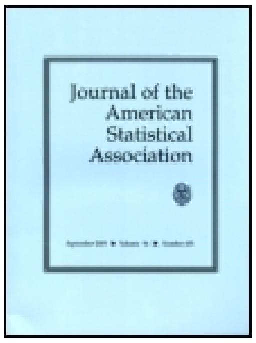


Journal of the American Statistical Association

ISSN: 0162-1459 (Print) 1537-274X (Online) Journal homepage: http://amstat.tandfonline.com/loi/uasa20

Robust Improper Maximum Likelihood: Tuning, Computation, and a Comparison With Other Methods for Robust Gaussian Clustering

Pietro Coretto & Christian Hennig

To cite this article: Pietro Coretto & Christian Hennig (2016) Robust Improper Maximum Likelihood: Tuning, Computation, and a Comparison With Other Methods for Robust Gaussian Clustering, Journal of the American Statistical Association, 111:516, 1648-1659, DOI: 10.1080/01621459.2015.1100996

To link to this article: http://dx.doi.org/10.1080/01621459.2015.1100996

JOURNAL OF THE AMERICAN STATISTICAL ASSOCIATION
2016, VOL. 111, NO. 516, 1648–1659, Theory and Methods
http://dx.doi.org/10.1080/01621459.2015.1100996

Taylor & Francis
Taylor & Francis Group

Robust Improper Maximum Likelihood: Tuning, Computation, and a Comparison With Other Methods for Robust Gaussian Clustering

Pietro Coretto [a] and Christian Hennig [b]

[a]Department of Economics and Statistics, University of Salerno, Fisciano, Italy; [b]Department of Statistical Science, University College London, London, UK

ABSTRACT
The two main topics of this article are the introduction of the "optimally tuned robust improper maximum likelihood estimator" (OTRIMLE) for robust clustering based on the multivariate Gaussian model for clusters, and a comprehensive simulation study comparing the OTRIMLE to maximum likelihood in Gaussian mixtures with and without noise component, mixtures of t-distributions, and the TCLUST approach for trimmed clustering. The OTRIMLE uses an improper constant density for modeling outliers and noise. This can be chosen optimally so that the nonnoise part of the data looks as close to a Gaussian mixture as possible. Some deviation from Gaussianity can be traded in for lowering the estimated noise proportion. Covariance matrix constraints and computation of the OTRIMLE are also treated. In the simulation study, all methods are confronted with setups in which their model assumptions are not exactly fulfilled, and to evaluate the experiments in a standardized way by misclassification rates, a new model-based definition of "true clusters" is introduced that deviates from the usual identification of mixture components with clusters. In the study, every method turns out to be superior for one or more setups, but the OTRIMLE achieves the most satisfactory overall performance. The methods are also applied to two real datasets, one without and one with known "true" clusters. Supplementary materials for this article are available online.

ARTICLE HISTORY
Received July 2014
Revised September 2015

KEYWORDS
Cluster analysis; EM-algorithm; Improper density; Mixture models; Model-based clustering; Robustness

1. Introduction

In this article, we introduce and investigate the "optimally tuned robust improper maximum likelihood estimator" (OTRIMLE), a method for robust clustering with clusters that can be approximated by multivariate Gaussian distributions. Its one-dimensional version was introduced in Coretto and Hennig (2010). We also present a simulation study comparing OTRIMLE and other approaches for (mostly robust) model-based clustering, which is, to our knowledge, the most comprehensive study in the field and involves a careful discussion of the issue of comparing methods based on different model assumptions.

The basic idea of OTRIMLE is to fit an improper density to the data that is made up by a Gaussian mixture density and a "pseudo mixture component" defined by a small constant density, which is meant to capture outliers in low density areas of the data. This is inspired by the addition of a uniform "noise component" to a Gaussian mixture (Banfield and Raftery 1993). Hennig (2004) showed that using an improper density improves the breakdown robustness of this approach. The OTRIMLE has been found to work well for one-dimensional data in Coretto and Hennig (2010).

As in many other statistical problems, violations of the model assumptions and particularly outliers may cause problems in cluster analysis. Our general attitude to the use of statistical models in cluster analysis is that the models should not be understood as reflecting some underlying but in practice unobservable "truth," but rather as thought constructs implying a certain behavior of methods derived from them (e.g., maximizing the likelihood), which may or may not be appropriate in a given application (more details on the general philosophy of clustering can be found in Hennig and Liao 2013). Using a model such as a mixture of multivariate Gaussian distributions, interpreting every mixture component as a "cluster," implies that we look for clusters that are approximately "Gaussian-shaped," but we do not want to rely on whether the data really were generated iid by a Gaussian mixture. We are interested in the performance of such methods in situations where one may legitimately look for Gaussian-shaped clusters, even if some data points do not belong to such clusters (called "noise" in the following), and even if the clusters are not precisely Gaussian. This reflects the fact that in practice, for example, mixtures of t-distributions are used for clustering the same datasets to which Gaussian mixtures are fitted as well, interpreting the resulting clusters in the same way.

For illustration of the outlier problem in model-based clustering, we use a five-dimensional dataset in which the 170 districts of the German city of Dortmund are characterized by a number of variables, which is discussed in detail in Section 6.1. Fitting a plain Gaussian mixture with $G = 4$ to all five variables by R's MCLUST package (Fraley et al. 2012), one cluster is a one-point cluster consisting only of an extreme outlier, and two further clusters fit two different varieties of moderate outliers. More

CONTACT Pietro Coretto pcoretto@unisa.it Department of Economics and Statistics, University of Salerno, 84084 Fisciano SA, Italy.
Color versions of one or more of the figures in the article can be found online at www.tandfonline.com/r/JASA.
Supplementary materials for this article are available online. Please go to www.tandfonline.com/r/JASA.

than 120 districts are collected in a single cluster. The task of robust clustering is to avoid having many or even most clusters dominated by outliers, and to produce a meaningful clustering structure also among the main bulk of nonextreme observations.

A number of model-based clustering methods that can deal with outliers have been proposed in recent years. An overview of these methods is given in Section 2. The OTRIMLE is introduced and discussed in Section 3, starting from the "RIMLE," in which the level of the improper constant density is a tuning constant. We then introduce a method for optimal tuning and discuss its computation. Section 5 presents a simulation study that uses a unified approach for defining elliptically shaped clusters with noise/outliers, presented in Section 4. The Dortmund dataset mentioned above is discussed in Section 6 along with a dataset of folk song melodies from two known regions. Some further issues including the estimation of the number of clusters are discussed in Section 7. Additional details about the example dataset (including full scatterplots), the simulation study, and computation of methods are provided in an online supplement (Coretto and Hennig in press b). Theoretical properties of the RIMLE with the tuning constant fixed are investigated in Coretto and Hennig (in press a) and cited here.

2. Methods from the Literature

In the following, assume an observed sample $\underline{x}_n = \{x_1, x_2, \dots, x_n\}$, where x_i is the realization of a random variable $X_i \in \mathbb{R}^p$ with $p \geq 1$; $X_1, \dots, X_n$ iid. The goal is to cluster the sample points into G distinct groups.

2.1. *Maximum Likelihood (ML) for Gaussian Mixtures (gmix)*

Let $\phi(x; \mu, \Sigma)$ be the density of a multivariate Gaussian distribution with mean vector $\mu \in \mathbb{R}^p$ and $p \times p$ covariance matrix Σ. Assume that the observed sample is iid drawn from the finite Gaussian mixture distribution having density

$$m(x; \theta) = \sum_{j=1}^{G} \pi_j \phi(x; \mu_j, \Sigma_j), \tag{2.1}$$

where $\pi_j \in [0, 1]$ for all $j = 1, 2, \dots, G$ and $\sum_{j=1}^{G} \pi_j = 1$, θ is the parameter vector containing the triplets π_j, μ_j, Σ_j for all $j = 1, 2, \dots, G$. Clustering coincides with assigning points to the mixture components based on ML parameter estimates. π_j can be interpreted as the expected proportion of points originated from the jth component. Let θ_n^{ml} be the ML estimator for θ, usually computed by the expectation–maximization (EM) algorithm (Dempster et al. 1977). The ML estimator under (2.1) exists only under appropriate constraints on the covariances matrices. These constraints (which are also relevant for the methods introduced below) will be discussed in detail in Section 3. Let τ_{ij}^{ml} be the estimated posterior probability that the observed point x_i has been drawn from the jth mixture component, that is,

$$\tau_{ij}^{\text{ml}} = \frac{\pi_{j,n}^{\text{ml}} \phi(x_i; \mu_{j,n}^{\text{ml}}, \Sigma_{j,n}^{\text{ml}})}{m(x_i; \theta_n^{\text{ml}})} \qquad \text{for all} \qquad j = 1, 2, \dots, G. \tag{2.2}$$

The point x_i can then be assigned to the kth cluster if $k = \arg\max_{j=1,2,\dots,G} \tau_{ij}^{\text{ml}}$. This assignment method is common to all model-based clustering methods introduced here. gmix is implemented in R's MCLUST package (Fraley et al. 2012). As illustrated in Section 1 and proven in Hennig (2004), the method can be strongly affected by outliers and deviations from the model assumptions, and we now turn to approaches that attempt to deal with this problem. For lack of space, we present these in detail in the online supplement and only give a short overview here.

2.2. *ML-Type Estimator for Gaussian Mixtures with Uniform Noise (gmix.u)*

Banfield and Raftery (1993) added a uniform mixture component on the smallest hyperrectangle covering the data to (2.1), calling it "*noise component*" to accommodate "noise."

2.3. *ML for Mixtures of Student t-Distributions (tmix)*

McLachlan and Peel (2000) replaced the Gaussian densities in (2.1) with multivariate Student-t densities, because they have heavier tails and can therefore accommodate outliers in a better way. Observations can be declared "noise" if they lie in a low density area of the t-distribution fitting their cluster. Hennig (2004) showed that neither tmix nor gmix.u is breakdown-robust.

2.4. *TCLUST*

TCLUST is based on maximizing a trimmed likelihood of a "fixed partition model" with cluster weights π_j. With $R = \cup_{j=1}^{G} R_j$, $\#\{R\} = [n(1-\alpha)]$ the number of nontrimmed points:

$$\theta^{\text{tclust}} := \underset{\theta\in\Theta, \#\{R\}=[n(1-\alpha)]}{\arg\max} \sum_{j=1}^{G} \sum_{i\in R_j} \left(\log \pi_j + \log \phi(x; \mu_j, \Sigma_j)\right). \tag{2.3}$$

For background, see Gallegos (2002), Gallegos and Ritter (2005), and García-Escudero et al. (2008). The TCLUST methodology is implemented in R's TCLUST package by Fritz, García-Escudero, and Mayo-Iscar (2012). Partition methods with trimming started with the trimmed k-means proposal of Cuesta-Albertos, Gordaliza, and Matrán (1997).

2.5. *Further Existing Work*

More approaches to robust model-based clustering can be found in the literature. Neykov et al. (2007) proposed and implemented a trimmed likelihood method. Qin and Priebe (2013) introduced an EM-algorithm adapted to maximum L_q-likelihood estimation and studied its behavior under a gross error model. References to other approaches to robust clustering are given in García-Escudero et al. (2010).

3. Optimally Tuned Robust Improper Maximum Likelihood

3.1. *Robust Improper Maximum Likelihood*

The robust improper maximum likelihood estimator (RIMLE) is based on the "noise component"-idea for robustification (gmix.u). The main idea is to use a pseudo-model where the noise is represented by an improper constant density over the whole Euclidean space:

$$\psi_\delta(x, \theta) = \pi_0\delta + \sum_{j=1}^{G} \pi_j \phi(x; \mu_j, \Sigma_j), \tag{3.1}$$

with $\pi_0, \pi_j \in [0, 1]$ for $j = 1, 2, \ldots, G$, $\pi_0 + \sum_{i=1}^{G} \pi_j = 1$. $\delta > 0$ is the improper constant density (icd). The parameter vector θ contains all Gaussian parameters plus all proportion parameters including π_0, i.e., $\theta = (\pi_0, \pi_1, \ldots, \pi_G, \mu_1, \ldots, \mu_G, \Sigma_1, \ldots, \Sigma_G)$. Given the sample improper pseudo-log-likelihood function

$$l_n(\theta) = \frac{1}{n}\sum_{i=1}^{n} \log \psi_\delta(x_i, \theta), \tag{3.2}$$

the RIMLE is defined as

$$\theta_n(\delta) = \arg\max_{\theta \in \Theta} l_n(\theta), \tag{3.3}$$

where Θ is an appropriate constrained parameter space discussed below. $\theta_n(\delta)$ is then used to cluster points as for model-based clustering methods. Define pseudo posterior probabilities in analogy with (2.2):

$$\tau_j(x_i, \theta) := \begin{cases} \dfrac{\pi_0\delta}{\psi_\delta(x_i, \theta)} & \text{if } j = 0 \\ \dfrac{\pi_j\phi(x_i, \mu_j, \Sigma_j)}{\psi_\delta(x_i, \theta)} & \text{if } j = 1, 2, \ldots, G; \end{cases} \quad \text{for } i = 1, 2, \ldots, n,$$

and assign the points based on the following rule

$$J(x_i, \theta) := \underset{j \in \{0,1,2,\ldots,G\}}{\arg\max}\ \tau_j(x_i, \theta). \tag{3.4}$$

Fixing δ, (3.1) does not define a proper probability model, but (3.3) yields a useful procedure for data modeled as a proportion of $(1 - \pi_0)$ of a mixture of Gaussian distributions plus a proportion of π_0 points not assigned to any meaningful cluster. Regions of high density are rather associated with clusters than with noise, so the noise regions should be those with the lowest density. This could be achieved by using the uniform density as in gmix.u, but for this the presence of gross outliers the dependence of the uniform distribution on the convex hull of the data still causes a robustness problem (Hennig 2004) .

The optimization problem in (3.3) requires that Θ is suitably defined, otherwise $\theta_n(\delta)$ may not exist. As discovered by Day (1969), the Gaussian mixture likelihood can degenerate. This problem extends to (3.1) as well. Let $\lambda_{k,j}$ be an eigenvalue of Σ_j for some $k = 1, 2, \ldots, p$ and $j = 1, 2, \ldots, G$. Take a sequence $(\theta_m)_{m\in\mathbb{N}}$ such that $\lambda_{k,j,m} \searrow 0$ and $\mu_{j,m} = x_1$, then $l_n(\theta_m) \to +\infty$. There are various ways to avoid this issue. Let $\lambda_{\max}(\theta)$ and $\lambda_{\min}(\theta)$ be, respectively, the maximum and the minimum eigenvalues of the covariance matrices in θ, Coretto and Hennig (in press a) adopted the "eigenratio constraint"

$$\lambda_{\max}(\theta)/\lambda_{\min}(\theta) \le \gamma < +\infty \tag{3.5}$$

with fixed $\gamma \ge 1$. $\gamma = 1$ constrains all component covariance matrices to be spherical and equal, as in k-means clustering, while $\gamma > 1$ restricts the relative scatter discrepancy among clusters. This type of constraint has been proposed by Dennis (1981) and studied by Hathaway (1985) for one-dimensional Gaussian mixtures. EM-algorithms for computing the ML of multivariate Gaussian mixtures under (3.5) have been studied by Ingrassia (2004) and Ingrassia and Rocci (2007), although asymptotic properties of the corresponding MLE have not been proved. The same constraints are used for TCLUST by García-Escudero et al. (2008). There are a number of alternative constraints, see Ingrassia and Rocci (2011) and Gallegos and Ritter (2009).

Although (3.5) prevents the unboundness of the likelihood in standard mixture models and TCLUST, for RIMLE this is not enough. Points not fitted by any of the Gaussian components can still be fitted by the improper uniform component. Therefore, Coretto and Hennig (in press a) proposed an additional "noise proportion constraint,"

$$\frac{1}{n}\sum_{i=1}^{n} \tau_0(x_i, \theta) \le \pi_{\max}, \tag{3.6}$$

for fixed $0 < \pi_{\max} < 1$. The quantity $n^{-1}\sum_{i=1}^{n} \tau_0(x_i, \theta)$ can be interpreted as the estimated proportion of noise points. Setting $\pi_{\max} = 0.5$ just implements a familiar condition in robust statistics that at most half of the data should be classified as "outliers/noise." The resulting restricted parameter space for RIMLE is then

$$\Theta := \left\{\theta:\ \pi_j \ge 0\ \forall j \ge 1;\ \pi_0 + \sum_{j=1}^{G}\pi_j = 1;\ \frac{1}{n}\sum_{i=1}^{n}\tau_0(x_i, \theta) \le \pi_{\max};\ \frac{\lambda_{\max}(\theta)}{\lambda_{\min}(\theta)} \le \gamma\right\}. \tag{3.7}$$

Coretto and Hennig (in press a) showed that $\theta_n(\delta)$ exists for any $\delta \ge 0$ if $\#(\underline{x_n}) > G + \lceil n\pi_{\max}\rceil$ and that $\theta_n(0)$ exists under the milder condition that $\#(\underline{x_n}) > G$. For $\delta = 0$, the RIMLE reduces to ML for plain Gaussian mixtures. Let $\mathrm{E}_P f(x)$ be the expectation of $f(x)$ under $x \sim P$. The RIMLE functional is defined as

$$\theta^\star(\delta) = \arg\max_{\theta \in \Theta_G} E_P \log \psi_\delta(x; \theta). \tag{3.8}$$

Existence of (3.8), consistency of $\theta_n(\delta)$ on the quotient space topology identifying all log-likelihood maxima and its breakdown point are shown in Coretto and Hennig (in press a).

3.2. Optimal Improper Density Level

Occasionally, subject matter knowledge may be available aiding the choice of δ, but such situations are rather exceptional. Here we suggest a data-dependent choice of δ. Note that δ is not treated as a model quantity to be estimated here, but rather as a tuning device to enable a good robust clustering. The aim of the RIMLE is to approximate the density of clustered regions of points when these regions look like those produced by a Gaussian distribution. We define the "optimal" δ value as minimizer of a criterion function measuring the discrepancy of the found clusters from the Gaussian prototype. Given $\theta_n(\delta)$, define the clusterwise squared Mahalanobis distances to clusters' centers as

$$d_{i,j,n} = (x_i - \mu_{j,n})'\Sigma_{j,n}^{-1}(x_i - \mu_{j,n}),$$

and the clusterwise weighted empirical distribution of $d_{i,j,n}$,

$$\mathbb{M}_j(t;\delta) = \frac{1}{\sum_{i=1}^n \tau_j(x_i, \theta_n(\delta))} \sum_{i=1}^n \tau_j(x_i, \theta_n(\delta)) \mathbf{1}\{d_{i,j,n} \leq t\}, \quad j = 1, 2, \ldots, G. \tag{3.9}$$

In $\mathbb{M}_j$, the ith point's distance is weighted according to the pseudo posterior probability that the ith observation has been generated by the jth mixture component. If the jth cluster is approximately Gaussian and $\mu_{j,n}$ and $\Sigma_{j,n}$ are good approximations of its location and scatter, we expect that squared Mahalanobis distances to $\mu_{j,n}$ of the points indeed belonging to mixture component no. j (for which $\tau_j(\cdot)$ indicates the estimated probability) will approximate a χ^2_p distribution. With $\chi^2_p(a)$ being the value of the cdf of the χ^2_p distribution at a, define the Kolmogorov-type distance for the jth cluster

$$\mathrm{K}_j(\delta) = \max_{i=1,\ldots,n} \left| \mathbb{M}_j(d_{i,j,n};\delta) - \chi^2_p(d_{i,j,n}) \right|. \tag{3.10}$$

The quality of the overall Gaussian approximation is then evaluated by weighting $\mathrm{K}_j(\cdot)$ with the estimated component proportion $\pi_{j,n}$:

$$\mathrm{D}(\delta) = \frac{1}{\sum_{j=1}^G \pi_{j,n}} \sum_{j=1}^G \pi_{j,n} \mathrm{K}_j(\delta). \tag{3.11}$$

For a given constant $\beta \geq 0$, define the optimal icd level as

$$\delta_n = \underset{\delta \in [0, \delta_{\max}]}{\arg\min} \left[\mathrm{D}(\delta) + \beta \pi_{0,n} \right]. \tag{3.12}$$

The corresponding optimally tuned RIMLE (OTRIMLE) will be denoted as $\theta_n(\delta_n)$. Existence and uniqueness of δ_n are not trivial, see Section 3.3. Section 5.4 is about how $D(\delta)$ behaves as a function of δ.

The straightforward choice for β, formalizing the "Gaussian cluster" concept, is $\beta = 0$. However, often in practice it is not so important that the clusters are of as precise Gaussian shape as possible. $\beta > 0$ (but normally smaller than 1) formalizes that less Gaussian shapes of clusters are tolerated if this brings the estimated noise proportion down. As can be seen in Section 5, choosing $\beta = \frac{1}{3}$ leads to improvements if the true clusters are t-distributed. Section 5.5 gives more details on the effect of choosing $\beta > 0$.

3.3. Computation

For a fixed δ, the RIMLE can be appropriately computed using an expectation–maximization (EM) algorithm. See Coretto and Hennig (in press a) for details, and the online supplement for details and background of the following. The outcome of the EM-algorithm depends on the initialization. We used an initialization method inspired by the MCLUST software. To avoid spurious clusters, we consider as valid initial partitions only those containing at least `min.pr`$\times n$ observations in each cluster (`min.pr`=0.005, say). As a first attempt to find such a valid partition, nearest neighbor-based clutter/noise detection proposed by Byers and Raftery (1998) is applied to identify an initial noise guess. Agglomerative hierarchical clustering based on ML criteria for Gaussian mixture models proposed by Banfield and Raftery (1993) is then used for finding initial Gaussian clusters among the nonnoise. See the online supplement for the case that the found partition is not "valid."

The OTRIMLE can be found by computing RIMLEs on a grid of δ values ranging from zero to some large enough $\delta_{\max}$. In practice, we solve the program (3.12) by the "golden section search" of Kiefer (1953) over the candidate set $\delta \in [0, \delta_{\max}]$. In most numerical experiments, we found that no more than 30 RIMLE evaluations are required. $\delta_{\max}$ can be chosen as highest density value occurring within an initialized cluster, discarding δ-values for which the RIMLE-solution ends up at the border of the parameter space.

4. Definition of "True" Clusters and Misclassification

In most simulation studies in cluster analysis, in which data are generated from mixture (or fixed partition) models, it is assumed that the "true" clusters are identified with the mixture components, and methods can then be evaluated by misclassification rates. But this can be problematic. Consider the comparison of ML estimators for Gaussian mixtures and for mixtures of t-distributions. In most applications, both approaches would be considered as potentially appropriate for doing the same thing, namely, finding clusters that are unimodal and elliptical. In applications in which the clustering is of main interest (as opposed to parameter estimation), researchers would not mind much whether the density around their cluster cores rather looks like a Gaussian or a t-distribution. But implications for which points are considered as "true outliers" versus "truly belonging to a cluster" would be different, because some points generated by a t-distribution with low degrees of freedom are indeed outlying with respect to the core of the t-distribution from which they are generated.

More generally, the identification of clusters and mixture components cannot be taken for granted. Hennig (2010) illustrated that the interpretation of Gaussian mixture components as clusters depends on whether the components are separated enough. And in robust clustering, one would often interpret a group of a few points with low density as "noise" even if they were generated by a Gaussian distribution.

We now define a "reference truth" for the mixture models that are used in our simulation study, from which misclassification rates then can be computed. For motivation consider Figure 1, which shows an artificial dataset drawn from a mixture of two Gaussians and a uniform distribution. Figure 1(a) shows the unlabeled dataset on which cluster analysis operates. Figure 1(b) shows the points labeled by the mixture components that generated them. Observe that there are three red stars (generated from the uniform noise) in the middle of the region where the two Gaussians have most of their mass. Furthermore, there are green points from the left Gaussian component with lower density that fall into the dense blue region. No method can be expected to reconstruct all the cluster memberships in such overlapping regions.

Figure 1(c) shows what we define as the "reference truth," defined next.

The idea is that we choose probability measures $P_1, P_2, \ldots, P_G$ to correspond to the G "true clusters" (implying that they "cluster," that is, generate clearly distinguishable,

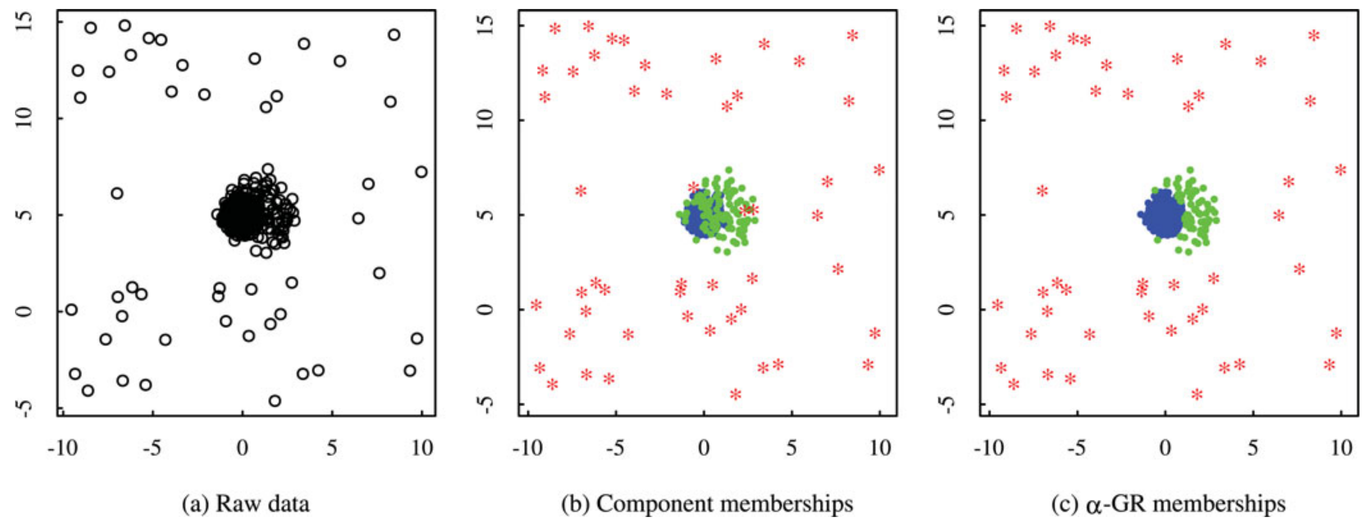


Figure 1. An artificial dataset consisting of 950 points drawn from two two-dimensional Gaussian distributions and 50 points from a uniform distribution on the square $[-10, 10] \times [-5, 15]$. (a) unlabeled points, (b) colored according to the mixture components, (c) colors represent the two GR_α (with $\alpha = 10^{-4}$); red stars belong to NR_α.

although not necessarily nonoverlapping, data patterns). For each of these, we define a region of points that can be considered as nonoutliers based on a mean and covariance matrix functional, which are Fisher-consistent at the Gaussian distribution, but robust and existing for other distributions, too. This defines a region of nonoutliers of Gaussian shape. We consider all points "noise" that are outliers according to this definition for all $P_1, P_2, \ldots, P_G$. Quadratic discriminant analysis assigns points to clusters that are nonoutliers with respect to more than one of the P_j. This means that points are assigned to clusters by optimal classification boundaries under the Gaussian assumption, even if the components are in fact not Gaussian. This formalizes using "Gaussian cluster prototypes" without assuming that clusters really have to be Gaussian.

To this end, let m_j and S_j be the minimum covariance determinant (MCD) center and scatter functional at P_j (Rousseeuw 1985). Cator and Lopuhaä (2012) proved existence of the MCD functional for a wide class of probability measures. The S_j can be corrected for achieving consistency at P_j equal to the Gaussian distribution (Croux and Haesbroeck 1999; Pison et al. 2002), so that when P_j is Gaussian, m_j and S_j are the corresponding mean vector and covariance matrix. Let π_j be the expected proportion of points generated from P_j. We allow $\sum_{j=1}^{G} \pi_j \leq 1$, so that points could be generated by (noise-)distributions other than $P_1, P_2, \ldots, P_G$. Define the quadratic discriminant score for assigning the point $y \in \mathbb{R}^p$ to the jth cluster by maximizing

$$\mathrm{qs}(y; \pi_j, m_j, S_j) : = \log(\pi_j) - \frac{1}{2}\log(\det(S_j)) \\ -\frac{1}{2}(y - m_j)' S_j^{-1}(y - m_j), \\ \text{for } j = 1, 2, \ldots, G.$$

If clusters are indeed Gaussian, this is equivalent to (3.4). Consider

$$\mathcal{E}_\alpha(m_j, S_j) := \{y : \quad (y - m_j)' S_j^{-1}(y - m_j) \leq \chi_p^2(1 - \alpha)\},$$

where $\chi_p^2(1-\alpha)$ is the $1-\alpha$ quantile of the χ_p^2 distribution. For a fixed α, the ellipsoid $\mathcal{E}_\alpha(m_j, S_j)$ defines the subset of $\mathbb{R}^p$ that hosts the jth cluster. The size of this ellipsoid is defined in terms of $\chi_p^2(1-\alpha)$, because for Gaussian P_j, $P_j(\mathbb{R}^p \setminus \mathcal{E}_\alpha(m_j, S_j)) = \alpha$. For a fixed level α, the α-Gaussian region is defined as the union of these ellipsoids:

$$\mathrm{GR}_\alpha := \bigcup_{j=1}^{G} \mathcal{E}_\alpha(m_j, S_j),$$

and the noise region is given by $\mathrm{NR}_\alpha := \mathbb{R}^p \setminus \mathrm{GR}_\alpha$.

