Open access • Journal Article • DOI:10.1080/03610918.2019.1626880

REPPlab: An R package for detecting clusters and outliers using exploratory projection pursuit

Daniel Fischer, Alain Berro, Klaus Nordhausen, Anne Ruiz-Gazen

Institutions: University of Toulouse, Vienna University of Technology

Published on: 02 Nov 2021 - Communications in Statistics - Simulation and Computation (Taylor & Francis)

Topics: Projection pursuit

WORKING PAPERS

N° TSE –1001

March 2019

"REPPlab: An R package for detecting clusters and outliers using exploratory projection pursuit"

Daniel Fischer, Alain Berro,
Klaus Nordhausen and Anne Ruiz-Gazen

Toulouse School of Economics

REPPlab: An R package for detecting clusters and outliers using exploratory projection pursuit

Daniel Fischer
Natural Resources Institute Finland(Luke)
31600 Jokioinen, Finland
E-mail: daniel.fischer@luke.fi

Alain Berro
Institut de Recherche en Informatique de Toulouse
University of Toulouse Capitole, France
E-mail: alain.berro@irit.fr

Klaus Nordhausen
Institute of Statistics & Mathematical Methods in Economics
Vienna University of Technology, Austria
E-mail: klaus.nordhausen@tuwien.ac.at

Anne Ruiz-Gazen
Toulouse School of Economics
University of Toulouse Capitole, France
E-mail: anne.ruiz-gazen@tse-fr.eu

February 27, 2018

Abstract

The R-package REPPlab is designed to explore multivariate data sets using one-dimensional unsupervised projection pursuit. It is useful as a preprocessing step to find clusters or as an outlier detection tool for multivariate data. Except from the packages tourr and rggobi, there is no implementation of exploratory projection pursuit tools available in R. REPPlab is an R interface for the Java program EPP-lab that implements four projection indices and three biologically inspired optimization algorithms. It also proposes new tools for plotting and combining the results and specific tools for outlier detection. The functionality of the package is illustrated through some simulations and using some real data.

Keywords: genetic algorithms, Java, kurtosis, particle swarm optimization, projection index, Tribes, projection matrix, unsupervised data analysis

1 Introduction

Exploratory Projection Pursuit (EPP) aims at finding potentially existing structures, typically clusters or outliers, in multivariate data sets by optimizing some index that reflects the interestingness of low dimensional linear projections. Principal Component Analysis (PCA) can be seen for example as a projection pursuit approach when using a scale measure as projection index. However, the main idea of EPP is to go beyond traditional PCA which only focuses on second-order moments and to consider other projection indices usually measuring non-gaussianity. The founding papers of EPP (Friedman and Tukey 1974), (Huber 1985) date back to the seventies and eighties and proposed already several projection indices together with different strategies about how to apply them for data exploration.

Recently, there is a renewed interest for EPP in several fields like for example hyperspectral imagery (Malpica et al. 2008), chemistry (Hou and Wentzell 2011) and genetics (Espezua et al. 2014). In these fields, the number of variables may be high with a limited number of samples and projection pursuit is suitable since it avoids the curse of dimensionality by projecting the data onto low dimensional subspaces. Nevertheless, as noticed by several authors, there exists almost no implementation of static EPP tools in statistical software. The R packages `tourr` (Wickham et al. 2011) and `tourrGui` (Huang et al. 2012), which contains a graphical user interface for `tourr`, are dedicated to tour animations while the `rrggobi` package (Cook and Swayne 2007) provides a command line interface to the interactive and dynamic graphical package GGobi. The package `pcaPP` (Filzmoser et al. 2014) is concerned with robust Principal Component Analysis.

The standalone Java program `EPP-lab` (Berro and Larabi Marie-Sainte 2014) is a program dedicated to EPP, described in (Larabi Marie-Sainte 2016) and freely available on GitHub. `EPP-lab` provides the possibility to run several times an optimization algorithm for a chosen one-dimensional projection index and to analyze the resulting projections in detail. Different projection indices and several biologically inspired algorithms are implemented in `EPP-lab`. The package `REPPlab` (Fischer et al. 2015) is an interface that gives R users access to all implemented projection indices and optimization algorithms of `EPP-lab`. Moreover, it provides several functions to visualize and analyze the results from `EPP-lab` to permit a thorough exploratory analysis of the obtained projections using R functionalities.

