are refined by the routine described in Subsection 3.3. We denote the resulting covariance matrix by $\widehat{\Sigma}_k(Z)$ and its location by $\widehat{\mu}_k(Z)$.
4. Each $(\widehat{\mu}_k(Z), \widehat{\Sigma}_k(Z))$ is used to start C-steps which are iterated to convergence. In each case the resulting scatter matrix is multiplied by $c(\alpha)$ as in (2), yielding the scatter estimate $C_k(Z)$.
5. The raw DetMCD covariance estimate $\widehat{\Sigma}_{raw}$ is chosen as the $C_k(Z)$ with the lowest determinant, with corresponding location estimate $\widehat{\mu}_{raw}$.
6. A reweighting step is applied to improve the statistical accuracy as in Ref. [7], yielding the final DetMCD estimates $(\widehat{\mu}_{rew}, \widehat{\Sigma}_{rew})$.
7. The robust distances $\mathrm{RD}_i = d(z_i, \widehat{\mu}_{rew}, \widehat{\Sigma}_{rew})$ then allow to classify the observations into **I**nliers and **O**utliers.

The DetMCD algorithm thus uses an ensemble of initial estimators to ensure high robustness against different contamination patterns. It is faster than the algorithm in Subsection 2.1, but not yet fast enough for real-time applications with high n. The main bottlenecks are the computation of some of the initial estimators S_k and the time taken by the C-steps. The next Section describes how these costs can be reduced.

3. An improved deterministic MCD

3.1. Standardizing the data

In the first step each variable is standardized by means of a robust estimator of location and scale. Whereas DetMCD used the median and an M-estimator of scale, we now use the univariate reweighted MCD estimator of [4] with coverage $\tilde{h} = \lfloor n/2 \rfloor + 1$. Note that for univariate data, the raw MCD estimates reduce to the mean and the standard deviation of the $\tilde{h}$-subset with smallest variance. They can be computed in $O(n \log(n))$ time as in Ref. [4] by sorting the data, followed by looping over contiguous $\tilde{h}$-subsets while updating their means and variances. We prefer the univariate MCD because methods that give zero-one weights to observations can be more robust against nearby contamination [12]. The standardized dataset Z then consists of the columns $Z_j = (X_j - \widehat{\mu}_{uni}(X_j))/\widehat{\sigma}_{uni}(X_j)$.

3.2. New initial estimators

The six initial estimates used by DetMCD are of several types. The first three estimators start by transforming the variables one by one, either by the sigmoid transformation $\tilde{Z}_j = \tanh(Z_j)$, the rank transformation, or the normal scores from the ranks. The resulting estimator is then the classical covariance matrix of the transformed variables. We will replace these three estimates by a single new one from Ref. [13], using the transformation

$$\tilde{z}_{ij} = g(z_{ij}) = \begin{cases} z_{ij} & \text{if } 0 \le |z_{ij}| \le b \\ q_1 \tanh\left(q_2 (c - |z_{ij}|)\right) \mathrm{sign}(z_{ij}) & \text{if } b < |z_{ij}| \le c \\ 0 & \text{if } |z_{ij}| > c \end{cases} \tag{6}$$

for $i = 1, \ldots, n$ and $j = 1, \ldots, p$. This transformation is called *wrapping*. The default choices are $b = 1.5$, $c = 4$, $q_1 = 1.541$ and $q_2 = 0.862$, which yield a continuous function g. These default choices strike a balance between accuracy for clean data and robustness for contaminated data. The choice $b = 1.5$ implies that for perfectly Gaussian data about 85% of the values are left unchanged, so that the subsequent computations remain accurate. The value $c = 4$ reflects that we do not trust measurements that lie more than 4 standard deviations away.

Next, we compute the new initial estimator $\tilde{S}_1$ as the covariance matrix of the wrapped data. In an extensive comparison study [13], this approach was shown to perform at least as well as the other three transformations, so we replace S_1, S_2 and S_3 by $\tilde{S}_1$.

The initial estimators S_4 and S_5 in DetMCD belong to the class of Generalized Spatial Sign Covariance Matrices (GSSCM) [14], which generalizes [15]. Among several versions [14], concluded that the so-called *linearly redescending* GSSCM performed very well, so we will use it as our second initial estimator $\tilde{S}_2$. It is defined as

$$\tilde{S}_2 = \frac{1}{n} \sum_{i=1}^{n} \xi^2(||z_i||)\, z_i z_i^T \tag{7}$$

where the weight function ξ is given by

$$\xi(r) = \begin{cases} 1 & \text{if } r \le A \\ (B - r)/(B - A) & \text{if } A < r \le B \\ 0 & \text{if } r > B \,. \end{cases}$$

The cutoffs A and B depend on the set of norms $||z_i||$ as detailed in Ref. [13]. In particular, A is roughly equal to the median of the $||z_i||$. We replace S_4 and S_5 by $\tilde{S}_2$, which achieves a breakdown value of 50%.

The final initial estimator S_6 was the OGK estimator [16]. Whereas S_6 performed quite well, it was by far the most computationally demanding among the six initial estimators of DetMCD. Fortunately simulations showed that the new $\tilde{S}_1$ and $\tilde{S}_2$ together are sufficient, so we can replace the six initial estimates by the fast methods $\tilde{S}_1$ and $\tilde{S}_2$ which saves computation time.

3.3. Refinement of initial estimates

As our initial estimators $\tilde{S}_k$ for $k = 1, 2$ may have inaccurate or tiny eigenvalues, we propose a refinement procedure similar to that in Ref. [8] which uses parts of [16].

1. $\tilde{S}_k$ is a symmetric matrix so it can be diagonalized as

$$\tilde{S}_k = VDV^T$$

where V is the matrix of eigenvectors of $\tilde{S}_k$ and D is the diagonal matrix with decreasing eigenvalues $\lambda_1 \ge \ldots \ge \lambda_p$. Compute the matrix T of principal component scores as

$$T = ZV \,.$$

2. If the condition number λ_1/λ_p of $\tilde{S}_k$ exceeds a predefined threshold of (say) $\kappa_{max} = 1000$, then $\tilde{S}_k$ is said to be ill-conditioned [17]. Then a warning is given and we do not continue with $\tilde{S}_k$.
3. Applying the univariate MCD estimator to the scores yields a new diagonal matrix

$$\tilde{D} = \mathrm{diag}\left(\widehat{\sigma}^2_{uni}(T_1), \ldots, \widehat{\sigma}^2_{uni}(T_p)\right)$$

from which we compute the refined scatter matrix as

$$\widehat{\Sigma}_k = V\tilde{D}V^T \,.$$

4. The center of Z is estimated by sphering the data, yielding $\tilde{Z} = \widehat{\Sigma}_k^{-1/2} Z$ with columns $\tilde{Z}_j$ for $j = 1, \ldots, p$. The univariate MCD estimator for location is then applied to each $\tilde{Z}_j$ and the result is transformed back, i.e.

$$\widehat{\boldsymbol{\mu}}_k(\boldsymbol{Z}) = \widehat{\boldsymbol{\Sigma}}_k^{1/2}\left(\widehat{\mu}_{uni}(\tilde{Z}_1), \ldots, \widehat{\mu}_{uni}(\tilde{Z}_p)\right)^T.$$

