Trimming Approach

to

Cluster Analysis

Pan Charoensuk V00827791

Songwan Joun V00729173

Department of Mathematics and Statistics

University of Victoria

Abstract

Clustering is a method of finding groups within data points that are similar to each other. One of the most famous clustering algorithm is K-means. However, in terms of robustness, K-means is heavily influenced by extreme observations. To solve this problem, we introduce robust clustering algorithm, TCLUST which has a restriction on cluster scatter matrices. We will describe the motivation of TCLUST algorithm and mathematical derivation of it on eigenvalue ratio restrictions. Then, we will show how TCLUST is implemented in R and the simulation study results.

Keywords: Robust Clustering, Trimming, TCLUST, eigenvalue ratio restriction

Mathematics Subject Classification 62H30

1. Introduction

K-means is an unsupervised learning algorithm widely used in cluster analysis. However, K-means relies heavily on four major assumptions. 1) Number of clusters, *k*, 2) SSE is the *right objective* to minimize, 3) every cluster has the *same* shape, and 4) every observation is *equally important* for each cluster. We will discuss the sensible choice of *k* in section 4.4. Minimizing SSE may not be the right objective, as clusters often have different variances. Assuming that SSE is minimized for any cluster leads to clusters having the same shape (e.g. scatter matrices are equal). This assumption does not hold when spurious observations or groups are introduced, which is the essence of the robustness of an estimator. We will illustrate the results of K-means with a simulation performed in this report. Similar to the classical means, trimming the data is one of the remedies we should always be looking to perform. Background noise and outlying groups could unnecessarily give too much information, leading to not being able to analyze what we are actually interested in. However, trimming approach on K-means also holds the same assumptions as

classical K-means. Even after discarding the spurious observations, limitation of having equally spherical shapes and weights on clusters would still prevent us from obtaining the optimal clustering results. TCLUST allows us to flexibly modify these assumptions, as we are able to put various types of constraints on the scatter matrices and control how strong the constraint is.

2. TCLUST on eigenvalue ratio constraints

In this report, we will illustrate mathematical framework for robust clustering by eigenvalue ratio constraints. The concepts and descriptions used in section 2 and section 3 are mainly from the work done by Fritz et al. (2013). Suppose we were given the sample observations $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ \dots, \ \boldsymbol{x}_n\}$ in $\boldsymbol{R}^p$, and the probability distribution function $\phi(\cdot; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ of p-variate normal distribution with mean μ and covariance matrix $\boldsymbol{\Sigma}$. Assume the number of clusters to be k and the trimming level to be α. Trimming level decides how much proportion of the data we want to treat as an outlier. For example, we should trim [nα] observations for trimming level α.

Let $R_0, R_1, \dots, R_k$ be a partition of the indices of n observations. For example, in the partition R_1, the observation $\{\boldsymbol{x}_2, \boldsymbol{x}_3\}$ can be included. Especially, R_0 is the partition for outlying observations. The TCLUST algorithm optimizes the set of parameters $R_0, R_1, \dots, R_k$, the centers $\boldsymbol{m}_1, \dots, \boldsymbol{m}_k$, p by p symmetric and positive semidefinite scatter matrices $\boldsymbol{S}_1, \ \dots, \boldsymbol{S}_k$ and weights $p_1, \dots, p_k$ such that it maximizes the objective function $\sum_{j=1}^{k} \sum_{i \in R_j} \log(p_j \phi(\boldsymbol{x}_i; \boldsymbol{\mu}_j, \boldsymbol{S}_j))$. Here, the weights should satisfy the condition $p_i \in [0,1]$ and $\sum_{j=1}^{k} p_j = 1$. This objective function can be interpreted as the sum of weighted log normal probability density function for each observation, giving the same weights if cluster is same.

Before we take a look at the eigenvalue constraints, let's think about the case where there are no restrictions on our objective function. Then, this gives a problem to clustering algorithm

because maximization of objective function $\sum_{j=1}^{k}\sum_{i\in R_j}\log(p_j\phi(\mathbf{x}_i;\boldsymbol{\mu}_j,\mathbf{S}_j))$ without any constraint on scatter matrices($\boldsymbol{S}_j$) is not a well-defined problem. For example, if $\boldsymbol{\mu}_j = \boldsymbol{x}_i$ and $\det(\boldsymbol{S}_j) \rightarrow 0$, then $\phi\left(\boldsymbol{x}_i;\boldsymbol{\mu}_j,\boldsymbol{S}_j\right) = \frac{exp\left\{-\frac{1}{2}\left(\mathbf{x}_i-\boldsymbol{\mu}_j\right)'\boldsymbol{S}_j^{-1}\left(\mathbf{x}_i-\boldsymbol{\mu}_j\right)\right\}}{\sqrt{2\pi}^{p}\det(\boldsymbol{S}_j)^{\frac{1}{2}}}$ goes infinite giving $\phi\left(\boldsymbol{x}_i;\boldsymbol{\mu}_j,\boldsymbol{S}_j\right)$ undefined. Therefore, imposing restriction is essential to make clustering problem solvable.

To make our maximization of objective function well defined problem, we can introduce eigenvalue ratio restrictions on the cluster scatter matrices $\boldsymbol{S}_1, \ldots, \boldsymbol{S}_k$. Let's denote $\lambda_l(\boldsymbol{S}_j)$ be a set of eigenvalues across all clusters scatter matrices $j = 1, \ldots, k$ and data dimension $l = 1, \ldots, p$. Then we can define the maximum of $\lambda_l(\boldsymbol{S}_j)$ as $\max_{j,l}\lambda_l(\boldsymbol{S}_j)$ and the minimum of $\lambda_l(\boldsymbol{S}_j)$ as $\min_{j,l}\lambda_l(\boldsymbol{S}_j)$. Eigenvalue ratio restriction is then defined as the ratio of maximum and minimum eigenvalue such that $\frac{\max_{j,l}\lambda_l(\boldsymbol{S}_j)}{\min_{j,l}\lambda_l(\boldsymbol{S}_j)} \leq c$ for some constant $c \geq 1$. Here, c decides the strength of the constraint. If c is big, there is a weak constraint, and if c is small, there is a strong constraint. For example, If we set c=1, this implies $\max_{j,l}\lambda_l(\boldsymbol{S}_j) = \min_{j,l}\lambda_l(\boldsymbol{S}_j)$. Therefore, all clusters should have same axis length.

3. TCLUST algorithm

TCLUST algorithm gives robust clustering methods by trimming and cluster matrices constraints. There are various types of constraints that can be applied on cluster scatter matrices such as cluster covariance determinant, eigenvalues or both. Here, we only described the TCLUST algorithm having eigenvalue ratio restriction. In other words, the algorithm approximately maximizes the objective function $\sum_{j=1}^{k}\sum_{i\in R_j}\log(p_j\phi(\mathbf{x}_i;\boldsymbol{\mu}_j,\mathbf{S}_j))$ under eigenvalue ratio

constraint $\frac{\max_{j,l} \lambda_l(\boldsymbol{S}_j)}{\min_{j,l} \lambda_l(\boldsymbol{S}_j)} \leq c$. The whole algorithm can be seen as a classification EM algorithm with concentration step. For each iteration we assign data points to the cluster (expectation step), trim the outlying observations (concentration step), and optimize parameters p, $\boldsymbol{\mu}$, $\mathbf{S}$ (maximization) according to the assignment result given on the concentration step. Our detailed examples for understanding TCLUST algorithm can be found on the supplementary documents section 1.