Definition 1 (α-Gaussian cluster memberships). Given $\alpha \in [0, 1)$, a data-generating process (DGP) with cluster parameters $\theta_C := \{(\pi_j, m_j, S_j),\ \ j = 1, 2, \ldots, G\}$, and a dataset $\underline{x_n} := \{x_1, x_2, \ldots, x_n\}$; the α-Gaussian cluster memberships are given by

$$\mathrm{AGR}_\alpha(x_i; \theta_C) : = \mathbf{1}\{x_i \in \mathrm{GR}_\alpha\} \times \underset{j=1,2,\ldots,G}{\arg\max} \\ j\,\mathbf{1}\{j = \arg\max_{g=1,\ldots,G} \ \mathrm{qs}(y; \pi_g, m_g, S_g)\}. \tag{4.1}$$

$\mathrm{AGR}_\alpha(x_i, \theta_C) = 0$ means that $x_i \in \mathrm{NR}_\alpha$.

This definition is inspired by the definition of outliers with respect to a reference model as in Davies and Gather (1993) and Becker and Gather (1999). A difference is that here the parameter α does not directly control the probability of the noise region. Once α and the triples (π_j, m_j, S_j) are fixed for all $j = 1, 2, \ldots, G$, the size of the noise region will depend on the degree of overlap and Gaussianity of the ellipsoids in GR_α. α needs to be small because the idea of an outlier implies that under the Gaussian distributions outliers are very rare. We choose $\alpha = 10^{-4}$, which implies that the probability that there is at least one outlier in $n = 500$ iid Gaussian observations is 0.0488.

The different robust clustering methods have different implicit ways of classifying points as "noise" (noise component, trimming, outlier identification in t-distributions). To make them comparable, we use (4.1) to unify the point assignment of the methods by computing $\mathrm{AGR}_\alpha(\cdot)$ based on the parameters estimated by the methods, from which we assume that estimators of the triples (π_j, m_j, S_j) (cluster proportion, center, and covariance matrix) can be computed (see Section 5.2). Let the estimated cluster parameters be $\hat{\theta}_{C,n}$. A misclassification rate can then be computed by applying an optimal permutation $\sigma\{\cdot\}$ of cluster labels:

$$\begin{aligned}\mathrm{mcr}(\theta_C, \hat{\theta}_{C,n}) &= \arg\min_{\sigma} \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}\{\mathrm{AGR}_\alpha(x_i; \theta_C) \\ &= \sigma\{\mathrm{AGR}_\alpha(x_i; \hat{\theta}_{C,n})\}\}. \qquad (4.2)\end{aligned}$$

See the online supplement for computation of the MCD functional for nonnormal distributions.

5. A Comparative Simulation Study

Here, we present a comprehensive simulation study comparing the OTRIMLE with the methods introduced in Section 2.

5.1. Data-Generating Processes

The methods are compared on a total of 24 DGPs with 1000 Monte Carlo replicates each. Half of the DGPs produce two-dimensional datasets. The remaining 12 DGPs are 20-dimensional versions that are constructed adding independent 18-dimensional uncorrelated zero-means unit-variance Gaussian and/or Student-t marginals. Therefore, clusters are always only defined on the first two marginals. Note that the aim of the simulation study is not variable selection; we designed the DGPs so that clustering information is only in the first two dimensions to be able to visualize and control the clustering patterns, but we compare clustering methods that use all variables (trying variable selection methods is beyond the scope of this article). We do not think that variable selection or dimension reduction is mandatory in clustering, because the meaning of the clusters is determined by the involved variables, see the discussion rejoinder in Hennig and Liao (2013). We choose $n = 1000$ for two-dimensional designs and $n = 2000$ for the 20-dimensional versions. DGPs have been designed to test a variety of "noise patterns," numbers of clusters G, and patterns of separation/overlap between different clusters.

We consider two main classes of DGP, namely, DGPs with a uniform noise component on the first two marginals, and DGPs that do not have a noise component. The first group includes the following setups, all of which have clusters generated from Gaussian distributions, and for $p = 20$ the 18 uninformative variables are Gaussian: (i) for "WideNoise" DGPs, the uniform noise component produces points that are widespread but overlap with the clustered regions entirely; (ii) for "SideNoise" DGPs the uniform noise component spreads points on a wide region that overlaps slightly with some of the clusters; (iii) in "SunSpot" DGPs there is a uniform component that produces few extremely outlying points. On the other hand, we consider DGPs that do not include a noise component (i.e., $\pi_0 = 0$). This second group can be divided into three further subgroups: (i)/(ii) in "GaussT" and "TGauss" DGPs, multivariate Student-t distributions with three degrees of freedom are used. In "GaussT" these are used as uninformative distributions for $p = 20$, whereas the first two clustered dimensions use Gaussians; in "TGauss" the clusters are generated by noncentral multivariate t_3-distributions and for $p = 20$ the 18 uninformative variables are Gaussian; (iii) in "Noiseless" DGPs, all points are drawn from Gaussian distributions.

For each of the six setups, there are variants with a lower and a higher number of clusters G, $p = 2$ (denoted by "l") and $p = 20$ (denoted by "h"), adding up to 24 DGPs. The nomenclature used in the following puts these at the end of the setup name, that is, "TGauss.5h" refers to the "TGauss"-setup with higher $G = 5$ and higher $P = 20$. For "WideNoise," "SideNoise," and "GaussT," the lower G was 2 and the higher G was 3. For "SunSpot," "TGauss," and "Noiseless," the lower G was 3 and the higher G was 5. The overlap between clusters as well as the combinations of cluster shapes varied between DGPs. Full details of the definition of the DGPs are given in the online supplement together with exemplary scatterplots of the first two dimensions of a dataset from every setup.

5.2. Implementation of Methods

Table 1 summarizes settings for the compared methods. TCLUST and RIMLE/OTRIMLE are based on eigenratio constraints, but this is not the case for the MCLUST software (Fraley et al. 2012) and the available implementation of mixtures of t-distributions (McLachlan and Peel 2000). To have full comparability of the solutions, the eigenratio constraints (see Section 3.3) have been implemented by us for OTRIMLE/RIMLE, gmix, tmix, and gmix.u; the latter is computed by use of the same routine that is used for RIMLE /OTRIMLE. For TCLUST, the constraints are in the original R-package.

For all methods, the eigenratio constraint has been set equal to 20 for each of the 24 DGPs. The latter choice is motivated by the fact that 20 is larger than the maximum true eigenratio across the designs involved in the comparison, and it is in general a value that often enables rather smooth optimization (obviously in reality the true eigenvalue ratios are not known, but for variables with comparable scales and value ranges, 20 gives the covariance matrices enough flexibility for most applications). OTRIMLE has been tested with and without the penalty term $\beta = \frac{1}{3}$ in (3.12), denoted by ot.rimle ($\beta = 0$) and ot.rimle.p, respectively. Other values for β, ranging from 0.1 to 0.5, have been tried, and results did not change much. A difficulty with TCLUST is that an automatic data-driven choice of the trimming level is currently not available. In tclust.f we set the trimming level to 10%. This choice is motivated by the fact the DGPs produced an average proportion of points belonging to the NR_α set in the range [0%, 23%] (see Table 1 in Coretto and Hennig in

Table 1. Summary of the main settings for the methods under comparison.

Method	Setup
gmix.u	RIMLE with $\delta = 1/V_n$, where V_n is volume of the smallest hyperrectangle that contains the data.
ot.rimle	OTRIMLE without penalty ($\beta = 0$).
ot.rimle.p	OTRIMLE with penalty term $\beta = 1/3$.
tclust	TCLUST with fixed trimming level set at 0.1.
ot.tclust	TCLUST with trimming level selected by the OTRIMLE criterion without penalty ($\beta = 0$).
ot.tclust.p	TCLUST with trimming level selected by the OTRIMLE criterion with penalty term $\beta = 1/3$.
tmix	ML for the Student-t mixture model (2.1) with $\upsilon = 3$ for all components plus eigenratio constraints.
gmix	ML for the Gaussian mixture model plus eigenratio constraints.

press b). Furthermore, since the trimming level plays a role similar to δ in RIMLE/OTRIMLE, there are two versions of TCLUST for which the same idea for automatic decisions the trimming level is used as proposed here for OTRIMLE, see Section 3. In ot.tclust and ot.tclust.p, the trimming level has been selected using (3.12) with the trimming level playing the role of δ, and the weights $\tau_j(\cdot)$ in $\mathbb{M}_j(\cdot)$ replaced by the 0–1 crisp weights of TCLUST, again using $\beta = 0$ or $\beta = \frac{1}{3}$. For t-mixtures, in the original proposal by McLachlan and Peel (2000), the degrees of freedom are assumed to be equal across the mixture components and are estimated from the data and covariance matrix constraints are not considered. In tmix, we fix the degrees of freedom to 3 and we incorporate eigenratio constraints, as for the other methods. This is motivated as follows: (i) for some of the DGPs (particularly the SunSpot designs) constraints were needed to avoid spurious solutions; (ii) for some of the sampling designs not based on Student-t distributions, estimation of the degrees of freedom of the t-distribution produced an extremely large variability in the misclassification rates; (iii) since a number of designs are based on Student-t with 3 degrees of freedom, the decision to fix this parameter to 3 gives the t-mixture a slight advantage, which seems fair given that the majority of setups rather seem to favor noise component/trimming.

For some of the DGPs, the experience suggested that solutions may depend strongly on the initialization. To reduce the bias introduced by different initializations, all methods are initialized from the same partition, see Section 3.3 (an additional set of R functions has been provided by TCLUST's authors to allow for this).

5.3. Results

The methods are compared using misclassification rates as defined in (4.2). These are more relevant in clustering tasks than parameter estimates. The results are graphically summarized in Figure 2, while average misclassification rates with standard errors are given in Table 2. Each square in the plot is a color-coded representation of the misclassification rate averaged over the 1000 Monte Carlo replicates for a given method-DGP pair. Further details about average misclassification rates are given in the online supplement. It also contains boxplots of the misclassification rates for all method-DGP pairs. Figure 2 shows clear evidence that using robust methods is important. The gmix method only performs well for Noiseless and some DGPs with Gaussians and t-distributions, but most other methods work well (although slightly worse at times) for these DGPs, too. tmix works well for most DGPs involving t-distributions, but for the other DGPs with noise/outliers, it is often seriously worse than gmix.u, OTRIMLE, and TCLUST.

gmix.u performs relatively well, although for a number of DGPs it suffers strongly from high dimensionality. For WideNose in two dimensions, gmix.u will equal in many cases a proper ML estimator, so the method should be advantageous here, and gmix.u is indeed best for these DGPs. However, for the 20-dimensional WideNoise, taking a uniform distribution over the smallest hyperrectangle containing the data can no longer be the ML estimate for the noise-generating mixture component, and its performance deteriorates strongly.

Regarding the TCLUST methods, even though the automatic trimming level of ot.tclust and ot.tclust.p did not always improve the results, it demonstrated to provide a reasonable choice for the trimming level. In fact, for all situations where the true average noise proportion is about equal to the trimming level of tclust, performances of tclust, ot.tclust, and ot.tclust.p are very similar, meaning that the OTRIMLE criterion is a good starting point for fixing the trimming rate. Compared to the RIMLE-type methods, the performance of TCLUST suffers in situations where there is a considerable degree of overlap between clusters. For DGPs with overlap (such as WideNoise.2, SunSpot.5, GaussT.2, and Noiseless.5), the misclassification

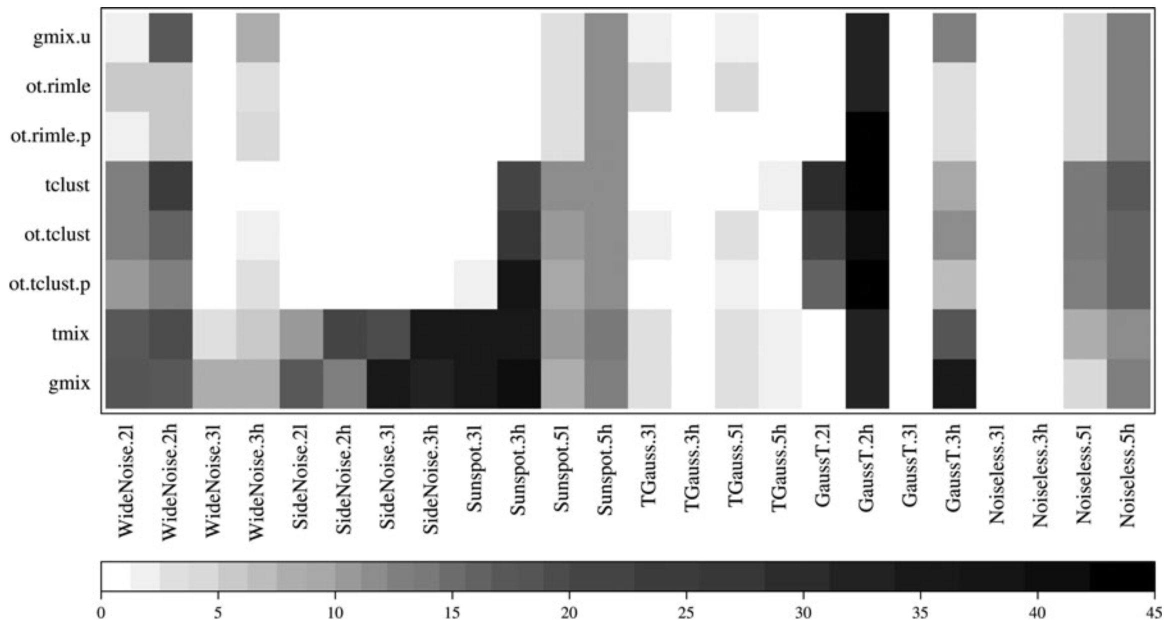


Figure 2. Level plot representing the sample mean of the Monte Carlo distribution of misclassification rates (percentage scale) for each DGP-method pair. Each square of the plot represents the average misclassification according to the bottom gray color scale.

Table 2. Monte Carlo average misclassification rates (%) with their standard errors in brackets. Misclassification rates are computed as in (4.1). Notice that both averages (and standard errors) are reported in percentage scale.

	Method							
DGP	gmix.u	ot.rimle	ot.rimle.p	tclust	ot.tclust	ot.tclust.p	tmix	gmix
WideNoise.2l	1.33(0.02)	5.00(0.29)	1.35(0.02)	12.70(0.33)	13.11(0.32)	10.20(0.32)	17.48(0.10)	18.40(0.04)
WideNoise.2h	17.36(0.04)	5.28(0.05)	5.40(0.06)	22.94(0.32)	16.20(0.33)	12.85(0.25)	18.90(0.03)	17.43(0.04)
WideNoise.3l	0.34(0.01)	0.42(0.01)	0.40(0.01)	0.44(0.01)	0.38(0.01)	0.39(0.01)	3.10(0.02)	7.79(0.13)
WideNoise.3h	7.59(0.18)	3.18(0.09)	4.27(0.09)	0.72(0.01)	1.31(0.02)	3.04(0.06)	5.91(0.04)	8.56(0.15)
SideNoise.2l	0.01(0.00)	0.03(0.01)	0.04(0.02)	0.18(0.01)	0.02(0.00)	0.01(0.00)	10.76(0.11)	16.68(0.16)
SideNoise.2h	0.03(0.01)	0.09(0.02)	0.15(0.03)	0.20(0.01)	0.20(0.01)	0.66(0.02)	20.03(0.03)	12.61(0.15)
SideNoise.3l	0.06(0.00)	0.08(0.01)	0.11(0.03)	0.19(0.01)	0.07(0.00)	0.06(0.00)	18.89(0.47)	36.60(0.40)
SideNoise.3h	0.13(0.00)	0.19(0.03)	0.24(0.04)	0.25(0.01)	0.21(0.01)	0.53(0.02)	34.23(0.16)	31.59(0.31)
Sunspot.3l	0.11(0.00)	0.12(0.00)	0.11(0.00)	0.21(0.00)	0.13(0.01)	2.17(0.36)	34.65(0.11)	35.38(0.19)
Sunspot.3h	0.24(0.00)	0.24(0.00)	0.24(0.00)	21.38(0.95)	27.19(1.00)	36.83(1.01)	34.75(0.06)	40.00(0.37)
Sunspot.5l	3.32(0.12)	3.39(0.12)	3.37(0.12)	11.25(0.13)	10.01(0.14)	9.53(0.14)	10.01(0.10)	7.92(0.16)
Sunspot.5h	11.72(0.10)	11.63(0.10)	11.65(0.10)	11.88(0.10)	11.63(0.10)	11.61(0.10)	13.84(0.12)	13.30(0.13)
TGauss.3l	1.84(0.02)	4.56(0.09)	0.90(0.01)	0.81(0.01)	1.63(0.03)	0.95(0.01)	3.28(0.02)	3.33(0.06)
TGauss.3h	0.49(0.01)	0.53(0.01)	0.51(0.01)	0.77(0.01)	0.51(0.01)	0.49(0.01)	0.83(0.01)	0.85(0.02)
TGauss.5l	1.77(0.02)	4.65(0.07)	1.20(0.02)	1.21(0.01)	2.55(0.04)	1.32(0.02)	3.37(0.02)	3.59(0.05)
TGauss.5h	0.92(0.02)	0.97(0.01)	0.94(0.01)	1.39(0.01)	0.96(0.01)	0.89(0.01)	1.43(0.02)	1.65(0.04)
GaussT.2l	0.57(0.01)	0.57(0.01)	0.56(0.01)	28.77(0.33)	21.50(0.48)	16.52(0.48)	0.83(0.02)	0.57(0.01)
GaussT.2h	32.45(0.05)	33.60(0.14)	44.91(0.10)	42.42(0.13)	41.72(0.15)	43.13(0.13)	31.32(0.16)	33.17(0.03)
GaussT.3l	0.11(0.00)	0.11(0.00)	0.11(0.00)	0.27(0.01)	0.13(0.00)	0.12(0.00)	0.17(0.01)	0.11(0.00)
GaussT.3h	12.32(0.12)	3.69(0.03)	3.44(0.03)	9.87(0.03)	12.22(0.04)	6.85(0.08)	18.34(0.03)	36.23(0.49)
Noiseless.3l	0.12(0.00)	0.12(0.00)	0.12(0.00)	0.28(0.01)	0.15(0.01)	0.13(0.00)	0.18(0.01)	0.12(0.00)
Noiseless.3h	0.25(0.00)	0.25(0.00)	0.25(0.00)	0.38(0.00)	0.28(0.00)	0.28(0.00)	0.29(0.00)	0.25(0.00)
Noiseless.5l	4.53(0.08)	4.47(0.08)	4.43(0.08)	14.43(0.13)	13.33(0.13)	13.14(0.14)	8.23(0.14)	4.58(0.09)
Noiseless.5h	12.39(0.10)	12.63(0.10)	12.62(0.10)	16.69(0.12)	16.13(0.12)	16.10(0.12)	11.92(0.08)	12.37(0.10)

rate of TCLUST is completely dominated by misclassifications between clusters (to see this consider Tables 2 and 4 in the online supplement). The reason is that the TCLUST parameters are based on a classification-type likelihood, which relies on the separation between clusters. The TCLUST also seems to not tolerate the large number of Student-t marginals of GaussT.2h and GaussT.3h. TCLUST performs well for a number of DGPs and is clearly best in WideNoise.2h.

The OTRIMLE methods show a very good overall performance. They produce high misclassification rates only for some 20-dimensional DGPs for which all methods are in trouble (performances for GaussT.2h are generally bad with even tmix, the best method there, producing an average misclassification rate of more than 30%), and they are best for a number of DGPs, particularly 20-dimensional WideNoise and some TGauss-DGPs. The comparison between ot.rimle and ot.rimle.p is mixed (as between ot.tclust and ot.tclust.p), with $\beta = \frac{1}{3}$ improving matters clearly for TGauss.2l and TGauss.3l (the shape of the t-distribution encourages ot.rimle to assign too many points to the noise; see Tables 2 and 3 in the online supplement, but being significantly worse for WideNoise.3h.

Comparing OTRIMLE with gmix.u, there are a number of DGPs for which gmix.u has a slightly lower misclassification rate than one or both of ot.rimle and ot.rimle.p. In all of these DGPs, all of gmix.u, ot.rimle, and ot.rimle.p basically produce the same clustering structure, with disagreement only about the classification of some borderline points. Differences are more substantial in the setups in which gmix.u is worse. For WideNoise.2h, gmix.u in most cases does not detect any noise, so that one of the clusters consists mainly of noise. For WideNoise.3h, sometimes all or much noise is merged into a cluster, with some impact on the clustering structure. For GaussT.3h, substantial amounts of outliers are integrated into the clusters.

5.4. Behavior of $D(\delta)$

Here, we investigate the behavior of $\mathrm{D}(\delta)$ as a function of δ via Monte Carlo experiments under various DGPs from Section 5. For each of these DGPs, we produced 100 independent samples. We computed $\mathrm{D}(\delta)$ for a grid of δ values taken from an interval $[2.22 \times 10^{-308}, 1]$, adding $\delta = 0$. In Figure 3, we report Monte Carlo averages$\pm$standard errors for $\mathrm{D}(\delta)$ for two selected DGPs: WideNoise.3h and GaussT.3h. These are defined in Section 5.1, both with $p = 20$. The main difference is that noise is produced by a two-dimensional uniform distribution in WideNoise.3h, whereas in GaussT.3h dimensions 3–20 are from centered t_3 distributions, generating some outliers. Figure 3 reports the behavior of $\mathrm{D}(\delta)$ for $\log(\delta) > -200$. For smaller values of δ (including $\delta = 0$) the behavior of the curves was basically constant. In both graphs there is a clear minimum, although for GaussT.3h this minimum lies on the border. For WideNoise.3h the OTRIMLE criterion has a nice convex behavior around its minimum. In GaussT.3h, in dimensions 3–20 the distributional shape of the clusters deviated from Gaussianity by heavier tails, although the core of the distribution looks similar to a Gaussian. $\mathrm{D}(\cdot)$ then enforces a Gaussian shape by assigning many points to the noise component. The result is that $\pi_{0,n}$ becomes large, and the optimal δ happens at point where larger values of δ do not produce parameter estimates within the constrained set anymore (unless the constraint is enforced). The latter is a reason for the use of the penalty term in (3.12) (see also Section 5.5). For the remaining 22 DGPs of Section 5.1, we found similar patterns (mostly similar to Widenoise.3h here). Another observation in Figure 3 is that around the minimum

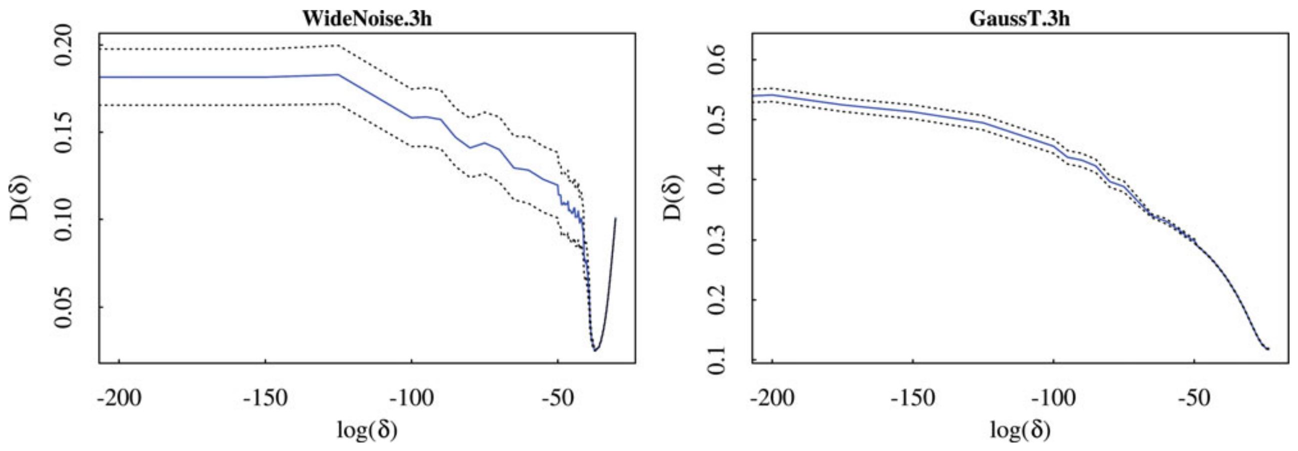


Figure 3. Monte Carlo average for OTRIMLE criterion $D(\delta)$ (blue solid line) $\pm$ standard errors (dotted lines) computed over a grid of values for $\delta \in \{0\} \cup [2.22 \times 10^{-308}, 1]$. The two plots refer to DGPs "WideNoise.3h" and "GaussT.3h" in Section 5.

$D(\delta)$ seems to be quite stable for different datasets from the same design.

5.5. *Effect of β*

For all 24 DGPs considered in Section 5.1, we also investigated the behavior of the noise proportion $\pi_{0,n}$ as a function of β, see (3.12). For each design, we produced 100 independent samples, and for each of these we computed the OTRIMLE solution $\theta_n(\delta_n)$ for various values of $\beta \in [0, 1]$. Figure 4 reports the Monte Carlo average of the estimated noise proportion $\pi_{0,n}\pm$ standard error. For both WideNoise.3h and GaussT.3h, an increase in β reduces smoothly the estimated noise proportion. There is some difference, however, in scales. The impact of β is much stronger for GaussT.3h, and the same happens for all those sampling designs in which within-cluster distributions deviate from Gaussianity. Results for other DGPs are quite similar.

The discovery of the overall clustering structure was never affected by changing β between 0 and 0.5 for the DGPs from Section 5. In the majority of cases, there was no change of the clustering at all. The only difference was that larger β sometimes produced a lower percentage of data classified as noise. In the case that t_3-distributions were involved, this was good, because with $\beta = 0$ only fairly small central cores (60% –70%) of the points generated by t-distributions were assigned to the clusters, whereas for larger β only points were classified as noise that were really quite "outlying." On the other hand, with a larger β, in data from DGPs with Gaussian clusters and some noise that was not clearly separated from the clusters, more "true" noise points were assigned to the clusters. There is no objective rule for what percentage of points from a t-distribution should be considered as "truly" outlying, and therefore it needs to be decided by the user how "tolerant" OTRIMLE is desired to be toward heavier distributional tails than the Gaussian ones.

Figure 5 shows a situation in which the choice of β affects the clustering a lot. The dataset consists of 100 observations each of $\mathcal{N}(0, 1)$ and $\mathcal{N}(3, 1)$ along the x-axis and 12 observations from $\mathcal{N}(12, 25)$. OTRIMLE was fitted with $G = 2$. The first two mixture components are not very well separated. $\beta = 0$ does not penalize noise, and therefore the observations from the third component are declared "noise" and the first two components are separated. However, $\beta = \frac{1}{3}$ merges the first two components and declares the third one the second cluster. The "switching point" between these two ways of "interpreting" the clustering structure is at about $\beta = 0.3$; larger values of β do not change the clustering anymore. Despite the "true" G being 3 here, regarding interpretation, depending on the application it may well make sense to either treat the smallest mixture component as

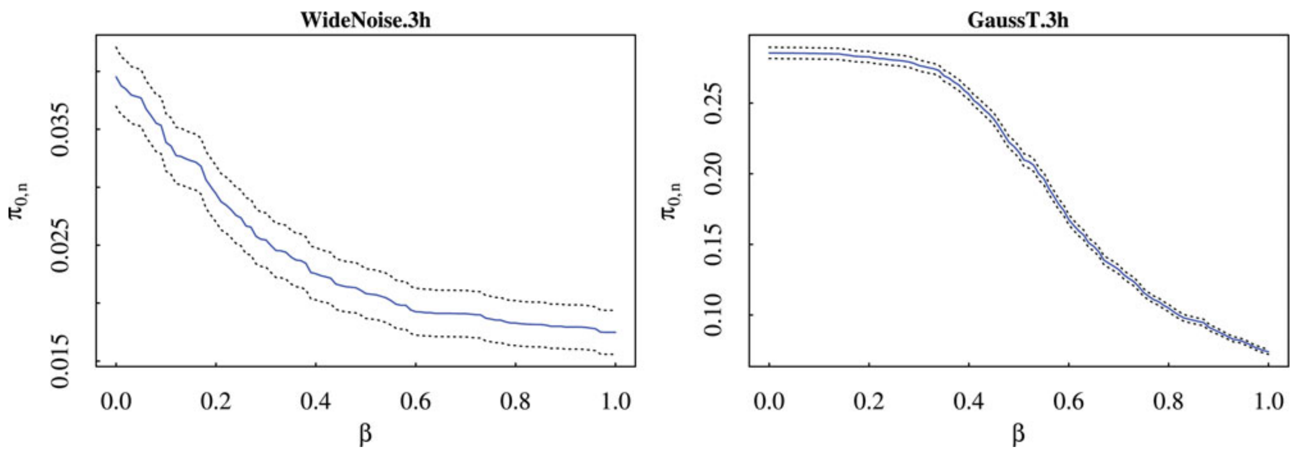


Figure 4. Monte Carlo average for the estimated noise proportion $\pi_{0,n}$ (blue solid line) $\pm$ standard errors (dotted lines) computed over a grid of values for $\beta \in [0, 1]$. The two plots refer to DGPs called "WideNoise.3h" and "GaussT.3h" in Section 5.