3.4. Speeding up the C-step by Cholesky decomposition

Starting from both refined estimators $\widehat{\boldsymbol{\Sigma}}_k$ we then iterate C-steps as in the DetMCD algorithm. The main cost of a C-step is the computation of the distances (3) based on the inverse of the covariance matrix $\widehat{\boldsymbol{\Sigma}}_{old}$. For this we propose to use the Cholesky decomposition, i.e.

$$\widehat{\boldsymbol{\Sigma}}_{old} = \boldsymbol{L}\boldsymbol{L}^T$$

with $\boldsymbol{L}$ a lower triangular $p \times p$ matrix. We then compute $\boldsymbol{y}_i = \boldsymbol{L}^{-1}(\boldsymbol{z}_i - \widehat{\boldsymbol{\mu}}_{old})$ by forward substitution. It can easily be verified that

$$d(\boldsymbol{z}_i, \widehat{\boldsymbol{\mu}}_{old}, \widehat{\boldsymbol{\Sigma}}_{old}) = ||\boldsymbol{y}_i|| \ .$$

We prefer the Cholesky decomposition over other approaches as it is fast and very stable numerically [18]. It immediately yields the determinant by $\det(\widehat{\boldsymbol{\Sigma}}_{old}) = \left(\prod_{j=1}^{p} L_{jj}\right)^2$ with L_{jj} the diagonal elements of $\boldsymbol{L}$.

The Cholesky decomposition also allows us to monitor the condition number, following Algorithms 4.1 and 5.1 in Ref. [19]. If

$$||\widehat{\boldsymbol{\Sigma}}_{old}||_1 \ ||\widehat{\boldsymbol{\Sigma}}_{old}^{-1}||_1 \geq \kappa_{max}$$

we approach singularity, and then the C-step is not taken. We thus monitor the condition number in two different stages of the algorithm: in the refinement procedure of $\tilde{\boldsymbol{S}}_k$ (Subsection 3.3) and in each C-step.

3.5. Further speedup by updating

To further speed up the C-step, we avoid redoing all computations for the new h-subset. Let H_{old} be the current h-subset, and H_{new} the new one obtained by sorting distances. We describe the changes in going from H_{old} to H_{new} by an n-dimensional vector $\boldsymbol{\delta} = (\delta_1, \ldots, \delta_n)^T$ in which δ_i in $\{+1, 0, -1\}$ indicates whether observation i enters, stays in, or leaves H_{old}. Obviously $\sum_i \delta_i = 0$. We will use the sum of squares and cross-products (sscp) matrix $\boldsymbol{\Lambda}_{old} = (h-1)\widehat{\boldsymbol{\Sigma}}_{old}$ which is the covariance matrix $\widehat{\boldsymbol{\Sigma}}_{old}$ without denominator. Initially $\widehat{\boldsymbol{\mu}}_{new} = \widehat{\boldsymbol{\mu}}_{old}$ and $\boldsymbol{\Lambda}_{new} = \boldsymbol{\Lambda}_{old}$. We then update the center and the sscp matrix sequentially [20–22] as follows. For each i with $\delta_i \neq 0$:

1. The total number of observations in the subset is updated:

$$h \leftarrow h + \delta_i \ .$$

2. The center $\widehat{\boldsymbol{\mu}}_{new}$ is updated, and the contribution of $\boldsymbol{z}_i$ before and after the update is computed:

$$\boldsymbol{u}_i = \boldsymbol{z}_i - \widehat{\boldsymbol{\mu}}_{new}$$

$$\widehat{\boldsymbol{\mu}}_{new} \leftarrow \widehat{\boldsymbol{\mu}}_{new} + \frac{\delta_i}{h}\boldsymbol{u}_i$$

$$\boldsymbol{v}_i = \boldsymbol{z}_i - \widehat{\boldsymbol{\mu}}_{new} \ .$$

3. Finally the sscp matrix $\boldsymbol{\Lambda}_{new}$ is updated as

$$\boldsymbol{\Lambda}_{new} \leftarrow \boldsymbol{\Lambda}_{new} + \delta_i \boldsymbol{u}_i \boldsymbol{v}_i^T \ .$$

This one-pass loop replaces (4) and (5) of the original C-step procedure, and accounts for a noteworthy speedup.

When $\sum_i |\delta_i| = 2$, i.e. when only two cases are interchanged, it is even faster to update the inverse directly. From the Sherman-Morrison-Woodbury identity

$$(\boldsymbol{A} + \boldsymbol{u}\boldsymbol{v}^T)^{-1} = \boldsymbol{A}^{-1} - \frac{\boldsymbol{A}^{-1}\boldsymbol{u}\boldsymbol{v}^T\boldsymbol{A}^{-1}}{1 + \boldsymbol{v}^T\boldsymbol{A}^{-1}\boldsymbol{u}}$$

we obtain

$$\left(\boldsymbol{\Lambda}_{new} + \delta_i \boldsymbol{u}_i \boldsymbol{v}_i^T\right)^{-1} = \boldsymbol{\Lambda}_{new}^{-1} - \frac{\delta_i}{\Delta_i}\left(\boldsymbol{\Lambda}_{new}^{-1}\boldsymbol{u}_i\boldsymbol{v}_i^T\boldsymbol{\Lambda}_{new}^{-1}\right)$$

with $\Delta_i := (1 + \delta_i \boldsymbol{v}_i^T \boldsymbol{\Lambda}_{new}^{-1}\boldsymbol{u}_i)$. Finally, we update the determinant for each change in a case i using the identity

$$\det\left(\boldsymbol{\Lambda}_{new} + \delta_i \boldsymbol{u}_i \boldsymbol{v}_i^T\right) = \Delta_i \det(\boldsymbol{\Lambda}_{new}) \ .$$

After the C-steps have converged, we multiply $\widehat{\boldsymbol{\Sigma}}_{new} = \boldsymbol{\Lambda}_{new}/(h-1)$ by $c(\alpha)$ as in (2).

4. Parallel computation and aggregation

Our final computational improvement stems from parallelization. Let $\boldsymbol{X}$ denote the dataset of n observations in p dimensions as before. We then randomly partition the dataset in q disjoint blocks $\boldsymbol{X}^{(l)}$ of $m = \lfloor n/q \rfloor$ cases (discarding the remaining cases if n is not divisible by q). Next, we standardize the blocks by

$$z_{ij}^{(l)} = \frac{x_{ij}^{(l)} - \widehat{\mu}_{uni}(X_j)}{\widehat{\sigma}_{uni}(X_j)}$$

where $l = 1, \ldots, q$ and $\widehat{\mu}_{uni}(.)$ and $\widehat{\sigma}_{uni}(.)$ are the univariate MCD estimators of location and scale (Subsection 3.1). As in Fig. 2 we then use the available processing threads as follows.