In general, EPP has two essential ingredients: a projection index and an optimization algorithm. Concerning the projection index, it is widely accepted in the EPP literature that it should measure the non-gaussianity of a projection, the Gaussian distribution being considered as the most uninteresting. There exist several families of indices as detailed in (Caussinus and Ruiz-Gazen 2009), (Rodriguez-Martinez et al. 2010) and (Koch 2014) that are aimed at revealing different non-gaussian structures. The four indices implemented in `EPP-lab` are the Friedman-Tukey, the Friedman, the discriminant and the kurtosis (either maximized or minimized) indices, see (Berro et al. 2010) for details concerning these indices. They belong to different families and, except for the so-called discriminant index which is a new proposal, they are well-known indices that have been studied in detail in the statistical literature.

As for the optimization algorithms, many proposals have been made for EPP (see (Berro et al. 2010) for some references and also (Tu et al. 2003), (Gou et al. 2010), (Espezua et al. 2014)). Following (Berro et al. 2010) and (Larabi Marie-Sainte et al. 2010), `EPP-lab` implements genetic and particle swarm optimization (PSO) algorithms that are biologically inspired. Such algorithms do not rely on any smoothness assumption of the function to optimize. Moreover, the Tribe algorithm, which is a particular PSO algorithm, is especially suited to find local optima which are of interest given the exploratory strategy we propose to follow and describe now.

The exploration philosophy in `EPP-lab` is not one of the traditional strategies in EPP. Usually, either a dynamical approach as in `tourr` or `rggobi`, or a global optimization method together with a structure removal (Friedman 1987b) is used. A dynamic approach is clearly of interest, but it may be considered as tedious for the data analyst. The usual alternative strategy consists in iterating a two-step procedure: look for the best projection direction, namely the one associated with the global optimum of the projection index, and remove the structure found from the data set. However, such a strategy may have some drawbacks. For example, finding a global optimum for a projection index is usually not a trivial task. This strategy also disregards all the possible different local optima found during this optimization process and may be time-consuming. Moreover, as stated in (Friedman 1987b), the structure removal based on orthogonal projections may miss some interesting projections, and other proposals are time-consuming. The strategy in `EPP-lab` differs from the previous one in the sense that all the local optima are saved for further analysis (Ruiz-Gazen et al. 2010). Such an approach has also been recently discussed in the discussion (Villa-

Vialaneix and Ruiz-Gazen 2015; Wickham et al. 2015a) of the paper (Wickham et al. 2015b). The use of genetic and particle swarm optimization methods helps in exploring efficiently the space of one-dimensional projections in this context.

The implementation of `EPP-lab` follows the work by (Berro et al. 2010) and (Larabi Marie-Sainte et al. 2010) who showed the value of considering several projection indices and several optimization algorithms to get the most of exploratory projection pursuit. It is described in (Larabi Marie-Sainte 2016).

Given the many potentially interesting projections obtained by using several projection indices and optimization algorithms, it is necessary to summarize the information because several directions may contain the same information as already stated in (Friedman 1987a). One way to do this is by combining the different projection directions as proposed in (Liski et al. 2016). This method combines several projection matrices with possibly different ranks to obtain an average projection matrix which also automatically chooses the rank for the average projection. Using this approach for EPP leads to summarize the similar projections in only one direction, while different directions that may correspond to some other local minima from the same or other projection indices or optimization algorithms, are combined in separate directions. The method is illustrated in the present paper through simulations and on a real data set.

The package `REPPlab` can be used as a preliminary step before clustering. First, it may help the data analyst in checking whether the data contains some clusters and thus helps in validating the use of some clustering method. EPP can also give guidelines for the choice of the number of clusters. Moreover, clustering may be performed on projection directions rather than on the original variables. This idea will be detailed in the simulations section. Another interesting feature of `REPPlab` is its ability to detect multivariate outliers. Observations that differ from the main bulk of the data are likely to be revealed by some projections associated with local maxima of the kurtosis for instance. Moreover, the package incorporates some specific functions to tag the observations that are far from the mean, or other location estimates, regarding the number of standard deviations, or other scale estimates, on the selected projection directions.

The outline of the paper is the following. The main features of the package are given in the second section. The advantage of combining several methods and local minima is illustrated in the third section using a simulation study in a clustering context. The data exploratory process using

REPPlab is detailed for clusters and outlier detection in Section 4 using two data sets. The last section concludes the paper.

2 The R-package REPPlab

The package REPPlab is freely available from the Comprehensive R Archive Network (CRAN) at http://CRAN.R-project.org/package=REPPlab and is published under the GNU General Public Licence (GPL) 2.0 or higher licence. For the package to work it is necessary to have besides R also the Java Development Kit JDK, version 1.4 or higher installed.