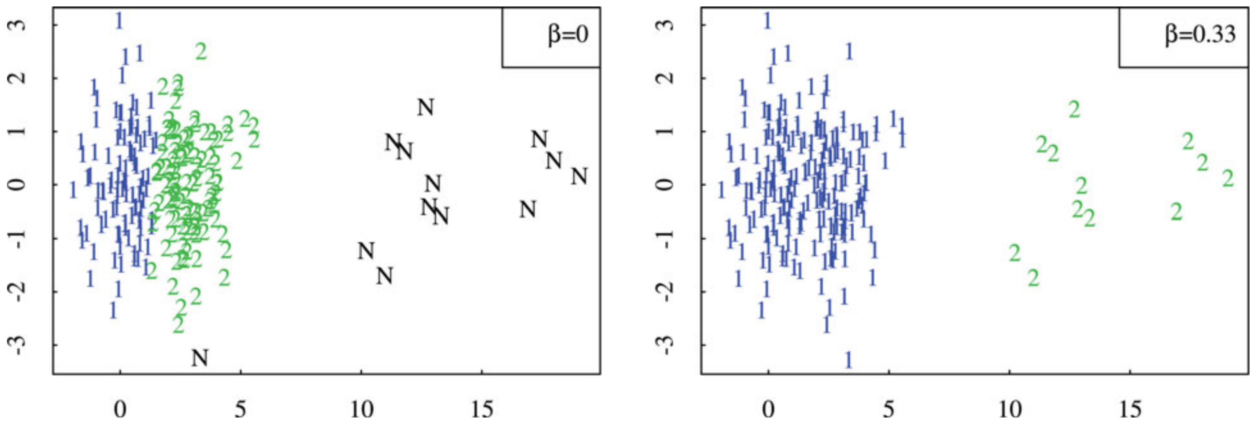


Figure 5. Clusterings with $\beta = 0$ and $\beta = \frac{1}{3}$ on artificial data example.

noise/outliers, or to merge the first two components to a single cluster. β tunes the method to rather produce noise, or to rather tolerate not-so-Gaussian components in cases like this.

6. Applications

In this section, we apply OTRIMLE and the alternative methods mentioned before to two real datasets. The first example does not come with any "ground truth," whereas the second one has "true" classes.

6.1. Dortmund Data

We here analyze a dataset giving information about 170 districts of the German city of Dortmund, which is described in Sommerer and Weihs (2005). We used five sociological key variables and transformed them in such a way that fitting Gaussian distributions within clusters makes sense. The resulting variables are the logarithm of the unemployment rate ("unemployment"), the birth/death balance divided by number of inhabitants ("birth.death"), the migration balance divided by number of inhabitants ("moves.in.out"), the logarithm of the rate of employees paying social insurance ("soc.ins.emp"), and the square root of the number of inhabitants ("inhabitants").

All variables were centered and standardized by the median absolute deviation. Figure 6 shows a scatterplot of `birth.death` and `moves.in.out`. To deal with overplotting, an existing extreme outlier with values $\approx (-200, 50)$ is not shown. Figure 7 shows a scatterplot of `unemployment` and `soc.ins.emp`. Figure 7 shows some more moderate outliers. The left side of Figure 6 shows a clustering from fitting a plain Gaussian mixture with $G = 4$ to all five variables by R's MCLUST package. Cluster 4 is a one-point cluster made of the extreme outlier. Clusters 1 and 3 basically fit two different varieties of moderate outliers, whereas all the more than 120 districts that are not extreme regarding these two variables are put together in a single cluster. Clearly, it would be more desirable to have a clustering that is not dominated so much by a few odd districts, given that there is some meaningful structure among the other districts. Such a clustering is produced by the OTRIMLE method, shown on the right side of Figure 6 and on the left side of Figure 7. The clustering is nicely interpretable with cluster no. 3 collecting a group of districts with higher migration balance and very scattered birth/death rate, cluster no. 1 being a high variation cluster characterized by high unemployment or high number of employees paying social insurance, cluster no. 2 being a homogenous group with medium number of employees paying social insurance and rather high but not very high unemployment, and cluster no. 4 collecting most

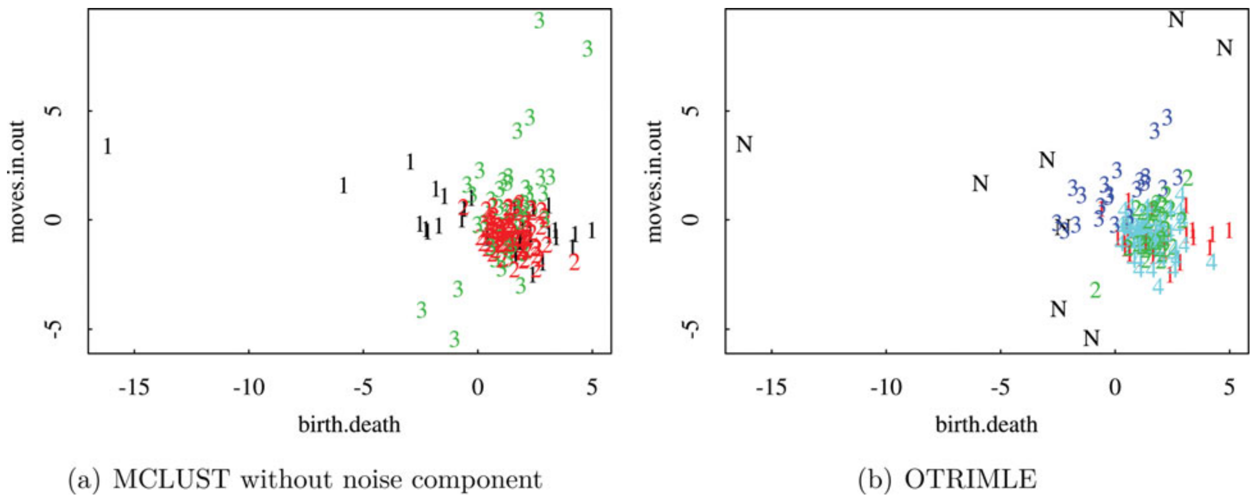


Figure 6. Scatterplot of `birth.death` and `moves.in.out` from Dortmund dataset with MCLUST clustering (left) and OTRIMLE clustering (right) with $G = 4$.

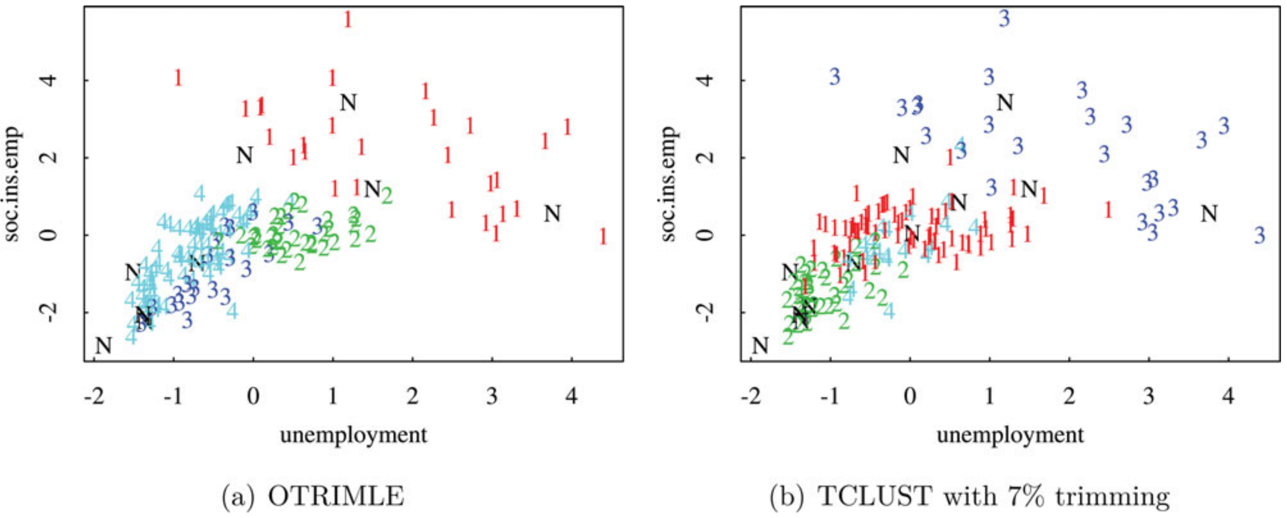


Figure 7. Scatterplot of `soc.ins.emp` and `unemployment` from Dortmund dataset with OTRIMLE clustering (left) and TCLUST clustering with trimming rate 7% (right) with $G = 4$.

districts with low values on both of these variables. For this dataset, values between $\beta = 0$ and $\beta = 0.5$ yield the same clustering (with eigenratio constraint $\gamma = 20$), despite the fact that $\beta = 0$ leads to $\log(\delta) = -11.5$ and $\pi_{0,n} = 0.055$, whereas for $\beta = 0.5$, (3.12) produces $\log(\delta) = -11.9$ and $\pi_{0,n} = 0.054$.

Looking at the methods introduced in Section 2, it turns out that substantial disagreement may exist between different robust clustering methods. Applying the methods implemented in `tclust` and discussed in García-Escudero et al. (2011), $\alpha = 0.07$ was found to be a good trimming rate for TCLUST with $G = 4$ here. The resulting clustering is compared with OTRIMLE's in Figure 7. Although what was trimmed is almost identical to OTRIMLE's "noise," the clustering is somewhat different, with TCLUST's clusters no. 1 and 4 ranging into the area of "high variation characterized by high unemployment or high number of employees paying social insurance" as mainly represented by cluster no. 3 of TCLUST and cluster no. 1 of OTRIMLE. It is hard to interpret OTRIMLE's cluster no. 4 using any pair of variables. This can be seen in the online supplement (Coretto and Hennig in press b), as well as solutions of the other methods.

6.2. Folk Song Data

The second dataset was provided by Daniel Müllensiefen. The observations are 776 folk song melodies, 586 of which are from Luxembourg and the remaining 190 are from Warmia in Poland. These are the "true" classes. The melodies are originally from the ESAC melody database (Schaffrath 1992). The 18 features (see the online supplement (Coretto and Hennig in press b) for a list) were computed by the software "FANTASTIC" (Müllensiefen 2009).

Visual inspection reveals that there are many unusual melodies, that is, outliers in the dataset. The main bulks of melodies from Luxembourg and Warmia differ systematically from each other, although there is much overlap and no strong separation. For measuring to what extent clusterings computed with $G = 2$ coincided with the two regions, we used the adjusted Rand index (ARI; Hubert and Arabie 1985) with an expected value of 0 for two random clusterings and a maximum of 1 for perfect agreement. OTRIMLE with settings as above ($\beta = 0$, $\gamma = 20$) classified 36.9% of the observations as "noise." The ARI between the OTRIMLE solution and the original regions is 0.155. For this (as for the other clustering methods), the OTRIMLE solution was interpreted as a three-cluster solution with "noise" as third cluster. Default MCLUST yields an ARI $= -0.045$, MCLUST with noise yields ARI $= -0.017$, ot.tclust yields ARI $= 0.016$ (the original TCLUST function with trimming rate 0.369 as suggested by OTRIMLE above achieves ARI $= 0.089$), and tmix yields ARI $= 0.083$. OTRIMLE's ARI-value, though clearly better than that of the other methods, is not particularly high, but computing the ARI only on the observations that were not classified as noise achieves ARI $= 0.392$ (the solution with $\beta = \frac{1}{3}$ is slightly worse here but slightly better above regarding the ARI including the noise points), which suggests that there is a clear correspondence between OTRIMLE's clustering and melodies that are typical for the regions.

7. Concluding Remarks

Despite our effort to make the simulation study fair, ultimately it would be good to have comparisons of methods run by researchers who did not have their hand in the design of any of the methods. Every method was best for certain DGPs in the simulation study, and simulation studies could be designed that make any method "win." Readers need to make up their own mind about to what extent our study covered situations that are important to them. One of our major aims was to confront all methods with DGPs that do not exactly match their model assumptions, but for which the methods nevertheless could be legitimately used. In fact, we incorporated crucial ideas from both MCLUST (initialization) and TCLUST (eigenvalue ratio constraints), and the combination of these ideas used here could actually be beneficial for all methods. The methods presented in this article will soon be available in the new R-package OTRIMLE.

The problem of determining the number of clusters G is very important in practice. Here are two possible approaches for RIMLE. First, the most popular approach for fitting plain mixture models, namely, the Bayesian Information Criterion, can be used (treating RIMLE/OTRIMLE improper constant density as a proper one), Fraley and Raftery (1998). Second, G could be decided in an exploratory way by monitoring the changes of the pseudo-likelihood over different values of G in a similar way to what is done for TCLUST in García-Escudero et al. (2011). Investigating these approaches in depth is beyond the scope of this article.

Supplementary Materials

Supplementary materials contain: (i) additional details about methods and algorithms; (ii) detailed definitions of the sampling designs for the simulation study along with boxplots and summary tables for misclassification rates (iii) additional plots for real data applications; (iv) R software with instructions to reproduce the comparative simulation study and applications to real data.

Funding

The authors gratefully acknowledge financial support from MIUR research grant PRIN 2010J3LZEN, and the use of the high-performance computing infrastructure funded by University of Salerno (research program ASSA098434). The authors gratefully acknowledge the support from the EPSRC grant EP/K033972/1.

ORCID

Pietro Coretto http://orcid.org/0000-0002-6972-9671
Christian Hennig http://orcid.org/0000-0003-1550-5637

References

Banfield, J. D., and Raftery, A. E. (1993), "Model-based Gaussian and Non-Gaussian Clustering," *Biometrics*, 49, 803–821. [1648,1649,1651]

Becker, C., and Gather, U. (1999), "The Masking Breakdown Point of Multivariate Outlier Identification Rules," *Journal of the American Statistical Association*, 94, 947–955. [1652]

Byers, S., and Raftery, A. E. (1998), "Nearest-Neighbor Clutter Removal for Estimating Features in Spatial Point Processes," *Journal of the American Statistical Association*, 93, 577–584. [1651]

Cator, E. A., and Lopuhaä, H. P. (2012), "Central Limit Theorem and Influence Function for the MCD Estimators at General Multivariate Distributions," *Bernoulli*, 18, 520–551. [1652]

Coretto, P., and Hennig, C. (2010), "A Simulation Study to Compare Robust Clustering Methods Based on Mixtures," *Advances in Data Analysis and Classification*, 4, 111–135. [1648]

——— (in press a), "A Consistent and Breakdown Robust Model-based Clustering Method," available at *http://arxiv.org/pdf/1309.6895*. [1649,1650,1651]

——— (in press b), Supplement to: "Robust Improper Maximum Likelihood: Tuning, Computation, and a Comparison With Other Methods for Robust Gaussian Clustering," available at *http://arxiv.org/src/1406.0808/anc/supplement.pdf*. [1649,1654,1658]

Cuesta-Albertos, J. A., Gordaliza, A., and Matrán, C. (1997), "Trimmed k-means: An Attempt to Robustify Quantizers," *Annals of Statistics*, 25, 553–576. [1649]

Croux, C., and Haesbroeck, G. (1999), "Influence Function and Efficiency of the Minimum Covariance Determinant Scatter Matrix Estimator," *Journal of Multivariate Analysis*, 71, 161–190. [1652]

Davies, L., and Gather, U. (1993), "The Identification of Multiple Outliers," *Journal of the American Statistical Association*, 88, 782–792. [1652]

Day, N. E. (1969), "Estimating the Components of a Mixture of Normal Distributions," *Biometrika*, 56, 463–474. [1650]

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977), "Maximum Likelihood From Incomplete Data via the EM Algorithm," *Journal of the Royal Statistical Society*, Series B, 39, 1–47. [1649]

Dennis, J. E. J. (ed.) (1981), *Algorithms for Nonlinear Fitting, (NATO Advanced Research Symposium)*, Cambridge, UK: Cambridge University Press. [1650]

Fraley, C., and Raftery, A. E. (1998), "How Many Clusters? Which Clustering Method? Answers via Model-based Cluster Analysis," *The Computer Journal*, 41, 578–588. [1658]

Fraley, C., Raftery, A. E., Murphy, T. B., and Scrucca, L. (2012), *mclust Version 4 for R: Normal Mixture Modeling for Model-Based Clustering, Classification, and Density Estimation*, Technical Report no. 597, Department of Statistics, University of Washington, Seattle, WA. [1648,1649,1653]

Fritz, H., García-Escudero, L. A., and Mayo-Iscar, A. (2012), "tclust: An R Package for a Trimming Approach to Cluster Analysis," *Journal of Statistical Software*, 47, 1–26. [1649]

Gallegos, M. T. (2002), "Maximum Likelihood Clustering With Outliers," in *Classification, Clustering, and Data Analysis*, eds. K. Jajuga, A. Sokolowski, and H.-H. Bock, Berlin: Springer, pp. 247–255. [1649]

Gallegos, M. T., and Ritter, G. (2005), "A Robust Method for Cluster Analysis," *Annals of Statistics*, 33, 347–380. [1649]

——— (2009), "Trimmed ML Estimation of Contaminated Mixtures," *Sankhya (Series A)*, 71, 164–220. [1650]

García-Escudero, L. A., Gordaliza, A., Matrán, C., and Mayo-Iscar, A. (2008), "A General Trimming Approach to Robust Cluster Analysis," *Annals of Statistics*, 38, 1324–1345. [1649,1650]

——— (2010), "A Review of Robust Clustering Methods," *Advances in Data Analysis and Classification*, 4, 89–109. [1649]

——— (2011), "Exploring the Number of Groups in Robust Model-Based Clustering," *Statistics and Computing*, 21, 585–599. [1658]

Hathaway, R. J. (1985), "A Constrained Formulation of Maximum-Likelihood Estimation for Normal Mixture Distributions," *The Annals of Statistics*, 13, 795–800. [1650]

Hennig, C. (2004), "Breakdown Points for Maximum Likelihood Estimators of Location-Scale Mixtures," *The Annals of Statistics*, 32, 1313–1340. [1648,1649,1650]

——— (2010), "Methods for Merging Gaussian Mixture Components," *Advances in Data Analysis and Classification*, 4, 3–34. [1651]

Hennig, C., and Liao, T. F. (2013), "How to Find an Appropriate Clustering for Mixed Type Variables With Application to Socioeconomic Stratification" (with discussion), *Journal of the Royal Statistical Science*, Series C, 62, 309–369. [1648,1653]

Hubert, L., and Arabie, P. (1985), "Comparing Partitions," *Journal of Classification*, 2, 193–218. [1658]

Ingrassia, S. (2004), "A Likelihood-based Constrained Algorithm for Multivariate Normal Mixture Models," *Statistical Methods & Applications*, 13, 151–166. [1650]

Ingrassia, S., and Rocci, R. (2007), "Constrained Monotone EM Algorithms for Finite Mixture of Multivariate Gaussians," *Computational Statistics and Data Analysis*, 51, 5339–5351. [1650]

——— (2011), "Degeneracy of the EM Algorithm for the MLE of Multivariate Gaussian Mixtures and Dynamic Constraints," *Computational Statistics & Data Analysis*, 55, 1715–1725. [1650]

Kiefer, J. (1953), "Sequential Minimax Search for a Maximum," *Proceedings of the American Mathematical Society*, 4, 502–506. [1651]

McLachlan, G. J., and Peel, D. (2000), "Robust Mixture Modelling Using the t–distribution," *Statistics and Computing*, 10, 339–348. [1649,1653]

Müllensiefen, D. (2009), *Fantastic: Feature ANalysis Technology Accessing STatistics (In a Corpus): Technical Report v1.5*, London: Goldsmiths University of London. [1658]

Neykov, N., Filzmoser, P., Dimova, R., and Neytchev, P. (2007), "Robust Fitting of Mixtures Using the Trimmed Likelihood Estimator," *Computational Statistics and Data Analysis*, 17, 299–308. [1649]

Pison, G., Aelst, S. V., and Willems, G. (2002), "Small Sample Corrections for LTS and MCD," *Metrika*, 55, 111–123. [1652]

Qin, Y., and Priebe, C. E. (2013), "Maximum lq-likelihood Estimation via the Expectation-Maximization Algorithm: A Robust Estimation of Mixture Models," *Journal of the American Statistical Association*, 108, 914–928. [1649]

Rousseeuw, P. J. (1985), "Multivariate Estimation With High Breakdown Point," *Mathematical Statistics and Applications*, 8, 283–297. [1652]

Schaffrath, H. (1992), "The esac Databases and Mappet Software," *Computing in Musicology*, 8, 66. [1658]

Sommerer, E.-O., and Weihs, C. (2005), "Introduction to the Contest "Social Milieus in Dortmund," in *Classification - the Ubiquitous Challenge*, Berlin: Springer, pp. 667–673. [1657]

SUPPLEMENT TO
Robust Improper Maximum Likelihood: Tuning, Computation, and a Comparison With Other Methods for Robust Gaussian Clustering

Pietro Coretto[*]
University of Salerno, Italy
e-mail: pcoretto@unisa.it

Christian Hennig[†]
University College London, UK
e-mail: c.hennig@ucl.ac.uk

The present manuscript contains material that is supplementary to Coretto and Hennig (2016). Unless otherwise stated, notation refers to the main paper.

Contents

1 Methods from the literature **2**

2 RIMLE/OTRILME computing **4**

2.1 RIMLE numerical approximation 4

2.2 OTRIMLE numerical approximation 5

2.3 Choice of initial values 6

2.4 Computation of the MCD functional 8

2.5 Additional remarks for the computation of misclassification rates 8

3 Definition of DGPs **8**

4 Plots for the DGPs and misclassification rates from the simulation study **12**

5 Detailed summary tables for the simulation study **24**

6 Plots for the analysis of Dortmund data **28**

[*]The author gratefully acknowledges financial support from MIUR research grant PRIN 2010J3LZEN, and the use of the high-performance computing infrastructure funded by University of Salerno (research program ASSA098434).

[†]The author gratefully acknowledges support from the EPSRC grant EP/K033972/1.

7 Information on folk song data **33**

7.1 Features 33

7.2 Plot 33

References **33**

1 Methods from the literature

Here is a more detailed description of some robust clustering methods from the literature that were presented in Section 2 of Coretto and Hennig (2016) that are used in the simulation study.

In order to settle the notation, we repeat some bits of Section 2 of Coretto and Hennig (2016). In the following, assume an observed sample $\underline{x_n} = \{x_1, x_2, \dots, x_n\}$, where x_i is the realization of a random variable $X_i \in \mathbb{R}^p$ with $p \geq 1$; $X_1, \dots, X_n$ i.i.d. The goal is to cluster the sample points into G distinct groups.

Maximum Likelihood (ML) for Gaussian mixtures (gmix). Let $\phi(x; \mu, \Sigma)$ be the density of a multivariate Gaussian distribution with mean vector $\mu \in \mathbb{R}^p$ and $p \times p$ covariance matrix Σ. Assume that the observed sample is i.i.d. drawn from the finite Gaussian mixture distribution having density

$$m(x; \theta) = \sum_{j=1}^{G} \pi_j \phi(x; \mu_j, \Sigma_j), \tag{1.1}$$

where $\pi_j \in [0, 1]$ for all $j = 1, 2, \dots, G$ and $\sum_{j=1}^{G} \pi_j = 1$, θ is the parameter vector containing the triplets π_j, μ_j, Σ_j for all $j = 1, 2, \dots, G$. Clustering coincides with assigning points to the mixture components based on ML parameter estimates. π_j can be interpreted as the expected proportion of points originated from the jth component. Let θ_n^{ml} be the ML estimator for θ. Let τ_{ij}^{ml} be the estimated posterior probability that the observed point x_i has been drawn from the jth mixture component, i.e.,

$$\tau_{ij}^{\text{ml}} = \frac{\pi_{j,n}^{\text{ml}} \phi(x_i; \mu_{j,n}^{\text{ml}}, \Sigma_{j,n}^{\text{ml}})}{m(x_i; \theta_n^{\text{ml}})} \qquad \text{for all} \quad j = 1, 2, \dots, G. \tag{1.2}$$

The point x_i can then be assigned to the kth cluster if $k = \arg\max_{j=1,2,\dots,G} \tau_{ij}^{\text{ml}}$. This assignment method is common to all model-based clustering methods.

ML–type estimator for Gaussian mixtures with uniform noise (gmix.u). Banfield and Raftery (1993) suggested to accommodate "noise" such as outliers by adding a uniform mixture component to (1.1), calling it "*noise component*", which is implemented in the `mclust` package mentioned above. The resulting density for the data is

$$m_u(x; \theta) := \pi_0 \mathbb{1}\{x \in S\} \frac{1}{V_n} + \sum_{j=1}^{G} \pi_j \phi(x; \mu_j, \Sigma_j), \tag{1.3}$$

where $S \subset \mathbb{R}^p$ with volume V_n, $\pi_0 \in [0, 1)$, $\pi_0 + \sum_{j=1}^{G} \pi_j = 1$, θ now also includes π_0. The parameter π_0 represents the proportion of the uniform noise component. Banfield and Raftery (1993) proposed to fix S as the convex hull of the data, although the `mclust` package actually uses a hyper-rectangle that includes S (Coretto and Hennig (2011) noted that this does not normally yield a proper ML-estimator). θ is estimated by maximizing the log-likelihood function corresponding to (1.3). Denoting this estimator θ_n^{ml}, the estimated posterior probability that a point x_i belongs to the noise component is given by $\tau_{i0}^{\text{ml}} = (\pi_{0,n}^{\text{ml}}/V_n)/m_u(x_i;\theta_n^{\text{ml}})$. Points are then assigned to clusters or the noise component as above. A drawback is that V_n may seriously be affected by outlying points, reducing the formal breakdown point of the method to zero (Hennig, 2004).

ML for mixtures of Student–t distributions (tmix). McLachlan and Peel (2000b) propose to replace the Gaussian densities in (1.1) with multivariate Student-t densities, because they have heavier tails and can therefore accommodate outliers in a better way. Observations can be declared "noise" if they lie so far away from their cluster center that they are in a low density area with a probability lower than some pre-specified small α. Let $f(x;\mu,S,v)$ be the non-central Student-t density in $\mathbb{R}^p$ with expectation μ, positive definite scale matrix S, and v degrees of freedom. Consider the density

$$m_t(x;\theta) := \sum_{j=1}^{G} \pi_j f(x;\mu_j,S_j,v_j), \tag{1.4}$$

where θ contains all mixture component parameters to be fitted by ML. Degrees of freedom can be fixed a priori or can be estimated. The unknown parameter vector is estimated by ML. A point x_i assigned to the jth mixture component can be defined as outlier if

$$(x_i - \mu_j^{\text{ml}})'(S_j^{\text{ml}})^{-1}(x_i - \mu_j^{\text{ml}}) \geq q(\alpha), \tag{1.5}$$

where μ_j^{ml} and S_j^{ml} are the resulting ML estimates for the model (1.4). McLachlan and Peel (2000a) suggest $q(\alpha) = \chi^2_{p,(1-\alpha)}$, the $(1-\alpha)$-quantile of the χ^2 distribution with p degrees of freedom. Hennig (2004) showed that this method is not breakdown-robust.

TCLUST. Gallegos (2002) and Gallegos and Ritter (2005) proposed the "spurious outliers model" as a probabilistic framework for robust crisp cluster analysis based on Gaussian distributions (more recent theoretical results are given in Gallegos and Ritter (2013)). García-Escudero et al. (2008) proposed a modification allowing different clusters to have different weights. This amounts to maximizing the weighted likelihood function

$$\left\{ \prod_{j=1}^{G} \prod_{i\in R_j} \pi_j \phi(x_i;\mu_j,\Sigma_j) \right\} \left\{ \prod_{i\notin R} g_i(x_i) \right\}, \tag{1.6}$$

where $R = \cup_{j=1}^{G} R_j$ is the set of indexes of clustered observations with $\#\{R\} = [n(1-\alpha)]$, and $g_i(\cdot)$ are the densities modeling outlying observations, π_j is the jth cluster's weight with $\pi_j \in [0,1]$ and $\sum_{j=1}^{G} \pi_j = 1$. Let θ be the unknown parameter vector including all triplets π_j, μ_j, Σ_j for all $j = 1, 2, \ldots, G$ (in the original spurious outlier model, $\pi_j = 1\ \forall j$, defining a "fixed partition model" with parameters for the cluster membership of every point). $\alpha > 0$ is the trimming rate, i.e., the proportion of points interpreted as outliers/noise. The authors show that for maximization of (1.6) it suffices to maximize of the left-hand side term of (1.6) and to "trim" the outliers, so that the problem boils down to

$$\theta^{\text{tclust}} := \underset{\theta\in\Theta, \#\{R\}=[n(1-\alpha)]}{\arg\max} \sum_{j=1}^{G} \sum_{i\in R_j} \left(\log \pi_j + \log \phi(x;\mu_j,\Sigma_j)\right). \tag{1.7}$$

As for the previously introduced methods, constraints will be discussed in detail in Section 3.1 in Coretto and Hennig (2016). Under mild regularity conditions, García-Escudero et al. (2008) propose an algorithm for computing θ_n^{tclust} and provide consistency theory. The TCLUST methodology is implemented in R's `tclust` package by Fritz et al. (2012). Partition methods with trimming started with the trimmed k-means proposal of Cuesta-Albertos et al. (1997).

2 RIMLE/OTRILME computing

In this section we provide details about computation of the RIMLE/OTRIMLE. We also give a detailed description of the initialization strategy adopted in the comparative Monte Carlo experiment. References to equations in brackets in this section are to Coretto and Hennig (2016).