1. Compute the initial estimate $\tilde{\boldsymbol{S}}_1^{(l)}(\boldsymbol{Z}^{(l)})$ by wrapping (6), and $\tilde{\boldsymbol{S}}_2^{(l)}(\boldsymbol{Z}^{(l)})$ by the GSSCM method (7).
2. Both estimates are then refined using the procedure outlined in Subsection 3.3, which yields $\widehat{\boldsymbol{\Sigma}}_1(\boldsymbol{Z}^{(l)})$ and $\widehat{\boldsymbol{\Sigma}}_2(\boldsymbol{Z}^{(l)})$.
3. We then apply step 4 of the DetMCD algorithm in Subsection 2.2 to each, using the improvements of Section 3, yielding $\boldsymbol{C}_1(\boldsymbol{Z}^{(l)})$ and $\boldsymbol{C}_2(\boldsymbol{Z}^{(l)})$.
4. The raw DetMCD for the block $l = 1, \ldots, q$ is then given by

$$\left(\widehat{\boldsymbol{\mu}}_{raw}^{(l)}, \widehat{\boldsymbol{\Sigma}}_{raw}^{(l)}\right) := \begin{cases} \left(\widehat{\boldsymbol{\mu}}_1^{(l)}, \widehat{\boldsymbol{\Sigma}}_1^{(l)}\right) & \text{if } \det\left(\widehat{\boldsymbol{\Sigma}}_1^{(l)}\right) \leq \det\left(\widehat{\boldsymbol{\Sigma}}_2^{(l)}\right) \\ \left(\widehat{\boldsymbol{\mu}}_2^{(l)}, \widehat{\boldsymbol{\Sigma}}_2^{(l)}\right) & \text{otherwise,} \end{cases}$$

where the type of initial estimator can vary between blocks. Note that the percentage of inliers in the blocks fluctuates around the percentage in the overall dataset, so it is likely that a majority of the q fits $(\widehat{\boldsymbol{\mu}}_{raw}^{(l)}, \widehat{\boldsymbol{\Sigma}}_{raw}^{(l)})$ are robust, but some may not be.

5. We now need to aggregate these q fits in a robust way. They have many dimensions since the symmetric matrices $\widehat{\boldsymbol{\Sigma}}_{raw}^{(l)}$ contain $p(p-1)/2$ distinct entries, and the $\widehat{\boldsymbol{\mu}}_{raw}^{(l)}$ have p additional entries. Since the total dimension will often be higher than q, computing a typical robust estimate of the q fits is problematic. Therefore we compute the entrywise median of the q fits, yielding the entrywise median of the $\widehat{\boldsymbol{\mu}}^{(l)}$ denoted as

$$\widehat{\boldsymbol{\mu}}_{med} = \left(\text{median}_l\left(\left(\widehat{\boldsymbol{\mu}}_{raw}^{(l)}\right)_1\right), \ldots, \text{median}_l\left(\left(\widehat{\boldsymbol{\mu}}_{raw}^{(l)}\right)_p\right)\right)^T$$

and the entrywise median of all scatter matrices, given by

$$\left(\widehat{\boldsymbol{\Sigma}}_{med}\right)_{jk} = \text{median}_l\left(\left(\widehat{\boldsymbol{\Sigma}}_{raw}^{(l)}\right)_{jk}\right) \tag{8}$$

for $j, k = 1, \ldots, p$. (Instead of the median also other robust univariate

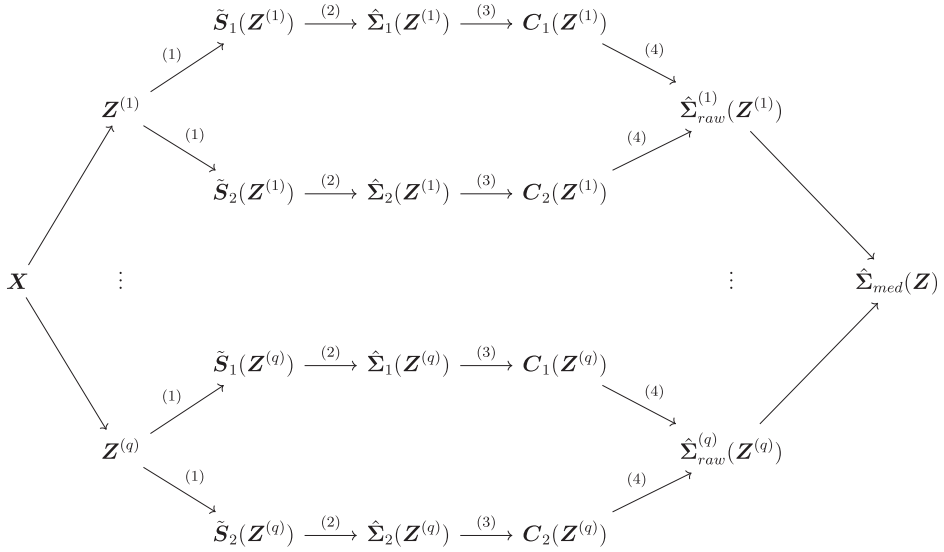

Fig. 2. First part of the parallel processing topology of RT-DetMCD, which computes q raw scatter estimates.

estimators could be used.) Note that the matrix $\widehat{\boldsymbol{\Sigma}}_{med}$ is a robust summary, but it does not have to be positive definite. Therefore, we cannot use $\widehat{\boldsymbol{\Sigma}}_{med}$ as a final aggregated outcome.

6. As a measure of how far the l-th fit $(\widehat{\boldsymbol{\mu}}^{(l)}_{raw}, \widehat{\boldsymbol{\Sigma}}^{(l)}_{raw})$ is from the entrywise median $(\widehat{\boldsymbol{\mu}}_{med}, \widehat{\boldsymbol{\Sigma}}_{med})$, each thread computes the Kullback-Leibler deviation $\mathrm{KL}[(\widehat{\boldsymbol{\mu}}_{med}, \widehat{\boldsymbol{\Sigma}}_{med}), (\widehat{\boldsymbol{\mu}}^{(l)}_{raw}, \widehat{\boldsymbol{\Sigma}}^{(l)}_{raw})]$ given by

$$\mathrm{KL}[(\boldsymbol{a},\boldsymbol{A}),(\boldsymbol{b},\boldsymbol{B})] := \mathrm{trace}\left(\boldsymbol{A}\boldsymbol{B}^{-1}\right) - p - \log\left(\det\left(\boldsymbol{A}\boldsymbol{B}^{-1}\right)\right) + (\boldsymbol{a}-\boldsymbol{b})^T\boldsymbol{B}^{-1}(\boldsymbol{a}-\boldsymbol{b}) \,. \tag{9}$$

The quantity $\mathrm{KL}[(\boldsymbol{a},\boldsymbol{A}),(\boldsymbol{b},\boldsymbol{B})]$ is nonnegative. It is zero when $\boldsymbol{a} = \boldsymbol{b}$ and $\boldsymbol{A} = \boldsymbol{B}$, low when $(\boldsymbol{a},\boldsymbol{A})$ deviates little from $(\boldsymbol{b},\boldsymbol{B})$, and high when they are very different.

Note that Formula (9) is not symmetric in its arguments, meaning that $\mathrm{KL}[(\boldsymbol{a},\boldsymbol{A}),(\boldsymbol{b},\boldsymbol{B})]$ need not be the same as $\mathrm{KL}[(\boldsymbol{b},\boldsymbol{B}),(\boldsymbol{a},\boldsymbol{A})]$. In fact, (9) requires $\boldsymbol{B}$ to be invertible but does not require $\boldsymbol{A}$ to be invertible. This is why we chose the matrix $\widehat{\boldsymbol{\Sigma}}^{(l)}_{raw}$ for $\boldsymbol{B}$ because it is invertible (its determinant is nonzero), whereas the entrywise median matrix $\widehat{\boldsymbol{\Sigma}}_{med}$ need not be.

