tclust: An R Package for a Trimming Approach to Cluster Analysis

Heinrich Fritz
Department of Statistics
and Probability Theory
Vienna University of Technology

Luis A. García-Escudero
Departamento de Estadística
e Investigación Operativa
Universidad de Valladolid

Agustín Mayo-Iscar
Departamento de Estadística
e Investigación Operativa
Universidad de Valladolid

Abstract

This introduction to the R package **tclust** is a (slightly) modified version of **?**, published in the *Journal of Statistical Software*.

Outlying data can heavily influence standard clustering methods. At the same time, clustering principles can be useful when robustifying statistical procedures. These two reasons motivate the development of feasible robust model-based clustering approaches. With this in mind, an R package for performing non-hierarchical robust clustering, called **tclust**, is presented here. Instead of trying to "fit" noisy data, a proportion α of the most outlying observations is trimmed. The **tclust** package efficiently handles different cluster scatter constraints. Graphical exploratory tools are also provided to help the user make sensible choices for the trimming proportion as well as the number of clusters to search for.

Keywords: Model-based clustering, trimming, heterogeneous clusters.

1. Introduction to robust clustering and tclust

Methods for cluster analysis attempt to detect homogeneous clusters with large heterogeneity among them. As happens with other (non-robust) statistical procedures, clustering methods may be heavily influenced by even a small fraction of outlying data. For instance, two or more clusters might be joined artificially, due to outlying observations, or "spurious" non-informative clusters may be composed of only a few outlying observations (see, e.g., **??**). Therefore, the application of robust methods in this context is very advisable, especially in fully automatic clustering (unsupervised learning) problems. Certain relations between cluster analysis and robust methods (**????**) also motivate interest in robust clustering techniques. For example, robust clustering techniques can be used to handle "clusters" of highly concentrated outliers which are especially dangerous in (non-robust) estimation. **?** provides a recent survey of robust clustering methods.

The **tclust** package for the R environment for statistical computing (**?**) implements different robust non-hierarchical clustering algorithms where trimming plays a key role. This package is available at `http://CRAN.R-project.org/package=tclust`. When trimming allows the removal of a fraction α of the "most outlying" data, the strong influence of outlying observations can be avoided. This trimming approach to clustering has been introduced in **?**, **?**, **?** and **?**. Trimming also serves to identify potentially interesting anomalous observations.

Trimming is not a new concept in statistics. For instance, the trimmed mean for one-dimensional data removes a proportion $\alpha/2$ each of the largest and smallest observations before computing the mean. However, it is not straightforward to extend this philosophy to cluster analysis, because most of these problems are of multivariate nature. Moreover, it is often the case that "bridge points" lying between clusters ought to be trimmed. Instead of forcing the statistician to define the regions to be trimmed in advance, the procedures implemented in **tclust** take the whole data structure into account in order to decide which parts of the sample should be discarded. By considering this type of trimming, these procedures are even able to trim outlying bridge points. The "self-trimming" philosophy behind these procedures is exactly the same as adopted by some well-known high breakdown-point methods (see, e.g., **?**).

As a first example of this trimming approach, let us consider the trimmed k-means method introduced in **?**. The function `tkmeans` from the **tclust** package implements this method. In the following example, this function is applied to a bivariate data set based on the Old Faithful geyser called `geyser2` that accompanies the **tclust** package. The code given below creates Figure 1.

```
R> library(robClus)
R> set.seed(100)
R> data(geyser2)
R> clus <- tkmeans2(geyser2, k = 3, alpha=0.03)
R> plot(clus, col = c(og, 2:4), tol.lwd = 1, tol.lty = 2)
```

In the data set `geyser2`, we are searching for $k = 3$ clusters and a proportion $\alpha = 0.03$ of the data is trimmed. The clustering results are shown in Figure 1. Among this 3% of trimmed data, we can see 6 anomalous "short followed by short" eruptions lengths. Notice that an observation situated between the clusters is also trimmed.

The package presented here adopts a "crisp" clustering approach, meaning that each observation is either trimmed or fully assigned to a cluster. In comparison, mixture approaches estimate a cluster pertinence probability for each observation. Robust mixture alternatives have also been proposed where noisy data is tried to be fitted through additional mixture components. For instance, package **mclust** (**???**) and the Fortran program **emmix** (**??**) implement such robust mixture fitting approaches. Mixture fitting results can be easily converted into a "crisp" clustering result by converting the cluster pertinence probabilities into 0-1 probabilities. Contrary to these mixture fitting approaches, the procedures implemented in the **tclust** package simply remove outlying observations and do not intend to fit them at all. Package **tlemix** (see **??**) also implements a closely related trimming approach. As described in Section 3, the **tclust** package focuses on offering adequate cluster scatter matrix constraints leading to a wide range of clustering procedures depending on the chosen constraint, and avoiding the occurrence of spurious non-interesting clusters.

The outline of the paper is as follows: In Section 2 we briefly review the so-called "spurious outliers" model and show how to derive two different clustering criteria from it. Different constraints on the cluster scatter matrices and their implementation in the **tclust** package are addressed in Section 3. Section 4 presents the numerical output returned by this package. Section 5 provides some brief comments concerning the algorithms implemented, and a comparison of **tclust** and several other robust clustering approaches are given in Section 6.

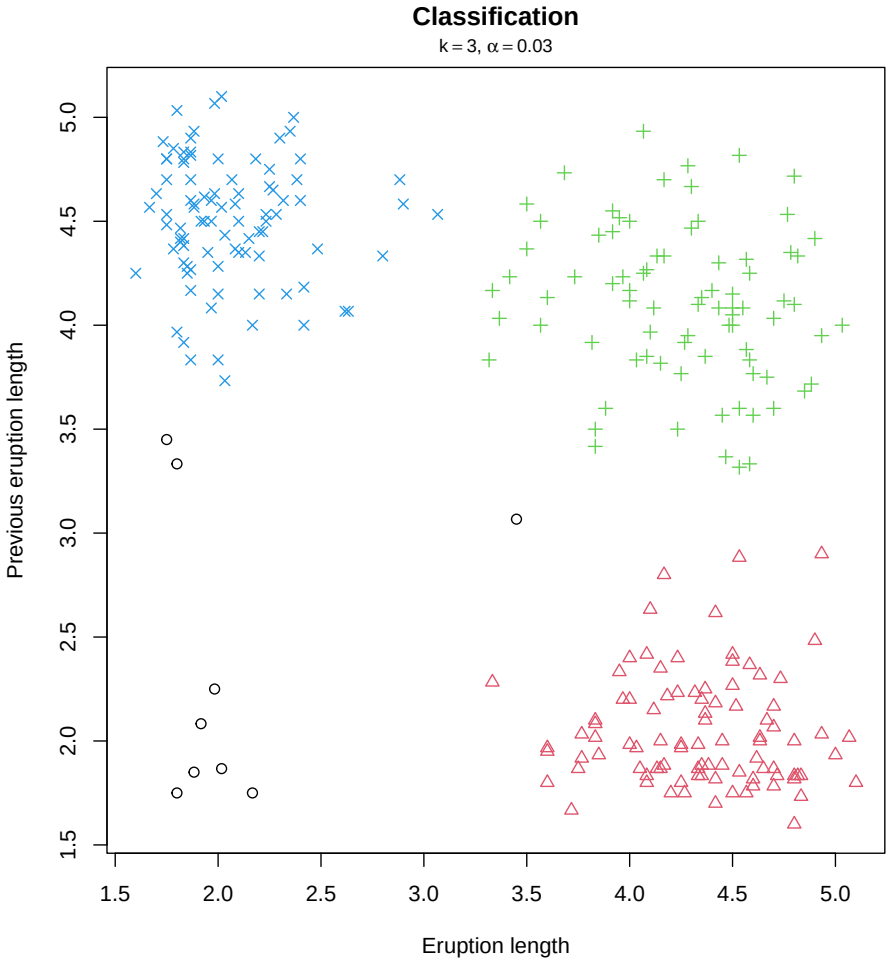

Figure 1: Trimmed k-means results with $k = 3$ and $\alpha = 0.03$ for the bivariate Old Faithful Geyser data. Trimmed observations are denoted by the symbol "$\circ$" (a convention followed in all the figures in this work).

Section 7 shows some graphical outputs that help advise the choice of the number of clusters and the trimming proportion. Other useful plots summarizing the robust clustering results are shown in Section 8. Finally, Section 9 applies the **tclust** package to a well-know real data set.

2. Trimming and the spurious outliers model

? and **?** propose the "spurious outliers model" as a probabilistic framework for robust crisp clustering. Let $f(\cdot; \mu, \Sigma)$ denote the probability density function of the p-variate normal distribution with mean μ and covariance matrix Σ. The "spurious-outlier model" is defined

through “likelihoods" like

$$\left[\prod_{j=1}^{k}\prod_{i\in R_j} f(x_i;\mu_j,\Sigma_j)\right]\left[\prod_{i\in R_0} g_i(x_i)\right] \tag{1}$$

with $\{R_0,...,R_k\}$ being a partition of the set of indices $\{1,2,...,n\}$ such that $\#R_0 = \lceil n\alpha\rceil$. R_0 are the indices of the “non-regular" observations generated by other (not necessarily normal) probability density functions g_i. “Non-regular" observations can be clearly considered as “outliers" if we assume certain sensible assumptions for the g_i (see details in **??**). Under these assumptions, the search of a partition $\{R_0,...,R_k\}$ with $\#R_0 = \lceil n\alpha\rceil$, vectors μ_j and positive definite matrices Σ_j maximizing (1) can be simplified to the same search (of a partition, vectors and positive definite matrices) by just maximizing

$$\sum_{j=1}^{k}\sum_{i\in R_j} \log f(x_i;\mu_j,\Sigma_j). \tag{2}$$

Notice that observations x_i with $i \in R_0$ are not taken into account in (2). Maximizing (2) with $k = 1$ yields the Minimum Covariance Determinant (MCD) estimator (**?**).