2.1 RIMLE numerical approximation

Fix three computing parameters `em.tol, min.det, min.prc`, with default values `em.tol` $= 10^{-6}$, and both `min.det` and `min.prc` with default value equal to the smallest positive floating point number representable on the computer (that is $2.225074 \times 10^{-308}$ for the R package used to code the numerical experiments in this paper). For any given $\delta, \pi_{\max}, \gamma$ define an initial choice of $\theta^{(0)} \in \Theta$. Let $s = 0, 1, 2, \ldots$ be the iteration index of the algorithm, so that $\theta^{(s)}$ is the value of the parameter θ computed at the sth iteration. In each step of the EM-algorithm perform the following substeps (complexity is caused by the requirement to deal with a number of potential degeneracies):

1. compute $\tau_j(x_i; \theta^{(s)})$ for all $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, G$ (these quantities are given as initial values for $s = 0$);
2. if for some $i = 1, 2, \ldots, n$, $\tau_j(x_i, \theta^{(s)}) = 0$ (numerically) for all $j = 0, 1, \ldots, G$;
   - 2.a. if $\delta > 0$ set $\tau_0(x_i, \theta^{(s)}) \hookleftarrow 1$ and $\tau_j(x_i, \theta^{(s)}) \hookleftarrow 0$ for all $j = 1, 2, \ldots, G$;
   - 2.b. if $\delta = 0$ assign x_i to the cluster $j*$ for which the Euclidean distance to its center is minimized, then set $\tau_{j*}(x_i, \theta^{(s)}) \hookleftarrow 1$ and $\tau_j(x_i, \theta^{(s)}) \hookleftarrow 0$ for all $j \neq j*$;
3. if for some $j \neq 0$, $\tau_j(x_i, \theta^{(s)}) = 0$ (numerically) for all $i = 1, 2, \ldots, n$, set $\theta^{em} \hookleftarrow \theta^{(s)}$ stop the algorithm and record a flag;
4. Compute

$$\pi_j^{(s+1)} = \frac{1}{n} \sum_{i=1}^{n} \tau_j(x_i, \theta^{(s)}) \qquad \text{for all } \ j = 0, 1, \ldots, G,$$

$$\mu_j^{(s+1)} = \frac{1}{n\pi_j^{(s)}} \sum_{i=1}^{n} \tau_j(x_i, \theta^{(s)})\, x_i \qquad \text{for all } \ j = 1, 2, \ldots, G,$$

$$\Sigma_j^{(s+1)} = \frac{1}{n\pi_j^{(s)}} \sum_{i=1}^{n} \tau_j(x_i, \theta^{(s)})\, (x_i - \mu_j^{(s+1)})(x_i - \mu_j^{(s+1)})' \quad \text{for all } \ j = 1, 2, \ldots, G.$$

5. if $\pi_0^{(s+1)} > \pi_{\max}$, set $\theta^{em} \hookleftarrow \theta^{(s)}$, stop the algorithm and record a flag;
6. if $\pi_j^{(s+1)} <$`min.prc`, set $\theta^{em} \hookleftarrow \theta^{(s)}$, stop the algorithm and record a flag;
7. For all $j = 1, 2, \ldots, G$ compute the spectral decomposition of $\Sigma_j^{(s+1)} = U_j^{(s+1)} \Lambda_j^{(s+1)} U_j^{(s+1)'}$, where $\Lambda_j^{(s+1)} = \mathrm{diag}(\lambda_{1,j}^{(s+1)}, \ldots, \lambda_{p,j}^{(s+1)})$ is the diagonal matrix of eigenvalues of $\Sigma_j^{(s+1)}$,

and $U_j^{(s+1)}$ is the matrix of the normalized corresponding eigenvectors. If $\prod_{k=1}^{p} \lambda_{k,j}^{(s+1)} <$ `min.det`: find the smallest $\underline{k}$ such that there exists $\underline{\lambda}$ for which $\underline{\lambda}^{\underline{k}} \prod_{h=\underline{k}+1}^{p} \lambda_{h,j}^{(s)} =$ `min.det`, and set $\lambda_{k,j}^{(s+1)} \hookleftarrow \underline{\lambda}$ for all $k = 1, 2, \ldots, \underline{k}$. Correct the covariance matrix by setting

$$\Sigma_j^{(s+1)} \hookleftarrow U_j^{(s+1)} \underline{\Lambda}_j^{(s+1)} U_j^{(s+1)\prime} \quad \text{with} \quad \underline{\Lambda}_j^{(s+1)} \hookleftarrow \operatorname{diag}\left(\underline{\lambda}, \ldots, \underline{\lambda}, \lambda_{\underline{k}+1,j}^{(s+1)}, \ldots, \lambda_{p,j}^{(s+1)}\right).$$

8. if $|l_n(\theta^{(s+1)}) - l_n(\theta^{(s)})| \leq$ `em.tol`:
   - 8.a. if $\lambda_{\max}(\theta^{(s+1)})/\lambda_{\min}(\theta^{(s+1)}) \leq \gamma$, set $\theta^{em} \hookleftarrow \theta^{(s+1)}$;
   - 8.b. if $\lambda_{\max}(\theta^{(s+1)})/\lambda_{\min}(\theta^{(s+1)}) > \gamma$, set $\lambda_{jk}^{em} \hookleftarrow \max\{\lambda_{jk}^{(s+1)}, \lambda_{\max}(\theta^{(s+1)})/\gamma\}$; for all $j = 1, 2, \ldots, G$ and $k = 1, 2, \ldots, p$. Then adjust the corresponding eigenvalue matrices setting $\Sigma_j^{em} \hookleftarrow U_j^{(s+1)} \Lambda_j^{em} U_j^{(s+1)\prime}$, and record a flag.
   - 8.c. stop iterating and return θ^{em}

2.2 OTRIMLE numerical approximation

Fix β and the following additional computing parameters: `lower.icd`, `upper.icd`, `opt.selector`. Solve the program (3.12) by "golden section search algorithm" (GSSA) of Kiefer (1953) over the candidate set $\{0\} \cup$[`lower.icd, upper.icd`]. Each step of the golden search requires a RIMLE evaluation as described previously. In the standard GSSA the algorithm starts from a point within (`lower.icd`, `upper.icd`) computed using the "golden ratio". If a better candidate is available this can be given as an additional input. At each RIMLE evaluation, do the following:

1. If `opt.selector=TRUE` (default): discard all δ values for which the RIMLE computation ends with one or more flags. If all δ values are discarded set `opt.selector=FALSE` and find again an OTRIMLE solution.

2. if `opt.selector=FALSE`: only discard δ values for which the RIMLE computation ends with an error flag related to $\pi_{\max}$ (the previous substep 5).

The OTRIMLE could be based on evaluating the RIMLE on a grid of δ values. We found that the GSSA increases efficiency by a big margin. In most numerical experiments we found that no more than 30 RIMLE evaluations are required even for extremely small `lower.icd` and large `upper.icd`. Notice that the GSSA only requires that the target function has at least a local minimum in the search domain. The golden search is performed over the interval [`lower.icd, upper.icd`], and then the solution is compared with the RIMLE at $\delta = 0$. This is because we always check against the solution without the noise component. One could set `lower.icd`=0, but there is a limit in approximating positive real numbers on a computer. For high dimensional problems, where density values are often extremely small, and there could still be a difference between $\theta_n(0)$ and $\theta_n(\varepsilon)$, where ε is the smallest positive number representable on a computer. The constraint implementation and the use of `opt.selector` in the OTRIMLE will be explained in detail below.

Remark 1. There is an intimate connection between $\delta, \gamma, \pi_{\max}$. When δ is too large it happens that too many points are assigned to the noise component. This will drain points away from Gaussians, which may cause small eigenvalues for them. For a given γ, $\pi_{\max}$ there will be $\delta_{\max}$ such that for $\delta > \delta_{\max}$ the RIMLE solution is on the border of the parameter space. Fixed γ and $\pi_{\max}$, such a $\delta_{\max}$ depends on the P that generates data, and this is unknown. So when $\pi_{0,n}$ is near $\pi_{\max}$, and/or $\lambda_{\min}(\theta_n(\delta))$ is near 0, this is an indication that δ is approaching $\delta_{\max}$. This means that for a given pair $(\gamma, \pi_{\max})$ there will be values of δ producing RIMLE solutions in the interior of Θ, and δ values producing RIMLE solutions on the border. Moreover the smallest eigenvalue that makes the RIMLE well defined also depends on δ.

The algorithm should be able to find small clusters with potentially small variance, and therefore we do not build the lower bound for the eigenvalues in the algorithm. Instead we avoid singularities by adjusting the minimum number of eigenvalues such that the determinant of the corresponding covariance matrix is positive at machine precision in any single EM step. The eigenratio constraint is then enforced only at the end of the EM run. This allows the algorithm to fully explore concentrated clusters. The parameter `min.prc` avoids RIMLE fits with fewer than G components, because this would mean that the OTRIMLE target function is not always evaluated at versions of $\theta_n(\cdot)$ with G mixture components.

The `opt.selector` switch has the role to discriminate between δ values that provide solutions on the border of the parameter space and δ values that provides solution in the interior. When `opt.selector=FALSE`, solutions violating the $\pi_{\max}$ condition are still discarded. This is because the violation of (3.7) is an indication that a too large δ has been explored. In other words $\pi_{\max}$ has two roles: (i) it makes the RIMLE functional well defined (together with the eigenratio constraint); (ii) from the practical viewpoint is is used to indirectly define the maximum level of the improper density. The default choice used in the main paper is to set `opt.selector=TRUE`. `opt.selector=FALSE` may be chosen if the researcher is interested in finding solutions on the border of the parameter space, which may particularly include very small (if not spurious) clusters with low within-cluster variation.

Regarding `opt.selector`, experience shows that within computation of the OTRIMLE at least if no random initializations are used, it is advantageous to discard δ-values for which the RIMLE-solution ends up at the border of the parameter space (unless this happens for all δ), enforcing (3.5), (3.6) or zero component proportions, because for these cases $D(\delta)$ may behave irregularly and the golden search may be led astray. This particularly rules out too large values of δ, which will trigger (3.6). Experience shows that the set of values of discarded δ-values is usually a single interval, often containing the largest values in the candidate set. Discarded solutions can be brought back if they are needed in later stages of the golden section search, which is not normally necessary.

2.3 Choice of initial values

Biernacki et al. (2003) and Karlis and Xekalaki (2003) studied the problem of initializing the EM for computing MLE of Gaussian mixtures models. Both works found that the EM solutions strongly depend on initial values. The EM convergence is rather slow, and often the EM approximates the best local maximum of the likelihood function in the proximity of the initial parameter values. One may assign the n points to G clusters at random and compute the EM solution for each of the many random initializations, and then one chooses the EM approximation with the largest likelihood. In this paper we are concerned with robustness. A small group of points not belonging to any Gaussian group may produce a bad local maximum in the objective function so that the EM doesn't move from the initials. So while many random initializations allow to explore different regions of the parameter space, in some data sets it is difficult to avoid random initialisations in which no cluster is contaminated in such a way that it leads to a bad local optimum. Furthermore when we compute the OTRIMLE, this is based on the RIMLE evaluation for several δ values. This means that the computational load implied by multiple random initials is multiplied by the number of δ values explored.

Instead of relying on random initials, we start partitioning the sample points into G groups plus a noise component. Then cluster memberships are used to initialize the pseudo-posterior probabilities. The main issue during initialization is not to start with too small clusters. Therefore we consider as valid initial partitions only those containing at least `min.pr`$\times n$ observations in each group (the default is `min.pr`=0.005). Nearest neighbor based clutter/noise detection proposed by Byers and Raftery (1998) is applied to identify an initial set of noise points. The latter is implemented in the `NNclean` function in R's `prabclus` package by Hennig and Hausdorf (2015). Agglomerative hierarchical clustering based on ML criteria for Gaussian mixture models proposed by Banfield and Raftery (1993) is then used for finding initial Gaussian groups in the non-noise. This has been implemented in the `hc` function of R's `mclust` package by Fraley et al. (2012). The initialization consists of the following steps:

1. Perform `NNclean` to isolate points likely to belong to the noise component. Typically this step finds more noise than it is actually needed.
2. On the subset of non-noise points obtained from step 1, perform `hc` from the `mclust` package, allowing for unequal covariance matrices.
3. If the partition obtained in steps 1 and 2 is valid, go to the final step 8, otherwise continue with the following step.
4. Add the too small clusters identified in step 3 to the noise component; on the remaining points do step 2 again. If the partition is now valid go to the final step 8, otherwise continue with the following step.
5. Increment the noise component identified in step 4 by adding the too small clusters from step 4. On the remaining points perform the k-means partitioning with G clusters with few random initializations. If the partition is now valid go to the final step 8, otherwise continue with the following step.
6. Add the too small clusters from step 5 to the noise component identified in 5. Repeat a maximum of 1000 random assignments and stop when the smallest cluster has at least `min.prc`$\times n$ points. For each random assignment, draw G data points at random, cluster points around the G random centers based on Euclidean distance minimization. If a valid partition is found, go to the final substep 8, otherwise continue with the following step.
7. If after 1000 random iteration in step 6 no valid partition has been obtained, return to the last (invalid) random partition and go to the final substep 8.
8. Set $\tau_j(x_i, \theta^{(0)}) = \mathbb{1}\{x_i \in j\text{th initial group}\}$ for $j = 1, 2, \ldots, G$, and $\tau_0(x_i, \theta^{(0)}) = \mathbb{1}\{x_i \in \text{initial noise component}\}$. Use the initial pseudo–posterior weights to initialize the RIMLE computation previously described.

The extensive simulation study discussed in Coretto and Hennig (2016), showed that this initialization performs much better than the random initialization. Usually steps 1–5 can produce sensible initial values even under the most adverse experimental design. We could hardly produce artificial data sets where steps 6–8 are needed in order to have an initial partition.

Note that rimle.o, rimle.op, tclust.o and tclust.op in the simulation study require to compute the RIMLE solution and the TCLUST solution for different values of δ and trimming level respectively. For a given dataset all RIMLE and TCLUST solutions are computed from the same starting partition as defined previously.

2.4 Computation of the MCD functional

This does not concern RIMLE and OTRIMLE computation, but the end of Section 4 of Coretto and Hennig (2016).

The vector of θ_C with cluster parameters is based on the MCD location and scatter computed at each P_j in the DGP. When P_j is the Gaussian distribution m_j and S_j coincide with the Gaussian expectation and covariance matrix. However, in the following simulation study deviations from Gaussianity are considered by exploring P_j that are: (i) non-central multivariate Student-t distributed, (ii) non-central multivariate Student-t on some marginals and Gaussian on other marginals. In these cases, appropriate MCD parameters have to be derived to obtain the reference θ_C. Although the required MCD parameters exist for the cases we will consider here (Cator and Lopuhaä, 2012), it is not easy to compute them in closed form. For non-Gaussian elements of the DGP we computed the MCD parameters by Monte Carlo integration based on 10^6 random draws, assuming zero mean and unit scale matrix and using affine equivariance of the MCD for transformation, by the `cov.rob` function of the R package `MASS` (Venables and Ripley, 2002).

2.5 Additional remarks for the computation of misclassification rates

Given the size of the comparative simulation study, some precaution is taken dealing with situations where the algorithm fails to provide a clustering solution for numerical reasons. In each Monte Carlo replicate, given the initial clustering, the software goes trough the following steps:

1. A clustering method is applied. If a valid solution is available go to step 3, otherwise perform step 2.
2. The method is applied setting its iteration number limit equal to 1. If a solution is available go to step 3, otherwise set the corresponding misclassification rate equal to `NA` (i.e. not available).
3. Compute the misclassification rate using formula (4.2).

`NA` cases only happened in 11 replicates out of 1000 for ot.tclust.p when applied to Sunspot.1h, and for no other method. These cases have not been considered for computing Monte Carlo averages (and related standard errors). OTRIMLE based methods always provided a solution based on previous steps 1 and 3.

Note that for the majority of the Monte Carlo replicates initialization was successfully performed based on steps 1–3 and 8 described in Section 2.3. The additional initialization step 4 was required for a minor fraction of replicates on few DGPs.

3 Definition of DGPs

The vector of θ_C with cluster parameters is based on the MCD location and scatter computed at each P_j in the DGP scaled for consistency at Gaussian distributions. When P_j is the Gaussian distribution m_j and P_j coincide with the expectation and covariance matrix under P_j. Apart from the normal case, we considered situations where P_j is: (i) non-central multivariate Student-t distributed, (ii) non-central multivariate Student-t on some marginals and normal on other marginals. When P_j is not normal, appropriate MCD parameters have to be derived. Even though existence of the MCD parameters is guaranteed for a wide class of distributions (Cator and Lopuhaä, 2012), it is not easy to compute them in closed form solutions. For non-Gaussian elements of the DGP we computed the MCD parameters by Monte Carlo integration based on 10^6 random draws. MCD has been computed with the `cov.rob` function of the `MASS` R package (Venables and Ripley, 2002).

For a covariance matrix Σ define $Q(\Sigma) = U\sqrt{\Lambda}$, where $\sqrt{\Lambda}$ is the square root of the diagonal matrix of eigenvalues of Σ, and U is the corresponding matrix of unit eigenvectors. Hence $\Sigma = Q(\Sigma)Q(\Sigma)'$. Let $\mathrm{N}(\mu, \Sigma)_p$ denote the p-dimensional Gaussian distribution with mean vector $\mu \in \mathbb{R}^p$ and $p \times p$ covariance matrix Σ. Let $t_v(\mu, \Sigma)_p$ be the p-dimensional non-central Student-t distribution with v degrees of freedom, expectation μ, and covariance matrix Σ. Notice that usually the non-central Student-t distribution is parametrized in terms of a scale matrix which is given by $(v-2)\Sigma/v$. Define 0_p as the p-dimensional zero vector, while I_p is the usual $p \times p$ identity matrix. According to the notation in Coretto and Hennig (2016) we will make use of four types of cluster generating distributions P_j for which MCD parameters are approximated as follows:

Gaussian: $P_j = \mathrm{N}(\mu_j, \Sigma_j)_p$. In this case $m_j = \mu_j$ and $S_j = \Sigma_j$.

Student-t: $P_j = \mathrm{t}_3(\mu_j, \Sigma_j)_2$. We computed the Monte Carlo approximation of the MCD center and scale for a $t_3(0_2, I_2)_2$. Let them be m_{MC} and S_{MC}, respectively. Using the affine equivariance we define $m_j = \mu_j + Q(\Sigma_j)m_{MC}$ and $S_j = Q(\Sigma_j)S_{MC,1}Q(\Sigma_j)'$. This results in $m_{MC} = 0_2$, whereas S_{MC} is diagonal matrix with $S_{MC}[k,k] = 0.3643$ for $k = 1, 2$.

GaussT: $P_j = \mathrm{GaussT}(\mu_j, \Sigma_j)$ is a 20-dimensional distribution with $\mathrm{N}(\mu_j, \Sigma_j)_2$ on the first two marginals and an independent $\mathrm{t}_3(0_{18}, I_{18})_{18}$ on the remaining marginals. Let m_{MC} and S_{MC} the Monte Carlo approximation of the MCD center and scale of a $\mathrm{GaussT}(0_{20}, I_{20})$ distribution. Let $\dot{\Sigma}_j$ be a 20×20 matrix obtained by replacing the elements of the block $\{I_{20}[r,c]\}_{r=1,2}^{c=1,2}$ with Σ_j; let $\dot{\mu}_j$ be the vector obtained by replacing the first two elements of 0_{20} with those of μ_j. Using the affine equivariance we define $m_j = \dot{\mu}_j + Q(\dot{\Sigma}_j)m_{MC}$ and $S_j = Q(\dot{\Sigma}_j)S_{MC}Q(\dot{\Sigma}_j)'$. Here $m_{MC} = 0_{20}$ while S_{MC} is diagonal matrix with $S_{MC}[k,k] = 0.9829$ for $k = 1, 2$, and $S_{MC}[k,k] = 0.3247$ for $k = 3, \ldots, 20$.

TGauss: $P_j = \mathrm{TGauss}(\mu_j, \Sigma_j)$ is 20-dimensional distribution with $\mathrm{t}_3(\mu_j, \Sigma_j)_2$ on the first two marginals, and an independent $\mathrm{N}(0_{18}, I_{18})_{18}$ on the remaining marginals. Let m_{MC} and S_{MC} be respectively Monte Carlo approximation of the MCD center and scale for a $\mathrm{TGauss}(0_{20}, I_{20})$ distribution. Let $\dot{\Sigma}_j$ be a 20×20 matrix obtained by replacing the elements of the block $\{I_{20}[r,c]\}_{r=1,2}^{c=1,2}$ with Σ_j; let $\dot{\mu}_j$ be the vector obtained by replacing the first two elements of 0_{20} with those of μ_j. Using the affine equivariance we define $m_j = \dot{\mu}_j + Q(\dot{\Sigma}_j)m_{MC}$ and $S_j = Q(\dot{\Sigma}_j)S_{MC}Q(\dot{\Sigma}_j)'$. In this case $m_{MC} = 0_{20}$, whereas S_{MC} is diagonal matrix with $S_{MC}[k,k] = 0.5023$ for $k = 1, 2$, and $S_{MC}[k,k] = 0.9739$ for $k = 3, \ldots, 20$.

Let $\mathrm{U}(a, b)$ with $a, b \in \mathbb{R}^2$ be the uniform distribution on the rectangle $[a_1, b_1] \times [a_2, b_2]$. Denote by $\mathrm{UGauss}(a, b)$ a 20-dimensional distribution that has a $\mathrm{Uniform}(a, b)$ on the first two marginals and an independent $\mathrm{N}(0_{18}, I_{18})_{18}$ on the remaining marginals. The 24 DGPs considered in the Monte Carlo comparison are defined as follows:

WideNoise.2l $G = 2$, $\pi_0 = 0.05$, $\pi_1 = 0.75$, $\pi_2 = 0.2$. $P_0 = \mathrm{U}(a, b)_2$ with $a = (-10, 10)'$ and $b = (-5, -15)'$. $P_j = \mathrm{N}(\mu_j, \Sigma_j)_2$ for $j = 1, 2$, where: $\mu_1 = (0, 5)'$, $\Sigma_1 = 0.2I_2$ $\mu_2 = (1, 5)'$, $\Sigma_2 = I_2$.

WideNoise.2h high dimensional version of WideNoise.2l with $P_0 = \mathrm{UGauss}(a, b)$ and $P_j = \mathrm{N}(\mu_j, \Sigma_j)_{20}$ for $j = 1, 2$. All parameters are the same as in WideNoise.2l except that now $\mu_{j,k} = 0$ and $\Sigma_j[k, k] = 1$ for $j = 1, 2$, and $k = 3, 4, \ldots, 20$, while $\Sigma_j[r, c] = 0$ for all $r \neq c$ with r and c taking values in $\{3, 4, \ldots, 20\}$.

WideNoise.3l $G = 3$, $\pi_0 = 0.1$, $\pi_j = 0.3$ for $j = 1, 2, 3$. $P_0 = \mathrm{U}(a, b)_2$ with $a = (-10, 10)'$ and $b = (-5, -15)'$. $P_j = \mathrm{N}(\mu_j, \Sigma_j)_2$ for $j = 1, 2, 3$, where: $\mu_1 = (0, 3)'$, $\Sigma_1[1, 1] = \Sigma_1[2, 2] = 1$, $\Sigma_1[1, 2] = 0.5$; $\mu_2 = (7, 1)'$, $\Sigma_2[1, 1] = \Sigma_2[2, 2] = 2$, $\Sigma_2[1, 2] = -1.5$ $\mu_3 = (5, 9)'$, $\Sigma_3[1, 1] = \Sigma_3[2, 2] = 2$, $\Sigma_3[1, 2] = 1.3$.

WideNoise.3h high dimensional version of WideNoise.3l with $P_0 = \mathrm{UGauss}(a, b)$ and $P_j = \mathrm{N}(\mu_j, \Sigma_j)_{20}$ for $j = 1, 2, 3$. All parameters are the same as in WideNoise.3l except that now $\mu_{j,k} = 0$ and $\Sigma_j[k, k] = 1$ for all $j = 1, 2, 3$ and $k = 3, 4, \ldots, 20$, while $\Sigma_j[r, c] = 0$ for all $r \neq c$ with r and c taking values in $\{3, 4, \ldots, 20\}$.

SideNoise.2l $G = 2$, $\pi_0 = 0.1$, $\pi_1 = 0.1$, $\pi_2 = 0.8$. $P_0 = \mathrm{U}(a, b)_2$ with $a = b = (-50, 5)'$. $P_j = \mathrm{N}(\mu_j, \Sigma_j)_2$ for $j = 1, 2$, where: $\mu_1 = (-10, 5)'$, $\Sigma_1 = 0.4I_2$ $\mu_2 = (3, 13)'$, $\Sigma_2[1, 1] = \Sigma_2[2, 2] = 1.5$, $\Sigma_2[1, 2] = -1.1$.

SideNoise.2h high dimensional version of SideNoise.2l with $P_0 = \mathrm{UGauss}(a, b)$ and $P_j = \mathrm{N}(\mu_j, \Sigma_j)_{20}$ for $j = 1, 2$. All parameters are the same as in SideNoise.2l except that now $\mu_{j,k} = 0$ and $\Sigma_j[k, k] = 1$ for $j = 1, 2$ and $k = 3, 4, \ldots, 20$, while $\Sigma_j[r, c] = 0$ for all $r \neq c$ with r and c taking values in $\{3, 4, \ldots, 20\}$.

SideNoise.3l $G = 3$, $\pi_0 = 0.1$, $\pi_1 = 0.15, \pi_2 = 0.35, \pi_3 = 0.4$. $P_0 = \mathrm{U}(a, b)_2$ with $a = b = (-50, 5)'$. $P_j = \mathrm{N}(\mu_j, \Sigma_j)_2$ for $j = 1, 2, 3$, with parameters set as in WideNoise.3l except that $\mu_1 = 0_2$.

SideNoise.3h high dimensional version of SideNoise.3l. Again $P_0 = \mathrm{UGauss}(a, b)$ with $a = b = (-50, 5)'$, and $P_j = \mathrm{N}(\mu_j, \Sigma_j)_{20}$ for $j = 1, 2, 3$. All Gaussian parameters are the same as in WideNoise.3h except that now $\mu_{1,1} = \mu_{1,1} = 0$.

SunSpot.3l $G = 3$, $\pi_0 = 0.025$, $\pi_1 = 0.325$ for $j = 1, 2, 3$. $P_0 = \mathrm{U}(a, b)_2$ with $a = b = (10^5, 10^5 + 10)'$. $P_j = \mathrm{N}(\mu_j, \Sigma_j)_2$ for $j = 1, 2, 3$, with all Gaussian parameters set as in WideNoise.3l.

SunSpot.3h high dimensional version of SunSpot.3l. Proportion parameters and a and b set as SunSpot.3l, but $P_0 = \mathrm{UGauss}(a, b)_2$ while $P_j = \mathrm{N}(\mu_j, \Sigma_j)_{20}$ for $j = 1, 2, 3$ with all parameters set as in WideNoise.3h.

SunSpot.5l $G = 5$, $\pi_0 = 0.002$, $\pi_1 = \pi_4 = 0.1497$, $\pi_2 = \pi_5 = 0.2994$, $\pi_3 = 0.0998$. $P_0 = \mathrm{U}(a, b)_2$ with $a = b = (30, 40)'$ $P_j = \mathrm{N}(\mu_j, \Sigma_j)_2$ with $\mu_1, \mu_2, \mu_3, \Sigma_1, \Sigma_2, \Sigma_3$ set as in WideNoise.3l, $\mu_4 = (-11, 5)'$, $\Sigma_4 = 0.5I_2$, $\mu_5 = (-9, 5)'$, $\Sigma_5 = 2.5I_2$.

SunSpot.5h high dimensional version of SunSpot.5l with $P_0 = \mathrm{UGauss}(a, b)$ and $P_j = \mathrm{N}(\mu_j, \Sigma_j)_{20}$ for $j = 1, 2, 3, 4, 5$. All parameters are the same as in Sunspot.5l except that now $\mu_{j,k} = 0$ and $\Sigma_j[k, k] = 1$ for all $j = 1, 2, 3, 4, 5$ and $k = 3, 4, \ldots, 20$, while $\Sigma_j[r, c] = 0$ for all $r \neq c$ with r and c taking values in $\{3, 4, \ldots, 20\}$.

TGauss.3l $G = 3$, $\pi_0 = 0$, $\pi_j = 1/3$ for $j = 1, 2, 3$. $P_j = t_3(\mu_j; \Sigma_j)_2$ with μ_j and Σ_j set as in WideNoise.3l for $j = 1, 2, 3$. Note that Σ_j here denotes the covariance matrix of the multivariate non-central t_3-distribution.

TGauss.3h high dimensional version of TGauss.3l. Everything is set as in the low dimensional version except now that $P_j = \text{TGauss}(\mu_j; \Sigma_j)$ for $j = 1, 2, 3$.

TGauss.5l $G = 5$, $\pi_0 = 0$, $\pi_1 = \pi_4 = 0.15$, $\pi_2 = \pi_5 = 0.3$; $\pi_3 = 0.1$, $P_j = t_3(\mu_j; \Sigma_j)_2$ for all $j = 1, 2, \ldots, 5$, with all parameters set as in SunSpot.5l except that $\mu_4 = (-10, 5)'$, $\mu_5 = (3, 13)$, and $\Sigma_5 = 2.5 I_5$.