7. Sort the deviations from lowest to highest and keep the first $q/2$ estimates. To simplify notation we pretend that these correspond to $l = 1, \ldots, q/2$. These are the block estimates closest to the robust summary $\widehat{\boldsymbol{\Sigma}}_{med}$. Since the $\widehat{\boldsymbol{\Sigma}}^{(l)}_{raw}$ are all positive definite we can now aggregate them. A simple way would be to average the matrices $\widehat{\boldsymbol{\Sigma}}^{(l)}_{raw}$ for $l = 1, \ldots, q/2$ and all the corresponding centers $\widehat{\boldsymbol{\mu}}^{(l)}_{raw}$.

Instead we can take the union of the corresponding h-subsets and compute its classical mean and covariance matrix. A faster way to do this is by a single-pass pooling method [20]. We initialize the sscp matrix $\boldsymbol{\Lambda}_{pooled}$ by $(m-1)\widehat{\boldsymbol{\Sigma}}^{(1)}_{raw}$ and $\widehat{\boldsymbol{\mu}}_{pooled}$ by $\widehat{\boldsymbol{\mu}}^{(1)}_{raw}$, and set $n_{pooled} = m$. Denoting the results from the next block by $(\widehat{\boldsymbol{\mu}}, \widehat{\boldsymbol{\Sigma}})$ we

(a) compute the difference in location $\widehat{\boldsymbol{\mu}}_\Delta = \widehat{\boldsymbol{\mu}} - \widehat{\boldsymbol{\mu}}_{pooled}$ and the sscp matrix $\boldsymbol{\Lambda} = (m-1)\widehat{\boldsymbol{\Sigma}}$.

(b) update the pooled sscp matrix, center and observation count by

$$\boldsymbol{\Lambda}_{pooled} \leftarrow \boldsymbol{\Lambda}_{pooled} + \boldsymbol{\Lambda} + \widehat{\boldsymbol{\mu}}_\Delta\widehat{\boldsymbol{\mu}}_\Delta^T \frac{n_{pooled}\, m}{n_{pooled} + m} \,,$$

$$\widehat{\boldsymbol{\mu}}_{pooled} \leftarrow \frac{n_{pooled}\, \widehat{\boldsymbol{\mu}}_{pooled} + m\, \widehat{\boldsymbol{\mu}}}{n_{pooled} + m} \,,$$

$$n_{pooled} \leftarrow n_{pooled} + m \,,$$

and we continue this way until all blocks have been pooled. We then put $\widehat{\boldsymbol{\Sigma}}_{raw}(\boldsymbol{Z}) := \boldsymbol{\Lambda}_{pooled}/(n_{pooled}-1)$.

8. Next we need to compute the reweighted MCD estimate $(\widehat{\boldsymbol{\mu}}_{rew}, \widehat{\boldsymbol{\Sigma}}_{rew})$ as described in Section 2. For this we compute the robust distances $\mathrm{RD}^{(l)}_i = d(\boldsymbol{z}^{(l)}_i, \widehat{\boldsymbol{\mu}}_{raw}, \widehat{\boldsymbol{\Sigma}}_{raw})$ for all blocks l and all cases $i = 1, \ldots, m$ in each. Doing this in the master thread would take too long, so we again distribute this computation over the threads. Each thread thus obtains a reweighted estimate $(\widehat{\boldsymbol{\mu}}^{(l)}_{rew}, \widehat{\boldsymbol{\Sigma}}^{(l)}_{rew})$.

9. The master thread receives all local weights and reweighted estimates, and combines them into the final overall reweighted estimate $(\widehat{\boldsymbol{\mu}}_{rew}, \widehat{\boldsymbol{\Sigma}}_{rew})$ by a pooling process similar to step 7 above.
10. Finally, each thread computes robust distances relative to the reweighted estimates and flags the outliers in parallel as those cases whose final robust distance $d(\boldsymbol{z}^{(l)}_i, \widehat{\boldsymbol{\mu}}_{rew}, \widehat{\boldsymbol{\Sigma}}_{rew})$ exceeds c_p .

The proposed aggregation strategy is depicted in Fig. 3.

Note that the final estimate $(\widehat{\boldsymbol{\mu}}^{(l)}_{rew}, \widehat{\boldsymbol{\Sigma}}^{(l)}_{rew})$ obtained at the end of step 9 can be used as a "warm start" input to step 3 in a subsequent run of the algorithm, when additional data require updating the result.

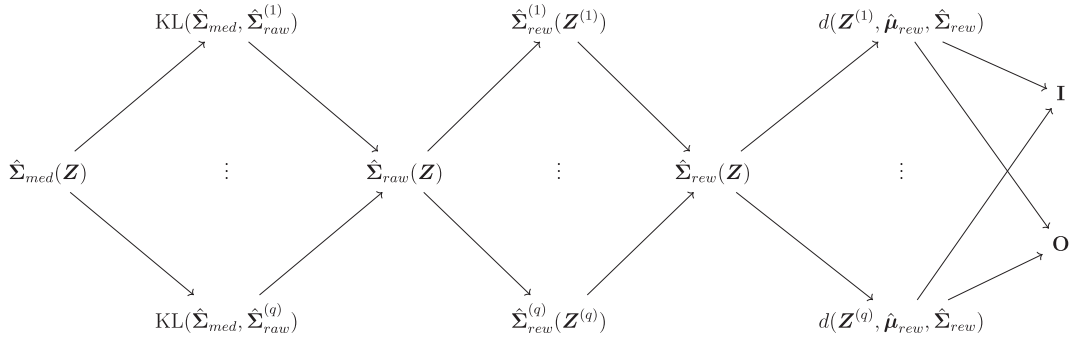

Fig. 3. Second part of the parallel processing topology of RT-DetMCD, responsible for the parallel aggregation (left), reweighting (middle) and the detection of outliers (right)

Table 1
The DetMCD algorithm and four increasingly modified versions.

Estimator	Section	Remark	Initial	Distance	C-steps	Parallelization
DetMCD	2	DetMCD	∘	∘	∘	∘
I	+ 3.1, 3.2, 3.3		•	∘	∘	∘
ID	+ 3.4		•	•	∘	∘
IDC	+ 3.5	Serial RT-DetMCD	•	•	•	∘
IDCP_q	+ 4	Parallel RT-DetMCD	•	•	•	•

5. Simulations

This section analyzes the statistical and computational performance of RT-DetMCD. We proposed three different algorithmic modifications in Section 3 and one in Section 4. Switching them on one after the other yields the five variations depicted in Table 1. The top row is DetMCD without any modifications. The next versions (rows) switch on modifications: new **I**nitial estimators (I), **D**istance calculation by Cholesky decomposition (D), update-based **C**-steps (C), and parallelization (P). Version IDC is the serial version of RT-DetMCD which does not require a parallel architecture. The parallel version of RT-DetMCD is abbreviated as IDCP_q where the subscript q denotes the number of blocks used. Comparing the computation times of the different versions is fair, as they share a common C++ codebase.