3.1 Initialization

As EM algorithm repeatedly optimizes the objective function and update parameters, we need to set the initial parameter values. We first randomly choose $(p+1)$ points for each cluster, therefore having select $k \times (p+1)$ observations. We choose $(p+1)$ points in each cluster to avoid singularity of covariance matrices. For each cluster, by using those selected data points, we can calculate cluster centers $\boldsymbol{m}_j^{\mathbf{0}}$ and cluster scatter matrices $\boldsymbol{S}_j^{\mathbf{0}}$ for $j = 1, \ldots, k$. Here, when we calculate cluster scatter matrices, we should apply the eigenvalue ratio constraints. For the j$^{\text{th}}$ cluster weights p_j^0, at first, it is chosen at random. This initialization procedure is repeated several times to find the best clustering results through TCLUST algorithm.

From the expectation step (section 3.2) to maximization step (section 3.4), those steps are executed until convergence. (i.e., $\theta^{l+1} = \theta^l$) or a maximum number of iterations is reached.

3.2 Expectation step

In the expectation step, the goal is to calculate posterior probabilities for each observation $\mathbf{x}_\mathrm{i}$ and assign observations to a cluster which provides maximum posterior probability. Let the distance metric from an observation $\mathbf{x}_\mathrm{i}$ to the center of cluster j ($\boldsymbol{\mu}_j^{\boldsymbol{l}}$) at l^{th} iteration be $D_j(\boldsymbol{x}_i; \theta^l) = p_j^l \phi\left(\mathbf{x_i}; \boldsymbol{\mu}_j^{\boldsymbol{l}}, \boldsymbol{S}_j^{\boldsymbol{l}}\right)$. If $D_j(\boldsymbol{x}_i; \theta^l)$ small, the distance of the $\mathbf{x}_\mathrm{i}$ to $\boldsymbol{\mu}_j^{\boldsymbol{l}}$ is large. The

posterior probabilities are defined as $\frac{D_j(\boldsymbol{x}_i;\theta^l)}{\sum_{j=1}^{k} D_j(\boldsymbol{x}_i;\theta^l)}$ for $j = 1, \ldots, k$ where θ^l is the set of cluster parameters in the iteration l.

3.3 Concentration step

After we calculate posterior probabilities for each observation, we can trim [nα] observations. Let $\mathrm{D}(\mathbf{x}_\mathrm{i};\theta^l)$ be the maximum of $D_j(\boldsymbol{x}_i;\theta^l)$ across all clusters for observation $\boldsymbol{x}_i$. Then, $\mathrm{D}(\mathbf{x}_\mathrm{i};\theta^l)$ is defined as $\mathrm{D}(\mathbf{x}_\mathrm{i};\theta^l) = \max\{D_1(\boldsymbol{x}_i;\theta^l), \ldots, D_k(\boldsymbol{x}_i;\theta^l)\}$. We can calculate $\mathrm{D}(\mathbf{x}_\mathrm{i};\theta^l)$ for all of the observations and sort it from smallest to largest. Then we trim the smallest [nα] observations as possible outliers. For the non-outlying observations, we can assign them to the clusters which provides the largest posterior probabilities (i.e. assign $\mathbf{x}_\mathrm{i}$ to cluster j if $\mathrm{D}_\mathrm{j}(\mathbf{x}_\mathrm{i};\theta^l)$=$\mathrm{D}(\mathbf{x}_\mathrm{i};\theta^l)$). As a result, we will now have the partition $R_0, R_1, \ldots, R_k$ for every observation. Here, R_0 is the partition for outlying observations with smallest [nα] $\mathrm{D}(\mathbf{x}_\mathrm{i};\theta^l)$'s.

3.4 Maximization step

In the maximizations step, the parameters $\theta^l = \left(p_1^l, \ldots, p_k^l, \boldsymbol{\mu}_1^l, \ldots, \boldsymbol{\mu}_k^l, \boldsymbol{S}_1^l, \ldots, \boldsymbol{S}_k^l\right)$ are updated, based on non-discarded observations and their cluster assignments. Let $n_j = \#R_j$ be the number of observations in cluster j. Then the weights are updated by $p_j^{l+1} = n_j/[n(1-\alpha)]$. In other words, we update weights proportional to the number of observations in cluster j. Also, the centers are updated by the sample means $\boldsymbol{m}_j^{l+1} = \frac{1}{n_j} \sum_{i \in R_j} \mathbf{x}_\mathrm{i}$. However, scatter matrices are not updated by sample covariance matrices because it may not satisfy eigenvalue ratio constraint. Instead, we use scatter matrices that are updated by truncated eigenvalues and spectral decomposition. First, we apply spectral decomposition of sample covariance matrix $\boldsymbol{T}_j = \mathbf{U}_\mathrm{j}'\mathbf{D}_\mathrm{j}\mathbf{U}_\mathrm{j}$ where $\mathbf{U}_\mathrm{j}$ being an orthogonal matrix and $\mathbf{D}_\mathrm{j} = diag(d_{j1}, d_{j2}, \ldots, d_{jp})$ a diagonal matrix consist

of eigenvalues. If we consider truncated eigenvalues such that

$$d_{\mathrm{jl}}^{m} = \begin{cases} d_{jl} & if\ d_{jl} \in [m,\ cm] \\ m & if\ d_{jl} < m \\ cm & if d_{jl} > cm \end{cases}$$

with m as some threshold value, the scatter matrices are updated as $\boldsymbol{S}_j^{l+1} = \mathbf{U}_{\mathrm{j}}'\mathbf{D}_{\mathrm{j}}^{*}\mathbf{U}_{\mathrm{j}}$ with $\mathbf{D}_{\mathrm{j}}^{*} = diag(d_{j1}^{m_{opt}}, d_{j2}^{m_{opt}}, \dots, d_{jp}^{m_{opt}})$ and m_{opt} minimizing $m \mapsto \sum_{j=1}^{k} n_j \sum_{l=1}^{p} \left(\log\left(d_{jl}^{m}\right) + \frac{d_{jl}}{d_{jl}^{m}}\right)$. These truncated eigenvalues with optimized m enable our updated scatter matrices meet the eigenvalue ratio restriction.

3.5 Evaluation

When the algorithm converges, we can calculate $\sum_{j=1}^{k} \sum_{i \in R_j} \log(p_j \phi(\mathbf{x}_{\mathrm{i}}; \boldsymbol{\mu}_{\mathrm{j}}, \mathbf{S}_{\mathrm{j}}))$ with parameters updated from the iterative expectation and maximization step.

4. Implementation in R

TCLUST allows us to modify the scatter matrix of the cluster by putting a constraint on relative sizes of the axes of clusters and relative volumes of clusters. The usage of TCLUST function in R is "*tclust(x, k, alpha, restr = c("eigen", "deter", "sigma"), restr.fact, equal.weights)*" where x is the observations, k is the number of clusters, *restr* is the type of the constraint, *restr.fact* is the strength of the constraint and *equal.weights* is an option to determine whether we allow clusters to have different weights.

4.1 Strength of constraints

The option *restr.fact* is a fixed value greater than 0 which determines the strength of the constraint in TCLUST function. The larger *restr.fact*, the looser is the restriction on the scatter matrices, allowing more heterogeneity among clusters. On the contrary, the closer *restr.fact* to 1,

the more equally scattered are the clusters.

4.2 Types of constraint

Constraining the eigenvalues allows us to simultaneously control the relative group sizes and also the deviation from sphericity in each cluster. This constraint is achieved when we set *restr = "eigen"*. Constraining determinants limits the relative volumes of the clusters. This type of constraint is done by setting *restr = "deter"*. Setting *restr = "sigma"* forces all cluster scatter matrices to be the same, and *restr.fact* is ignored when applying this type of constraint. The examples to these types of constraint are provided in the supplementary documents section 2.