TGauss.5h high dimensional version of TGauss.5l. Everything is set as in the low dimensional version except now that $P_j = \text{TGauss}(\mu_j; \Sigma_j)$ for $j = 1, 2, \ldots, 5$.

GaussT.2l $G = 2$, $\pi_0 = 0$, $\pi_1 = 0.15$, $\pi_2 = 0.85$; $P_j = \text{N}(\mu_j; \Sigma_j)_2$ for $j = 1, 2$ with $\mu_1 = (-1, 0.5)'$ $\Sigma_1 = 0.2 I_2$, and $\mu_2 = (0.3, 1.3)'$ $\Sigma_2[1, 1] = \Sigma_2[2, 2] = 1; \Sigma_2[1, 2] = -0.8$.

GaussT.2h high dimensional version of TGauss.2l. Everything is set as in the low dimensional version except now that $P_j = \text{GaussT}(\mu_j; \Sigma_j)$ for $j = 1, 2$.

GaussT.3l $G = 3$, $\pi_0 = 0$, $\pi_j = 1/3$ and $P_j = \text{N}(\mu_j; \Sigma_j)_2$ for $j = 1, 2, 3$ with pairs $(\mu_j; \Sigma_j)$ fixed as in WideNoise.3l.

GaussT.3h high dimensional version of GaussT.3l. Everything is set as in the low dimensional version except now that $P_j = \text{GaussT}(\mu_j; \Sigma_j)$ for $j = 1, 2, 3$.

Noiseless.3l $G = 3$, $\pi_0 = 0$, $\pi_j = 1/3$ and $P_j = N(\mu_j; \Sigma_j)_2$ with μ_j, Σ_j set as in WideNoise.3l for all $j = 1, 2, 3$.

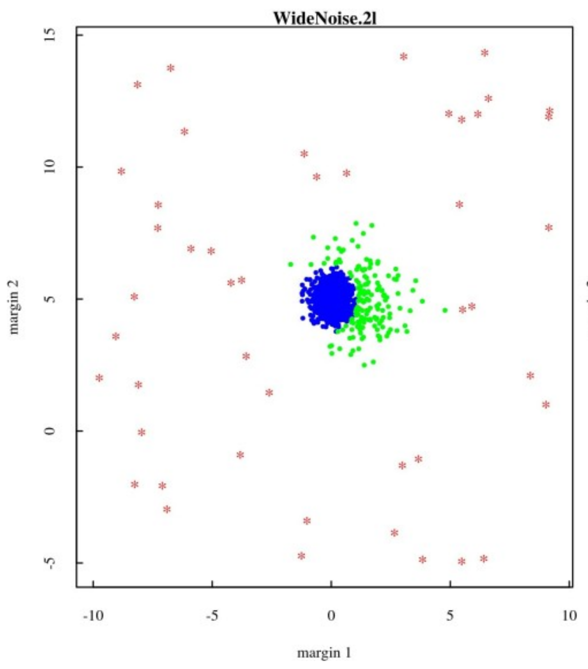
Noiseless.3h high dimensional version of Noiseless.3l. Everything is set as in the low dimensional version except now that $P_j = \text{N}(\mu_j; \Sigma_j)_{20}$ with parameters set as in WideNoise.3h for all $j = 1, 2, 3$.

Noiseless.5l $G = 5$, expected proportions set as in TGauss.5l; $P_j = \text{N}(\mu_j; \Sigma_j)_2$ with all μ_j and Σ_j parameters set as in TGauss.5l except that μ_j is divided by $\sqrt{3}$ for all $j = 1, 2, \ldots, 5$.

Noiseless.5h high dimensional version of Noiseless.5h. Difference with the lower dimensional version is that now $P_j = \text{N}(\mu_j; \Sigma_j)_{20}$ with $\mu_{j,k} = 0$ and $\Sigma_j[k, k] = 1$ for all $j = 1, 2, \ldots, 5$ while $\Sigma_j[c, r] = 0$ for all $r \neq c$ with the pair and r, c taking values in $\{3, 4, \ldots, 20\}$.

For a design with G clusters, a DGP produces n points by sampling from distributions $P_1, P_2, \ldots, P_G$ so that an expected number of points $n\pi_j$ is sampled under P_j for $j = 1, 2, \ldots, G$. For some designs noise is generated by adding a further component that is denoted by P_0. The expected proportion of points under P_0 is denoted by π_0. In each Monte Carlo replicate sampling is performed as follows: the vector $(n_0, n_1, \ldots, n_G)'$ is sampled from a Multinomial$(n, \pi_0, \pi_1, \ldots, \pi_G)$ distribution, then n_j points are independently sampled from P_j for all $j = 1, 2, \ldots, G$. Therefore, in each Monte Carlo sample is an i.i.d. sample from the mixture distribution $\pi_0 P_0 + \sum_{j=1}^{G} \pi_j P_j$.

4 Plots for the DGPs and misclassification rates from the simulation study



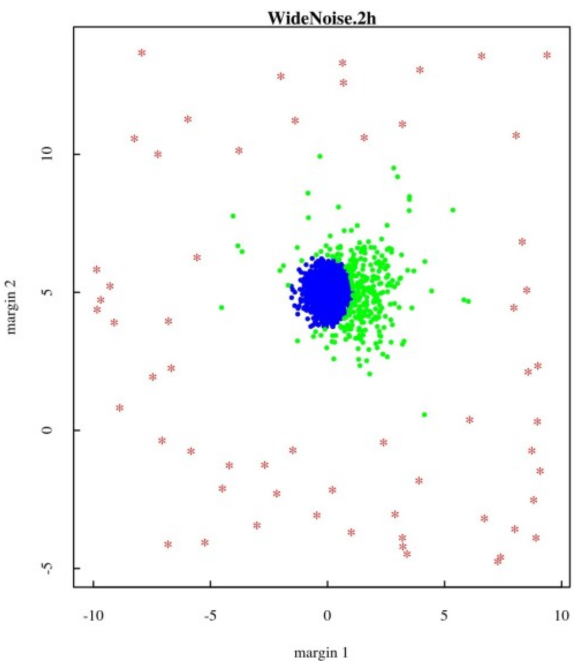


Figure 1: α-Gaussian Regions for WideNoise.2 designs, with $\alpha = 10^{-4}$. Red stars are points defined as noise while clusters are denoted by colored points.

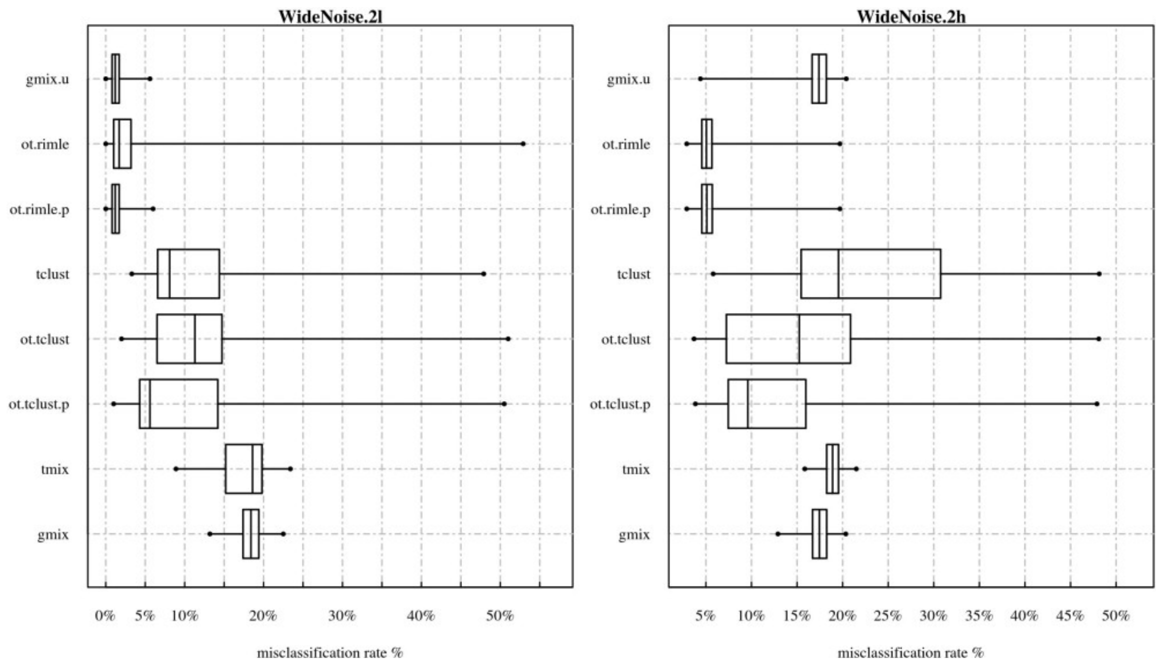


Figure 2: Boxplots for the Monte Carlo distribution of misclassification rates (%) for WideNoise.2 designs. Notice that the segment joining the whiskers' extremes always concides with the range of the distribution.

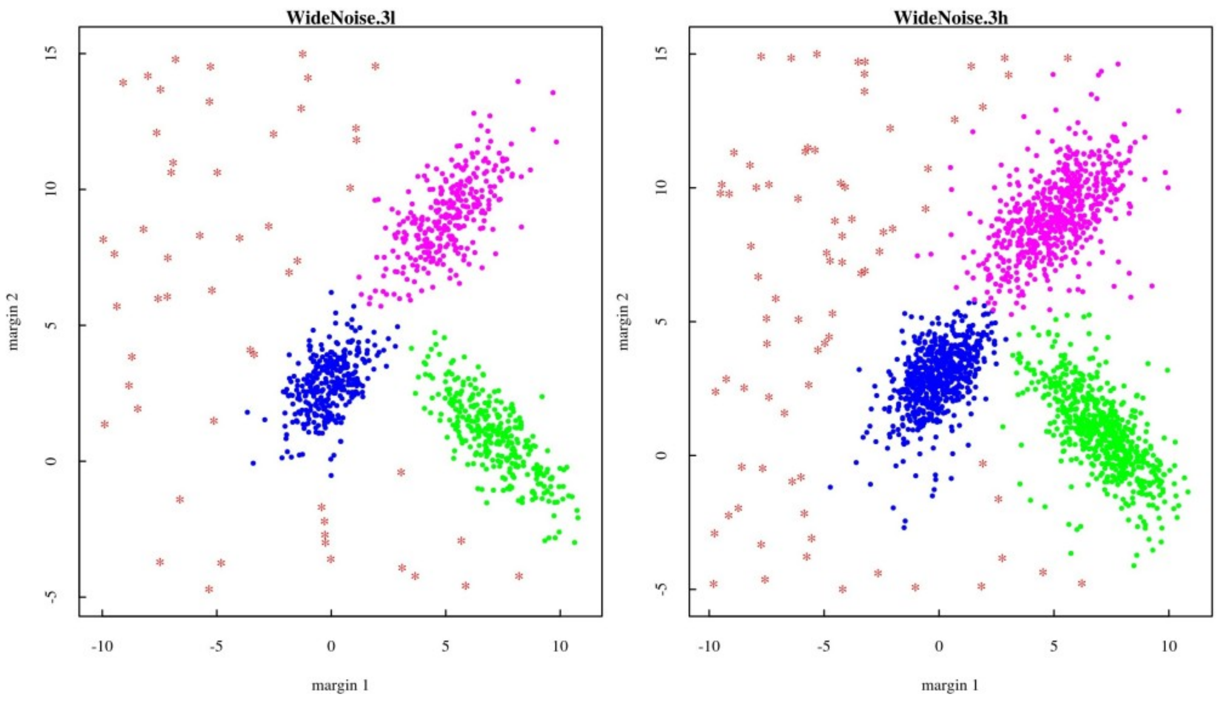


Figure 3: α-Gaussian Regions for WideNoise.3 designs, with $\alpha = 10^{-4}$. Red stars are points defined as noise while clusters are denoted by colored points.

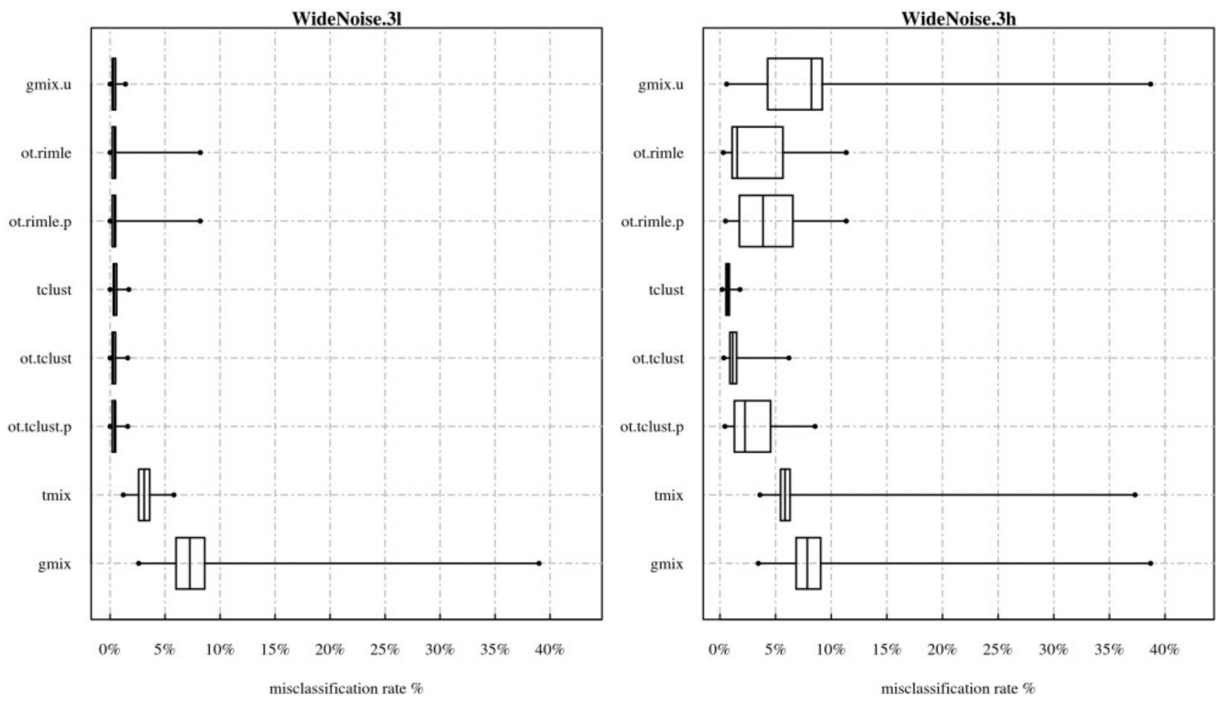


Figure 4: Boxplots for the Monte Carlo distribution of misclassification rates (%) for WideNoise.3 designs. Notice that the segment joining the whiskers' extremes always concides with the range of the distribution.

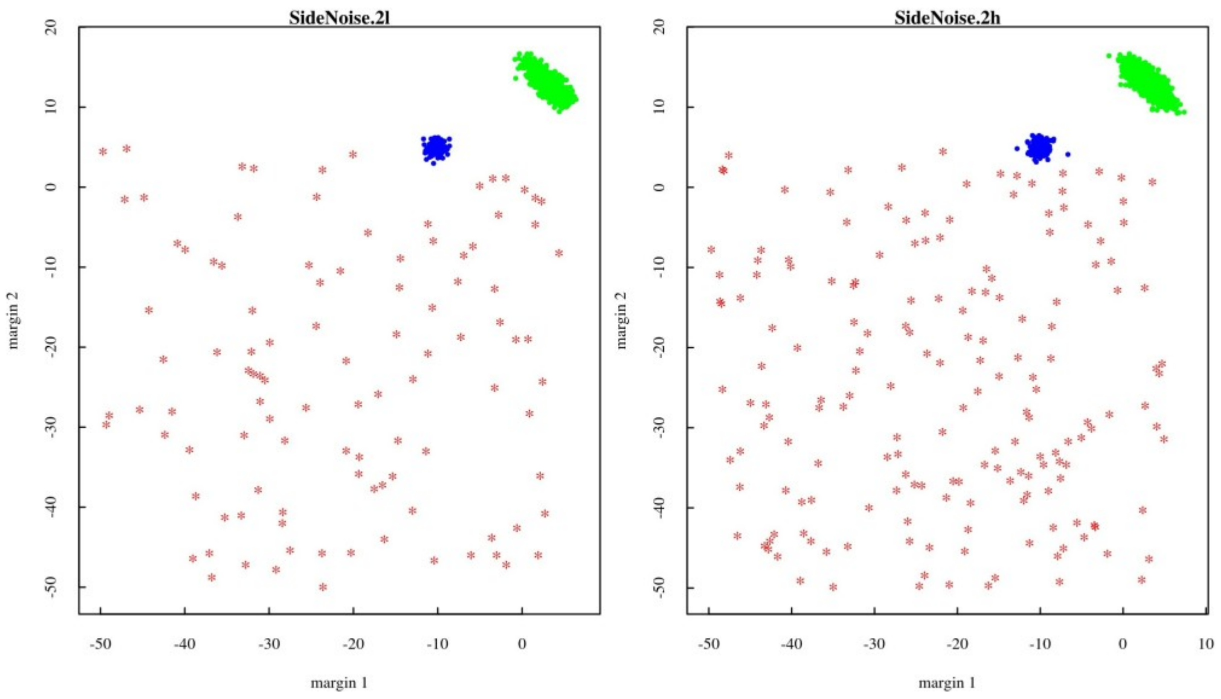


Figure 5: α-Gaussian Regions for SideNoise.2 designs, with $\alpha = 10^{-4}$. Red stars are points defined as noise while clusters are denoted by colored points.

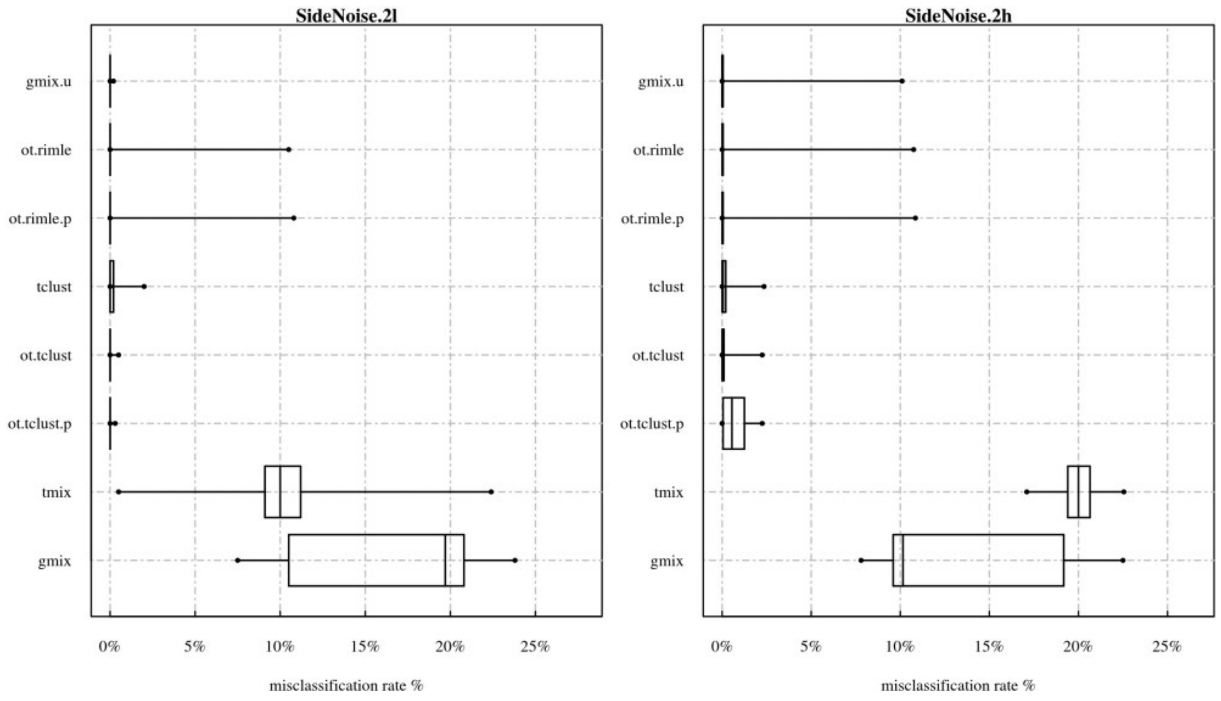


Figure 6: Boxplots for the Monte Carlo distribution of misclassification rates (%) for SideNoise.2 designs. Notice that the segment joining the whiskers' extremes always concides with the range of the distribution.

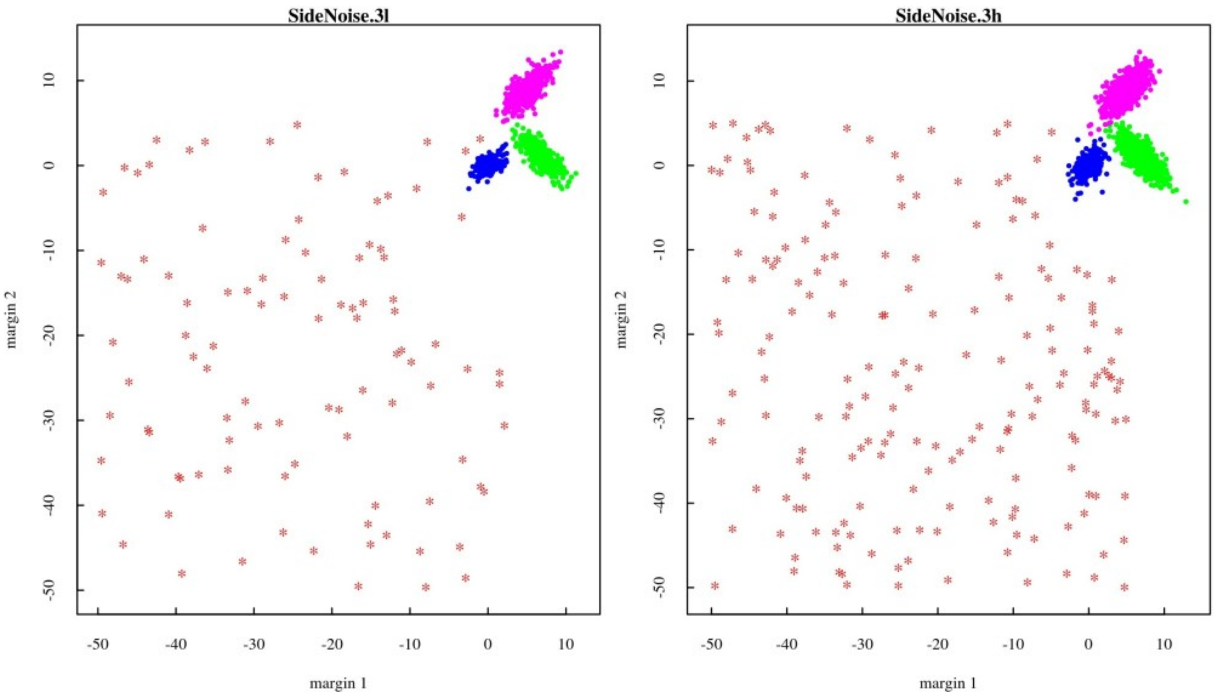


Figure 7: α-Gaussian Regions for SideNoise.3 designs, with $\alpha = 10^{-4}$. Red stars are points defined as noise while clusters are denoted by colored points.

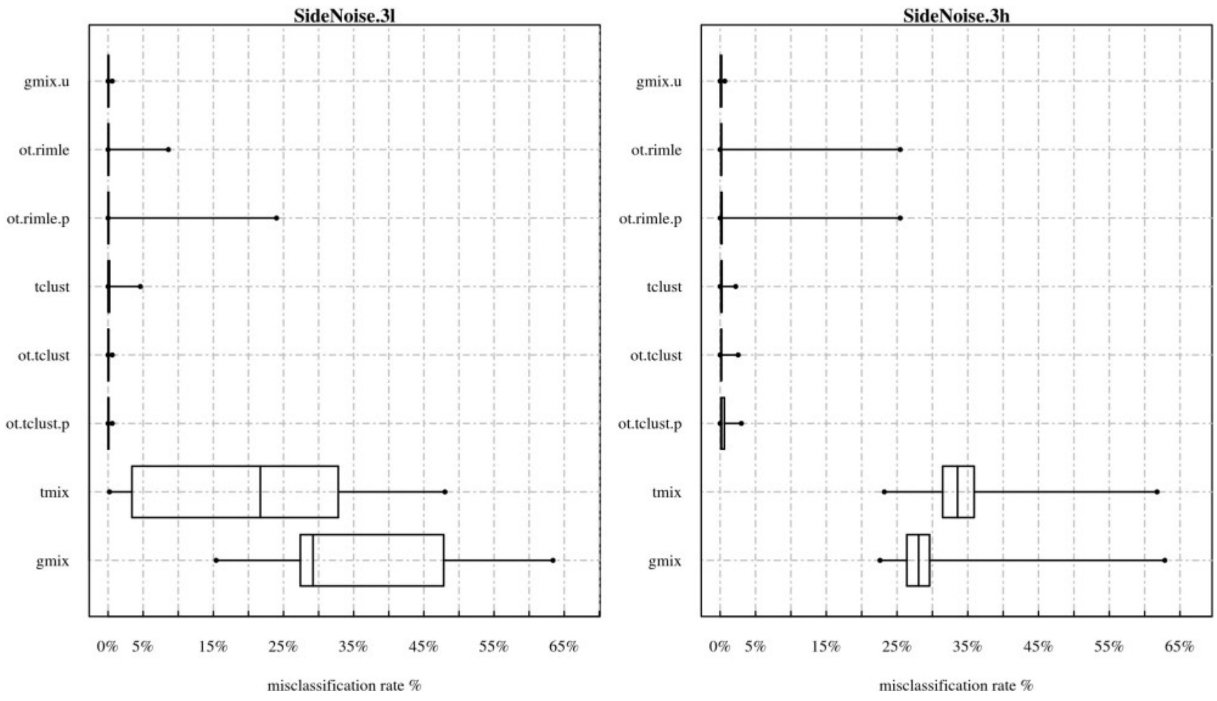


Figure 8: Boxplots for the Monte Carlo distribution of misclassification rates (%) for SideNoise.3 designs. Notice that the segment joining the whiskers' extremes always concides with the range of the distribution.

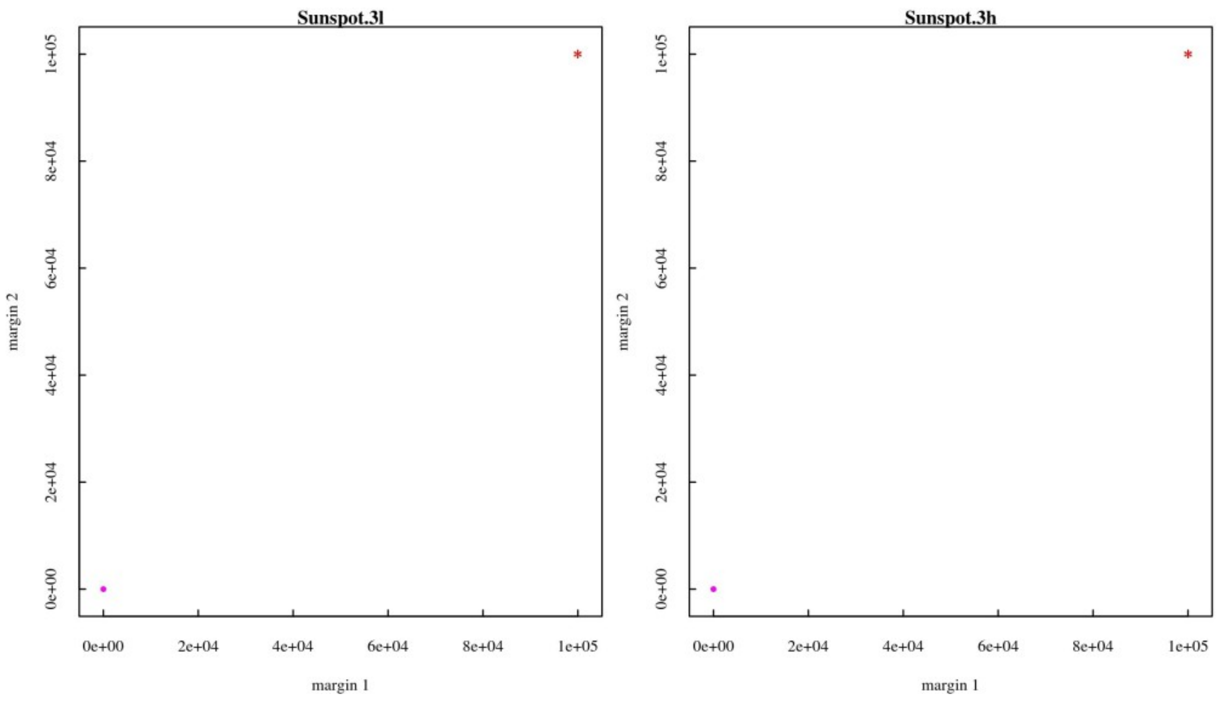


Figure 9: α-Gaussian Regions for SunSpot.3 designs, with $\alpha = 10^{-4}$. Red stars are points defined as noise while clusters are denoted by colored points.