We will generate n cases from a p-variate Gaussian distribution $N(0, \boldsymbol{\Sigma})$ with center zero, where p is set to 4, 8 or 16 and n depends on the experiment. Without loss of generality we set the diagonal of $\boldsymbol{\Sigma}$ to 1. Since the methods under consideration are not affine equivariant we cannot just set $\boldsymbol{\Sigma}$ equal to the identity matrix. Instead we consider matrices $\boldsymbol{\Sigma}$ of different types:

1. The ALYZ covariance matrices are generated as in Section 4 of [23], yielding a different $\boldsymbol{\Sigma}$ in each replication. These matrices typically contain relatively weak correlations.
2. The A09 type is defined by $\boldsymbol{\Sigma}_{jk} = (-0.9)^{|j-k|}$ for $j, k = 1, \ldots, p$. This allows for some strong correlations.

Next, we replace εn random cases by outliers of different types, where ε denotes the fraction of contamination. *Shift contamination* was generated from $N(\boldsymbol{\mu}_C, \boldsymbol{\Sigma})$ where $\boldsymbol{\mu}_C$ lies in the direction where the outliers are hardest to detect, namely that of the last eigenvector $\boldsymbol{v}$ of the true covariance matrix $\boldsymbol{\Sigma}$. We rescale $\boldsymbol{v}$ to the typical size of a data point by making $\boldsymbol{v}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{v} = E[Y^2] = p$ where $Y^2 \sim \chi_p^2$. Finally $\boldsymbol{\mu}_C = \gamma\boldsymbol{v}$ in which γ

Table 2
Kullback-Leibler deviation and speedup for $\boldsymbol{\Sigma}$ of type A09.

	Point contamination			Shift contamination			Cluster contamination		
	$p = 4$	$p = 8$	$p = 16$	$p = 4$	$p = 8$	$p = 16$	$p = 4$	$p = 8$	$p = 16$
A: KL deviation									
$\varepsilon = 0.1$									
DetMCD	0.0226	0.0243	0.0266	0.0227	0.0242	0.0266	0.0229	0.0241	0.0266
I	0.0225	0.0246	0.0266	0.0228	0.0244	0.0265	0.0230	0.0241	0.0264
ID	0.0226	0.0248	0.0266	0.0227	0.0245	0.0266	0.0230	0.0242	0.0265
IDC	0.0227	0.0248	0.0262	0.0227	0.0243	0.0271	0.0230	0.0241	0.0266
IDCP_4	0.0233	0.0258	0.0280	0.0233	0.0258	0.0287	0.0245	0.0252	0.0280
$\varepsilon = 0.3$									
DetMCD	0.373	0.347	0.336	0.373	0.345	0.336	0.373	0.344	0.336
I	0.373	0.348	0.336	0.376	0.345	0.337	0.373	0.345	0.336
ID	0.373	0.345	0.336	0.373	0.347	0.336	0.373	0.344	0.336
IDC	0.372	0.348	0.336	0.373	0.345	0.338	0.373	0.343	0.338
IDCP_4	0.376	0.349	0.340	0.390	0.351	0.343	0.375	0.348	0.341
B: Speedup factor									
$\varepsilon = 0.1$									
I	90	102	244	195	215	222	74	231	304
ID	104	123	203	231	273	269	88	261	240
IDC	113	137	291	270	291	325	97	297	333
IDCP_4	115	148	291	357	376	350	112	295	323
$\varepsilon = 0.3$									
I	336	419	432	96	134	227	119	285	297
ID	408	481	500	119	149	265	146	312	325
IDC	478	516	572	126	161.91	297	166	296	336
IDCP_4	574	557	687	140	177	405	183	365	395

can be varied. *Cluster contamination* stems from $N(\boldsymbol{\mu}_C, 0.05^2\,\boldsymbol{I})$ where $\boldsymbol{I}$ is the identity matrix. Finally, *point contamination* places all outliers in the point $\boldsymbol{\mu}_C$ so they behave like a tight cluster. These settings make the simulation consistent with those in Refs. [8,24].

The distance of an estimated $\widehat{\boldsymbol{\Sigma}}$ to the true $\boldsymbol{\Sigma}$ is measured by the Kullback-Leibler deviation $\mathrm{KL}(\widehat{\boldsymbol{\Sigma}}, \boldsymbol{\Sigma})$ using (9) without the centers, that is,

$$\mathrm{KL}(\boldsymbol{A}, \boldsymbol{B}) = \mathrm{trace}\left(\boldsymbol{A}\boldsymbol{B}^{-1}\right) - p - \log\left(\det\left(\boldsymbol{A}\boldsymbol{B}^{-1}\right)\right) .$$

This measure was used in several other simulation studies such as [14, 23,24]. We will compare the accuracy of the new methods to that of DetMCD, and also compute the speedup factor as

$$\mathrm{speedup} = \mathrm{time(DetMCD)} \,/\, \mathrm{time(new\ \ method)} .$$

The first experiment has $n = 2^{16} = 65536$ observations in $p = 4, 8, 16$ dimensions. In all versions of MCD we set $\alpha = 0.5$ so $h \approx n/\,2$ observations are covered, which is the most robust choice. Table 2 is for $\boldsymbol{\Sigma}$ of type A09 and $\gamma = 50$. The scenarios are point contamination (left), shift contamination (middle) and cluster contamination (right), both for 10% and 30% of outliers. The top panel presents the KL deviations and the bottom panel reports the corresponding speedup factors, each averaged over 50 replications. Table 3 shows the same results for $\boldsymbol{\Sigma}$ of type ALYZ.

The DetMCD method is in the first row of all panels. The next row contains the I version, which modifies the original DetMCD algorithm by incorporating the new data standardization described in Subsection 3.1 and replacing the six initial estimators by the two new ones of Subsection 3.2. The I version is much faster than the original DetMCD as seen in its substantial speedup factors in both Tables 2 and 3. This is due to replacing six initial estimators (including a slower one) by two fast ones.

Note that the accuracy of the I version (as measured by the KL deviation) is as good as that of the slower DetMCD. In some instances with lower γ (not shown) the I version was actually more accurate than DetMCD. This improvement stems from using redescending techniques, which assign zero weights to observations that lie far away from the majority of data, as in (6) and (7). The standardization (Subsection 3.1) and the refinement procedure (Subsection 3.3) both use the univariate MCD, and the new initial estimators are based on wrapping and the linearly redescending GSSCM. This makes the proposed algorithm even more robust against contamination.

The next version (ID) switches on the numerically more stable distance computation by Cholesky decomposition, followed by the IDC version which also incorporates the updating mechanism. These versions do not change the KL deviation much, because both would be equivalent to version I if numerical precision were perfect. But the new implementations do improve the speedup factor. Overall IDC was faster than ID which in turn was faster than I, so each modification has contributed to the speedup.

When the sample size n is large we need to speed up the computation even more. This can be achieved by adding the parallel computation architecture of Section 4, yielding the IDCP version. Tables 2 and 3 show IDCP_4 which splits up the data into 4 blocks. This indeed improves the speedup factor. However, in some situations (here for $\varepsilon = 0.3$ in Table 3) the speedup is at the expense of a higher KL deviation, i.e. a loss of accuracy. This is due to the fact that the blocks have a lower sample size (here $n/4$), and for high p (here for $p = 16$) there are not always enough cases per dimension to provide an accurate estimate of the underlying covariance matrix.