4.3 Warning

The obtained values for k and α and their associated clustering solutions must be explored carefully. Algorithm gives a warning if the ratio exceeds the upper bound. In which case, the upper bound may be increased stepwise until the warning disappears. TCLUST outputs point out which solutions are artificial. This allows us to easily search for clustering solutions which are not artificially restricted, if desired. If the type of constraint is not appropriate for the dataset, the algorithm would still run while warning us that there is an error in the formal argument. In particular, forcing the scatter matrices to be exactly the same (*restr = "eigen"*) for clusters that differ in shape would result in the algorithm returning the appropriate shape of the clusters, ignoring the restriction entirely. The example of this violation is demonstrated in the supplementary documents section 2.

4.4 Choosing k

One of the most difficult problem in clustering is choosing the number of clusters, k where k is, most of the time, unknown. The choice of trimming proportion, α is also dependent on the choice of k, where α has to be chosen without knowing the true contamination level. Increasing α

tends to trim entire clusters, decreasing k. Too low of an α might allow groups of outliers to form clusters, increasing k. TCLUST package allows us to choose sensible k and α simultaneously by carefully monitoring the maximum value attained by log-likelihood as we change k and α sequentially. The CTL-Curve (classification trimmed likelihood curve) allows us to determine whether k needs to be increased as α increases (elbow method).

5. Simulation studies

We conduct a simulation on multivariate normally distributed observations, with 3 main clusters and background noise. The CTL-curves suggest that we use $k = 4$ for K-means, conflicting with the true number of clusters. We then start to trim out 5% of the observations. Lastly, we apply TCLUST algorithm with the strength, *restr.fact* = 5 and 50. We will compare each algorithm's performance by looking at the misclassification rate for each method. Figure 1 illustrates the CTL-curves plot. R code is provided in the supplementary documents section 2.

Below are figure 2 to illustrate the simulation on multivariate-normal distributional data with different clustering algorithms, and table 1 to compare their performance. We see an improvement using TCLUST algorithm over K-means and Trimmed K-means in figure 2 as the background noise is discarded after being trimmed and allowing different shapes and weights on each cluster. Table 1 reinforces the results as we could see the misclassification numerically.

Table 1. The misclassification rate for each algorithm

Misclassification					
	K-means k = 4	K-means k = 3	Trimmed K-means k = 3	TCLUST restr.fact = 5	TCLUST restr.fact = 50
Misclassification	0.9963	0.1665	0.1423	0.1209	0.1200

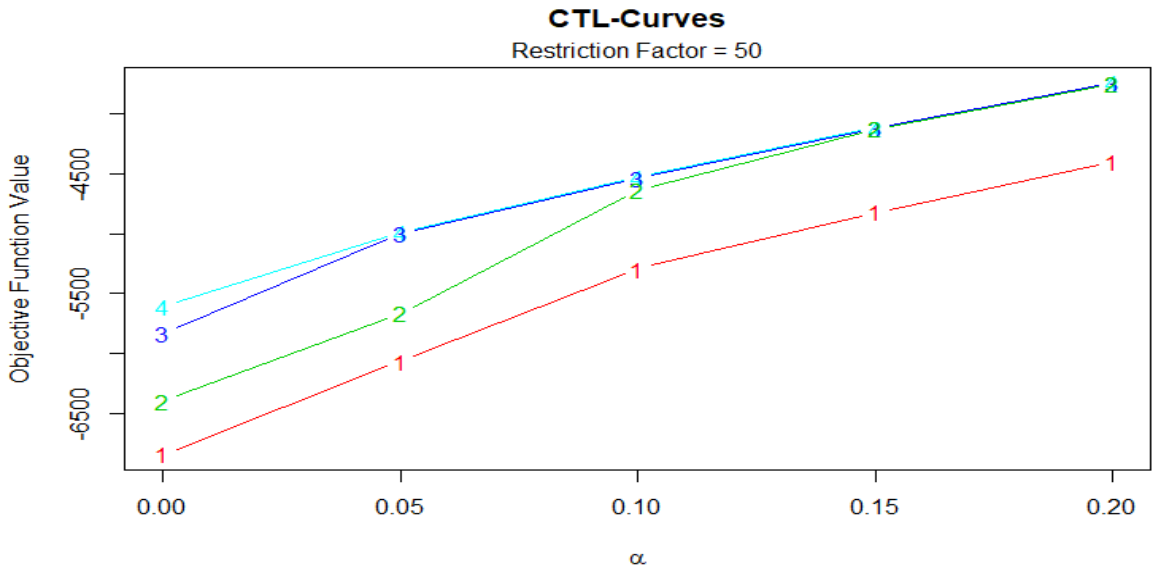


Figure 1. CTL-curves plot to determine sensible choices of k and α

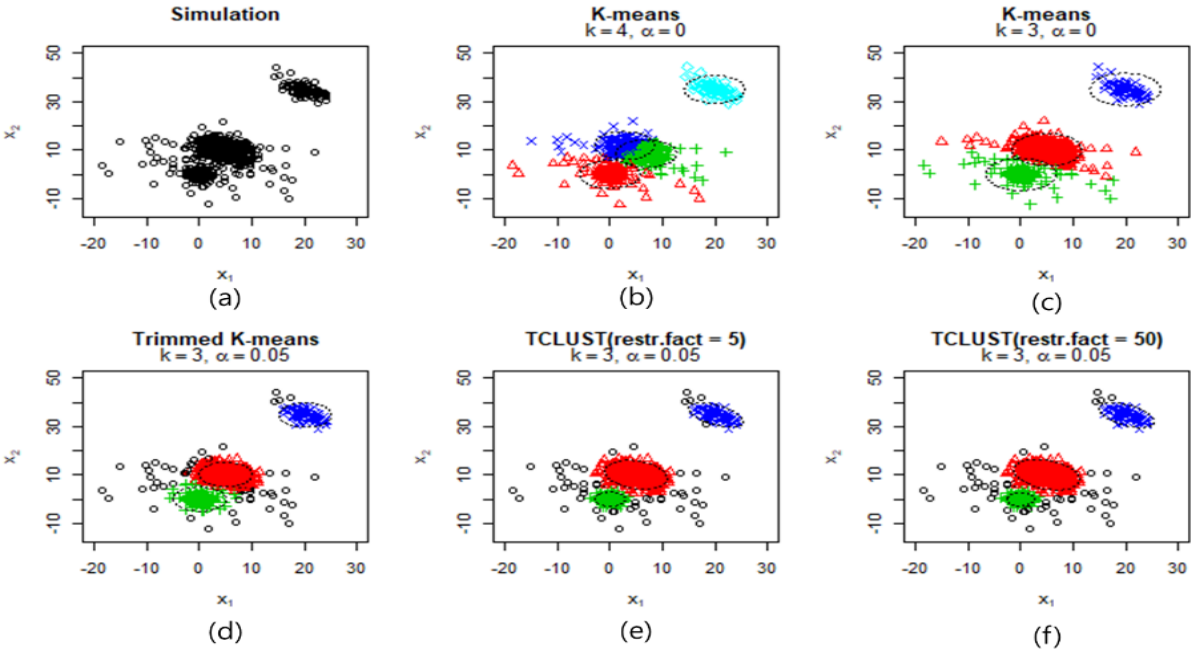


Figure 2. The results of each algorithm. Multivariate-normal data is generated (a). K-means with k = 4 give significantly misclassified clusters(b). K-means with k = 3 gives much better results, but includes the background noise(c). Trimming 5% of the data removes most of the noise(c). Lastly, applying TCLUST to allow different weights and relative sizes with stricter(e) and looser constraint(f).