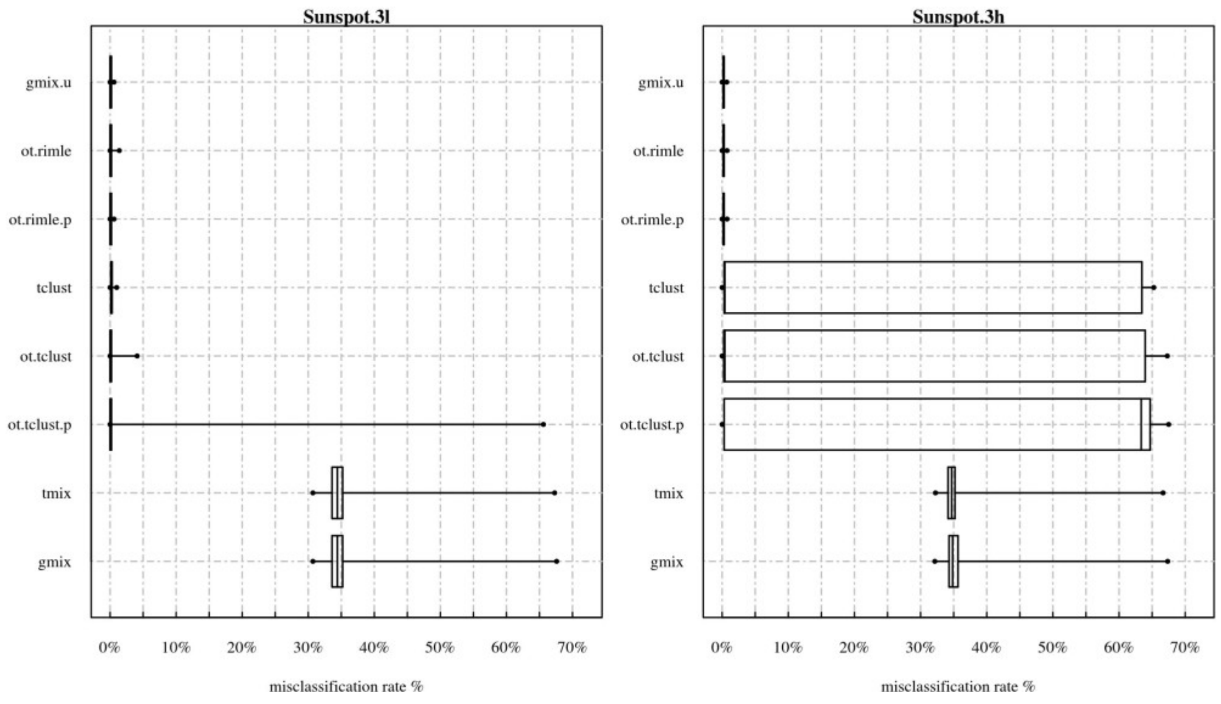


Figure 10: Boxplots for the Monte Carlo distribution of misclassification rates (%) for SunSpot.3 designs. Notice that the segment joining the whiskers' extremes always concides with the range of the distribution.

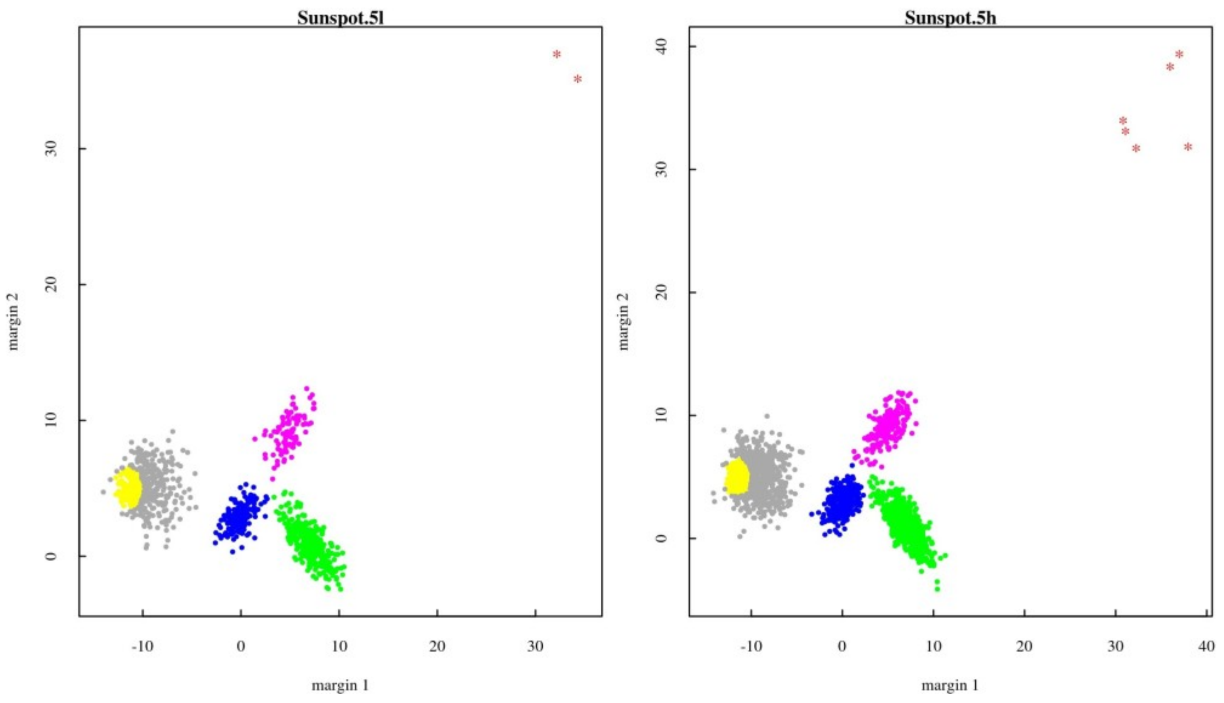


Figure 11: α-Gaussian Regions for SunSpot.5 designs, with $\alpha = 10^{-4}$. Red stars are points defined as noise while clusters are denoted by colored points.

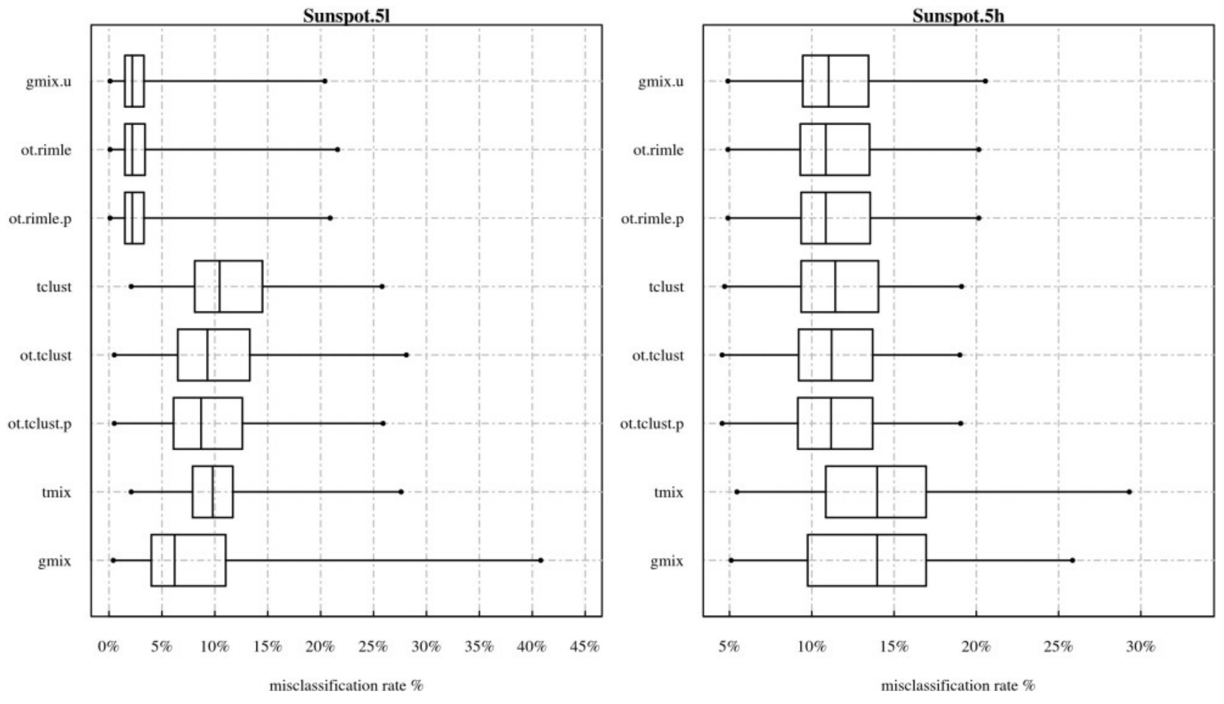


Figure 12: Boxplots for the Monte Carlo distribution of misclassification rates (%) for SunSpot.5 designs. Notice that the segment joining the whiskers' extremes always concides with the range of the distribution.

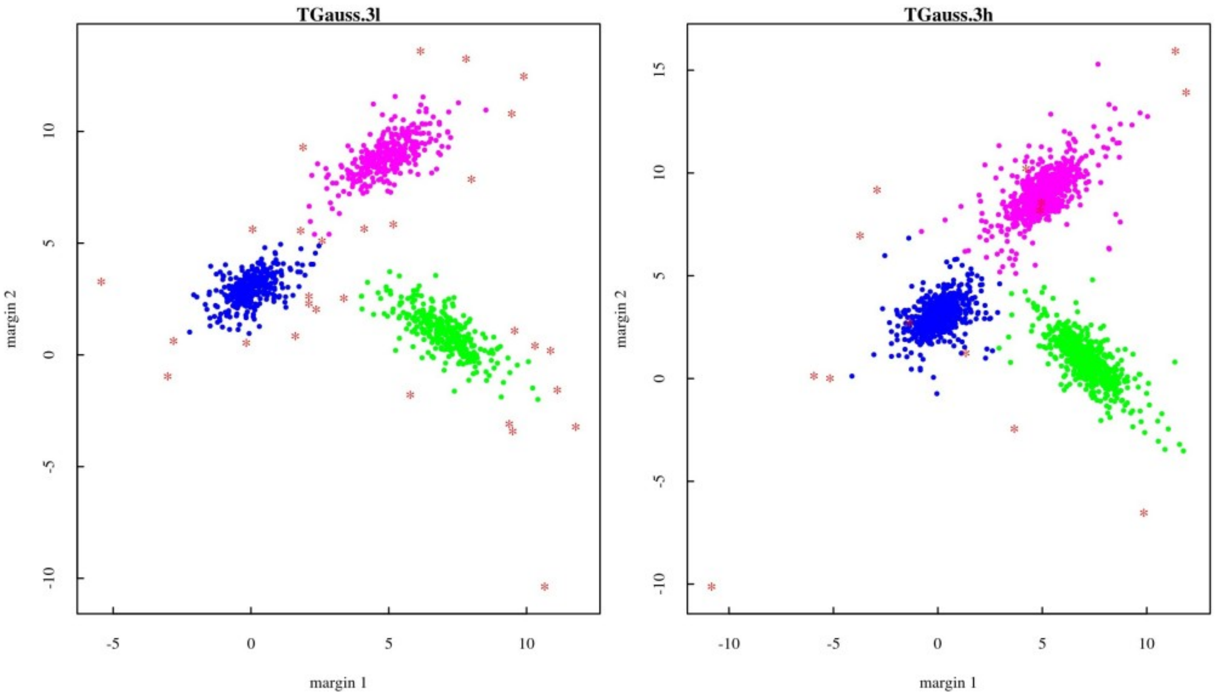


Figure 13: α-Gaussian Regions for TGauss.3 designs, with $\alpha = 10^{-4}$. Red stars are points defined as noise while clusters are denoted by colored points.

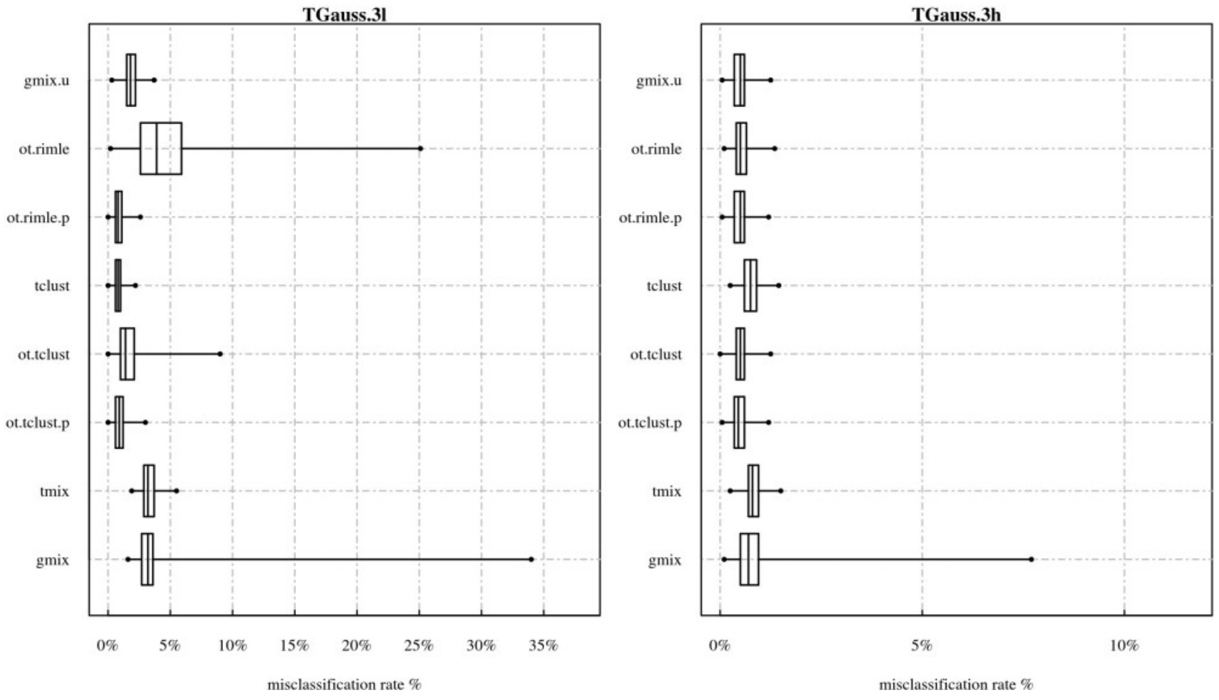


Figure 14: Boxplots for the Monte Carlo distribution of misclassification rates (%) for TGauss.3 designs. Notice that the segment joining the whiskers' extremes always concides with the range of the distribution.

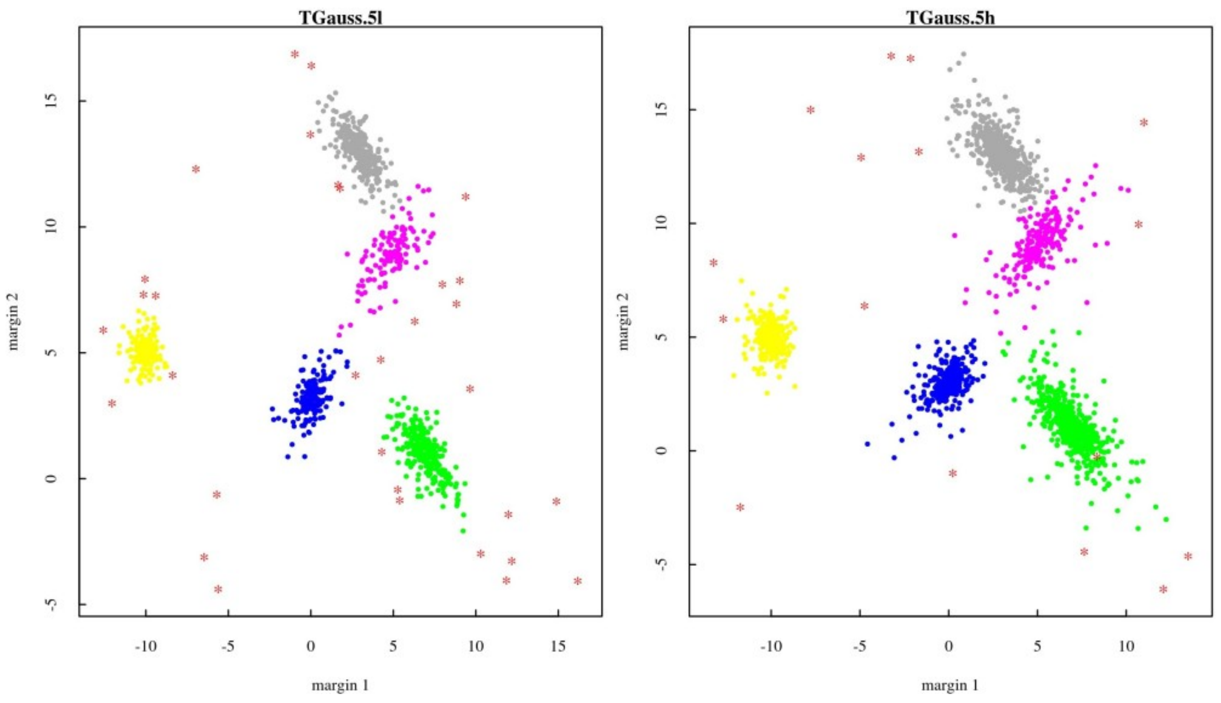


Figure 15: α-Gaussian Regions for TGauss.5 designs, with $\alpha = 10^{-4}$. Red stars are points defined as noise while clusters are denoted by colored points.

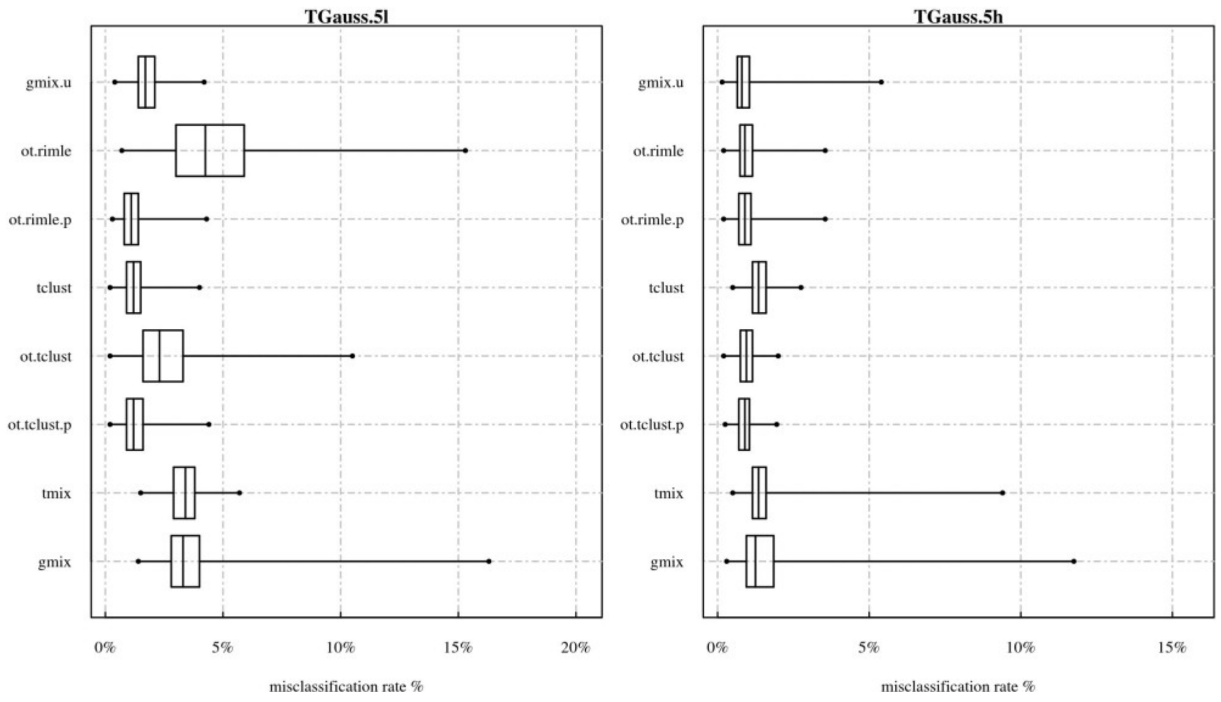


Figure 16: Boxplots for the Monte Carlo distribution of misclassification rates (%) for TGauss.5 designs. Notice that the segment joining the whiskers' extremes always concides with the range of the distribution.

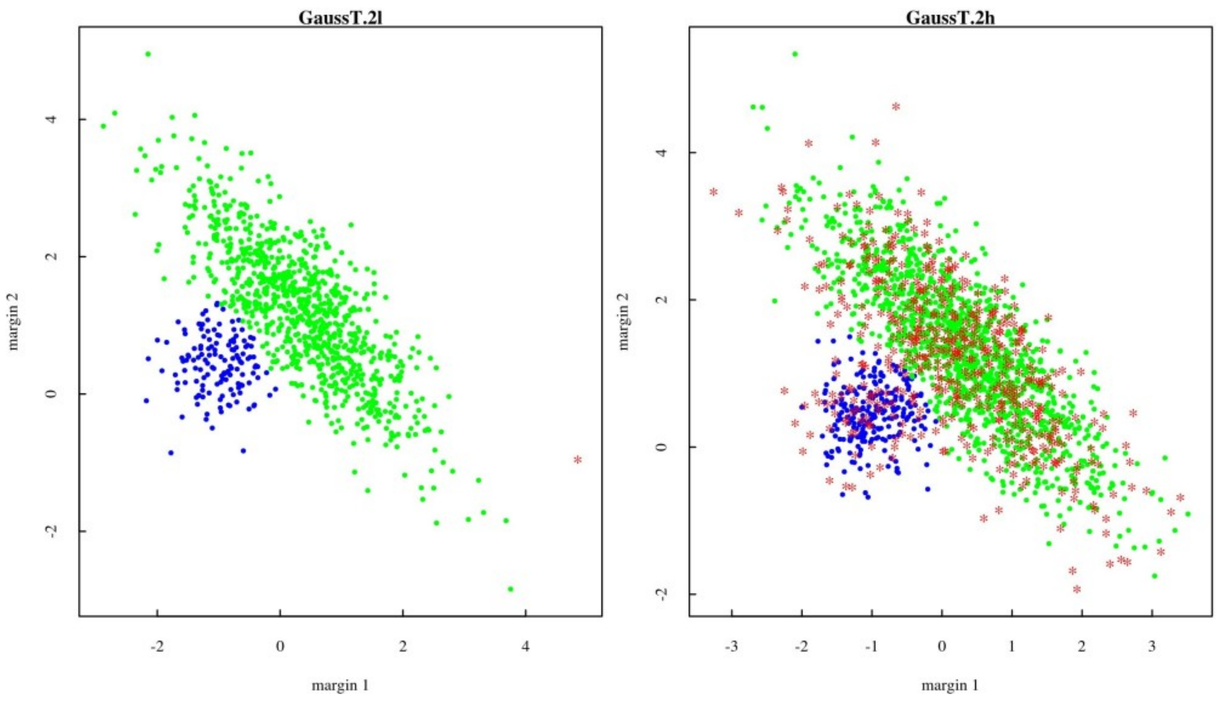


Figure 17: α-Gaussian Regions for GaussT.2 designs, with $\alpha = 10^{-4}$. Red stars are points defined as noise while clusters are denoted by colored points.

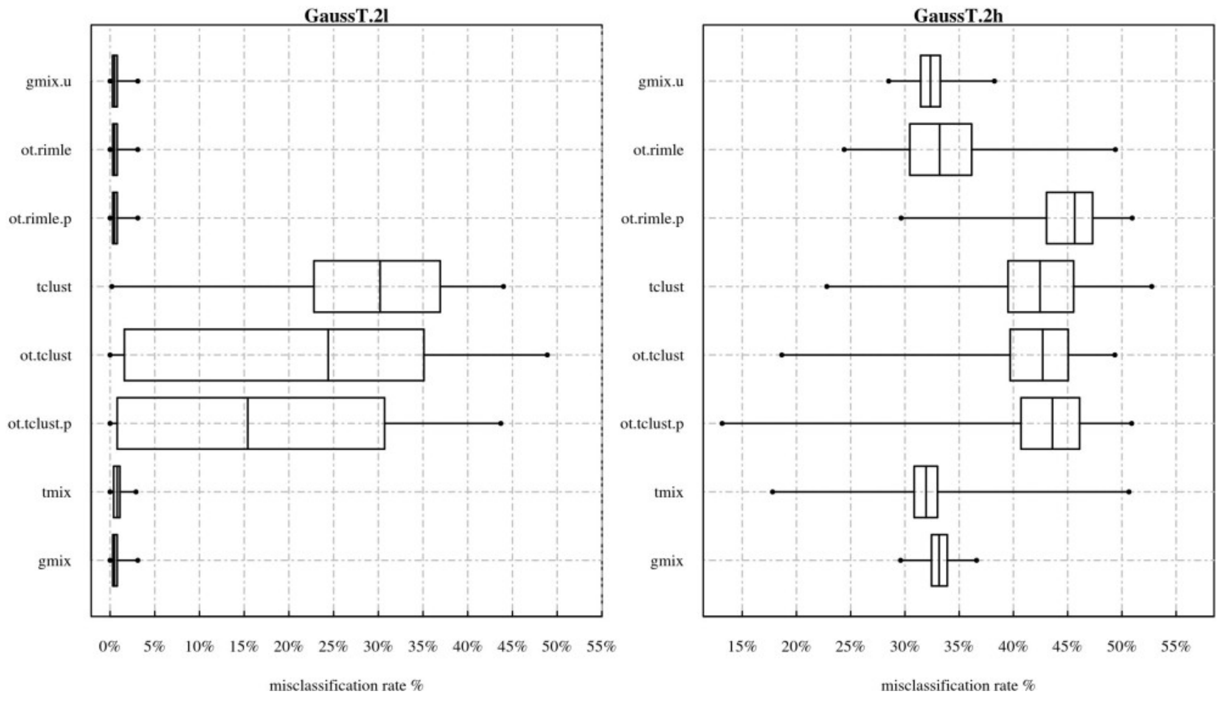


Figure 18: Boxplots for the Monte Carlo distribution of misclassification rates (%) for GaussT.2 designs. Notice that the segment joining the whiskers' extremes always concides with the range of the distribution.

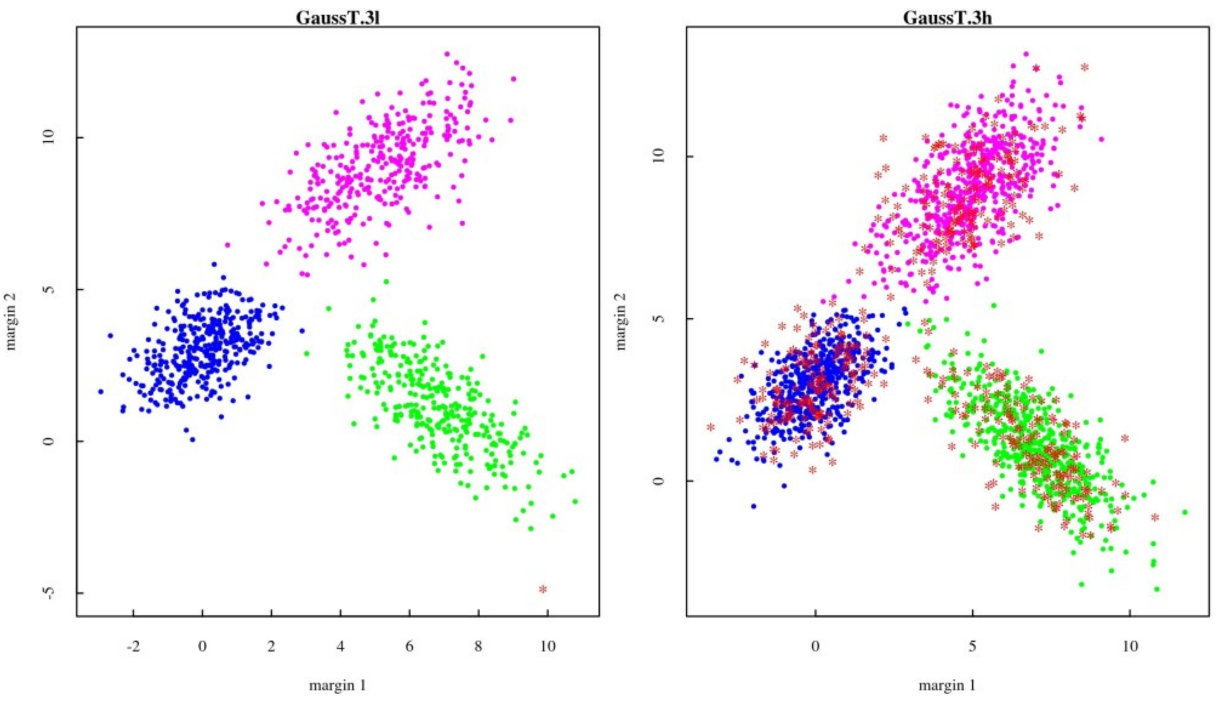


Figure 19: α-Gaussian Regions for GaussT.3 designs, with $\alpha = 10^{-4}$. Red stars are points defined as noise while clusters are denoted by colored points.