Unfortunately, the direct maximization of (2) is not a well-defined problem when $k > 1$. It is easy to see that (2) is unbounded without any constraint on the cluster scatter matrices Σ_j. The `tclust` function from the **tclust** package approximately maximizes (2) under different cluster scatter matrix constraints which will be shown in Section 3.

The maximization of (2) implicitly assumes equal cluster weights. In other words, we are ideally searching for clusters with equal sizes. The function `tclust` provides this option by setting the argument `equal.weights = TRUE`. The use of this option does not guarantee that all resulting clusters exactly contain the same number of observations, but the method hence prefers this type of solutions.

Alternatively, different cluster sizes or cluster weights can be considered by searching for a partition $\{R_0,...,R_k\}$ (with $\#R_0 = \lceil n\alpha\rceil$), vectors μ_j, positive definite matrices Σ_j and weights $\pi_j \in [0,1]$ maximizing

$$\sum_{j=1}^{k}\sum_{i\in R_j} (\log \pi_j + \log f(x_i;\mu_j,\Sigma_j)). \tag{3}$$

The (default) option `equal.weights = FALSE` is used in this case. Again, the scatter matrices also have to be constrained such that the maximization of (3) becomes a well-defined problem. Note that (3) simplifies to (2) when assuming `equal.weights = TRUE` and all weights are equally set to $\pi_j = 1/k$.

	`equal.weights = TRUE`	`equal.weights = FALSE`
`restr = "eigen"`	*k-means* **?**	**?**
`restr = "deter"`	**?**	This work

Table 1: Clustering methods handled by **tclust**. Names in cursive letters are untrimmed ($\alpha = 0$) methods.

3. Constraints on the cluster scatter matrices

As already mentioned, the function `tclust` implements different algorithms aimed at approximately maximizing (2) and (3) under different types of constraints which can be applied on the scatter matrices Σ_j. The type of constraint is specified by the argument `restr` of the `tclust` function. Table 1 gives an overview of the different clustering approaches implemented by the `tclust` function depending on the chosen type of constraint.

Imposing constraints is compulsory because maximizing (2) or (3) without any restriction is not a well-defined problem. Notice that an almost degenerated scatter matrix Σ_j would cause trimmed log-likelihoods (2) and (3) to tend to infinity. This issue can cause a (robust) clustering algorithm of this type to end up finding "spurious" clusters almost lying in lower-dimensional subspaces. Moreover, the resulting clustering solutions might heavily depend on the chosen constraint. The strength of the constraint is controlled by the argument `restr.fact` ≥ 1 in the `tclust` function. The larger `restr.fact` is chosen, the looser is the restriction on the scatter matrices, allowing for more heterogeneity among the clusters. On the contrary, small values of `restr.fact` close to 1 imply very "equally scattered" clusters. This idea of constraining cluster scatters to avoid spurious solutions goes back to **?**, who proposed it in mixture fitting problems.

Also arising from the spurious outlier model, other types of constraints have recently been introduced by **??**. These (closely related) constraints also serve to avoid degeneracy of trimmed likelihoods but they are not implemented in the current version of the **tclust** package.

3.1. Constraints on the eigenvalues

Based on the eigenvalues of the cluster scatter matrices, a scatter similarity constraint may be defined. With $\lambda_l(\Sigma_j)$ as the eigenvalues of the cluster scatter matrices Σ_j and

$$M_n = \max_{j=1,\ldots,k} \max_{l=1,\ldots,p} \lambda_l(\Sigma_j) \text{ and } m_n = \min_{j=1,\ldots,k} \min_{l=1,\ldots,p} \lambda_l(\Sigma_j) \tag{4}$$

as the maximum and minimum eigenvalues, the restriction `restr = "eigen"` constrains the ratio M_n/m_n to be smaller or equal than a fixed value `restr.fact` ≥ 1. A theoretical study of the properties of this approach with `equal.weights = FALSE` can be found in **?**.

This type of constraint limits the relative size of the axes of the equidensity ellipsoids defined through the obtained Σ_j when assuming normality. This way we are simultaneously controlling the relative group sizes and also the deviation from sphericity in each cluster.

Setting `equal.weights = TRUE`, `restr = "eigen"` and `restr.fact = 1` implies the most constrained case. In this case, the `tclust` function tries to solve the trimmed k-means problem as introduced by **?**. This problem simplifies to the well-known k-means clustering criterion

when no trimming is done (i.e., `alpha = 0`). The `tkmeans` function directly implements this most constrained application of the `tclust` function.

3.2. Constraints on the determinants

Another way of restricting cluster scatter matrices is constraining their determinants. Thus, if

$$M_n = \max_{j=1,...,k} |\Sigma_j| \text{ and } m_n = \min_{j=1,...,k} |\Sigma_j|$$

are the maximum and minimum determinants, we attempt to maximize (2) or (3) by constraining the ratio M_n/m_n to be smaller or equal than a fixed value `restr.fact`. This is done in the function `tclust` by using the option `restr = "deter"`.

Now, this type of constraint limits the relative volumes of the mentioned equidensity ellipsoids, but not the cluster shapes. The use of this type of constraint is particularly advisable when affine equivariance is required because this property is satisfied when `restr = "deter"`.

The untrimmed case `alpha = 0`, `restr = "deter"` and `restr.fact = 1` was already outlined in **?**, as the only sensible way to avoid (Mahalanobis distance modified) k-means type algorithms to return clusters of a few almost collinear observations. The possibility of trimming data was also considered in **?** who implicitly assumed $|\Sigma_1| = ... = |\Sigma_k|$ (and so `restr.fact = 1`). The package presented here extends her approach to more general cases (`restr.fact` > 1).

3.3. Example

In this example, we examine the influence of different constraints by applying the function `tclust` to the so-called `M5data` data set. This data set, which accompanies the **tclust** package, has been generated following the simulation scheme M5 introduced in **?**. Thus it is a bivariate mixture of three simulated gaussian components with very different scatters and a clear overlap between two of these components. A 10% proportion of outliers is also added in the outer region of the bounding rectangle enclosing the three gaussian components. See Figure 2 for a graphical representation and **?** for more details on the structure of this `M5data` data set. Executing the following code yields Figure 3.

```
R > data ("M5data")
R > x <- M5data[, 1:2]
R > res.a <- tclust(x, k = 3, alpha = 0.1, restr.fact = 1,  restr = "eigen",
    + equal.weights = TRUE)
R > res.b <- tclust(x, k = 3, alpha = 0.1, restr.fact = 50, restr = "eigen",
    + equal.weights = FALSE)
R > res.c <- tclust(x, k = 3, alpha = 0.1, restr.fact = 1,  restr = "deter",
    + equal.weights = TRUE)
R > res.d <- tclust(x, k = 3, alpha = 0.1, restr.fact = 50, restr = "deter",
    + equal.weights = FALSE)
R > plot(res.a, main = "/r")
R > plot(res.b, main = "/r")
R > plot(res.c, main = "/r")
R > plot(res.d, main = "/r")
```

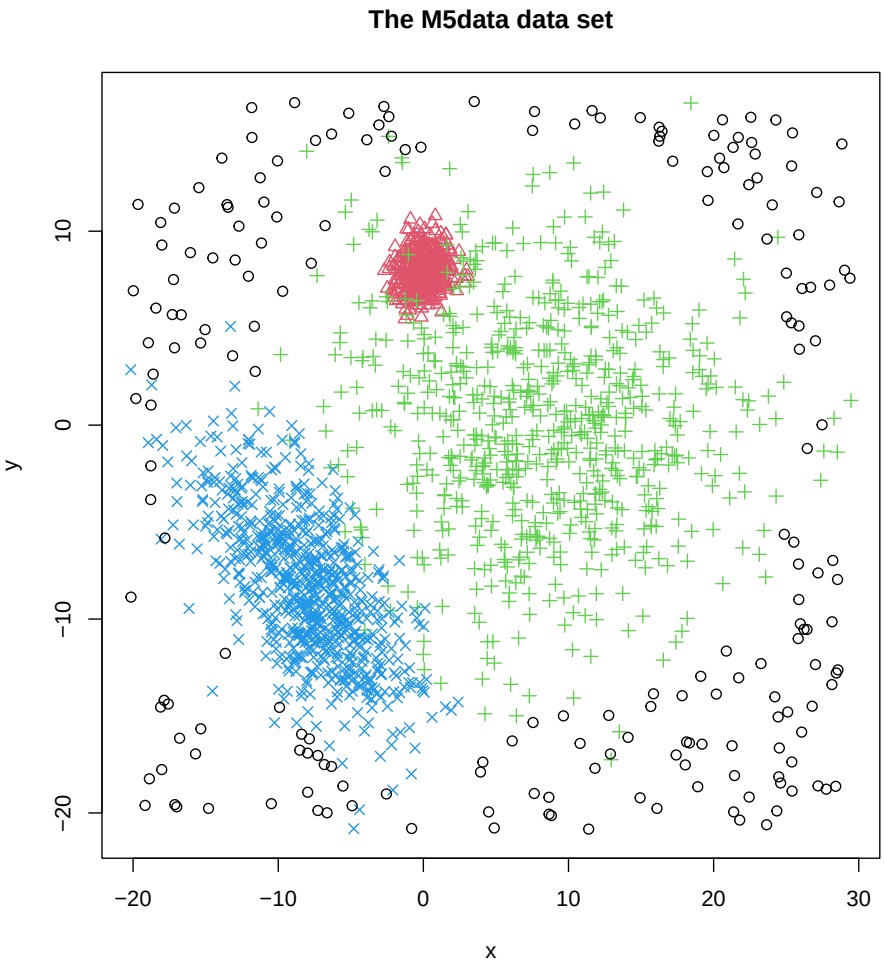

Figure 2: A scatter plot of the `M5data` data set. Different symbols are used for the data points generated by each of the three bivariate normal components and "○" for the added outliers.