6. Conclusion

Giving constraints on scatter matrices improves robustness on clustering and avoid unwell defined problem. Through the simulation study, we could conclude that TCLUST algorithm outperforms non robust k-means algorithm when there are some outliers or overlapping points in the bridge region. However, the problem where the algorithm would find the local maxima in the objective function still persists, similar to classical K-means. In which case, we would need to perform the TCLUST algorithm several times to ensure reasonable results. If time permits, we should also look into modifying the algorithm and propose a remedy to the problem such as better starting point in the initial steps, or using the robust covariance matrices for the clusters instead of the truncated covariance matrices.

References

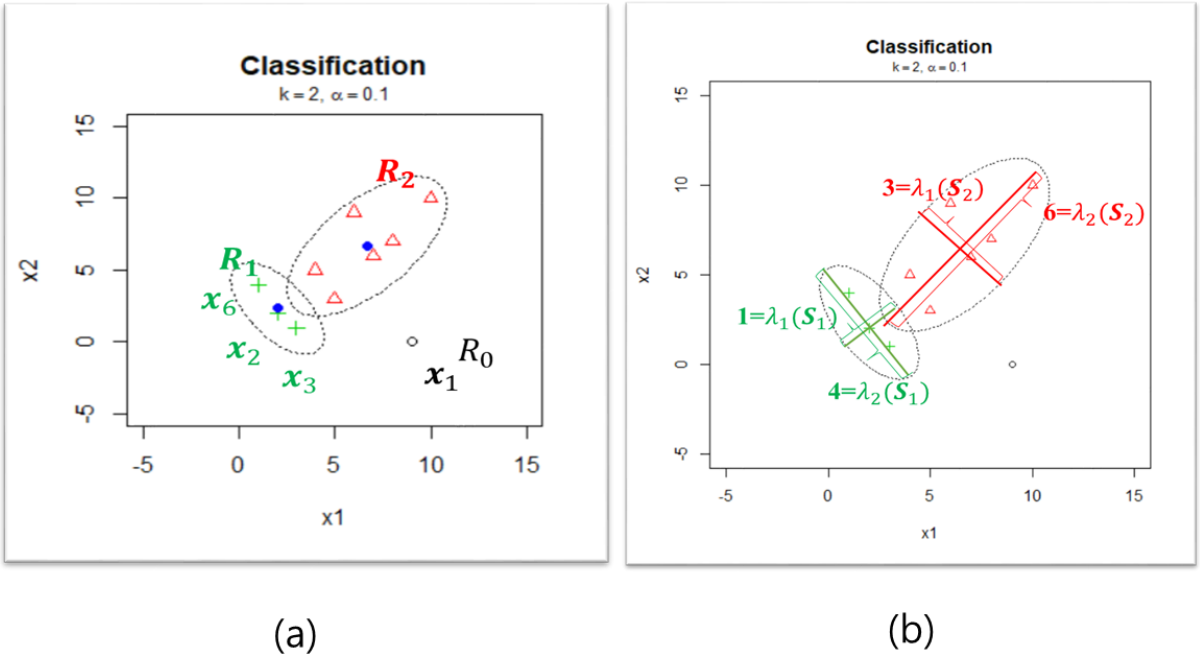
Fritz, H., Garcıa-Escudero, L. A., & Mayo-Iscar, A. (2012). tclust: An R package for a trimming approach to cluster analysis. *Journal of Statistical Software*, *47*(12), 1-26.

Fritz, H., GarcíA-Escudero, L. A., & Mayo-Iscar, A. (2013). A fast algorithm for robust constrained clustering. *Computational Statistics & Data Analysis*, *61*, 124-136.

García-Escudero, L. A., Gordaliza, A., Matrán, C., & Mayo-Iscar, A. (2008). A general trimming approach to robust cluster analysis. *The Annals of Statistics*, *36*(3), 1324-1345.

Supplementary documents

1. TCLUST Examples



(a) (b)

Supplementary figure 1 (a) The TCLUST result on eigenvalue restriction with 10 samples observations. Here, the number of clusters were set to 2 (k=2) and trimming level was set to 0.1. (b) Figure that shows the eigenvalues of 2 dimensional data for each cluster. Here, the length of axis represents eigenvalues.

1.1 Example 1

Suppose p=2 dimensional data $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ \ldots, \ \boldsymbol{x}_{10}\} = \begin{pmatrix} 9 & 2 & 3 & 4 & 6 & 1 & 8 & 8 & 5 & 10 \\ 0 & 2 & 1 & 5 & 9 & 4 & 6 & 7 & 3 & 10 \end{pmatrix}$ in $\boldsymbol{R}^2$ and a probability distribution of 2-variate normal density $\phi(\cdot; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. If we set number of clusters k=2 and trimming level : α = 0.1, then one observation should be trimmed because [nα] = [10*0.1] = 1 ($\#R_0 = 1$). The objective function is $\sum_{i \in R_1} \log(p_1 \phi(\boldsymbol{x}_i; \boldsymbol{\mu}_1, \boldsymbol{S}_1)) + \sum_{i \in R_2} \log(p_2 \phi(\boldsymbol{x}_i; \boldsymbol{\mu}_2, \boldsymbol{S}_2))$ and we want to find best R_0(for outliers), R_1, R_2 partition of the observation indices {1, …, 10}, centers $\boldsymbol{m}_1$, $\boldsymbol{m}_2$, 2x2

scatter matrices $\boldsymbol{S}_1$, $\boldsymbol{S}_2$, and the weights p_1, p_2

Plot (a) of supplementary figure 1 shows the result of TCLUST algorithm for the above example. Through TCLUST, we can assign 10 observations to two clusters and outlier region(R_0). The algorithm assigned $\boldsymbol{x}_2, \boldsymbol{x}_3, \boldsymbol{x}_6 \sim \phi_1(\cdot; \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ to green cluster (R_1), $\boldsymbol{x}_4, \boldsymbol{x}_5, \boldsymbol{x}_7, \boldsymbol{x}_8, \boldsymbol{x}_9 \sim \phi_2(\cdot; \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$ to red cluster (R_2) and the $\boldsymbol{x}_1$ as an outlier (R_0). This cluster assignment is the result that maximizes the objective function $\sum_{i \in R_1} \log(p_1 \phi(\boldsymbol{x}_i; \boldsymbol{\mu}_1, \boldsymbol{S}_1)) + \sum_{i \in R_2} \log(p_2 \phi(\boldsymbol{x}_i; \boldsymbol{\mu}_2, \boldsymbol{S}_2))$ with some cluster eigenvalue constraints. Here, the optimized parameter values for mean, weights, and scatter matrices are $\boldsymbol{\mu}_1 = (2.3,\ 2.0)', \boldsymbol{\mu}_2 = (6.6,\ 6.6)'$, $p_1 = 0.3$, $p_2 = 0.7$, $\boldsymbol{S}_1 = \begin{bmatrix} 1.0 & -0.8 \\ -0.8 & 1.7 \end{bmatrix}$ and $\boldsymbol{S}_2 = \begin{bmatrix} 2.8 & 2.1 \\ 2.1 & 3.9 \end{bmatrix}$ each.