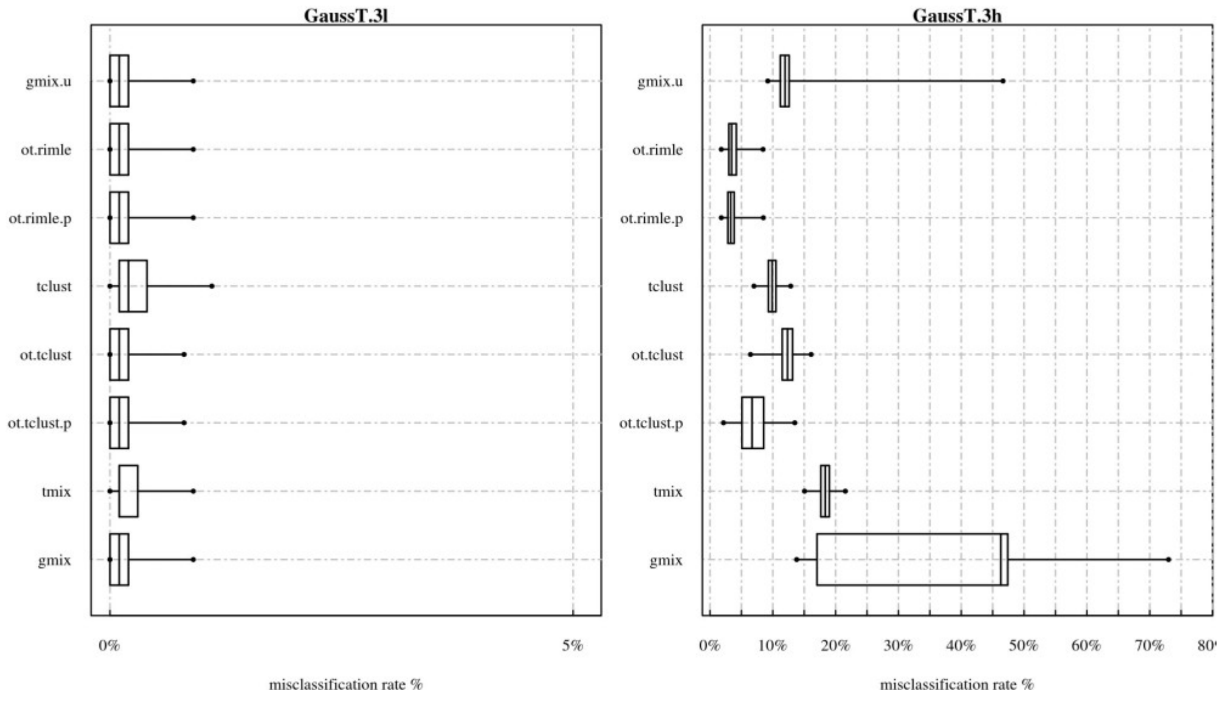


Figure 20: Boxplots for the Monte Carlo distribution of misclassification rates (%) for GaussT.3 designs. Notice that the segment joining the whiskers' extremes always concides with the range of the distribution.

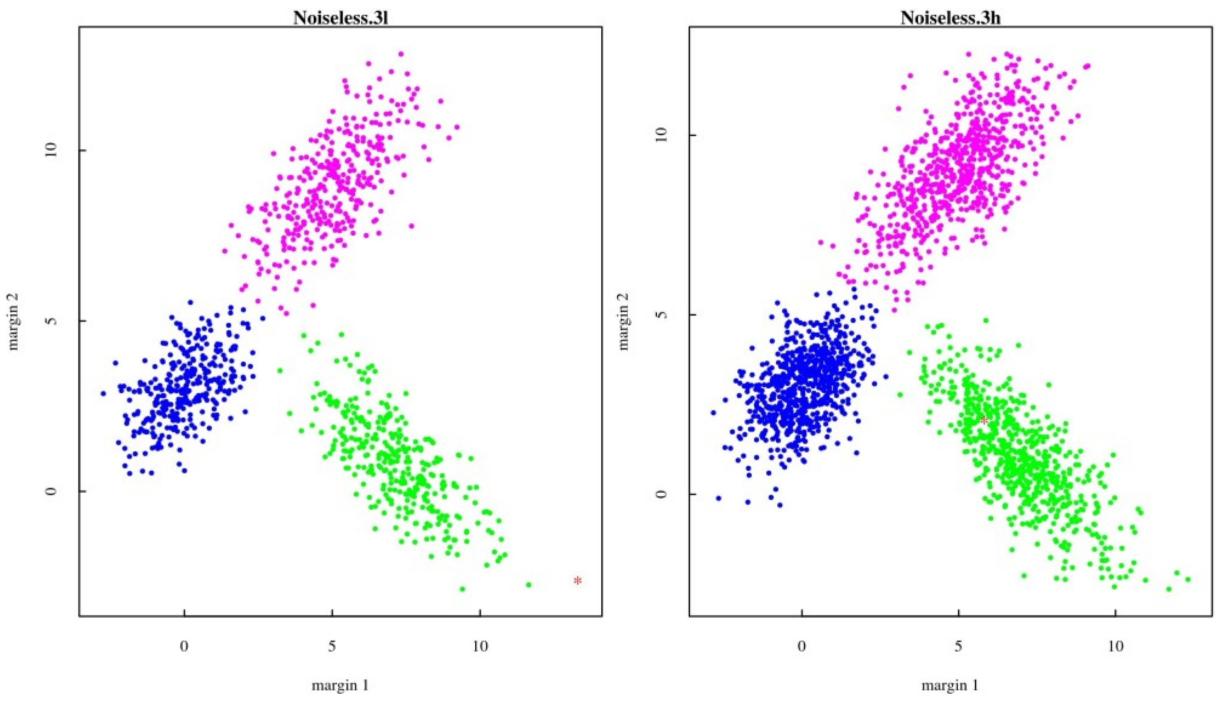


Figure 21: α-Gaussian Regions for Noiseless.3 designs, with $\alpha = 10^{-4}$. Red stars are points defined as noise while clusters are denoted by colored points.

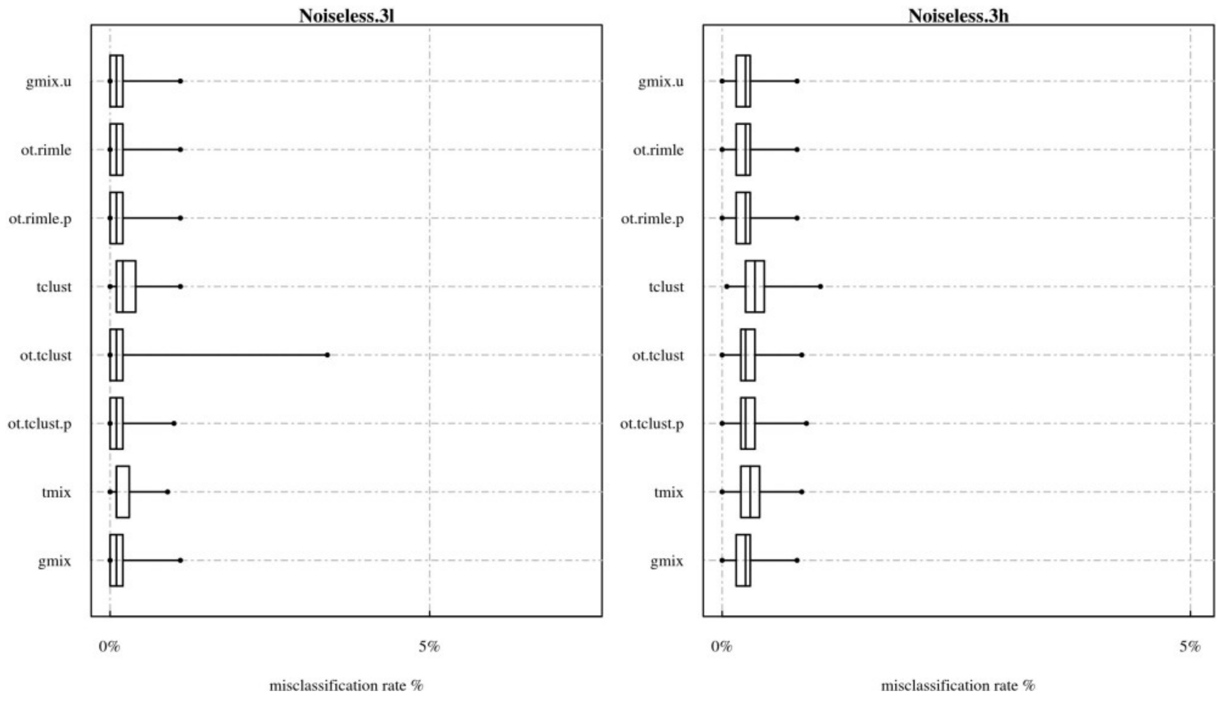


Figure 22: Boxplots for the Monte Carlo distribution of misclassification rates (%) for Noiseless.3 designs. Notice that the segment joining the whiskers' extremes always concides with the range of the distribution.

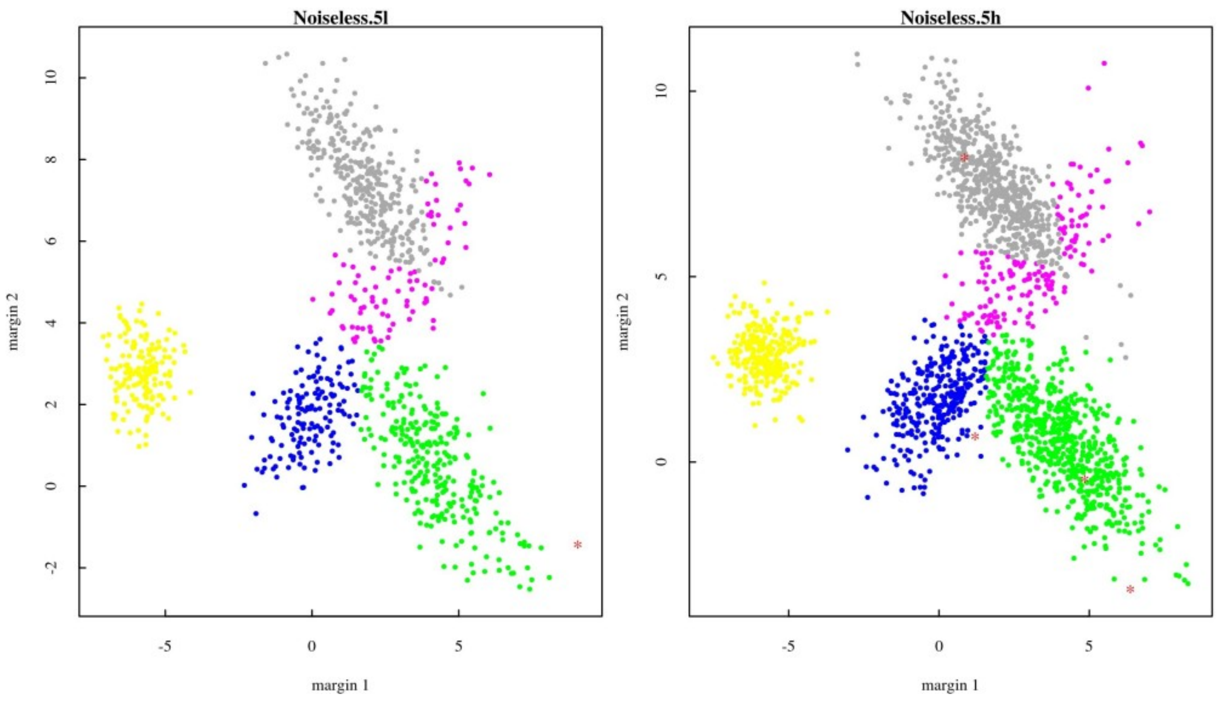


Figure 23: α-Gaussian Regions for Noiseless.5 designs, with $\alpha = 10^{-4}$. Red stars are points defined as noise while clusters are denoted by colored points.

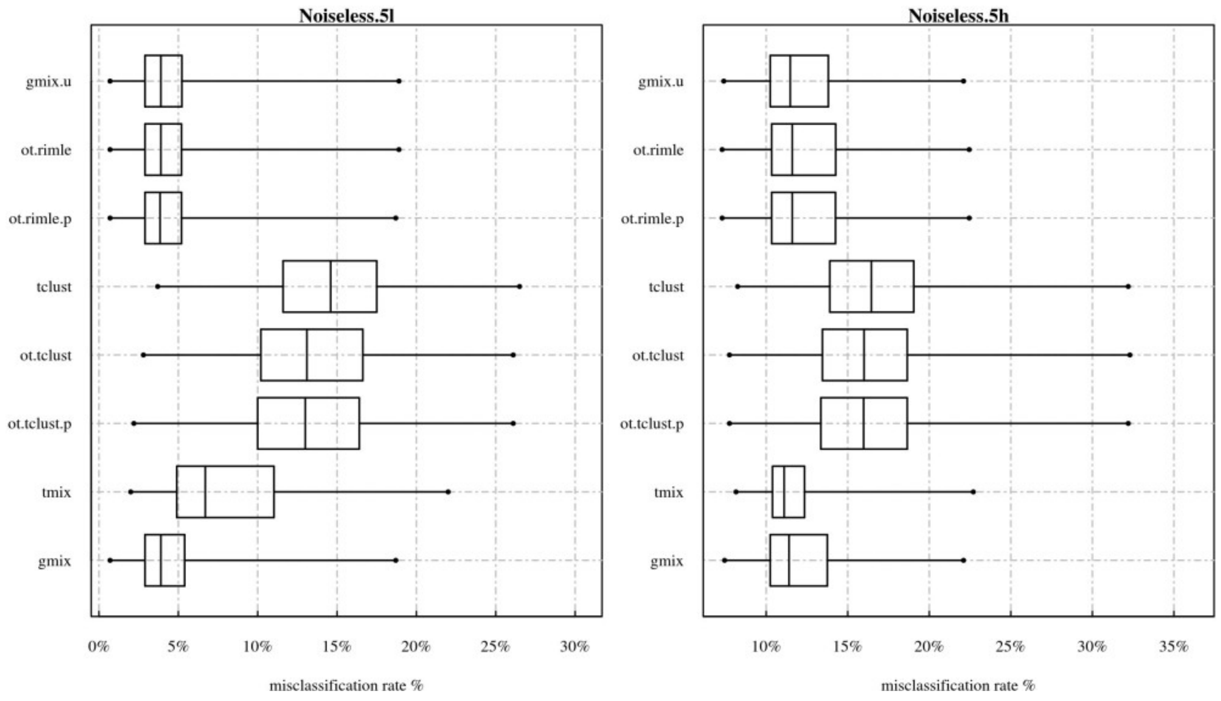


Figure 24: Boxplots for the Monte Carlo distribution of misclassification rates (%) for Noiseless.5 designs. Notice that the segment joining the whiskers' extremes always concides with the range of the distribution.

5 Detailed summary tables for the simulation study

Table 1: Monte Carlo estimates of the expected proportion of points belonging to noise region and the clusters. The index $j = 0$ denotes membership to the noise region, i.e. NR_α. Indexes $j = 1, 2, \dots, 5$ denote α-Gaussian regions. We set $\alpha = 10^{-4}$ for all designs. The table reports Monte Carlo averages and their standard errors multiplied by 10^4 in brackets.

DGP	Cluster Index					
	$j = 0$	$j = 1$	$j = 2$	$j = 3$	$j = 4$	$j = 5$
WideNoise.2l	0.04(2.02)	0.81(4.04)	0.15(3.62)	—	—	—
WideNoise.2h	0.04(1.31)	0.81(2.86)	0.16(2.60)	—	—	—
WideNoise.3l	0.06(2.33)	0.31(4.68)	0.31(4.72)	0.32(4.68)	—	—
WideNoise.3h	0.05(1.50)	0.32(3.24)	0.32(3.34)	0.32(3.35)	—	—
SideNoise.2l	0.10(3.03)	0.10(3.04)	0.80(3.89)	—	—	—
SideNoise.2h	0.10(2.08)	0.10(2.18)	0.80(2.84)	—	—	—
SideNoise.3l	0.10(2.98)	0.15(3.55)	0.35(4.76)	0.40(4.83)	—	—
SideNoise.3h	0.10(2.06)	0.15(2.65)	0.35(3.36)	0.40(3.56)	—	—
Sunspot.3l	0.02(1.53)	0.33(4.72)	0.32(4.52)	0.32(4.70)	—	—
Sunspot.3h	0.03(1.10)	0.33(3.26)	0.32(3.35)	0.32(3.19)	—	—
Sunspot.5l	0.00(0.44)	0.15(3.49)	0.30(4.60)	0.10(3.02)	0.17(3.72)	0.28(4.50)
Sunspot.5h	0.00(0.32)	0.15(2.53)	0.30(3.24)	0.10(2.04)	0.17(2.56)	0.28(3.21)
TGauss.3l	0.04(1.99)	0.32(4.64)	0.32(4.48)	0.32(4.78)	—	—
TGauss.3h	0.01(0.57)	0.33(3.32)	0.33(3.25)	0.33(3.32)	—	—
TGauss.5l	0.04(1.95)	0.14(3.61)	0.29(4.64)	0.10(3.03)	0.14(3.51)	0.28(4.42)
TGauss.5h	0.01(0.58)	0.15(2.52)	0.30(3.23)	0.11(2.25)	0.15(2.51)	0.29(3.30)
GaussT.2l	0.00(0.09)	0.15(3.56)	0.85(3.57)	—	—	—
GaussT.2h	0.22(3.02)	0.12(2.17)	0.66(3.41)	—	—	—
GaussT.3l	0.00(0.07)	0.33(4.79)	0.33(4.55)	0.33(4.61)	—	—
GaussT.3h	0.23(3.03)	0.26(3.15)	0.26(3.15)	0.26(3.01)	—	—
Noiseless.3l	0.00(0.00)	0.33(4.65)	0.33(4.54)	0.33(4.72)	—	—
Noiseless.3h	0.00(0.00)	0.33(3.26)	0.33(3.28)	0.33(3.38)	—	—
Noiseless.5l	0.00(0.00)	0.16(3.74)	0.30(4.68)	0.09(2.82)	0.15(3.60)	0.31(4.77)
Noiseless.5h	0.00(0.00)	0.16(2.64)	0.30(3.15)	0.09(2.00)	0.15(2.52)	0.31(3.36)

Table 2: Monte Carlo average “noise-to-cluster” misclassification rates (%) with their standard errors in brackets. The “noise-to-cluster” misclassification rate gives the proportion of points belonging to the noise region assigned to one of the Gaussian clusters. Notice that both averages and their standard errors are reported in percentage scale.

DGP	Method							
	gmix.u	ot.rimle	ot.rimle.p	tclust	ot.tclust	ot.tclust.p	tmix	gmix
WideNoise.2l	0.04(0.00)	0.03(0.00)	0.08(0.00)	0.00(0.00)	0.00(0.00)	0.05(0.01)	3.93(0.04)	4.28(0.02)
WideNoise.2h	3.63(0.02)	0.50(0.02)	0.68(0.02)	0.00(0.00)	0.52(0.03)	1.31(0.04)	3.66(0.01)	3.66(0.01)
WideNoise.3l	0.09(0.00)	0.11(0.01)	0.21(0.01)	0.04(0.00)	0.14(0.00)	0.18(0.01)	2.77(0.02)	5.02(0.04)
WideNoise.3h	3.81(0.04)	2.01(0.05)	2.74(0.05)	0.21(0.00)	0.83(0.02)	2.21(0.05)	4.58(0.02)	4.53(0.02)
SideNoise.2l	0.00(0.00)	0.02(0.01)	0.04(0.02)	0.17(0.01)	0.00(0.00)	0.01(0.00)	9.58(0.05)	10.00(0.03)
SideNoise.2h	0.02(0.01)	0.08(0.02)	0.14(0.03)	0.17(0.01)	0.18(0.01)	0.65(0.02)	9.92(0.02)	9.92(0.02)
SideNoise.3l	0.01(0.00)	0.01(0.00)	0.03(0.01)	0.10(0.01)	0.01(0.00)	0.01(0.00)	6.73(0.12)	9.82(0.03)
SideNoise.3h	0.02(0.00)	0.05(0.01)	0.09(0.01)	0.10(0.01)	0.07(0.01)	0.35(0.02)	9.69(0.02)	9.69(0.02)
Sunspot.3l	0.00(0.00)	0.00(0.00)	0.00(0.00)	0.00(0.00)	0.00(0.00)	0.03(0.01)	2.49(0.02)	2.49(0.02)
Sunspot.3h	0.00(0.00)	0.00(0.00)	0.00(0.00)	0.00(0.00)	0.16(0.02)	0.53(0.03)	2.52(0.01)	2.51(0.01)
Sunspot.5l	0.00(0.00)	0.00(0.00)	0.00(0.00)	0.00(0.00)	0.00(0.00)	0.00(0.00)	0.01(0.00)	0.02(0.00)
Sunspot.5h	0.01(0.00)	0.01(0.00)	0.01(0.00)	0.00(0.00)	0.00(0.00)	0.00(0.00)	0.10(0.00)	0.14(0.00)
TGauss.3l	1.82(0.02)	0.01(0.00)	0.63(0.01)	0.56(0.01)	0.10(0.00)	0.32(0.01)	3.25(0.02)	3.08(0.02)
TGauss.3h	0.19(0.00)	0.10(0.00)	0.14(0.00)	0.04(0.00)	0.09(0.00)	0.11(0.00)	0.60(0.01)	0.32(0.00)
TGauss.5l	1.59(0.02)	0.10(0.02)	0.81(0.02)	0.50(0.01)	0.10(0.00)	0.45(0.01)	3.05(0.02)	3.00(0.02)
TGauss.5h	0.18(0.00)	0.12(0.00)	0.17(0.00)	0.03(0.00)	0.07(0.00)	0.10(0.00)	0.60(0.01)	0.38(0.01)
GaussT.2l	0.00(0.00)	0.00(0.00)	0.00(0.00)	0.00(0.00)	0.00(0.00)	0.00(0.00)	0.01(0.00)	0.00(0.00)
GaussT.2h	19.39(0.03)	5.11(0.06)	11.45(0.08)	14.93(0.04)	10.15(0.09)	12.16(0.07)	19.74(0.05)	21.26(0.03)
GaussT.3l	0.00(0.00)	0.00(0.00)	0.00(0.00)	0.00(0.00)	0.00(0.00)	0.00(0.00)	0.00(0.00)	0.00(0.00)
GaussT.3h	11.64(0.04)	1.00(0.03)	1.28(0.03)	9.47(0.03)	0.05(0.00)	0.30(0.01)	17.86(0.03)	19.63(0.09)
Noiseless.3l	0.00(0.00)	0.00(0.00)	0.00(0.00)	0.00(0.00)	0.00(0.00)	0.00(0.00)	0.00(0.00)	0.00(0.00)
Noiseless.3h	0.00(0.00)	0.00(0.00)	0.00(0.00)	0.00(0.00)	0.00(0.00)	0.00(0.00)	0.00(0.00)	0.00(0.00)
Noiseless.5l	0.00(0.00)	0.00(0.00)	0.00(0.00)	0.00(0.00)	0.00(0.00)	0.00(0.00)	0.00(0.00)	0.00(0.00)
Noiseless.5h	0.00(0.00)	0.00(0.00)	0.00(0.00)	0.00(0.00)	0.00(0.00)	0.00(0.00)	0.00(0.00)	0.00(0.00)

Table 3: Monte Carlo average “cluster-to-noise” misclassification rates (%) with their standard errors in brackets. The “cluster-to-noise” misclassification rate gives the proportion of points belonging to Gaussian clusters assigned to the noise component. Notice that both averages and their standard errors are reported in percentage scale.

DGP	Method							
	gmix.u	ot.rimle	ot.rimle.p	tclust	ot.tclust	ot.tclust.p	tmix	gmix
WideNoise.2l	0.06(0.00)	1.01(0.07)	0.03(0.00)	2.55(0.03)	4.07(0.07)	2.23(0.08)	0.00(0.00)	0.00(0.00)
WideNoise.2h	0.00(0.00)	0.07(0.00)	0.05(0.00)	1.08(0.01)	0.58(0.01)	0.35(0.01)	0.00(0.00)	0.00(0.00)
WideNoise.3l	0.11(0.00)	0.17(0.01)	0.04(0.00)	0.24(0.01)	0.09(0.00)	0.06(0.00)	0.00(0.00)	0.00(0.00)
WideNoise.3h	0.00(0.00)	0.02(0.00)	0.01(0.00)	0.11(0.00)	0.02(0.00)	0.01(0.00)	0.00(0.00)	0.00(0.00)
SideNoise.2l	0.01(0.00)	0.01(0.00)	0.01(0.00)	0.01(0.00)	0.01(0.00)	0.01(0.00)	0.00(0.00)	0.00(0.00)
SideNoise.2h	0.01(0.00)	0.01(0.00)	0.01(0.00)	0.02(0.00)	0.02(0.00)	0.01(0.00)	0.00(0.00)	0.00(0.00)
SideNoise.3l	0.01(0.00)	0.03(0.01)	0.01(0.00)	0.01(0.00)	0.02(0.00)	0.01(0.00)	0.00(0.00)	0.00(0.00)
SideNoise.3h	0.01(0.00)	0.02(0.00)	0.01(0.00)	0.03(0.00)	0.02(0.00)	0.02(0.00)	0.00(0.00)	0.00(0.00)
Sunspot.3l	0.00(0.00)	0.01(0.00)	0.00(0.00)	0.07(0.00)	0.01(0.00)	0.00(0.00)	0.00(0.00)	0.00(0.00)
Sunspot.3h	0.00(0.00)	0.01(0.00)	0.00(0.00)	0.05(0.00)	0.01(0.00)	0.01(0.00)	0.00(0.00)	0.00(0.00)
Sunspot.5l	0.02(0.01)	0.03(0.01)	0.01(0.01)	0.68(0.01)	0.27(0.02)	0.10(0.01)	0.00(0.00)	0.00(0.00)
Sunspot.5h	0.00(0.00)	0.01(0.00)	0.01(0.00)	0.22(0.00)	0.05(0.00)	0.04(0.00)	0.00(0.00)	0.00(0.00)
TGauss.3l	0.00(0.00)	4.55(0.09)	0.26(0.01)	0.23(0.01)	1.53(0.03)	0.62(0.02)	0.00(0.00)	0.00(0.00)
TGauss.3h	0.06(0.00)	0.20(0.00)	0.13(0.00)	0.53(0.01)	0.21(0.00)	0.17(0.00)	0.00(0.00)	0.01(0.00)
TGauss.5l	0.02(0.00)	4.41(0.08)	0.25(0.01)	0.53(0.01)	2.25(0.04)	0.70(0.02)	0.00(0.00)	0.00(0.00)
TGauss.5h	0.07(0.00)	0.25(0.01)	0.14(0.00)	0.80(0.01)	0.33(0.00)	0.22(0.00)	0.00(0.00)	0.02(0.00)
GaussT.2l	0.00(0.00)	0.00(0.00)	0.00(0.00)	1.08(0.02)	1.17(0.06)	0.10(0.01)	0.00(0.00)	0.00(0.00)
GaussT.2h	0.00(0.00)	0.27(0.04)	0.00(0.00)	0.00(0.00)	0.07(0.02)	0.02(0.01)	0.00(0.00)	0.00(0.00)
GaussT.3l	0.00(0.00)	0.00(0.00)	0.00(0.00)	0.11(0.00)	0.01(0.00)	0.01(0.00)	0.00(0.00)	0.00(0.00)
GaussT.3h	0.00(0.00)	2.26(0.04)	1.73(0.04)	0.00(0.00)	11.75(0.04)	6.13(0.09)	0.00(0.00)	0.00(0.00)
Noiseless.3l	0.01(0.00)	0.01(0.00)	0.01(0.00)	0.11(0.00)	0.02(0.00)	0.01(0.00)	0.00(0.00)	0.01(0.00)
Noiseless.3h	0.01(0.00)	0.01(0.00)	0.01(0.00)	0.09(0.00)	0.02(0.00)	0.02(0.00)	0.00(0.00)	0.01(0.00)
Noiseless.5l	0.02(0.00)	0.01(0.00)	0.01(0.00)	0.63(0.01)	0.20(0.01)	0.12(0.00)	0.00(0.00)	0.01(0.00)
Noiseless.5h	0.01(0.00)	0.02(0.00)	0.01(0.00)	0.22(0.00)	0.08(0.00)	0.07(0.00)	0.00(0.00)	0.01(0.00)

Table 4: Monte Carlo average "cluster-to-cluster" misclassification rates (%) with their standard errors in brackets. The "cluster-to-cluster" misclassification rate gives the proportion of points belonging to Gaussian clusters not assigned consistently to a Gaussian cluster. Notice that both averages and their standard errors are reported as percentages.