Although different constraints are imposed, we are searching for $k = 3$ clusters and the trimming proportion is set to $\alpha = 0.1$ in all the cases. Note that only the clustering procedure introduced in **?**, shown in Figure 3,(d), with a sufficiently large value of `restr.fact` approximately returns the three original clusters in spite of the very different cluster scatters and different cluster sizes. Moreover, this clustering procedure adequately handles the severe overlap of two clusters. The value `restr.fact = 50` has been chosen in this case because the eigenvectors of the covariance matrices of the three gaussian components satisfy restriction (4) for this value. Due to their underlying assumptions, the other three clustering methods (trimmed k-means in Figure 3,(a), **?** in (b), **?** in (c)) return rather similarly structured clusters. In fact, we found spherical clusters in (a), clusters with the same scatter matrix in (b) and clusters with the same cluster scatter matrix determinant in (c). The `M5data` is perhaps a very "extreme" situation and restriction settings in (a), (b) and (c) can be useful (and easier to be interpreted) with not so extreme data sets and where the assumptions implied by these

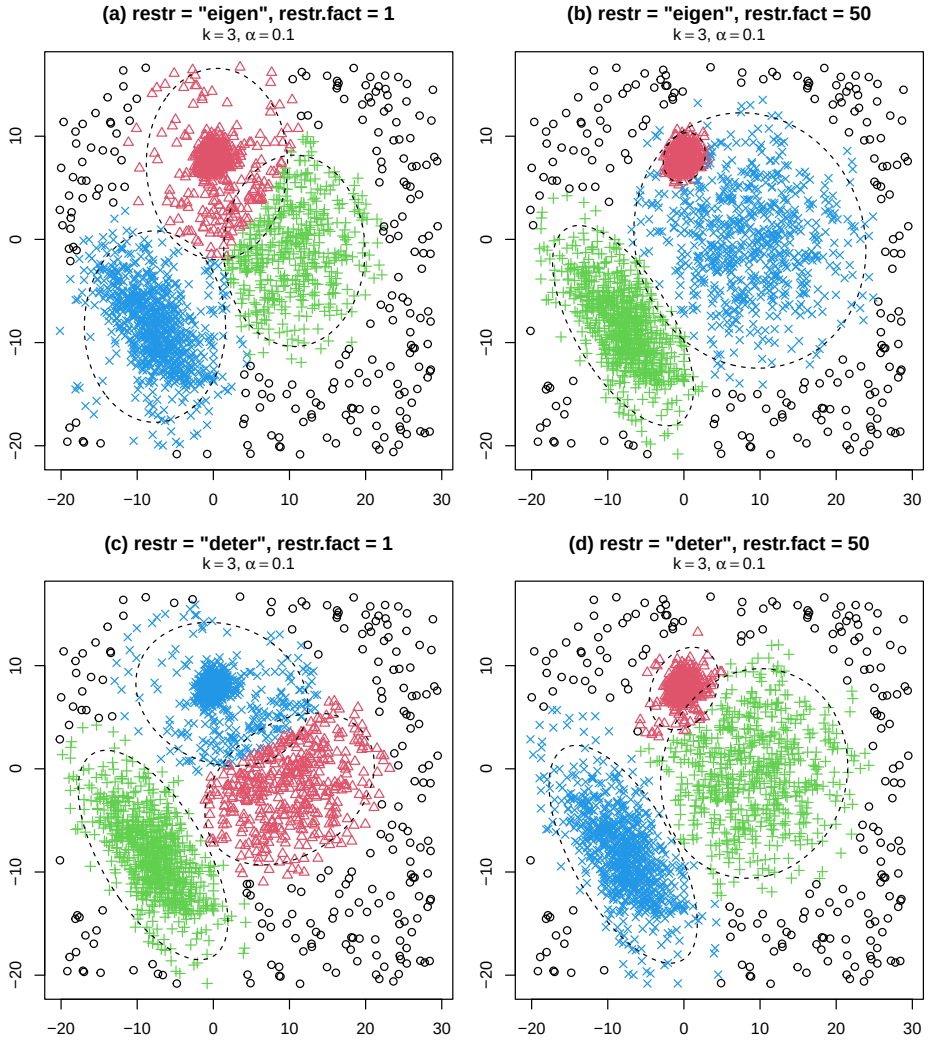

Figure 3: Results of the clustering processes for the `M5data` data set for different constraints on the cluster scatter matrices and the parameters $\alpha = 0.1$ and $k = 3$. Different colors and symbols represent each observation's individual cluster assignment.

restriction settings hold.

4. Numerical output

The function `tclust` returns an S3 object containing the cluster centers μ_j by columns (`$centers`), scatter matrices Σ_j as an array (`$cov`), the weights (`$weights`), the number of observations in each group (`$size`) and the maximum value found for the trimmed log-likelihood objective function (2) or (3) (`$obj`). The vector `$cluster` provides the cluster assignment of each observation, whereas an artificial cluster "`0`" (without location and scatter information) is introduced which holds all trimmed data points.

Sometimes, (2) and (3) maximize with some clusters remaining empty (see Figure 4). In this

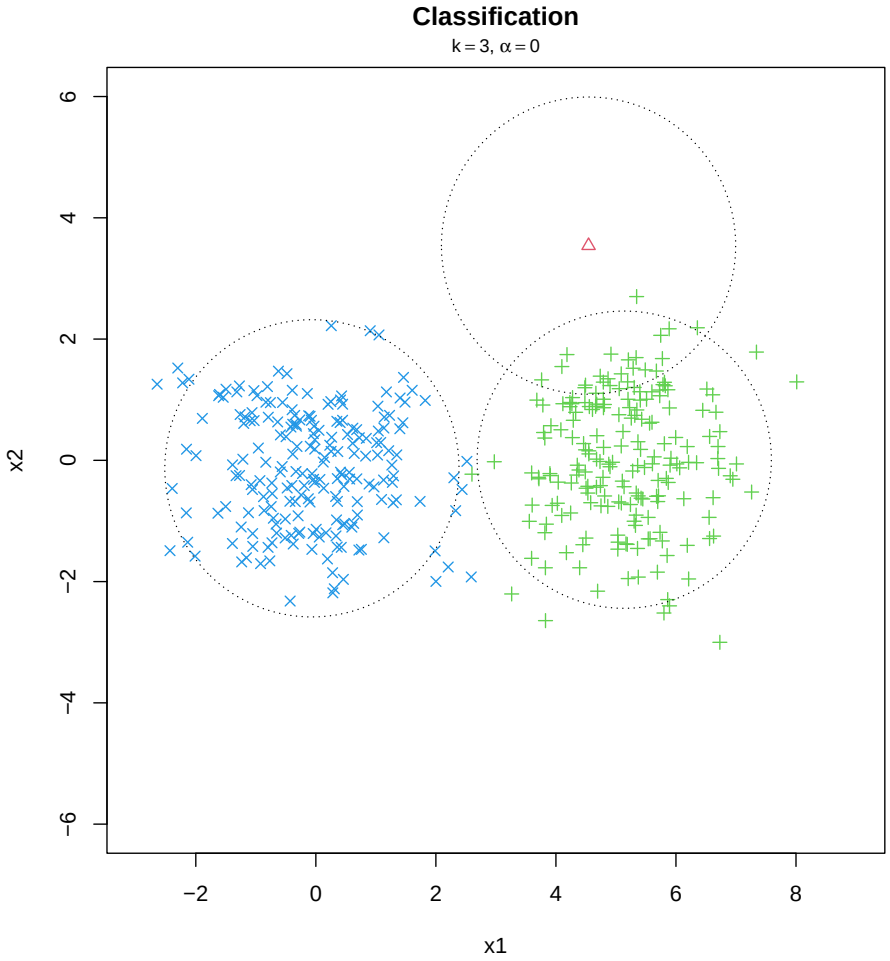

Figure 4: Applying `tclust` with $k = 3$ and $\alpha = 0$ on a simulated data set which originally consists of 2 clusters when `equal.weights = FALSE`.

case, only information on the non-empty groups is returned. Notice that, if we are searching for k clusters, empty clusters can be found when a clustering solution for a number of clusters strictly smaller than k attains a higher value for (2) or (3) than the best solution found with k clusters. In this case, artificial empty clusters may be defined by considering sufficiently remote centers μ_j and scatter matrices Σ_j satisfying the specific constraints that are assigned to these empty clusters. They are chosen such that $f(\cdot; \mu_j, \Sigma_j)$ gives almost null density to all the observations in the sample. These artificially added centers and scatter matrices are not returned as output by the `tclust` function and a warning is issued. For instance, let us consider the following code

```
R > set.seed(10)
R > x <- rbind(MASS::mvrnorm(200, c (0, 0), diag (2)),
    +          MASS::mvrnorm(200, c (5, 0), diag (2)))
R > clus <- tclust(x, k = 3, alpha = 0, restr.fact = 1)
```

```
R > plot(clus)
```

Although we are searching for $k = 3$ clusters, Figure 4 and the issued warning show that only 2 clusters are found. Notice that $k = 2$ is surely a more sensible choice for the number of clusters than $k = 3$ for this generated data set. Therefore, the detection of empty clusters, or clusters with few data points, can be helpful, providing valuable tools for making sensible choices for k as we will see in Section 7. On the other hand, the detection of empty clusters is very unlikely to happen when the argument `equal.weights = TRUE` is provided in the call to `tclust`.