We therefore need to choose the number of blocks carefully. Parallelization splits up the $n \times p$ dataset $\boldsymbol{X}$ into q blocks $\boldsymbol{X}^{(l)}$, each with $m = n/q$ observations. When choosing q we should take care that the blocks have enough observations per dimension to yield accurate estimates, so we impose

$$m \,/\, p \geq \omega$$

and we will try various choices of ω, starting from $2^{12} = 4096$. We only consider values of q that satisfy this condition, i.e. $q \leq n/(p\,\omega)$. In particular, if $n/p < \omega$ we will not parallelize. On the other hand we want to choose q as high as possible to obtain the best speedup. Combining these constraints yields the choice

$$q = \max\left(\frac{n}{p\,\omega}\,,\ 1\right) . \tag{10}$$

When this rule yields $q = 1$ we use the serial algorithm IDC. In practice, q is further bounded from above in terms of the available number of CPU cores.

In view of these considerations we carried out a new experiment with increasing total numbers of observations. We generated datasets with $n =$

Table 3
Kullback-Leibler deviation and speedup for $\boldsymbol{\Sigma}$ of type ALYZ.

	Point contamination			Shift contamination			Cluster contamination		
	$p = 4$	$p = 8$	$p = 16$	$p = 4$	$p = 8$	$p = 16$	$p = 4$	$p = 8$	$p = 16$
A: KL deviation									
$\varepsilon = 0.1$									
DetMCD	0.0227	0.0242	0.0265	0.0227	0.0241	0.0267	0.0229	0.0244	0.0264
I	0.0230	0.0242	0.0264	0.0233	0.0241	0.0269	0.0230	0.0244	0.0265
ID	0.0227	0.0247	0.0267	0.0233	0.0254	0.0268	0.0230	0.0244	0.0263
IDC	0.0228	0.0242	0.0272	0.0229	0.0247	0.0264	0.0229	0.0244	0.0264
IDCP_4	0.0236	0.0257	0.0286	0.0237	0.0326	0.0292	0.0237	0.0262	0.0283
$\varepsilon = 0.3$									
DetMCD	0.372	0.348	0.339	0.373	0.345	0.336	0.374	0.347	0.334
I	0.372	0.346	0.339	0.375	0.347	0.336	0.375	0.348	0.337
ID	0.372	0.345	0.339	0.373	0.346	0.336	0.373	0.348	0.335
IDC	0.372	0.347	0.339	0.373	0.345	0.337	0.373	0.348	0.335
IDCP_4	0.375	0.351	1.62	0.379	0.349	0.343	0.382	0.354	1.02
B: Speedup factor									
$\varepsilon = 0.1$									
I	83	197	238	176	158	238	203	227	183
ID	98	219	304	211	190	241	241	256	249
IDC	109	267	378	269	214	307	268	272	239
IDCP_4	134	330	464	347	219	395	325	360	349
$\varepsilon = 0.3_{\mathrm{SS}}$									
I	256	263	281	188	233	258	195	190	267
ID	318	301	318	216	270	281	239	223	297
IDC	347	326	364	241	286	315	264	236	323
IDCP_4	473	357	379	349	320	383	358	292	416

Table 4
Kullback-Leibler deviation and speedup factor for $\mathbf{\Sigma}$ of type ALYZ with fraction $\varepsilon = 0.3$ of point contamination, where the number of parallel blocks q is given by (10), for various dataset dimensions and values of ω.

n	$\omega = 2^{12} = 4096$			$\omega = 2^{13} = 8192$			$\omega = 2^{14} = 16384$		
	$p = 4$	$p = 8$	$p = 16$	$p = 4$	$p = 8$	$p = 16$	$p = 4$	$p = 8$	$p = 16$
A: KL deviation									
2^{10}	0.380	0.593	0.847	0.429	0.617	0.778	0.447	0.490	0.879
2^{11}	0.415	0.378	0.571	0.397	0.413	0.515	0.405	0.411	0.544
2^{12}	0.328	0.349	0.433	0.352	0.375	0.444	0.360	0.393	0.445
2^{13}	0.362	0.351	0.386	0.368	0.338	0.369	0.362	0.346	0.362
2^{14}	0.360	0.341	0.358	0.362	0.349	0.362	0.360	0.359	0.352
2^{15}	0.374	0.350	0.346	0.375	0.349	0.347	0.383	0.349	0.354
2^{16}	0.370	0.349	0.345	0.377	0.349	0.333	0.367	0.344	0.343
2^{17}	0.370	0.342	0.329	0.373	0.339	0.333	0.371	0.343	0.331
2^{18}	0.370	0.342	0.332	0.371	0.346	0.326	0.370	0.346	0.326
2^{19}	0.371	0.345	0.335	0.371	0.344	0.334	0.370	0.344	0.333
B: Speedup factor									
2^{10}	6.75	10.8	13.9	7.54	9.91	14.5	7.44	10.9	13.6
2^{11}	9.15	12.7	17.3	11.0	12.7	16.3	9.35	13.0	16.8
2^{12}	13.8	19.3	22.9	14.9	18.789	23.2	15.0	18.8	23.3
2^{13}	25.8	32.3	37.3	23.6	31.7	37.0	26.9	31.7	36.4
2^{14}	49.1	63.2	72.9	47.5	61.8	66.5	50.8	61.8	68.7
2^{15}	160	122	129	96.8	110	128	93.8	121	124
2^{16}	490	387	229	301	203	233	174	214	225
2^{17}	1190	1060	769	838	715	396	547	384	389
2^{18}	2490	2450	2150	2080	2010	1360	1680	1250	766
2^{19}	5020	5250	5140	4660	4730	3860	4090	3670	2610
C: Number of blocks									
2^{10}	1	1	1	1	1	1	1	1	1
2^{11}	1	1	1	1	1	1	1	1	1
2^{12}	1	1	1	1	1	1	1	1	1
2^{13}	1	1	1	1	1	1	1	1	1
2^{14}	1	1	1	1	1	1	1	1	1
2^{15}	2	1	1	1	1	1	1	1	1
2^{16}	4	2	1	2	1	1	1	1	1
2^{17}	8	4	2	4	2	1	2	1	1
2^{18}	16	8	4	8	4	2	4	2	1
2^{19}	32	16	8	16	8	4	8	4	2

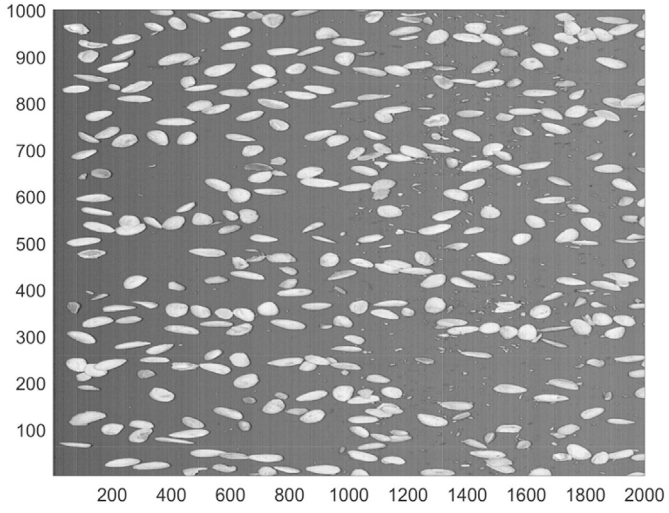

Fig. 4. 1000 × 2000 pixel region of the classifier training set. The image contains almonds as well as almond shells and dust.