1.2 Example 2

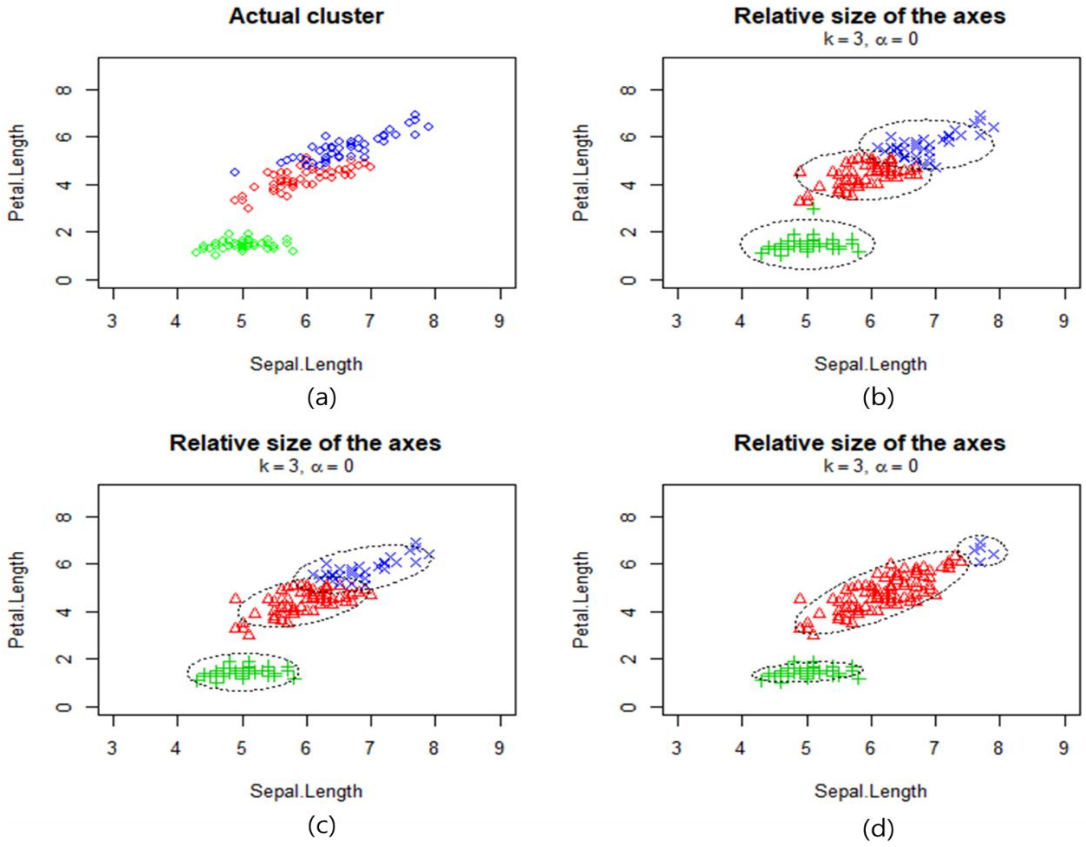
Suppose we have result from TCLUST algorithm such that the ratio of the minimum and maximum eigenvalue should be less or equal to c=2. From the TCLUST output, the cluster $\boldsymbol{S}_1$ has $\lambda_1 = 1,\ \lambda_2 = 4$ and the cluster $\boldsymbol{S}_2$ has $\lambda_1 = 3,\ \lambda_2 = 6$. Plot (b) of supplementary figure 1 shows the plot of two clusters $\boldsymbol{S}_1, \boldsymbol{S}_2$ and corresponding eigenvalues. Here, $\max_{\mathrm{j,l}} \lambda_{\mathrm{l}}(\mathbf{S}_{\mathrm{j}}) = \lambda_2(\mathbf{S}_2) = 6$ and $\min_{\mathrm{j,l}} \lambda_{\mathrm{l}}(\mathbf{S}_{\mathrm{j}}) = \lambda_1(\mathbf{S}_1) = 1$. Therefore, $\frac{\max_{j,l} \lambda_l(\boldsymbol{S}_j)}{\min_{j,l} \lambda_l(\boldsymbol{S}_j)} = \frac{6}{4} < 2 = \mathrm{c}$ and this implies that the algorithm made two output clusters such that the length of the longest axis is smaller than two times the minimum of shortest axis length.

1.3 Example 3

Plot (a) of the supplementary figure 1 is the TCLUST results from k=2, α = 0.1 case with 10 observations. Suppose p_1, p_2 are randomly chosen and $\boldsymbol{\mu}_1$, $\mathbf{S}_1$, $\boldsymbol{\mu}_2$, $\mathbf{S}_2$ are given. If we look at the observation $\boldsymbol{x}_3$, for each cluster, we can calculate $D_1(\mathbf{x}_3;\theta) = p_1\phi(\mathbf{x_1};\boldsymbol{\mu}_1,\mathbf{S}_1)$ and $D_2(\mathbf{x}_3;\theta) = p_2\phi(\mathbf{x_2};\boldsymbol{\mu}_2,\mathbf{S}_2)$. Therefore, posterior probabilities are $P_{1,\,post}(\boldsymbol{x_3}) = \frac{D_1(\mathbf{x}_3;\theta)}{\sum_{j=1}^{2} D_j(\mathbf{x}_3;\theta)} = \frac{p_1\phi(\mathbf{x_3};\boldsymbol{\mu}_1,\mathbf{S}_1)}{p_1\phi(\mathbf{x_3};\boldsymbol{\mu}_1,\mathbf{S}_1)+p_2\phi(\mathbf{x_3};\boldsymbol{\mu}_2,\mathbf{S}_2)}$ and $P_{2,post}(\boldsymbol{x}_3) = \frac{D_2(\mathbf{x}_3;\theta)}{\sum_{j=1}^{2} D_j(\mathbf{x}_3;\theta)} = \frac{p_2\phi(\mathbf{x_3};\boldsymbol{\mu}_2,\mathbf{S}_2)}{p_1\phi(\mathbf{x_3};\boldsymbol{\mu}_1,\mathbf{S}_1)+p_2\phi(\mathbf{x_3};\boldsymbol{\mu}_2,\mathbf{S}_2)}$ each. It is easy to see that $P_{j,\,post}(\boldsymbol{x}_i) \propto D_j(\mathbf{x}_\mathrm{i};\theta)$. Now that we have posterior probabilities for $\mathbf{x}_3$, we can calculate $\mathrm{D}(\mathbf{x}_3;\theta) = \max\{D_1(\mathbf{x}_3;\theta), D_2(\mathbf{x}_3;\theta)\} \propto \max\{P_{1,\,post}(\boldsymbol{x_3}), P_{2,post}(\boldsymbol{x}_3)\}$. Here, $\boldsymbol{x}_3$ will be assigned to the cluster which provides maximum posterior probability. For defining outliers, after calculating $\mathrm{D}(\mathbf{x}_\mathrm{i};\theta)'s$ for every observation $\mathrm{i} = 1, \ldots, \mathrm{n}$, [nα] observations $\mathbf{x}_\mathrm{i}$ with smallest values of $\mathrm{D}(\mathbf{x}_\mathrm{i};\theta)$ will be discarded as possible outliers. Finally, we update parameters based on non-discarded observations and new cluster assignment.