5. Algorithms

The maximization of (2) or (3) considering different cluster scatter matrix constraints is not straightforward because of the combinatorial nature of the associated maximization problems.

The algorithm presented in **?** can be adapted to approximately solve all these problems. The methods implemented in **tclust** could be seen as Classification EM algorithms (**??**), whereas a certain type of "concentration" steps (see the fast-MCD algorithm in **?**) is also applied. In fact, the concentration steps applied by the package **tclust** can be considered as an extension of those applied by the batch-mode k-means algorithm (**??**). It can be seen that the target function always increases throughout the application of concentration steps, whereas several random start configurations are needed in order to avoid ending trapped in local maxima. Therefore, `nstart` random initializations and `iter.max` concentration steps are considered. The probability that the algorithm converges close to the global optimum maximizing (2) or (3) increases with larger values of `nstart` and `iter.max`. The drawback of high values of `nstart` and `iter.max` obviously is the increasing computational effort.

In the concentration step, the centers and scatter matrices are updated by considering the cluster sample means and cluster sample covariance matrices. New cluster assignments are obtained by gathering the "closest" observations to the new centers. Mahalanobis distances, based on the computed cluster sample covariance matrices, are used in order to decide which are the closest observations to each center. If needed, in the updating step, the cluster sample covariance matrices are modified as little as possible but in such a way that they satisfy the desired constraints (**?**). The main idea behind these "constrained" concentration steps is to replace the eigenvalues of the sample covariance matrices by optimally truncated eigenvalues, which satisfy the desired constraint. A more detailed description of the algorithm applied by `tclust` and the way the restrictions are forced onto the cluster scatter matrices can be found in **?**.

6. Comparison with other robust clustering proposals

In this section, we briefly compare the performance of the clustering procedures implemented in the **tclust** package with respect to other robust clustering proposals in the literature.

The Partitioning Around Medoids (PAM) clustering method (**?**) has been proposed as a robust alternative to k-means clustering. It can be seen that the effect of the 6 anomalous "short followed by short" eruptions lengths in the lower left corner of Figure 1 do not affect the position of the k-medoid centers (see Figure 5,(a)) too much, nor the resulting clusters.

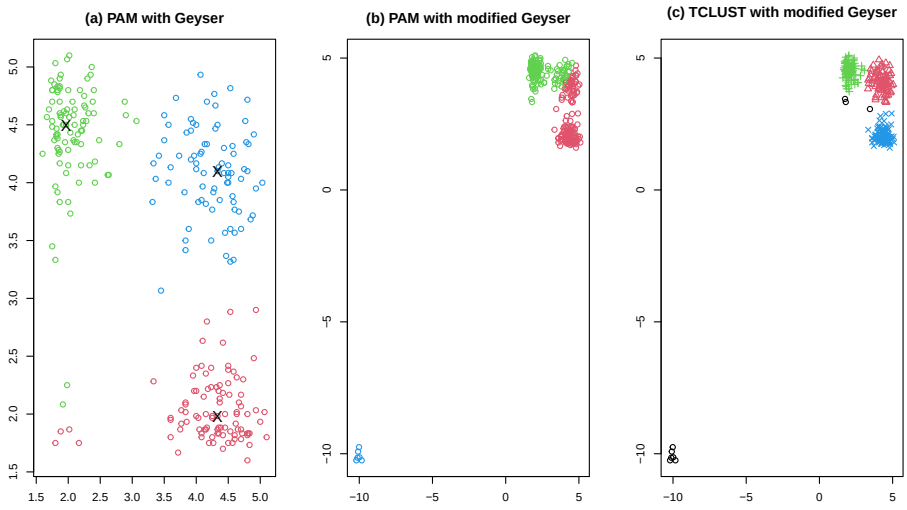

Figure 5: PAM's clustering results for the `geyser2` data with 3-medoids denoted by the symbol "×" in (a). PAM's clustering results for a modified `geyser2` data set in (b) and when applying `tkmeans` with $k = 3$ and $\alpha = 0.03$ in (c).

However, in Figure 5,(b), we see that the clustering results with $k = 3$ are strongly affected when moving these 6 anomalous points toward a more distant position. On the other hand, in Figure 5,(c), we can see that these outlying data points do not affect the trimmed k-means based clustering at all once that they are trimmed.

In fact, only one single outlier placed in a very remote position is able to completely break down the PAM method (**?**). This also happens when applying **emmix**, which has a breakdown point of zero (**?**). The **emmix** approach is able to obtain appropriate clustering results for the two data sets made of mixtures of symmetric and asymmetric t components as those shown in Figure 6. These two data sets contain three main clusters with some distant observations in the heavy tails of these t components, which would be considered as outliers when assuming normality. When applying the classification EM algorithm without trimming to these data sets, we are not able to find the three cluster structures and two main clusters are artificially joined together. However, when considering $\alpha = 0.05$, **tclust** perfectly avoids the harmful effect of the observations in the tails and still discovers the three clusters. In fact, almost all non-trimmed observations are correctly clustered in both, symmetric and asymmetric, cases. Any $\alpha > 0$ discarding the most outlying observations would give similar results. Moreover, it may be seen that the shape of the elliptical clusters are essentially discovered in the case of the symmetric-elliptical t components. In this example, we see that applying **tclust** to data sets including non-normally distributed components as those in Figure 6 may result in proper clustering solutions, this however cannot be guaranteed if the underlying distributions differ too much from normality. The closely related **tlemix** package allows to consider other non-normal models by taking advantage of the flexibility provided by the **FlexMix** package (**?**). On the other hand, **tclust** focuses on normally distributed components and on the implementation of appropriate cluster scatter matrix constraints while **tlemix** does not. The **tlemix** mainly controls the minimum number of observations in a cluster.

The widely used **mclust** package considers a uniformly distributed component for explaining

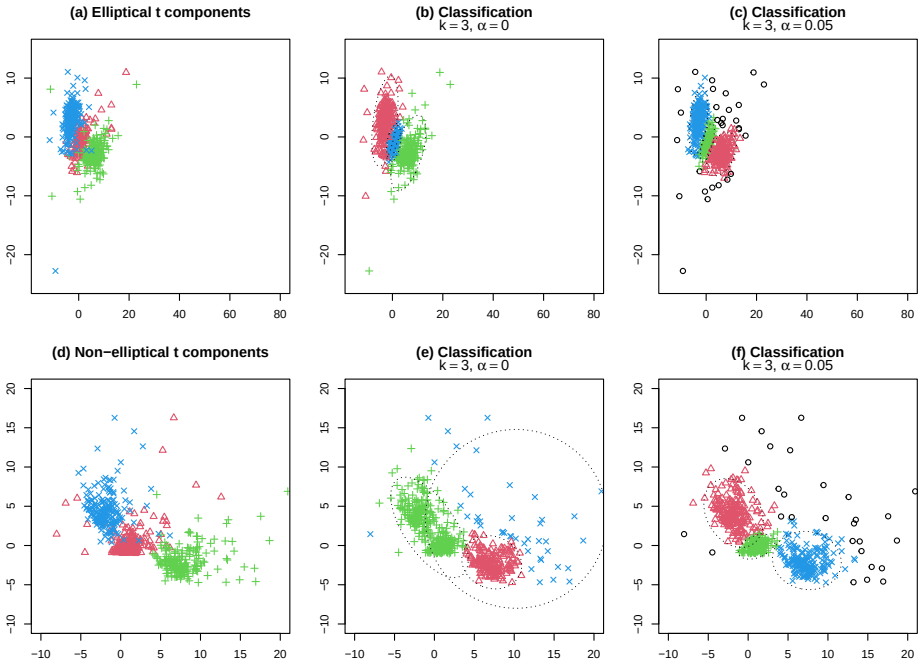

Figure 6: A data set made up of three elliptical t components is shown in (a) and three asymmetrical t components in (d). The associated clustering results when applying the `tclust` function with $k = 3$ and $\alpha = 0$ are shown in (b) and (e) and with $k = 3$ and $\alpha = 0.05$ in (c) and (f).

outlying data points. As we can see, this uniform component successfully accommodates the 10% "background noise" as seen in Figure 7,(a). However, it is not able to cope with a more structured noise pattern like the "helix" in Figure 7,(c) which also accounts for 10% of the data, although the information of a 10% contamination level was passed to **mclust**. Alternatively, the **tclust** package with $k = 2$ and $\alpha = 0.1$ properly discovers the outlying data points without trying to fit them.

Since groups of outliers may be considered as further clusters, it could be argued that robust clustering problems can always be solved by increasing the number of groups we are searching for. However, as explained in **?**, this is not necessarily the best strategy. Firstly, sometimes the researcher fixes the number of clusters in advance, not being aware of the presence of a small amount of outlying observations. Secondly, it could lead to a quite large number of clusters when very scattered outliers are present in the data set.