$2^{10}, 2^{11}, \ldots, 2^{19}$ with $\mathbf{\Sigma}$ of type ALYZ and fraction $\varepsilon = 0.3$ of point contamination with $\gamma = 35$. We let ω range from 2^{12} to 2^{14}. Table 4 summarizes the results, with the same panels for the KL deviation and speedup as before. The bottom panel shows the number of blocks q as determined from (10), noting that it is 1 for the smaller sample sizes n.

In Table 4 we see that the KL deviation remained stable over all dataset sizes. This indicates that provided q is chosen by (10), i.e. the blocks have enough observations per dimension, the accuracy of parallel RT-DetMCD is comparable to that of the serial version. At the same time the parallel version achieves much higher speedup factors than the serial version. We also note that the estimation accuracy was rather stable across the three values of ω considered. It thus appears that $\omega = 2^{12}$ (which yields the best speedup factors) is a reasonable default choice.

6. Industrial application of RT-DetMCD

Industrial food inspection machines scan millions of individual objects per hour, yielding faster and more accurate results than manual inspection. Mechanical sorting boosts the processing capacity of a production line, enabling the food producer to simultaneously provide consistent food quality and safety guarantees. We illustrate the feasibility of anomaly detection by RT-DetMCD in this context. The example is an almond inspection setting, where the machine measures the object response on $p = 4$ wavelengths using a line scan image acquisition system. Each incoming scan line consists of 4096 pixels and has to be classified within milliseconds to comply with the production throughput. The goal is the adequate detection of foreign material (such as shells, hulls, wood, stones and pieces of glass) between the almonds, so the foreign material can be removed in real time.

We use the RT-DetMCD method for unsupervised classification. This is considerably different from the customary classification setting, where training sets from each individual product must first be analyzed carefully by hand in order to assign its objects to different types of material. Instead, we assume that the training sets are contaminated by defects, that is, outliers.

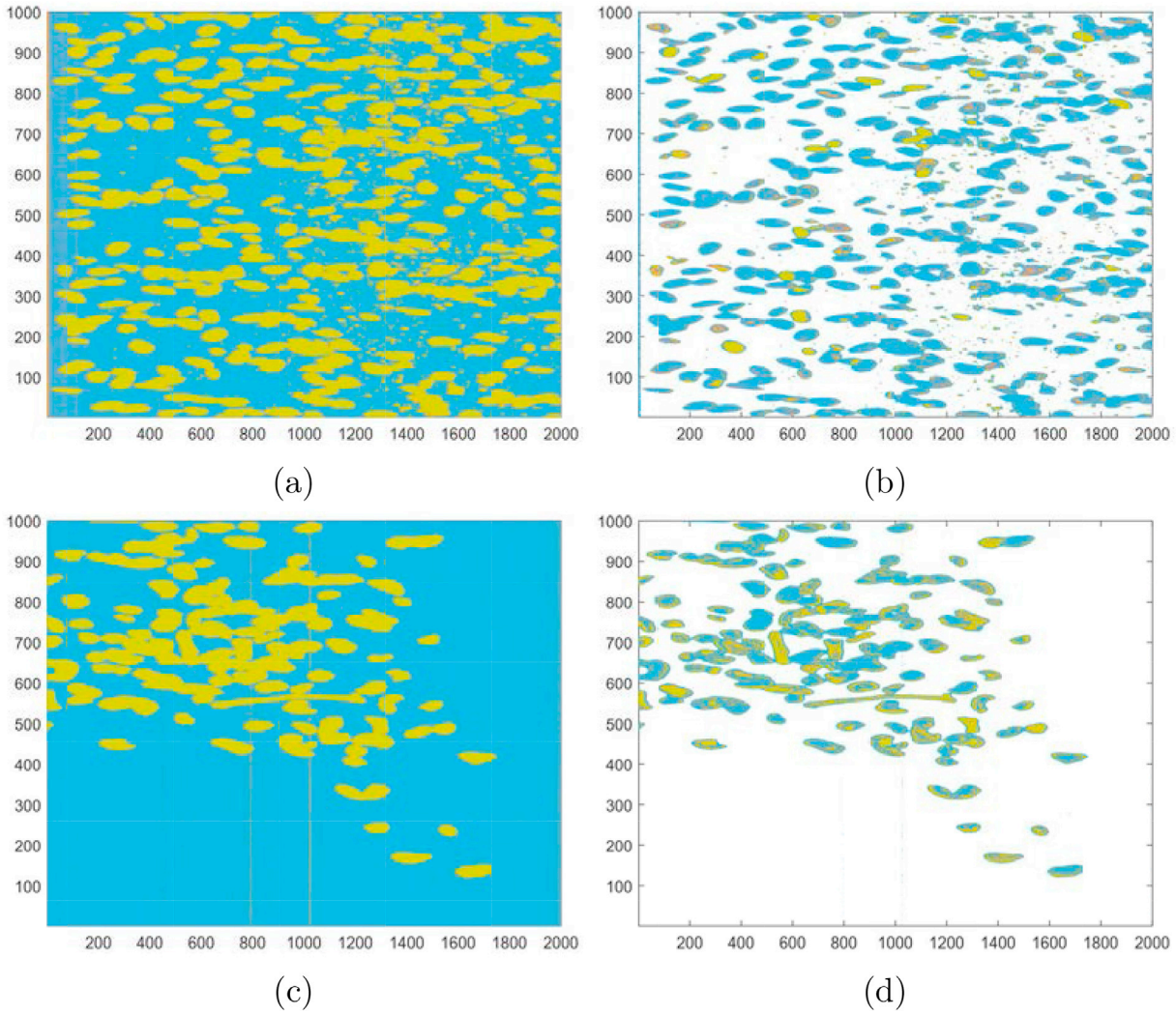

Fig. 5. Industrial almond dataset: (a) segmenting the training dataset of Fig. 4 into foreground and background by RT-DetMCD with foreground shown in yellow; (b) detecting outliers among the foreground pixels reveals foreign material shown in yellow; correctly detected foreground (c) and defects (d) in a test dataset. (For interpretation of the references to color in this figure legend, the reader is referred to the Web version of this article.)

In the example the training set consists of 2048 sequentially stacked scan lines of 4096 pixels which captured the incoming product flow, totaling over 8 million observations (pixels) with $p = 4$ dimensions each. The first dimension of the dataset is visualized in black and white in Fig. 4. All the images of this example were clipped to a region of interest of 1000×2000 pixels so the image resolution can be rendered here.

We first extract the relevant foreground objects by training parallelRT-DetMCD on all eight million observations, yielding a fit $(\widehat{\boldsymbol{\mu}}_1, \widehat{\boldsymbol{\Sigma}}_1)$. As the majority of these observations consist of background (i.e. the dark pixels in Fig. 4), RT-DetMCD identified the foreground material as anomalies, shown in Fig. 5a. Next, RT-DetMCD was trained on the 3 127 973 foreground objects, yielding a fit $(\widehat{\boldsymbol{\mu}}_2, \widehat{\boldsymbol{\Sigma}}_2)$ in seconds, which revealed non-almond material (Fig. 5b). Closer inspection showed that entire shells were adequately detected as outliers, as well as almond discolorations and damaged almond skins.