A schematic overview of the functions of REPPlab and their corresponding methods are provided in Figure 1. The package consists mainly of the three functions EPPlab, EPPlabOutlier and WhitenSVD where EPPlab is the main function of the package being the R interface to the actual Java program. The most important arguments of EPPlab are the data matrix x, the desired projection index PPindex which should be computed using the algorithm PPalg, and how often the index should be computed (n.simu). The default projection index is KurtosisMax. The names of the three possible algorithms are GA for the genetic algorithm and PSO and Tribe for the two particle swarm optimization algorithms.

2.1 Preliminary standardization

The function EPPlab always centers and scales the data, by subtracting the column means and dividing by the columns standard deviations, and the argument sphere controls whether or not the data should also be uncorrelated. The default is FALSE, but if set to TRUE the function WhitenSVD is used to whiten the data. WhitenSVD uses a singular value decomposition and estimates the rank of the data as the number of eigenvalues larger than 1E-06. Therefore whitening can also be done if there are fewer observations than dimensions.

2.2 Input parameters for the optimization algorithms

To fine-tune the algorithms it is possible to specify the number of particles (particles) for the standard particle swarm optimization algorithm and the number of individuals (individuals) for the genetic algorithm. Concerning the stopping criterion, it can be modified either by specifying

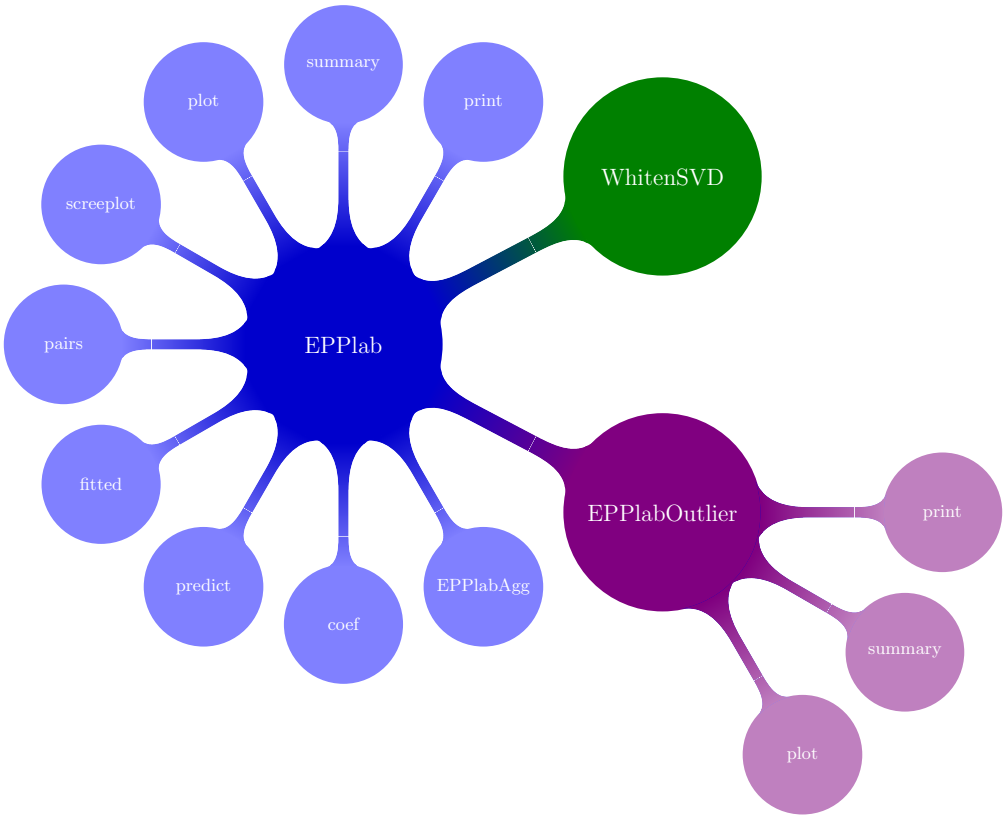

Figure 1: Schematic overview over the functions in the `REPPlab` package.

the maximum number of iterations (`maxiter`) or by specifying the convergence criteria `step_iter` and `eps`. The algorithms stop as soon as one of the two following conditions holds: the maximum number of iterations is reached or the relative difference between the index value of the present iteration i and the value of iteration i-`step_iter` is less than `eps`. In the last situation, the algorithm is said to converge and the function `EPPlab` will return the number of iterations needed to attain convergence. If the convergence is not reached, but the maximum number of iterations is attained, the function will return a warning.