A clear limitation of **tclust** is that it is not applicable on high-dimensional data sets, as the method in its current definition definitely needs a data set containing more observations than dimensions.

7. Selecting the number of groups and the trimming size

Perhaps one of the most complex problems when applying cluster analysis is the choice of the number of clusters, k. In some cases one might have an idea of the number of clusters in

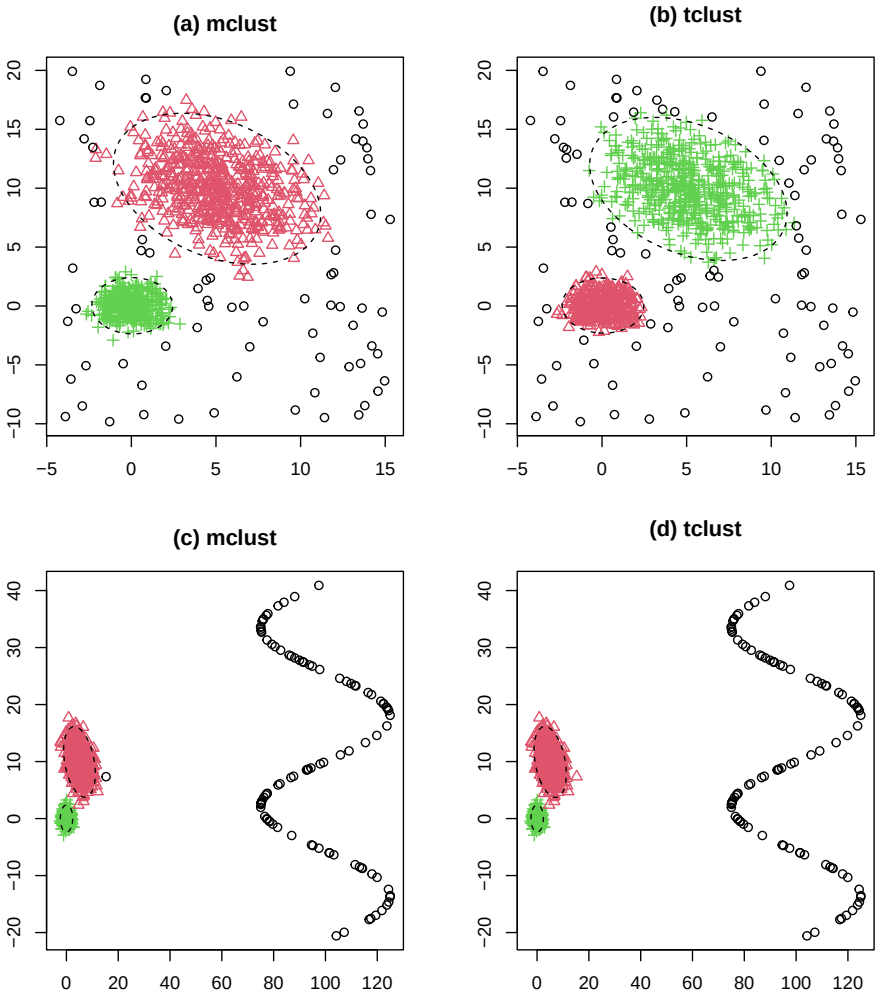

Figure 7: Clustering results for two simulated data sets when applying **mclust** with $k = 2$ in (a) and (c) and **tclust** with $k = 2$ and $\alpha = 0.1$ in (b) and (d).

advance, but usually k is completely unknown. Moreover, in the approach proposed here, the trimming proportion α has also to be chosen without knowing the true contamination level.

As we will see through the following example, the choices for k and α are related problems that should be addressed simultaneously. It is important to see that a particular trimming level implies a specific number of clusters and vice versa. This dependency can be explained as entire clusters tend to be trimmed completely when increasing α. On the other hand, when choosing α too low, groups of outliers might form new spurious clusters and thus it appears that the number of clusters found in the data set is higher. Moreover, the simultaneous choice of k and α depends on the type of clusters we are searching for and on the allowed differences between cluster sizes. These two aspects can be controlled by the choice of arguments `restr` and `restr.fact`.

To demonstrate the relation between α, k and `restr.fact`, let us consider `restr = "eigen"` and the data set in Figure 8 which could either be interpreted as a mixture of three components

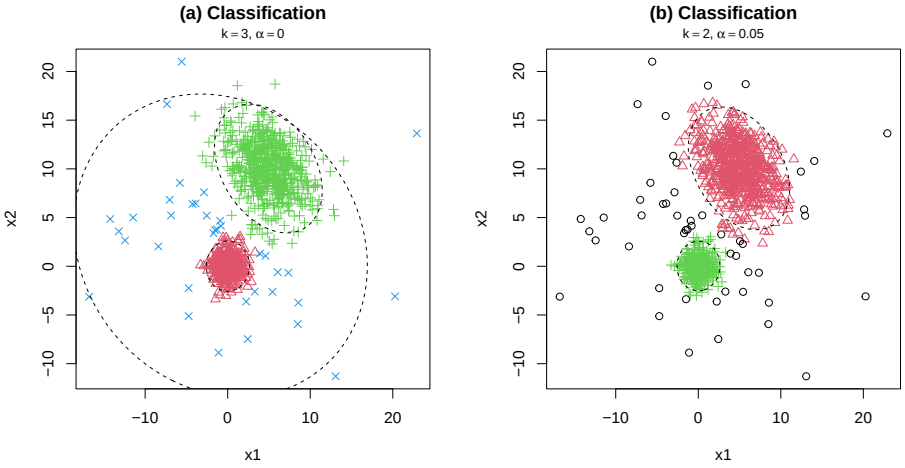

Figure 8: Clustering results for the simulated data set `mixt` with $k = 3$, $\alpha = 0$ and `restr.fact` $= 50$ (a) and $k = 2$, $\alpha = 0.1$ and `restr.fact` $= 8$ (b).

(a) or a mixture of two components (b) with a 10% outlier proportion. Both clustering solutions shown in Figure 8 are perfectly sensible and the final choice of α and k only depends on the value given to `restr.fact`. The code used to obtain Figure 8 is the following:

```
R > sigma1 <- diag(2)           ## EigenValues: 1, 1
R > sigma2 <- diag(2) * 8 - 2   ## EigenValues: 8, 4
R > sigma3 <- diag(2) * 50      ## EigenValues: 50, 50
R > mixt <- rbind(
    + MASS::mvrnorm(360, mean = c (0.0,  0), sigma = sigma1),
    + MASS::mvrnorm(540, mean = c (5.0, 10), sigma = sigma2),
    + MASS::mvrnorm(100, mean = c (2.5,  5), sigma = sigma3))
R > plot(tclust(mixt, k = 3, alpha = 0.00, restr.fact = 50))
R > plot(tclust(mixt, k = 2, alpha = 0.05, restr.fact = 12))
```

Considering `sigma1` and `sigma2`, the quotient of the largest and smallest eigenvalue is 8, whereas the maximal quotient is 50 if we consider `sigma1`, `sigma2` and `sigma3`. Thus `restr.fact` $= 8$ would allow to consider two clusters while `restr.fact` $= 50$ would also allow to assume three groups there. Although the proportion of "contaminated" data is equal to 10%, the trimming level must be reduced to 5%, when considering $k = 2$, because the third (more scattered) gaussian component partially overlaps with the other two components.

Let us assume first that `restr` and `restr.fact` have been fixed in advance by the researcher who applies the robust clustering method. Even with this information and assuming $\alpha = 0$, choosing the appropriate number of clusters is not an easy task. The careful monitoring of the maximum value attained by log-likelihoods like those in (2) and (3) while changing k has traditionally been applied as a method for choosing the number of clusters when $\alpha = 0$. Moreover **?** stated that the use of "weighted" log-likelihoods (3) is preferred to the use of log-likelihoods assuming equal weights (2). Notice that increasing k always causes the maximized log-likelihood (2) to increase too, and this could lead to "overestimate" the appropriate number of clusters (see **?**).

In this trimming framework, let us consider $\mathcal{L}^{\Pi}_{\texttt{restr.fact}}(\alpha, k)$ as the maximum value reached by (3) for each combination of a given set of values for k and α. **?** propose to monitor the "classification trimmed likelihoods" functionals

$$(\alpha, k) \mapsto \mathcal{L}^{\Pi}_{\texttt{restr.fact}}(\alpha, k)$$

while altering α and k, which yields an exploratory graphical tool for making sensible choices for parameters α and k. In fact, it is proposed to choose the number of clusters as the smallest value of k such that

$$\mathcal{L}^{\Pi}_{\texttt{restr.fact}}(\alpha, k+1) - \mathcal{L}^{\Pi}_{\texttt{restr.fact}}(\alpha, k) \tag{5}$$

is (close to) 0 except for small values of α. Once the number of clusters is fixed, a good choice for the trimming level is the first α_0 such that (5) is (close to) 0 for every $\alpha \geq \alpha_0$. Although we are convinced that monitoring the classification trimmed likelihoods functionals is very informative, no theoretical statistical procedures are available yet for determining when (5) can be formally considered as "close to 0".

The function `ctlcurves` in package **tclust** approximates the classification trimmed likelihoods by successively applying the `tclust` function for a sequence of values of k and α. A default value `restr.fact = 50` is considered but, if desired, other values of `restr.fact` can be passed to `tclust` via `ctlcurves` too.