The next task was to classify a variety of unknown material in a test dataset, i.e. a previously unseen image of material. This was achieved by computing robust distances of new observations from the existing fit, and checking when they exceed the cutoff. The computation was done in parallel, using the third part of the flowchart in Fig. 3 corresponding to step 10 in the algorithm in Section 4. This construction forms an anomaly detector that uses the fits trained on the image shown in Fig. 4. The robust distances from the background segmentation fit $(\widehat{\boldsymbol{\mu}}_1, \widehat{\boldsymbol{\Sigma}}_1)$ performed as expected, detecting all foreground material on the fly (Fig. 5c). It also revealed the presence of water droplets on the image acquisition lens, seen as vertical stripes around columns 800 and 1000. Presented with the foreground objects, the second detector based on $(\widehat{\boldsymbol{\mu}}_2, \widehat{\boldsymbol{\Sigma}}_2)$ revealed all non-almond material (e.g. almond tree wood), with the output shown in Fig. 5d.

Segmenting the entire new image (the test dataset) with over 8 million observations into background and foreground only took 8.4 ms, whereas segmenting the approximately 3 million foreground cases took 3.3 ms.

Note that in industrial settings the computation speed of RT-DetMCD is an important advantage since it means that the classifier can be re-trained quickly, even on-the-fly whenever new data are observed. In this particular application it was sufficient to run RT-DetMCD at regular intervals.

7. Conclusions and outlook

Real-time industrial processes are very demanding in terms of computation speed. Often the detection of anomalies is of crucial importance, e.g. for food sorting machines that need to remove foreign material on the fly. This paper focused on anomaly detection by robust estimation using the minimum covariance determinant (MCD) approach.

Although the existing DetMCD algorithm is fast enough for off-line statistical analysis, it cannot cope with the huge sample sizes and stringent speed requirements of industrial processes. Therefore we constructed an improved method called RT-DetMCD by incorporating several new ideas, resulting in high speedup factors without loss of accuracy. A major speedup is obtained by parallel processing, which splits up the data into blocks that are analyzed separately. Combining these results into an overall fit required the development of a novel aggregation approach.

The performance of RT-DetMCD was studied by simulation, which showed that each improvement contributed to the overall speedup. Its ability to handle real-time industrial processes was illustrated by a case study on the automated sorting of almonds. The industrial C++ code of RT-DetMCD used in the simulation and application is proprietary, but a research-level Matlab version which mimics its results is available from the webpage http://wis.kuleuven.be/statdatascience/robust/software.

The output of the new RT-DetMCD technique can be used as a basis for other multivariate techniques such as robust principal component analysis and classification in industrial settings.

Declaration of competing interest

The authors have no conflicts of interest.

Acknowledgements

We thank Johan Speybrouck for providing the industrial datasets and Tim Wynants for his support throughout the project. We also acknowledge the financial support of VLAIO grant HBC.2016.0208 as well as project C16/15/068 of Internal Funds KU Leuven.

Appendix A. Supplementary data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.chemolab.2020.103957.

References

[1] M. Daszykowski, K. Kaczmarek, Y. Vander Heyden, B. Walczak, Robust statistics in data analysis - a review: basic concepts, Chemometr. Intell. Lab. Syst. 85 (2007) 203–219.
[2] M. Hubert, M. Debruyne, P.J. Rousseeuw, Minimum covariance determinant and extensions, Wiley Interdiscip. Rev.: Comput. Stat. 10 (3) (2018) e1421.
[3] P.J. Rousseeuw, M. Debruyne, S. Engelen, M. Hubert, Robustness and outlier detection in chemometrics, Crit. Rev. Anal. Chem. 36 (2006) 221–242.
[4] P.J. Rousseeuw, A. Leroy, Robust Regression and Outlier Detection, Wiley-Interscience, New York, 1987.
[5] P.J. Rousseeuw, Least median of squares regression, J. Am. Stat. Assoc. 79 (1984) 871–880.
[6] P.J. Rousseeuw, Multivariate estimation with high breakdown point, in: W. Grossmann, G. Pflug, I. Vincze, W. Wertz (Eds.), Mathematical Statistics and Applications, B, Reidel Publishing Company, Dordrecht, 1985, pp. 283–297.
[7] P.J. Rousseeuw, K. Van Driessen, A fast algorithm for the Minimum Covariance Determinant estimator, Technometrics 41 (1999) 212–223.
[8] M. Hubert, P.J. Rousseeuw, T. Verdonck, A deterministic algorithm for robust location and scatter, J. Comput. Graph Stat. 21 (2012) 618–637.
[9] J. Zhu, Z. Ge, Z. Song, F. Gao, Review and big data perspectives on robust data mining approaches for industrial process modeling with outliers and missing data, Annu. Rev. Contr. 46 (2018) 107–133.
[10] M. Hubert, P.J. Rousseeuw, K. Vanden Branden, ROBPCA: a new approach to robust principal component analysis, Technometrics 47 (2005) 64–79.
[11] C. Croux, G. Haesbroeck, Influence function and efficiency of the Minimum Covariance Determinant scatter matrix estimator, J. Multivariate Anal. 71 (1999) 161–190.
[12] J. Raymaekers, P.J. Rousseeuw, I. Vranckx, Discussion of "The power of monitoring: how to make the most of a contaminated multivariate sample", Stat. Methods Appl. 27 (2018) 589–594.
[13] J. Raymaekers, P.J. Rousseeuw, Fast robust correlation for high dimensional data, Technometrics (2019), https://doi.org/10.1080/00401706.2019.1677270.
[14] J. Raymaekers, P.J. Rousseeuw, A generalized spatial sign covariance matrix, J. Multivariate Anal. 171 (2019) 94–111.
[15] S. Visuri, V. Koivunen, H. Oja, Sign and rank covariance matrices, J. Stat. Plann. Inference 91 (2000) 557–575.
[16] R. Maronna, R. Zamar, Robust estimates of location and dispersion for high-dimensional data sets, Technometrics 44 (2002) 307–317.
[17] J.-H. Won, J. Lim, S.-J. Kim, B. Rajaratnam, Condition-number-regularized covariance estimation, J. Roy. Stat. Soc. B 75 (2013) 427–450.
[18] M. Lira, R. Iyer, A. Trindade, V. Howle, QR versus Cholesky: a probabilistic analysis, Int. J. Numer. Anal. Model. 13 (2016) 114–121.
[19] N.J. Higham, Fortran codes for estimating the one-norm of a real or complex matrix, with applications to condition estimation, ACM Trans. Math Software 14 (1988) 381–396.
[20] J. Bennett, R. Grout, P. Pébay, D. Roe, D. Thompson, Numerically stable, single-pass, parallel statistics algorithms, IEEE Int. Conf. Cluster Comput. (2009) 1–8.
[21] C. Hertzog, On pooling covariance matrices for multivariate analysis, Educ. Psychol. Meas. 46 (1986) 349–352.
[22] M. Riani, D. Perrotta, A. Cerioli, The forward search for very large datasets, J. Stat. Software 67 (2015) 1–20.
[23] C. Agostinelli, A. Leung, V.J. Yohai, R.H. Zamar, Robust estimation of multivariate location and scatter in the presence of cellwise and casewise contamination, Test 24 (2015) 441–461.
[24] K. Boudt, P.J. Rousseeuw, S. Vanduffel, T. Verdonck, The minimum regularized covariance determinant estimator, Stat. Comput. 30 (1) (2020) 113–128.