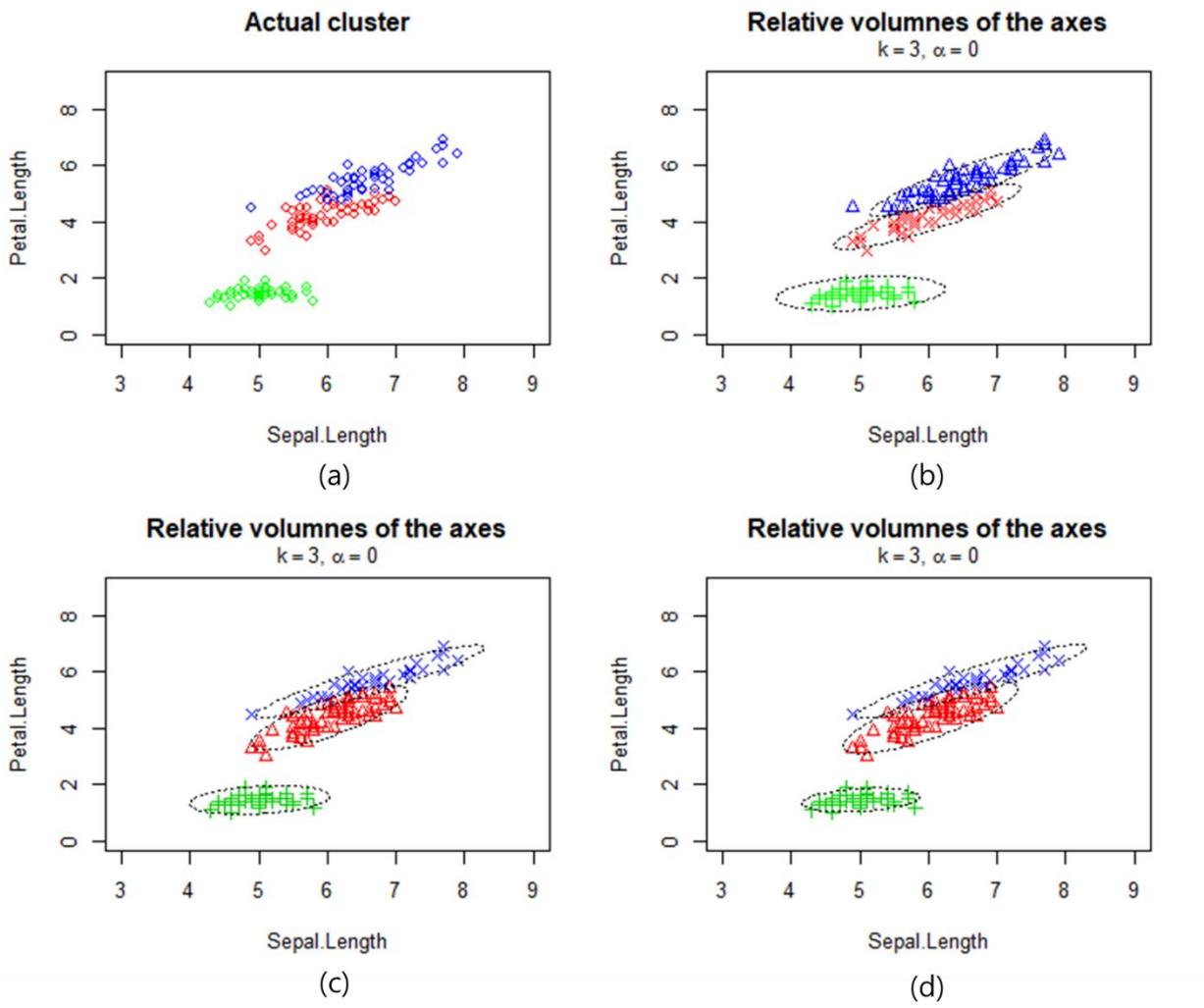
2. Practical examples for TCLUST

We will perform various constraint algorithms on Iris dataset, available in R, to compare each algorithm visually. Only two of the variables, sepal length and petal length of the flowers are considered.



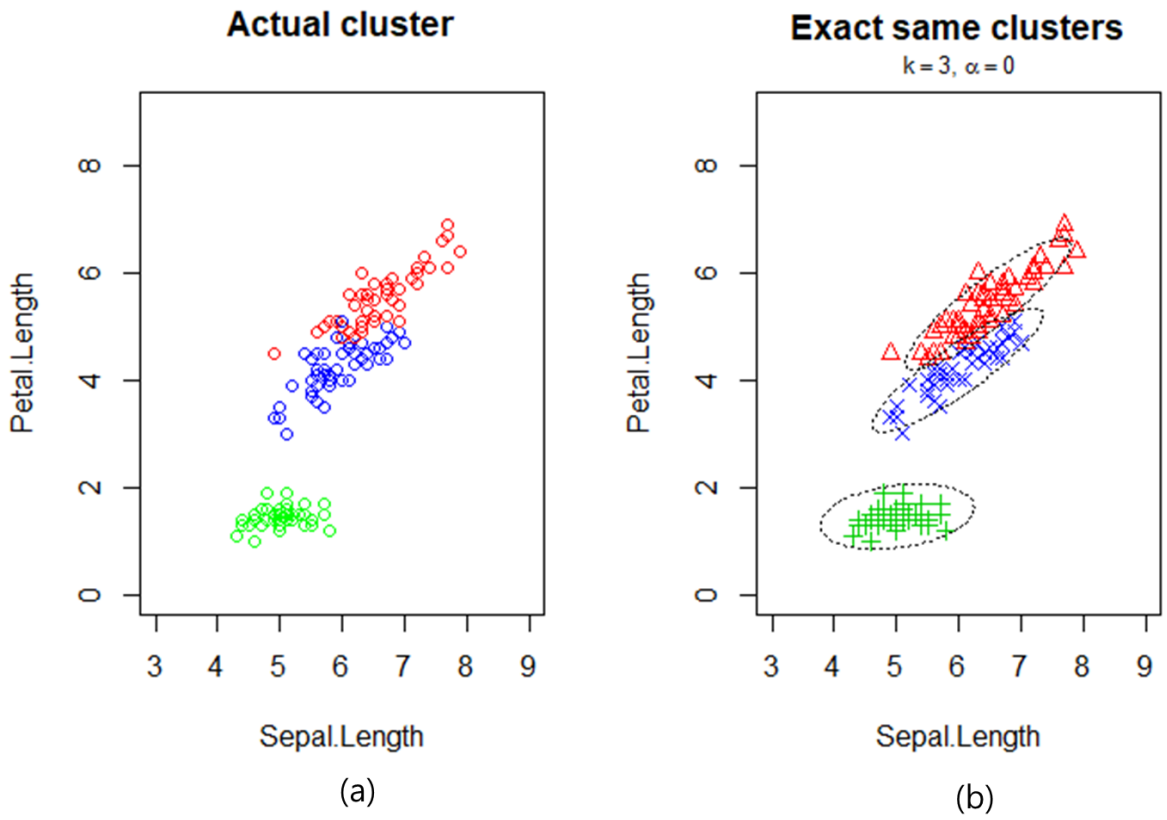
Supplementary figure 2. TCLUST with the restriction on the eigenvalues, *restr = "eigen"*

Supplementary figure 2 above shows the plots of the clusters where we put constraints on the eigenvalues of the scatter matrices. plot(a) is the true clusters for Iris data. We then restrict the ratio of the eigenvalues equal to 1 (b), leading to the clusters having similar shapes. Loosen the restriction allows the clusters to take different shapes. (c) has the ratio of less than or equal to 2.5, and (d) with ratio of less than or equal to 30.



Supplementary figure 3. TCLUST with the restriction on the eigenvalues, *restr = "deter"*

In supplementary figure 3, we now put constraints on the determinants of the scatter matrices. We restrict the ratio of the determinants equal to 1 (b), leading to the clusters having similar shapes. Loosen the restriction allows the clusters to have different volumes. (c) has the ratio of less than or equal to 2.5, and (d) with ratio of less than or equal to 10.



Supplementary figure 4. TCLUST with the restriction on every scatter matrix, *restr = "sigma"*

Lastly, in supplementary figure 4, we force the scatter matrices to be the same, achieving by setting *restr = "sigma"*. The strength of the ratio is ignored here. Visually, we notice that the green cluster in (b) is different than the other two. This is due to the fact that imposing the restriction on the entirety of the clusters is not appropriate in the Iris data. (a) illustrates that the green cluster clearly takes on a more spherical and horizontal shape with respect to *sepal.length* axis . The algorithm would give a warning that the argument is mismatched, then ignore the restriction entirely in order to optimize and fit the appropriate shape of the clusters instead.