DGP	Method							
	gmix.u	ot.rimle	ot.rimle.p	tclust	ot.tclust	ot.tclust.p	tmix	gmix
WideNoise.2l	1.24(0.02)	3.96(0.23)	1.23(0.02)	10.16(0.32)	9.04(0.28)	7.92(0.26)	13.55(0.07)	14.12(0.04)
WideNoise.2h	13.72(0.04)	4.71(0.04)	4.67(0.05)	21.85(0.32)	15.10(0.33)	11.20(0.26)	15.24(0.03)	13.77(0.03)
WideNoise.3l	0.14(0.00)	0.15(0.00)	0.15(0.00)	0.16(0.00)	0.15(0.00)	0.15(0.00)	0.32(0.01)	2.77(0.12)
WideNoise.3h	3.78(0.16)	1.15(0.04)	1.52(0.04)	0.40(0.01)	0.46(0.01)	0.82(0.02)	1.32(0.03)	4.02(0.14)
SideNoise.2l	0.00(0.00)	0.00(0.00)	0.00(0.00)	0.00(0.00)	0.00(0.00)	0.00(0.00)	1.17(0.08)	6.68(0.16)
SideNoise.2h	0.00(0.00)	0.00(0.00)	0.00(0.00)	0.00(0.00)	0.00(0.00)	0.00(0.00)	10.11(0.02)	2.69(0.15)
SideNoise.3l	0.04(0.00)	0.04(0.00)	0.07(0.02)	0.08(0.01)	0.04(0.00)	0.04(0.00)	12.16(0.36)	26.78(0.40)
SideNoise.3h	0.09(0.00)	0.12(0.02)	0.14(0.02)	0.12(0.00)	0.11(0.00)	0.16(0.00)	24.53(0.16)	21.89(0.31)
Sunspot.3l	0.11(0.00)	0.11(0.00)	0.11(0.00)	0.15(0.00)	0.12(0.00)	2.14(0.35)	32.17(0.11)	32.89(0.18)
Sunspot.3h	0.23(0.00)	0.23(0.00)	0.23(0.00)	21.33(0.95)	27.02(0.99)	36.29(1.00)	32.23(0.06)	37.49(0.37)
Sunspot.5l	3.30(0.12)	3.36(0.12)	3.36(0.12)	10.57(0.12)	9.74(0.14)	9.43(0.14)	10.00(0.10)	7.90(0.16)
Sunspot.5h	11.71(0.10)	11.61(0.10)	11.63(0.10)	11.66(0.10)	11.57(0.10)	11.58(0.10)	13.74(0.11)	13.16(0.13)
TGauss.3l	0.02(0.00)	0.00(0.00)	0.01(0.00)	0.02(0.00)	0.00(0.00)	0.01(0.00)	0.02(0.00)	0.25(0.05)
TGauss.3h	0.24(0.00)	0.23(0.00)	0.24(0.00)	0.21(0.00)	0.21(0.00)	0.22(0.00)	0.23(0.00)	0.52(0.02)
TGauss.5l	0.16(0.01)	0.14(0.01)	0.14(0.00)	0.18(0.01)	0.20(0.01)	0.17(0.01)	0.31(0.01)	0.59(0.03)
TGauss.5h	0.66(0.01)	0.60(0.01)	0.63(0.01)	0.56(0.01)	0.56(0.01)	0.56(0.01)	0.83(0.01)	1.26(0.04)
GaussT.2l	0.56(0.01)	0.56(0.01)	0.56(0.01)	27.69(0.32)	20.33(0.45)	16.42(0.47)	0.82(0.02)	0.56(0.01)
GaussT.2h	13.06(0.05)	28.22(0.10)	33.46(0.12)	27.50(0.15)	31.49(0.14)	30.96(0.15)	11.58(0.15)	11.91(0.02)
GaussT.3l	0.11(0.00)	0.11(0.00)	0.11(0.00)	0.16(0.00)	0.12(0.00)	0.12(0.00)	0.17(0.01)	0.11(0.00)
GaussT.3h	0.68(0.09)	0.43(0.00)	0.43(0.00)	0.40(0.00)	0.41(0.00)	0.42(0.01)	0.48(0.01)	16.60(0.40)
Noiseless.3l	0.12(0.00)	0.12(0.00)	0.11(0.00)	0.17(0.00)	0.13(0.00)	0.12(0.00)	0.18(0.01)	0.11(0.00)
Noiseless.3h	0.24(0.00)	0.24(0.00)	0.24(0.00)	0.29(0.00)	0.26(0.00)	0.26(0.00)	0.29(0.00)	0.24(0.00)
Noiseless.5l	4.51(0.08)	4.47(0.08)	4.43(0.08)	13.80(0.12)	13.13(0.13)	13.02(0.13)	8.23(0.14)	4.57(0.09)
Noiseless.5h	12.38(0.10)	12.61(0.10)	12.60(0.10)	16.47(0.11)	16.05(0.12)	16.04(0.12)	11.92(0.08)	12.36(0.10)

6 Plots for the analysis of Dortmund data

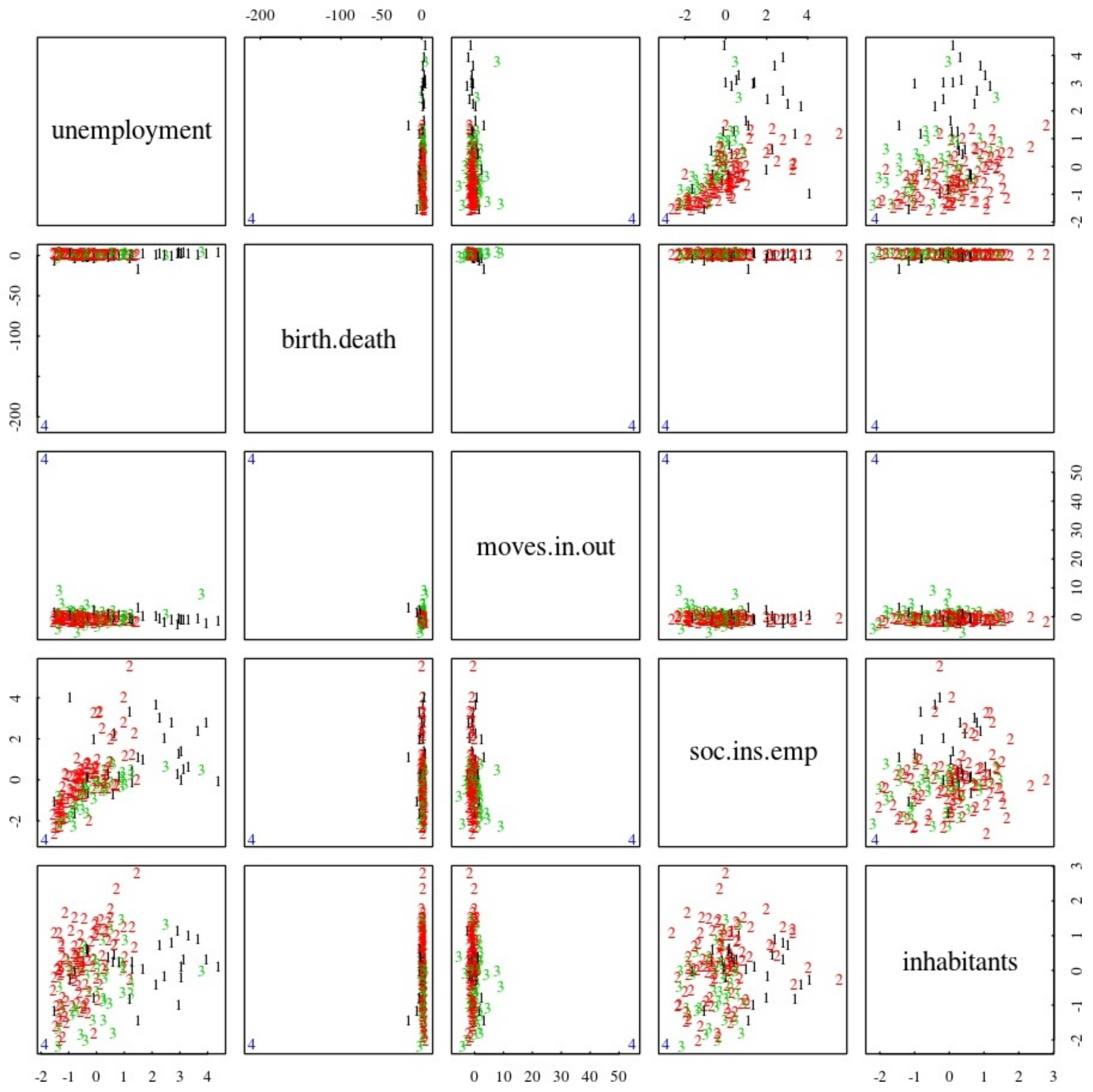


Figure 25: MCLUST clustering without the uniform noise component for the Dortmund data. This clustering is based on a pure Gaussian mixture model.

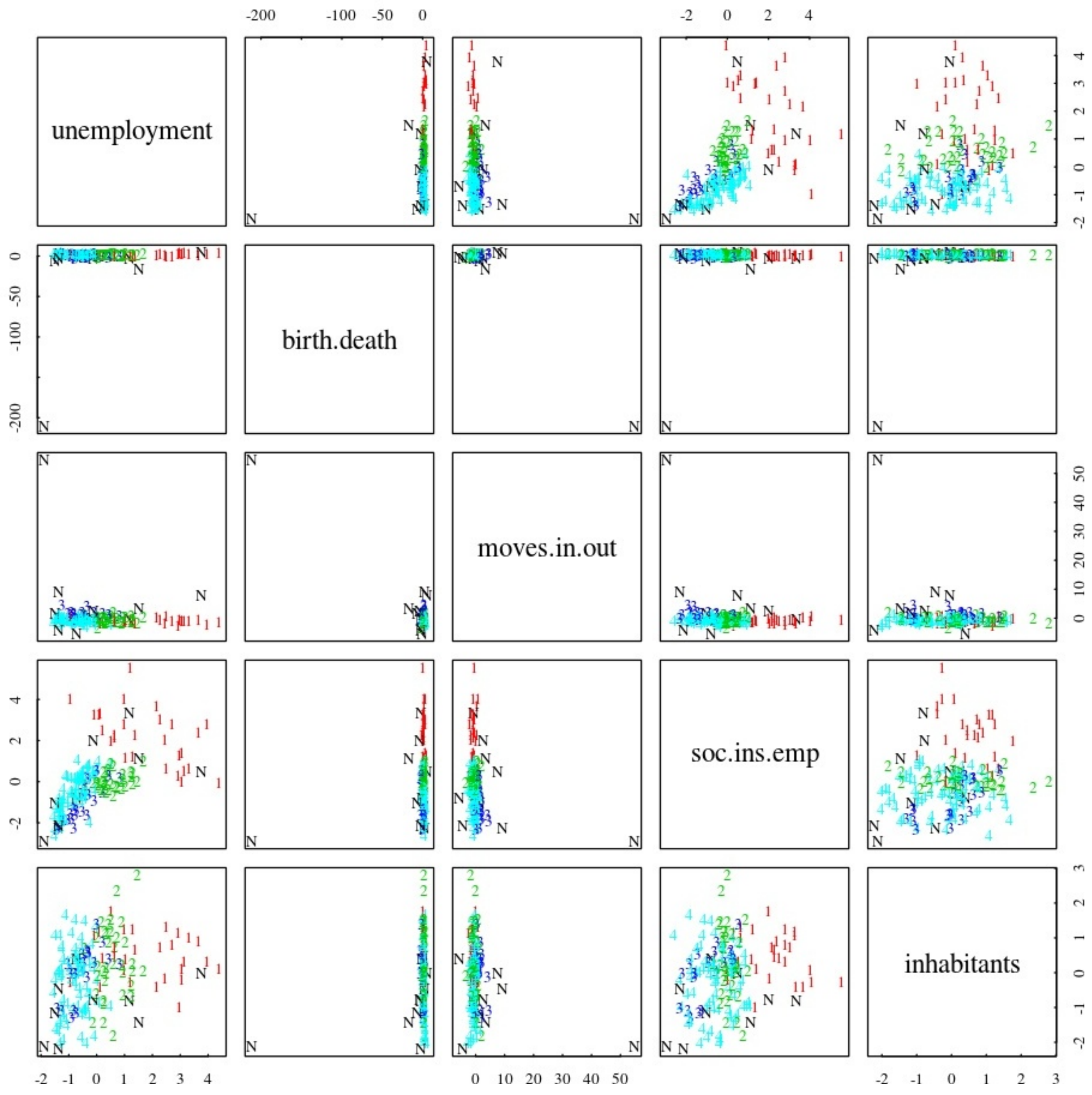


Figure 26: OTRIMLE clustering for the Dortmund data obtained without the penalty term (that is $\beta = 0$).

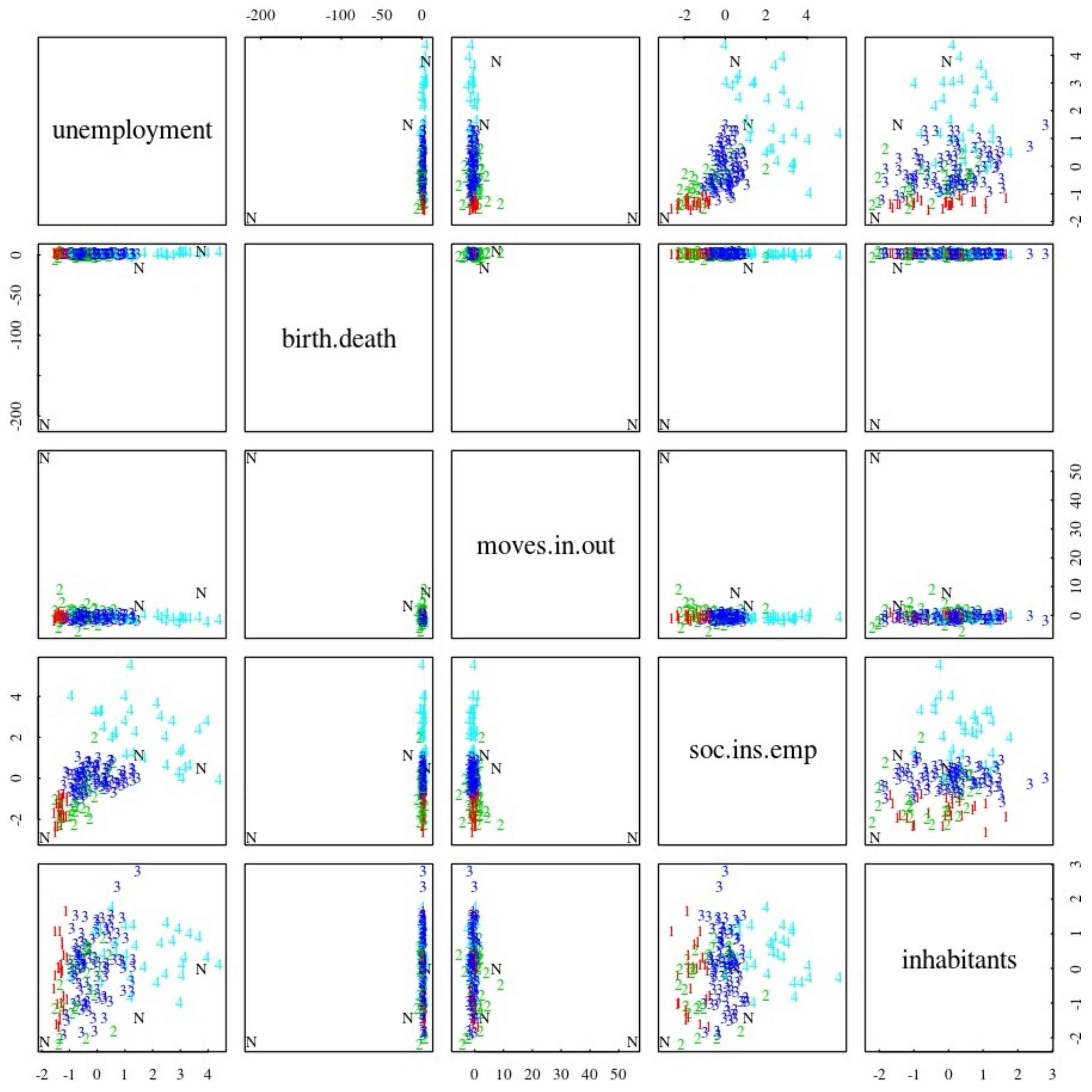


Figure 27: MCLUST clustering with the uniform noise component for the Dortmund data.

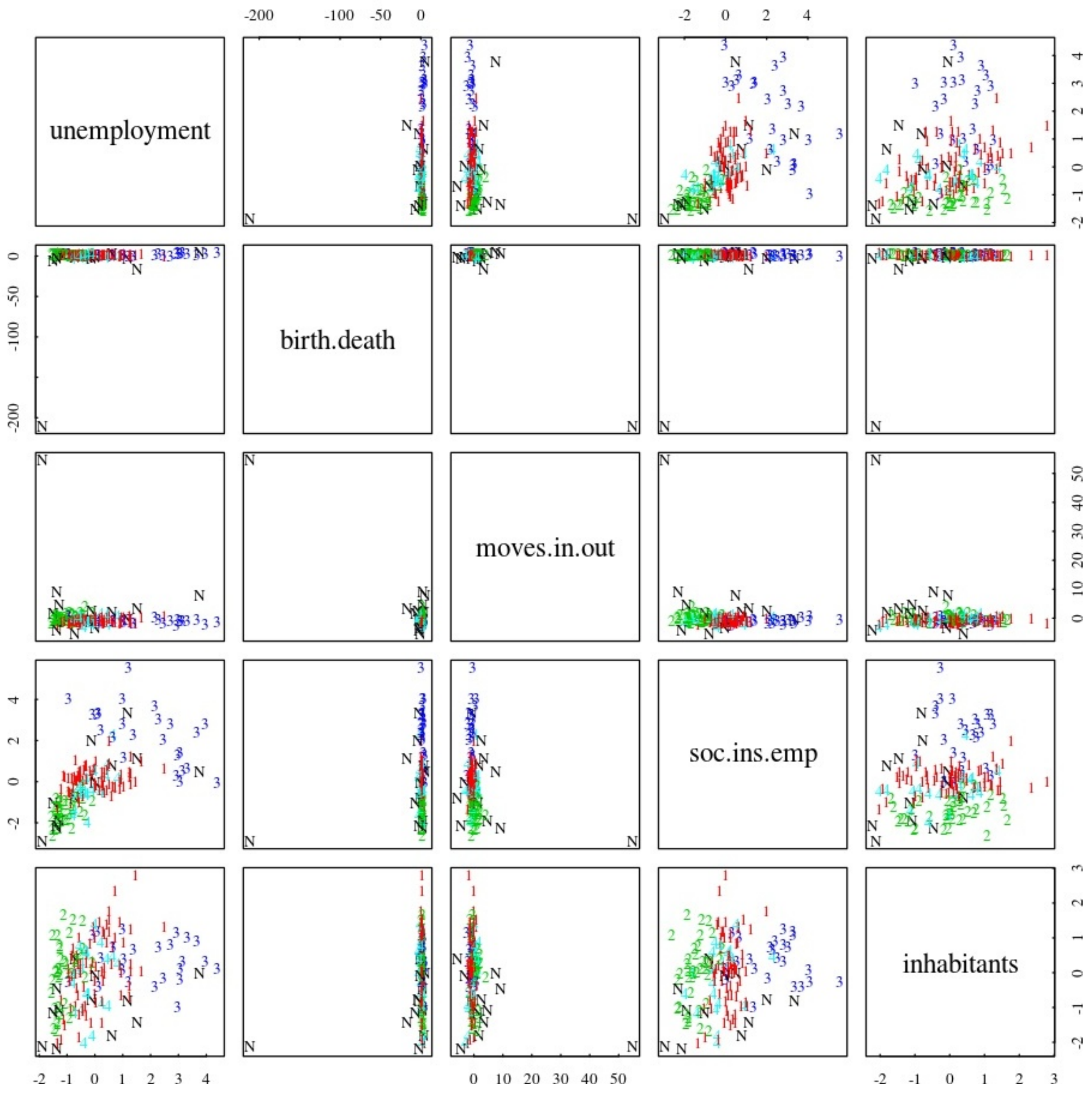


Figure 28: TCLUST clustering for the Dortmund data with trimming level set at 20%.

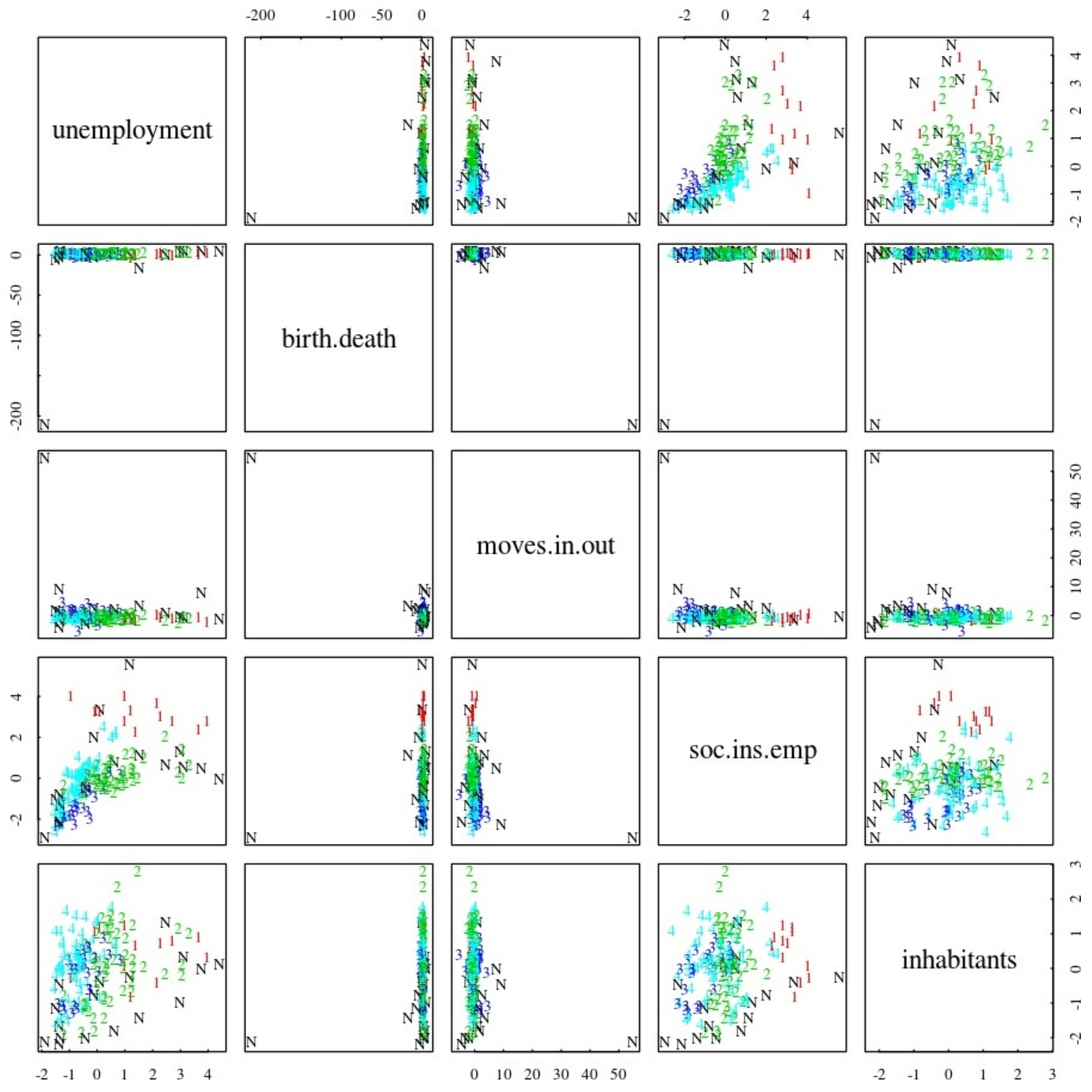


Figure 29: Clustering based on a mixture of Student-t distribution with 4 degrees of freedom for the Dortmund data. Noise identified as in McLachlan–Peel's original proposal. Note that a mixture of t_4-distributions was fitted because numerical problems were encountered fitting a $t-$mixture with lower degrees of freedom.

7 Information on folk song data

7.1 Features

The 18 features were computed by the software "FANTASTIC" (Müllensiefen (2009)). Originally, this software computed 40 features. We discarded some features that were either discrete or partly discrete in a way not compatible with clustering based on continuous distributions. Some more features were discarded because, looking at correlations and scatterplots, they carried almost identical information to some other features. The final list of used features (see Müllensiefen (2009)) was mean.entropy, mean.Yules.K, mean.Sichels.S, p.range, p.entropy, i.abs.mean, i.abs.std, i.entropy, note.dens, tonalness, tonal.clarity, tonal.spike, int.cont.grad.mean, step.cont.glob.dir, step.cont.loc.var, poly.coeff1, poly.coeff2, poly.coeff3. Features were standardized to unit MAD.

7.2 Plot

The following plot is a two-dimensional projection of the 18-dimensional folk song data, which was obtained by local discriminant 50-nearest-neighbour based dimension reduction (Hastie and Tibshirani, 1996), discriminating the Luxemburg and Warmia classes, as implemented in the R-package `fpc`.

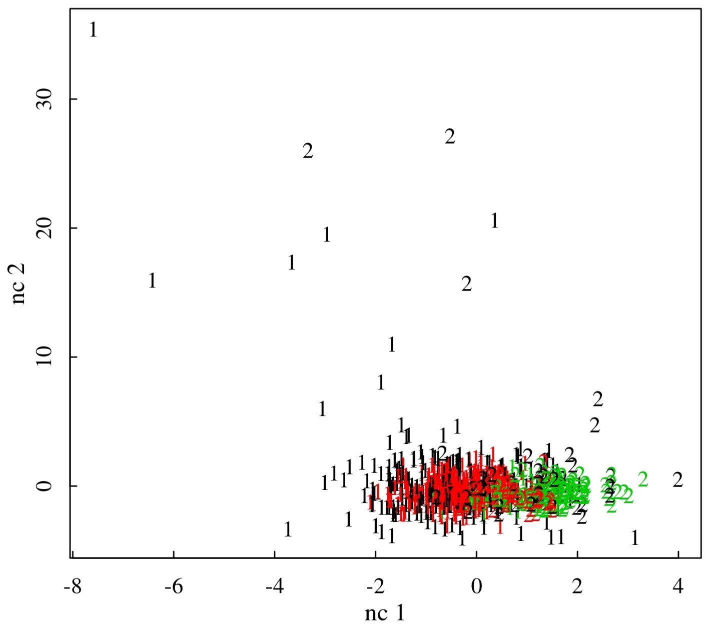


Figure 30: Local neighborhood-based discrimination plot for folk song data with songs from Luxemburg and Warmia denoted by symbols "1" and "2". The OTRIMLE clustering with $\beta = 0$ is indicated by colors red and green for the two clusters and black for points classified as noise.

References

Banfield, J. D. and A. E. Raftery (1993). Model-based gaussian and non-gaussian clustering. *Biometrics 49*, 803–821.

Biernacki, C., G. Celeux, and G. Govaert (2003). Choosing starting values for the EM algorithm for getting the highest likelihood in multivariate gaussian mixture models. *Computational Statistics & Data Analysis 41*(3), 561–575.

Byers, S. and A. E. Raftery (1998). Nearest-neighbor clutter removal for estimating features in spatial point processes. *Journal of the American Statistical Association 93*(442), 577–584.

Cator, E. A. and H. P. Lopuhaä (2012). Central limit theorem and influence function for the MCD estimators at general multivariate distributions. *Bernoulli 18*(2), 520–551.

Coretto, P. and C. Hennig (2011). Maximum likelihood estimation of heterogeneous mixtures of gaussian and uniform distributions. *Journal of Statistical Planning and Inference 141*(1), 462–473.

Coretto, P. and C. Hennig (2016). Robust improper maximum likelihood: Tuning, computation, and a comparison with other methods for robust gaussian clustering. Preprint available at https://arxiv.org/pdf/1406.0808.

Cuesta-Albertos, J. A., A. Gordaliza, and C. Matrán (1997). Trimmed k-means: An attempt to robustify quantizers. *Annals of Statistics 25*, 553–576.

Fraley, C., A. E. Raftery, T. B. Murphy, and L. Scrucca (2012). *mclust Version 4 for R: Normal Mixture Modeling for Model-Based Clustering, Classification, and Density Estimation.*

Fritz, H., L. A. García-Escudero, and A. Mayo-Iscar (2012). tclust: An R package for a trimming approach to cluster analysis. *Journal of Statistical Software 47*(12), 1–26.

Gallegos, M. T. (2002). Maximum likelihood clustering with outliers. In *Classification, Clustering, and Data Analysis*, pp. 247–255. Springer.

Gallegos, M. T. and G. Ritter (2005). A robust method for cluster analysis. *Annals of Statistics 33*(5), 347–380.

Gallegos, M. T. and G. Ritter (2013). Strong consistency of k-parameters clustering. *Journal of Multivariate Analysis 117*, 14–31.

García-Escudero, L. A., A. Gordaliza, C. Matrán, and A. Mayo-Iscar (2008). A general trimming approach to robust cluster analysis. *Annals of Statistics 38*(3), 1324–1345.

Hastie, T. and R. Tibshirani (1996). Discriminant adaptive nearest neighbor classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence 18*, 607–616.

Hennig, C. (2004). Breakdown points for maximum likelihood estimators of location-scale mixtures. *The Annals of Statistics 32*(4), 1313–1340.

Hennig, C. and B. Hausdorf (2015). *prabclus: Functions for clustering of presence-absence, abundance and multilocus genetic data.* CRAN. R package version 2.2-6.

Karlis, D. and E. Xekalaki (2003). Choosing initial values for the EM algorithm for finite mixtures. *Computational Statistics & Data Analysis 41*(3-4), 577–590.

Kiefer, J. (1953). Sequential minimax search for a maximum. *Proceedings of the American Mathematical Society 4*(3), 502–506.

McLachlan, G. J. and D. Peel (2000a). *Finite Mixture Models.* New York: Wiley.

McLachlan, G. J. and D. Peel (2000b). Robust mixture modelling using the t–distribution. *Statistics and Computing 10*(4), 339–348.

Müllensiefen, D. (2009). *Fantastic: Feature ANalysis Technology Accessing STatistics (In a Corpus): Technical Report v1.5*. London, United Kingdom: Goldsmiths University of London.

Venables, W. N. and B. D. Ripley (2002). *Modern Applied Statistics with S* (Fourth ed.). New York: Springer. ISBN 0-387-95457-0.