For instance, the following code applied to the previously simulated `mixt` data set

```
R > plot (ctlcurves (mixt, k = 1:4, alpha = seq (0, 0.2, by = 0.05)))
```

results in Figure 9. This figure shows that increasing k from 2 to 3 is needed when $\alpha = 0$, as the objective functions value differs noticeably between $k = 2$ and $k = 3$. On the other hand, increasing k from 2 to 3 is not needed anymore as the third (more scattered) "cluster" vanishes when trimming 5% of the most outlying observations. Thus, there is no discernable difference of the objective functions value with $\alpha \geq \alpha_0 = 0.05$ and $k \geq 2$. Increasing k from 3 to 4 is not needed in any case.

The previously described procedure for making sensible choices for parameters k and α requires an active role from the researcher. The type of restriction and the allowed restriction factor, which do not necessarily depend on the given data set, must be specified in advance. For instance, some specific clustering applications like "location-facilities" problems require almost spherical clusters that can be obtained by setting `restr = "eigen"` and a `restr.fact` close to 1. The researcher's decision on the restriction consequently modifies the proper determination of parameters k and α.

Due to the important role of the statement of `restr` and `restr.fact`, some general guideline for fixing them will be given here. For instance, as already commented, fixing `restr = "deter"` is recommended when only the relative cluster sizes shall be constrained, or when affine equivariance is clearly needed. On the other hand, using `restr = "eigen"` is advised when we want to simultaneously constrain relative cluster sizes and shapes.

With respect to the choice of `restr.fact`, we recommend to initially use large values when applying `ctlcurves`, thus, providing high flexibility to the clustering method. The default value `restr.fact = 50` is suggested for `ctlcurves`, as it worked well with a lot of data sets (especially if the variables have been properly standardized through the `scale` argument in the `tclust` function). The so obtained "sensible" values for k and α and their associated

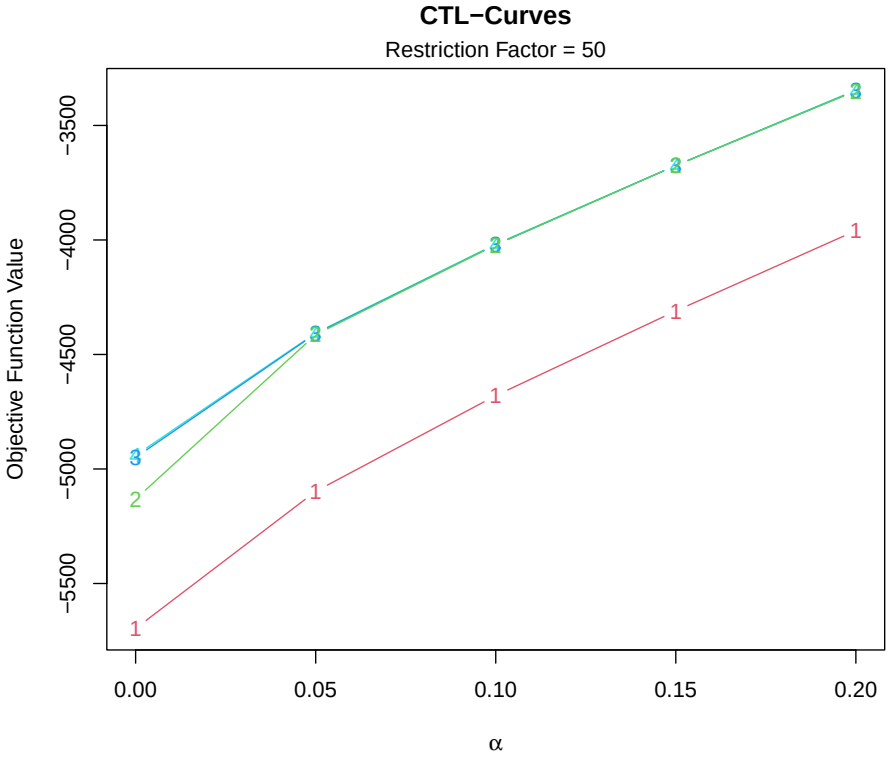

Figure 9: Classification trimmed likelihoods with $k = 1, ..., 4$, $\alpha = 0$, 0.05, ..., 0.2 and `restr.fact` $= 50$ for the `mixt` data set in Figure 8.

clustering solutions must be explored carefully. For instance, `tclust` issues a warning when the returned clustering solution has been "artificially restricted" by the algorithm, as shown in Section 9. This means, that the values M_n and m_n (see Section 3.1 and Section 3.2) derived from the returned scatter matrices satisfy $M_n/m_n =$ `restr.fact`, because the algorithm has forced the chosen constraint, since the (unconstrained) group sample covariance matrices do not satisfy $M_n/m_n \leq$ `restr.fact`. In this situation, if no specific constraints are required, `restr.fact` may be increased stepwise until this warning disappears. Moreover, printing the object returned by the `ctlcurves` function points out all "artificially restricted" solutions for each computed combination of parameters k and α. In this way, if desired, we can easily search for clustering solutions which are not artificially restricted and do not contain spurious clusters. Finally, the exploratory tools in Section 8 also help to evaluate whether all these parameters are reasonably chosen.

Note that arguments `nstart` and `iter.max` may be provided in the call to `ctlcurves` and they are internally passed to function `tclust`.

The curves presented in **?** can be considered as precedents of those we obtain by using the `ctlcurves` function. Trimmed likelihoods have also been taken into account in **?** for choosing k and α by using a BIC criterion.

8. Graphical displays

As seen in previous examples, the package **tclust** provides functions for visualizing the computed cluster assignments. One-dimensional, two-dimensional and higher-dimensional cases are visualized differently:

$p = 1$**:** The one-dimensional data set with the corresponding cluster assignments is displayed along the x-axis. Setting the argument `jitter = TRUE` jitters the data along the y-axis in order to increase the visibility of the actual data structure. Additionally, a (robust) scatter estimation of each cluster is also displayed.

$p = 2$**:** Tolerance ellipsoids are plotted additionally in order to visualize the estimated cluster scatter matrices.

$p > 2$**:** The first two Fisher's canonical coordinates are displayed in this case, which are computed based on the estimated cluster scatter matrices. Notice that trimmed observations are not taken into account when computing these coordinates, since they have been completely discarded. The implementation of these canonical coordinates is derived from the function `discrcoord` as implemented in the package **fpc** (**?**).

A simple example demonstrates how the `plot` function works in different dimensions. The code:

```
R > geyser1 <- geyser2[, 1, drop = FALSE]
R > geyser3 <- cbind (geyser2, rnorm (nrow (geyser2)))
R > plot (tkmeans (geyser1, k = 2, alpha = 0.03), jitter = TRUE)
R > plot (tkmeans (geyser3, k = 3, alpha = 0.03))
```

yields Figure 10. For demonstrating the different plotting modes, we have selected one single variable from `geyser2` to obtain a one-dimensional data set (`geyser1`), and, added an additional normally distributed variable to `geyser2`, yielding a three-dimensional data set (`geyser3`). Figure 10 plots the results of the trimmed k-means robust clustering method for these two generated data sets.

Given a `tclust` object, some additional exploratory graphical tools can be applied in order to evaluate the quality of the cluster assignments and the trimming decisions. This is done by applying the function `DiscrFact`.

Let $\widehat{R} = \{\widehat{R_0}, \widehat{R_1}, ..., \widehat{R_k}\}$, $\widehat{\theta} = (\widehat{\theta_1}, ..., \widehat{\theta_k})$ and $\widehat{\pi} = (\widehat{\pi_1}, ..., \widehat{\pi_k})$ be the values obtained by maximizing (2) or (3) (we set $\widehat{\pi_j} = 1/k$ when maximizing (2)). $D_j(x_i; \widehat{\theta}, \widehat{\pi}) = \widehat{\pi_j}\phi(x_i, \widehat{\theta_j})$ is a measure of the degree of affiliation of observation x_i with cluster j. These values can be ordered as $D_{(1)}(x_i; \widehat{\theta}, \widehat{\pi}) \leq ... \leq D_{(k)}(x_i; \widehat{\theta}, \widehat{\pi})$. Thus the quality of the assignment decision of a non trimmed observation x_i to the cluster j with $D_{(k)}(x_i; \widehat{\theta}, \widehat{\pi}) = D_j(x_i; \widehat{\theta}, \widehat{\pi})$ can be evaluated by comparing its degree of affiliation with cluster j to the best second possible assignment. That is

$$\mathrm{DF}(i) = \log\left(D_{(k-1)}(x_i; \widehat{\theta}, \widehat{\pi}) / D_{(k)}(x_i; \widehat{\theta}, \widehat{\pi})\right) \text{ for } x_i \text{ not trimmed.}$$

Let $x_{(1)}, ..., x_{(n)}$ be the observations in the sample after being sorted according to their $D_{(k)}(\cdot; \widehat{\theta}, \widehat{\pi})$ values, i.e., $D_{(k)}(x_{(1)}; \widehat{\theta}, \widehat{\pi}) \leq ... \leq D_{(k)}(x_{(n)}; \widehat{\theta}, \widehat{\pi})$. It is not difficult to see that

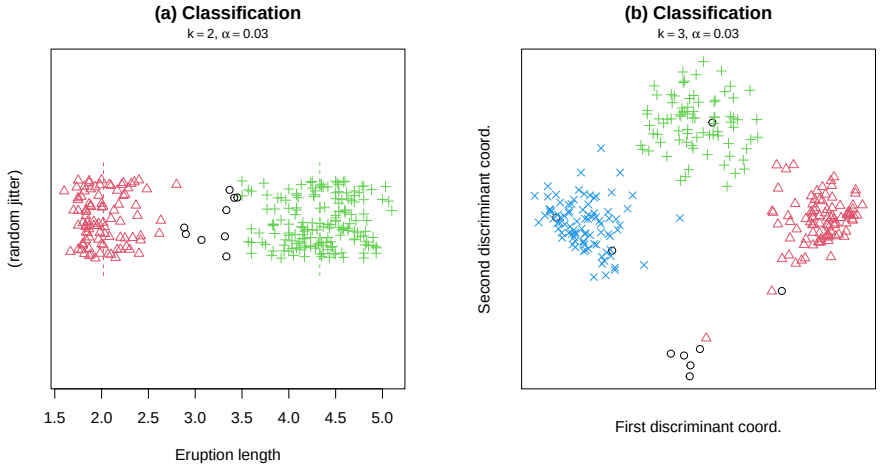

Figure 10: Trimmed k-means clustering results for `geyser1` (one-dimensional) in (a) and for `geyser3` (three-dimensional) in (b). These two data sets are based on `geyser2`. $k = 2$ is fixed in (a) and $k = 3$ in (b) while $\alpha = 0.03$ is fixed in both cases.