3. R codes

3.1 R code for the practical examples (2).

TCLUST on the full Iris dataset is also provided.

```
################################
### Robust Cluster Analysis ###
################################
# Pan Charoensuk
# Songwan Joun
# Stat 454/556 - Robust Statistics
# Project 1 - Robust Clustering
################################


################################
######### EXAMPLES ############
################################


################################
#KMEANS
################################
set.seed(5)
k_iris = kmeans(iris[,c(1,3)], centers =3)

par(mfrow=c(1,1))
#Let's see 1st and 2nd column
plot(iris[,1], iris[,3], col=k_iris$cluster, pch=19,
     xlab="Sepal.Length", ylab="Petal.Length", main="K-means (k=3)")
points(k_iris$centers[,1], k_iris$centers[,2], col="blue", pch=20, cex=5)
idx_kmeans = which(as.numeric(iris$Species)!=k_iris$cluster)
points(iris[idx_kmeans,1], iris[idx_kmeans,3], col=iris[idx_kmeans,5], cex=2, pch=1)
legend("bottomright", legend = c("True", "Kmeans","Cluster center"),
       pch=c(1,19,19), col=c("black","black","blue"))
text(k_iris$centers, c("51","58","41"),col="white")



################################
#Trimming
################################
#install.packages("tclust")
library ("tclust")

## Warning: package 'tclust' was built under R version 3.5.2

set.seed(5)
tk_iris <- tkmeans (iris[,c(1,3)], k = 3, alpha = 0.1)
par(mfrow=c(1,1))
plot (tk_iris, xlab="Sepal.Length", ylab="Petal.Length", main="Trimmed K-means (k=3)")

points(tk_iris$centers[1,], tk_iris$centers[2,], col="black", pch=20, cex=5)
text(tk_iris$centers[1,],tk_iris$centers[2,],
c(sum(tk_iris$cluster==1),sum(tk_iris$cluster==2),sum(tk_iris$cluster==3)),col="white")

idx_tkmeans = which(as.numeric(iris$Species)!=tk_iris$cluster & tk_iris$cluster!=0)
```

```r
points(iris[idx_tkmeans,1], iris[idx_tkmeans,3], col=rep(c("green","blue"), c(5,11)),
     pch=1, cex=2)
legend("bottomright", legend = c("True", "Kmeans"), pch=c(1,19))

par(mfrow=c(2,2))
  for(alpha in c(0.05, 0.1, 0.2, 0.4)){
    tk_iris <- tkmeans (iris[,c(1,3)], k = 3, alpha = alpha)
    plot (tk_iris, xlab="Sepal.Length", ylab="Petal.Length", main="Trimmed K-means (k=3)")
    points(tk_iris$centers[1,], tk_iris$centers[2,], col="black", pch=20, cex=5)
    text(tk_iris$centers[1,],tk_iris$centers[2,],
c(sum(tk_iris$cluster==1),sum(tk_iris$cluster==2),sum(tk_iris$cluster==3)),col="white")

  }

################################
#TCLUST
################################

#constraint on eigenvalues
set.seed(0)
a = tclust(iris[,c(1,3)], alpha=0, rest=c("eigen"), restr.fact=1)

## Warning in .tclust.warn(O, O$ret): The result is artificially constrained
## due to restr.fact = 1.

b = tclust(iris[,c(1,3)], alpha=0, rest=c("eigen"), restr.fact=2.5)

## Warning in .tclust.warn(O, O$ret): The result is artificially constrained
## due to restr.fact = 2.5.

c = tclust(iris[,c(1,3)], alpha=0, rest=c("eigen"), restr.fact=30)

## Warning in .tclust.warn(O, O$ret): The result is artificially constrained
## due to restr.fact = 30.

par(mfrow=c(2,2))
plot(iris[,c(1,3)], col=c(rep("green", sum(iris$Species=="setosa")),
                          rep("red", sum(iris$Species=="versicolor")),
                          rep("blue", sum(iris$Species=="virginica"))),
     main = "Actual cluster",xlim=c(3,9), ylim=c(0,9))
plot(a, main="Relative size of the axes",xlim=c(3,9), ylim=c(0,9))
plot(b, main="Relative size of the axes",xlim=c(3,9), ylim=c(0,9))
plot(c, main="Relative size of the axes",xlim=c(3,9), ylim=c(0,9))
```

```r
#constraint on determinants
set.seed(0)
a = tclust(iris[,c(1,3)], alpha=0, rest=c("deter"), restr.fact=1)

## Warning in .tclust.warn(O, O$ret): The result is artificially constrained
## due to restr.fact = 1.

b = tclust(iris[,c(1,3)], alpha=0, rest=c("deter"), restr.fact=2.5)

## Warning in .tclust.warn(O, O$ret): The result is artificially constrained
## due to restr.fact = 2.5.

c = tclust(iris[,c(1,3)], alpha=0, rest=c("deter"), restr.fact=10)
par(mfrow=c(2,2))
plot(iris[,c(1,3)], col=c(rep("green", sum(iris$Species=="setosa")),
                          rep("red", sum(iris$Species=="versicolor")),
                          rep("blue", sum(iris$Species=="virginica"))),
     main = "Actual cluster",xlim=c(3,9), ylim=c(0,9))
plot(a, main="Relative volumnes of the axes",xlim=c(3,9), ylim=c(0,9), col=c("red","blue","green"))
#default had bad cluster colors
```

```r
plot(b, main="Relative volumnes of the axes",xlim=c(3,9), ylim=c(0,9))
plot(c, main="Relative volumnes of the axes",xlim=c(3,9), ylim=c(0,9))

#forcing clusters to be the same
set.seed(0)
#a = tclust(iris[,c(1,3)], alpha=0, rest="sigma")
par(mfrow=c(1,2))
plot(iris[,c(1,3)], col=c(rep("green", sum(iris$Species=="setosa")),
                          rep("blue", sum(iris$Species=="versicolor")),
                          rep("red", sum(iris$Species=="virginica"))),
     main = "Actual cluster",xlim=c(3,9), ylim=c(0,9))
plot(a, main="Exact same clusters",xlim=c(3,9), ylim=c(0,9))

#TCLUST gives an error..this explains why this example did not show the correct result in our the
presentation...

################################################################################################
############################### MULTIDIMENSIONAL IRIS ##########################################
################################################################################################


set.seed(5)

# plotting CTL-Curves to determine k and alpha
plot(ctlcurves(iris[,1:4], k = 1:5, alpha = seq(0,0.3, by = 0.03)))

## Depending on arguments x, k and alpha, this function needs some time to compute.
## (Remove this message by setting "trace = 0")

#no significant improvement when increase k from 3 to 4,5. Choose k = 3

# defining true clusters
true_iris = data.frame(cbind(iris[,1:4], true_clust = c(rep(2,50), rep(1,50), rep(3,50))))
names(true_iris) = c("x1","x2","x3","x4","true_clust")

#10 simulations
num_sim <- 10
misclass_table <- matrix(NA, nrow = num_sim, ncol = 4)
for(i in 1:num_sim){
  set.seed(i)
  #kmeans 3 clusters
  iris_kmeans <- tclust(iris[,1:4], k = 3, alpha = 0, equal.weights = TRUE, restr.fact = 1)
  #calculate misclassification rate
  k_means_mr=sum(iris_kmeans$cluster!=true_iris$true_clust)/length(true_iris$true_clust)


  #trimmed kmeans
  iris_tkmeans <- tclust(iris[,1:4], k = 3, alpha = 0.03, equal.weights = TRUE, restr.fact = 1)
  #calculate misclassification rate
  tk_means_mr=sum(iris_tkmeans$cluster!=true_iris$true_clust)/length(true_iris$true_clust)

  #tclust
  ## eigen, restr.fact = 5
  iris_tclust.5 <- tclust(iris[,1:4], k = 3, alpha = 0, equal.weights = FALSE, restr.fact = 5)
  #calculate misclassification rate
  tclust.5_mr=sum(iris_tclust.5$cluster!=true_iris$true_clust)/length(true_iris$true_clust)

  ## eigen, restr.fact = 10
  iris_tclust.10 <- tclust(iris[,1:4], k = 3, alpha = 0, equal.weights = FALSE, restr.fact = 10)
  #calculate misclassification rate
  tclust.10_mr=sum(iris_tclust.10$cluster!=true_iris$true_clust)/length(true_iris$true_clust)

  ## misclassification rate table
  misclass_table[i,1] <- k_means_mr
  misclass_table[i,2] <- tk_means_mr
  misclass_table[i,3] <- tclust.5_mr
```

```
  misclass_table[i,4] <- tclust.10_mr

}

colnames(misclass_table) <- c("K-means", "TkMeans", "TCLUST(upp_bound=5)", "TCLUST(upp_bound=10)")
misclass_table
```