2.3 Description of the outputs of the `EPPlab` function

The function `EPPlab` returns an object of class `epplab` which contains the estimated `n.simu` directions and their criterion value as well as other useful information. The found directions are ordered according to their criterion (index) value. For convenient working with an object of class `epplab`, the methods `print`, `summary`, `plot`, `pairs`, `screeplot`, `fitted`, `predict` and `coef` are provided. Since in common applications of the function `EPPlab` the number of times the algorithms are called (`n.simu`) will be large it will often not be meaningful to look at all results at once. For example `print` will present only information about the direction with the largest index criterion whereas all the other functions usually have an argument `which` to decide which directions should be investigated. By default often only the first ten directions will be presented. A natural first option to explore the results is to investigate them graphically. The function `screeplot` can be used to compare the objective criterion values from the different runs. The function plots the run number against its criterion value, and this should give an impression which directions might be the same. The `plot` function offers three types of plots controlled by the `type` argument. For `type = "angles"`, the angles between the run with the largest objective criterion and all other runs are plotted. This again should give an idea which directions might differ and which not. The default plotting type is however `type = "density"` which plots for the chosen directions marginal kernel density estimates while `type = "hist"` will give the corresponding histograms. Both options indicate if the corresponding directions yield anything interesting like clusters or outliers. Similarly the function `pairs` produces a scatter plot matrix of the directions included in `which` to evaluate correlations between directions or possible multidimensional structures. For further analysis of the desired directions, the projected data can be easily extracted using the `fitted` function, for

new observations the function `predict` computes the corresponding projections. The projecting directions can be obtained using the `coef` function.

2.4 Description of the `EPPlabOutlier` function

If the purpose of the analysis is outlier detection the function `EPPlabOutlier` is the best way to extract the relevant information from an `epplab` object. In the `EPPlabOutlier` function the user can specify which location and scale measure are to be used and what is the factor `k` that classifies an observation as an outlier based on the location and scale used. The `location` and `scale` arguments take as input, functions which return the corresponding quantities for a vector. The defaults are `mean` and `sd` but, for example, also `median` and `mad` might be used. The `EPPlabOutlier` creates an object of class `epplabOutlier` for which `print`, `summary` and `plot` functions are available. The output of these three functions is most meaningful if the data given to `EPPlab` before calling `epplabOutlier` has row labels for the observations. Just printing the object of class `epplabOutlier` returns a binary matrix where 1 in its ij-th element indicates that the observation i is considered an outlier in direction j. A visual presentation of this binary matrix is obtained using `plot` where the user can choose the colors for outliers and non-outliers and whether only those rows should be plotted which are considered at least in one direction as an outlier. A maybe more informative overview is provided using `summary` which informs about the total number of outliers found and in how many directions each of the identified outliers are considered atypical.

2.5 Description of the `EPPlabAgg` function

This function is designed to combine and summarize the different projection directions. The input for the function is either an `epplab` object or a list of such objects when, for example, results from different calls with different indices or algorithms should be combined. The combining idea is quite simple. Denote as u_i, $i = 1, \ldots, N$ the N unit projection vectors which should be combined and as P_i, $i = 1, \ldots, N$ the corresponding projection matrices which all have rank 1. This function performs an eigenvalue - eigenvector decomposition of

$$P^* = \frac{1}{N} \sum_{i=1}^{N} P_i$$

obtaining the p eigenvalues $\lambda_1 \geq \ldots \geq \lambda_p$ and corresponding eigenvectors. The idea is then to keep those k eigenvectors which explain "most" of P^*. For a detailed description and the theoretic foundation see (Liski et al. 2016) who also suggest methods to automatically decide upon the value k. The function `EPPlabAgg` offers then three options to decide upon k using the `method` argument. Two automatic ones denoted `"inverse"` and `"sq.inverse"` follow (Liski et al. 2016) or as alternative the method `"cumulative"`, which chooses the minimal k such that $\sum_{i=1}^{k} \lambda_i / \sum_{i=1}^{N} \lambda_i \geq c$ where c is the percentage given by the argument `percentage`.