$x_{(1)}, ..., x_{(\lceil n\alpha \rceil)}$ are the trimmed observations which are not assigned to any cluster. Nevertheless, it is possible to compute the degree of affiliation $D_{(k)}(x_i; \widehat{\theta}, \widehat{\pi})$ of a trimmed observation x_i to its nearest cluster. Thus, the quality of the trimming decision on this observation can be evaluated by comparing $D_{(k)}(x_i; \widehat{\theta}, \widehat{\pi})$ to $D_{(k)}(x_{(\lceil n\alpha \rceil+1)}; \widehat{\theta}, \widehat{\pi})$, with $x_{(\lceil n\alpha \rceil+1)}$ being the non-trimmed observation with smallest value of $D_{(k)}(\cdot; \widehat{\theta}, \widehat{\pi})$. That is

$$\mathrm{DF}(i) = \log\left(D_{(k)}(x_i; \widehat{\theta}, \widehat{\pi}) / D_{(k)}(x_{(\lceil n\alpha \rceil+1)}; \widehat{\theta}, \widehat{\pi})\right) \text{ for } x_i \text{ trimmed.}$$

Hence, discriminant factors $\mathrm{DF}(i) \leq 0$ are obtained for every observation in the data set, whether trimmed or not.

Observations with large $\mathrm{DF}(i)$ values (i.e., values close to zero) indicate doubtful assignments or trimming decisions. The use of this type of discriminant factors was already suggested in **?** in a clustering problem without trimming. "Silhouette" plots (**?**) can be used for summarizing the obtained ordered discriminant factors. Clusters in the silhouette plot with many large $\mathrm{DF}(i)$ values indicate the existence of not very "well-determined" clusters. The most "doubtful" assignments with $\mathrm{DF}(i)$ larger than a $\log($`threshold`$)$ value are highlighted by the function `DiscrFact`.

Figure 11 shows the result of applying the `DiscrFact` function to a clustering solution found for the `mixt` data set appearing in Figure 8. The following code is used to obtain this figure:

```
R > clus.w <- tclust (mixt, k = 3, alpha = 0.1, restr.fact = 1,
    + equal.weights = TRUE)
R > discr.clus.w <- DiscrFact (clus.w, threshold = 0.1)
R > plot (discr.clus.w)
```

The choice `threshold` $= 0.1$ means that a decision on a particular observation x_i is considered as doubtful, if the quality of the second best possible decision ($D_{(k-1)}(x_i; \widehat{\theta}, \widehat{\pi})$ or

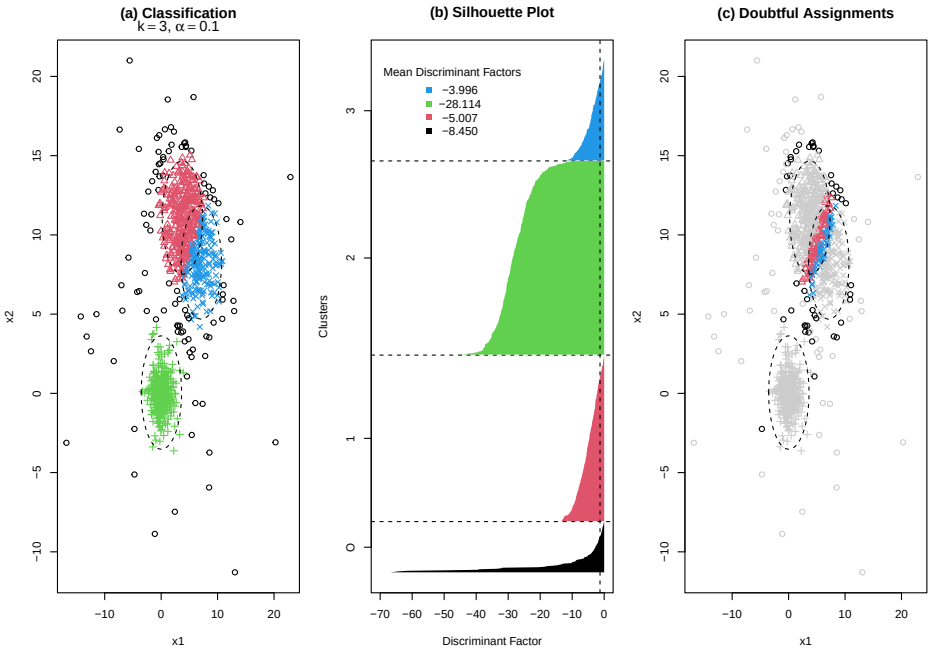

Figure 11: Graphical displays based on the DF(i) values for a `tclust` cluster solution obtained with $k = 3$, $\alpha = 0.1$, `restr.fact` $= 1$ and `equal.weights = TRUE` for the `mixt` data set.

$D_{(k)}(x_{(\lceil n\alpha\rceil+1)}; \widehat{\theta}, \widehat{\pi})$ for trimmed observations) is larger than one tenth of the quality of the actually made decision ($D_{(k)}(x_i; \widehat{\theta}, \widehat{\pi})$).

Although Figure 9 suggests to choose $k = 2$, k has been increased to 3 in order to show how such a change leads to doubtful cluster assignment decisions which can be visualized by `DiscrFact`. Figure 11,(a) simply illustrates the cluster assignments and trimming decisions. The mentioned silhouette plot is presented in (b), whereas the doubtful decisions are marked in (c). All observations with DF(i) $\geq \log(0.1)$ are highlighted as they are plotted darker/in color. Most of the doubtful decisions are located in the overlapping area of the two artificially found clusters (highlighted symbols "×" and "+"). Some doubtfully trimmed observations (highlighted symbol "○") are located in the boundaries of these two clusters.

9. Swiss Bank notes data

The well-known "Swiss Bank notes" data set includes 6 numerical measurements (six-dimensional data set) made on 100 genuine and 100 counterfeit old Swiss 1000-franc bank notes (**?**). The following code can be used to obtain the classification trimmed likelihoods shown in Figure 12.

```
R > data ("swissbank")
R > plot (ctlcurves (swissbank, k = 1:4, alpha = seq (0, 0.3, by = 0.025)))
```

This figure indicates the clear existence of $k = 2$ main clusters ("genuine" and "forged" bills). Moreover, considering the clear difference between $\mathcal{L}^{\Pi}_{50}(0, 3)$ and $\mathcal{L}^{\Pi}_{50}(0, 2)$, we can see that a

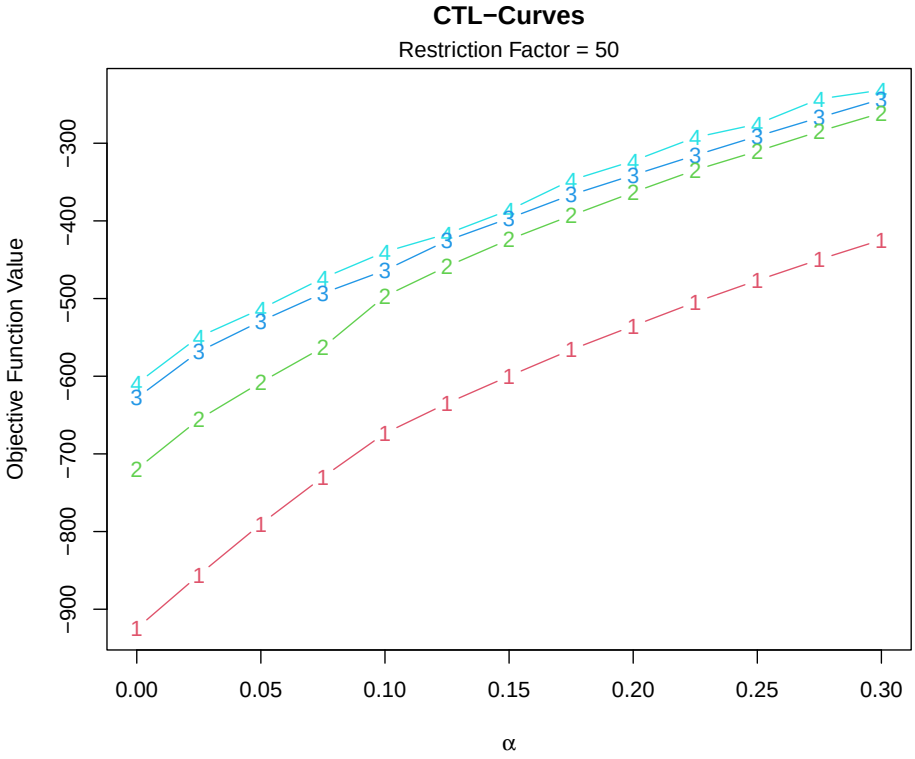

Figure 12: Classification trimmed likelihoods for $k = 1, ..., 4$ and $\alpha = 0, .025, ..., .3$ when `restr.fact` $= 50$ for the "Swiss Bank notes" data set.

further cluster, i.e., $k = 3$, is needed when no trimming is allowed. This extra cluster can be justified by the heterogeneity of the group of forgeries (perhaps due to the presence of different sources of forged bills).