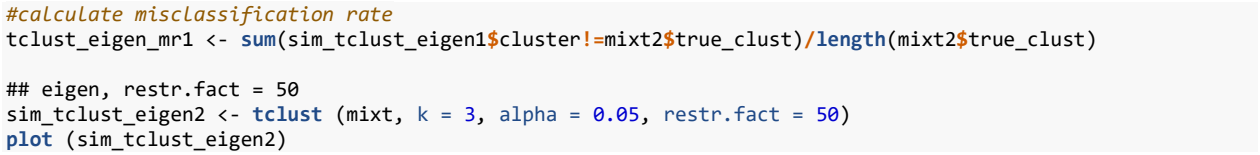
```
##        K-means TkMeans TCLUST(upp_bound=5) TCLUST(upp_bound=10)
##  [1,] 0.1066667    0.14         0.09333333          0.08666667
##  [2,] 0.1066667    0.14         0.09333333          0.64666667
##  [3,] 0.1066667    0.14         0.09333333          0.64000000
##  [4,] 0.1066667    0.14         0.09333333          0.08666667
##  [5,] 0.1066667    0.14         0.09333333          0.64666667
##  [6,] 0.1066667    0.14         0.09333333          0.08666667
##  [7,] 0.1066667    0.14         0.09333333          0.64000000
##  [8,] 0.1066667    0.14         0.09333333          0.64666667
##  [9,] 0.1066667    0.14         0.09333333          0.08666667
## [10,] 0.1066667    0.14         0.09333333          0.08666667
```

```
#TkMeans and TCLUST gives inconsistent results. We will investigate more.
```

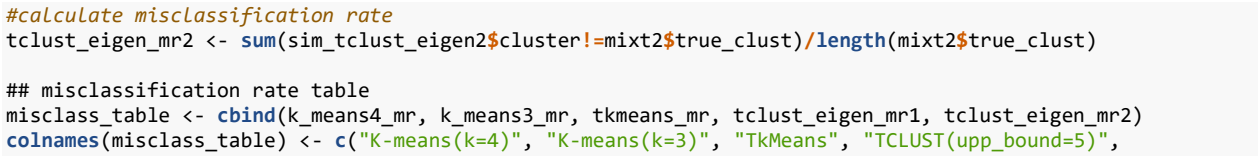
3.2 R code for simulation study (Section 5)

```
##################################
######### SIMULATION #############
##################################
par(mfrow=c(1,1))
library(mvtnorm)

#generate multivariate normal data
set.seed (10)
sigma1 <- diag (2) ## EigenValues: 1, 1
sigma2 <- diag (2) * 8 - 2 ## EigenValues: 8, 4
sigma3 <- diag (2) * 50 ## EigenValues: 50, 50
sigma4 <- diag(2) * 10-4
mixt <- rbind (rmvnorm (360, mean = c (0.0, 0), sigma = sigma1),
  rmvnorm (540, mean = c (5.0, 10), sigma = sigma2),
  rmvnorm (100, mean = c (2.5, 5), sigma = sigma3),
  rmvnorm(75, mean = c(20,35), sigma = sigma4))

plot(mixt, main = "Simulation", xlab = bquote(x[1]), ylab = bquote(x[2]))

mixt2 = data.frame(cbind(mixt, true_clust = c(rep(2,360), rep(1,540), rep(0,100))))

## Warning in cbind(mixt, true_clust = c(rep(2, 360), rep(1, 540), rep(0,
## 100))): number of rows of result is not a multiple of vector length (arg 2)

names(mixt2) = c("x","y","true_clust")

#select k , alpha
#ctl-curves
plot(ctlcurves(mixt, k = 1:4, alpha = seq(0, 0.20, by =  0.05)))

## Depending on arguments x, k and alpha, this function needs some time to compute.
## (Remove this message by setting "trace = 0")

#kmeans 4 clusters
sim_kmeans4 <- tclust(mixt, k = 4, alpha = 0, equal.weights = TRUE, restr.fact = 1)

## Warning in .tclust.warn(O, O$ret): The result is artificially constrained
## due to restr.fact = 1.
```

```r
plot(sim_kmeans4,main = "K-means", xlab = bquote(x[1]), ylab = bquote(x[2]), ylim = c(-15,50),
     xlim = c(-20,30))

#calculate misclassification rate
k_means4_mr=sum(sim_kmeans4$cluster!=mixt2$true_clust)/length(mixt2$true_clust)

#kmeans 3 clusters
sim_kmeans3 <- tclust(mixt, k = 3, alpha = 0, equal.weights = TRUE, restr.fact = 1)

## Warning in .tclust.warn(O, O$ret): The result is artificially constrained
## due to restr.fact = 1.

plot(sim_kmeans3,main = "K-means", xlab = bquote(x[1]), ylab = bquote(x[2]), ylim = c(-15,50),
     xlim = c(-20,30))

#calculate misclassification rate
k_means3_mr=sum(sim_kmeans3$cluster!=mixt2$true_clust)/length(mixt2$true_clust)

#trimmed kmeans
sim_tkmeans <- tclust(mixt, k = 3, alpha = 0.05, equal.weights = TRUE, restr.fact = 1)

## Warning in .tclust.warn(O, O$ret): The result is artificially constrained
## due to restr.fact = 1.

plot(sim_tkmeans, main = "Trimmed K-means")

#calculate misclassification rate
tkmeans_mr <- sum(sim_tkmeans$cluster!=mixt2$true_clust)/length(mixt2$true_clust)


#tclust
## eigen, restr.fact = 5
sim_tclust_eigen1 <- tclust (mixt, k = 3, alpha = 0.05, restr.fact = 5)

## Warning in .tclust.warn(O, O$ret): The result is artificially constrained
## due to restr.fact = 5.

plot (sim_tclust_eigen1)
```



```r
#calculate misclassification rate
tclust_eigen_mr1 <- sum(sim_tclust_eigen1$cluster!=mixt2$true_clust)/length(mixt2$true_clust)

## eigen, restr.fact = 50
sim_tclust_eigen2 <- tclust (mixt, k = 3, alpha = 0.05, restr.fact = 50)
plot (sim_tclust_eigen2)
```



```r
#calculate misclassification rate
tclust_eigen_mr2 <- sum(sim_tclust_eigen2$cluster!=mixt2$true_clust)/length(mixt2$true_clust)

## misclassification rate table
misclass_table <- cbind(k_means4_mr, k_means3_mr, tkmeans_mr, tclust_eigen_mr1, tclust_eigen_mr2)
colnames(misclass_table) <- c("K-means(k=4)", "K-means(k=3)", "TkMeans", "TCLUST(upp_bound=5)",
"TCLUST(upp_bound=50)")
misclass_table

##      K-means(k=4) K-means(k=3)   TkMeans TCLUST(upp_bound=5)
## [1,]    0.9962791    0.1665116 0.1423256          0.1209302
##      TCLUST(upp_bound=50)
## [1,]                 0.12
```