Considering Figure 12, the choice $k = 2$ and a value of α close to 0.1 also seem sensible. Notice that $\mathcal{L}^{\Pi}_{50}(\alpha, 3)$ is clearly larger than $\mathcal{L}^{\Pi}_{50}(\alpha, 2)$ for $\alpha < 0.1$ while these differences are not so big when $\alpha \geq 0.1$. We can even see smaller differences in the classification trimmed likelihood curves when increasing k from 3 to 4. However, these differences are less significant than those previously commented. More spurious clusters can be surely found but they have less entity and importance.

Figure 13 shows the clustering results with $k = 2$, $\alpha = 0.1$ and `restr.fact = 50` obtained by executing the code:

```
R > clus <- tclust (swissbank, k = 2, alpha = 0.1, restr.fact = 50)
R > plot (DiscrFact (clus, threshold = 0.0001))
```

Notice that, in this example, we did not want to impose a specific constraint on the solution. Thus, the default parameter `restr.fact = 50` has initially been used in `ctlcurves`. After choosing the combination $\alpha = 0.10$ and $k = 2$, we could try to reduce the restriction factor which resulted in a warning:

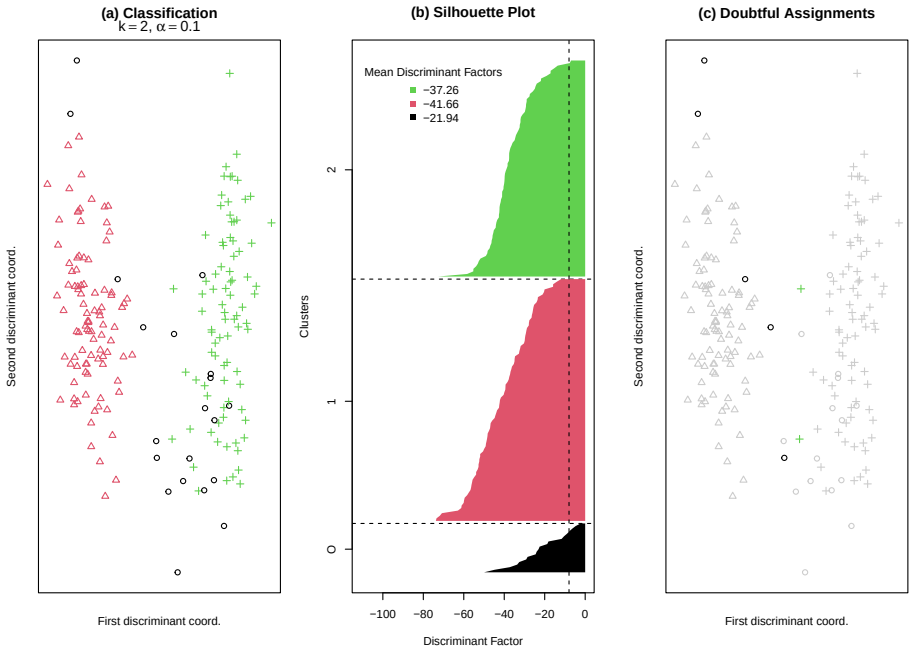

Figure 13: Clustering results with $k = 2$, $\alpha = 0.1$ and `restr.fact` $= 50$ summarized by the use of `DiscrFact` function for the "Swiss Bank notes" data set. The threshold value is chosen in order to highlight the 7 most doubtful cluster assignments.

```
R > tclust(swissbank, k = 2, alpha = 0.1, restr.fact = 40)
In .tclust.warn(warnings, ret):
  The result is artificially constrained due to restr.fact = 40.
```

Thus the choice `restr.fact = 50` seems appropriate as it does not artificially restrict the result, whereas a slightly smaller restriction factor (40) does. By examining the sizes of the obtained groups, we see that no spurious groups are found with `restr.fact = 50`:

```
R > clus$size
  [1] 95 85
```

We have used `restr = "eigen"` in this example, but `restr = "deter"` can be also successfully applied with smaller values of `restr.fact`.

We also use the function `DiscrFact` to summarize the obtained clustering results. The two first Fisher's canonical coordinates derived from the final cluster assignments are plotted. The threshold value 0.0001 is chosen in order to highlight the 7 most doubtful decisions.

Finally, Figure 14 shows a scatterplot of the fourth ("Distance of the inner frame to lower border") against the sixth variable ("Length of the diagonal") with the corresponding cluster assignments. We use the symbols "`G`" for the genuine bills and "`F`" for the forged ones. The 7 most doubtful decisions (i.e., the observations with largest $\mathrm{DF}(i)$ values that were highlighted in Figure 13,(c)) are surrounded by circles in this figure. We can see that "Cluster 1" essentially includes most of the "forged" bills while "Cluster 2" includes most of the "genuine" ones. Among the trimmed observations, we can find a subset of 15 forged bills following a clearly

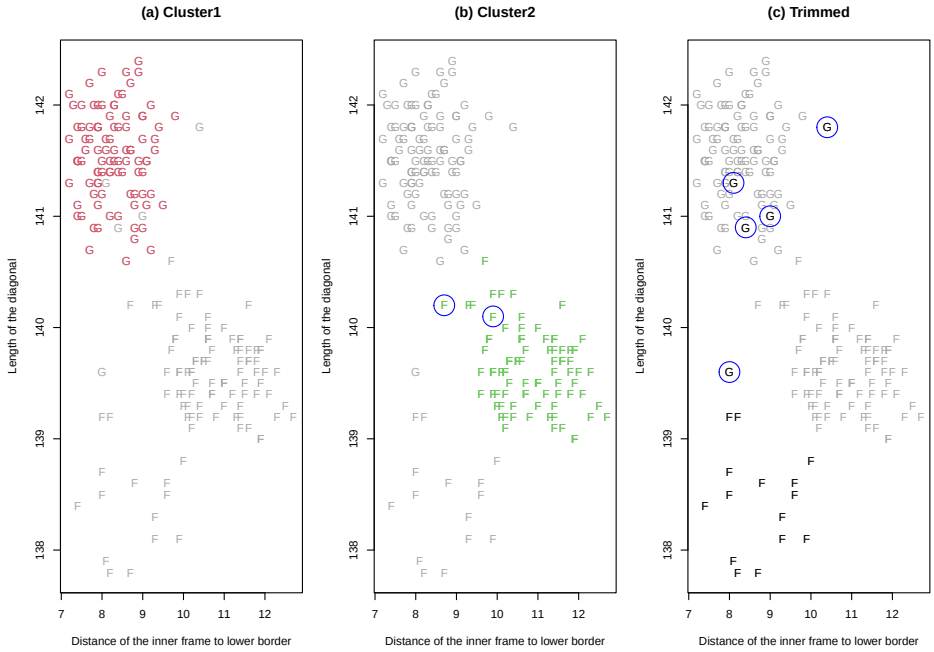

Figure 14: Clustering results with $k = 2$, $\alpha = .1$ and `restr.fact` $= 50$ for the "Swiss Bank notes" data set. Only the fourth and sixth variables are plotted. The 7 most doubtful decisions are rounded by a circle symbol.

different forgery pattern that has been previously commented by other authors (see, e.g., **??**). These most doubtful assignments include 5 "genuine" bills that have perhaps been wrongly trimmed.

10. Conclusion

This paper presents a package called **tclust** for robust (non-hierarchical) clustering. The implementation is flexible, so only the restrictions on the cluster scatters have to be changed in order to carry out different robust clustering algorithms. Robustness is achieved by trimming a specific proportion of observations which are identified as the "most outlying" ones.

Although this R-package implements robust clustering approaches which have already been described in the literature, some of these approaches have been extended to provide increased flexibility. The package also provides some graphical tools which on the one hand help to chose appropriate parameters (`ctlcurves`) and on the other hand help to estimate the adequacy of a particular clustering solution (`DiscrFact`).

Future work on this package focuses on implementing additional types of scatter restrictions, making the algorithm even more flexible, and on providing numerical tools for automatically choosing the number of clusters and the trimming proportion.

Acknowledgements:

This research is partially supported by the Spanish Ministerio de Ciencia e Innovación, grant MTM2011-28657-C02-01. We would like to thank the reviewers and the editor whose comments and suggestions greatly helped in improving the content and the presentation of this paper.

Affiliation:

Luis A. García-Escudero
Departamento de Estadística e Investigación Operativa. Universidad de Valladolid
Prado de la Magdalena s/n, 47005 Valladolid, SPAIN
E-mail: lagarcia@eio.uva.es