3 Simulations

EPP is often used as preprocessing step to find clusters in data. In the following simulation study, we will compare different indices and combination methods implemented in `REPPlab` for that purpose. For the comparison we consider two settings each having three underlying clusters. In the first setting the cluster sizes are balanced with $n_1 = n_2 = n_3 = 100$ whereas in the second setting the sizes are unbalanced with $n_1 = 200$, $n_2 = 80$, $n_3 = 20$. For the cluster design we follow (Hou and Wentzell 2011) and have three 10-variate normal populations with $N_{10}(\mu_i, \Sigma)$, where $\mu_1 = (-1, -0.58, 0, \ldots, 0)^\top$, $\mu_2 = (1, -0.58, 0, \ldots, 0)^\top$, $\mu_3 = (0, 1.15, 0, \ldots, 0)^\top$ and $\Sigma = \text{diag}(0.1, 0.2, 1, \ldots, 1)$. In order to make the design not simply visible, we then rotate the observations with a random orthogonal matrix. Hence when performing dimension reduction, k=2 directions should be sufficient to find the clusters. The success of the different preprocessing methods will be evaluated using the adjusted Rand index (Rand 1971) when k-means clustering is used when searching for the three clusters. The index has the range 0 to 1 and the larger, the better.

For both settings we simulated 1000 times all indices in the package `epplab` with the settings `PPalg = "Tribe"`, `n.simu = 100`, `maxiter = 200` and `sphere = TRUE`.

To get an idea of the computational complexity the computation times for each setting and all the indices are shown in Figure 2. As can be seen here, the setting does not matter, but there are considerable differences between the indices. Both kurtosis indices (kmin and kmax) are the fastest to compute whereas the discriminant index (disc) and the Friedman Tukey index (ft) are quite slow to compute. The Friedman index (fried) is intermediate regarding the computation time.

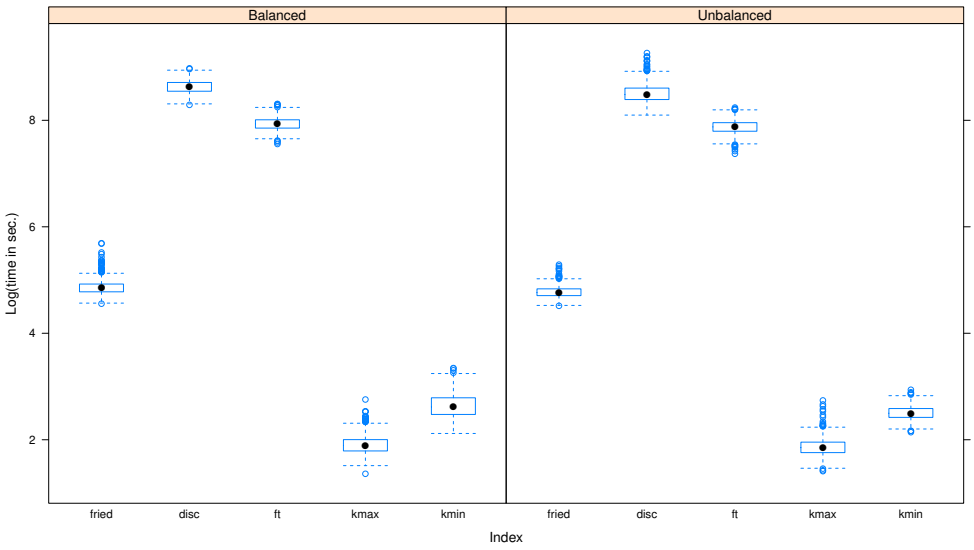

Figure 2: Computation times for the 100 projection directions based on 1000 repetitions in both cluster settings.

Therefore, when evaluating the performance of combination methods we either combine all indices or only the fast indices designed for clustering (kmin and fried) and the two combinations will be denoted as 'all' and 'fast' respectively in what follows.

When combining results we used the in-build methods as provided by `EPPlabAgg`. We used options `"inverse"`, `"sq.inverse"` and `"cumulative"` with percentages 85% and 95%, denoted as 'cum85' and 'cum95'. We will show however only the results of inverse and cum85 as using `sq.inverse` was a little worse than inverse and cum95 was clearly the worst from all of them.

Figure 3 shows the performance for both cases. Obviously, maximizing kurtosis is not a good idea as preparation for clustering since the index is designed for finding outliers. The Friedman-Tukey index is also not very good, and the discriminant index differs a lot between the two settings.

However when combining all indices the performance is in general good, but better with only the fast indices. Note that, when combining all methods, the performance does not fail although bad performing indices are included. When comparing the two aggregation methods, it seems that for the balanced case cum85 is better while for the unbalanced case the inverse method is better.

Figure 4 shows how many directions the different aggregation methods choose. Recall that the number of clusters is 3 and so $k = 2$ should be right but naturally from the figure one cannot conclude if the right two directions are chosen. The Figure nevertheless gives the number of

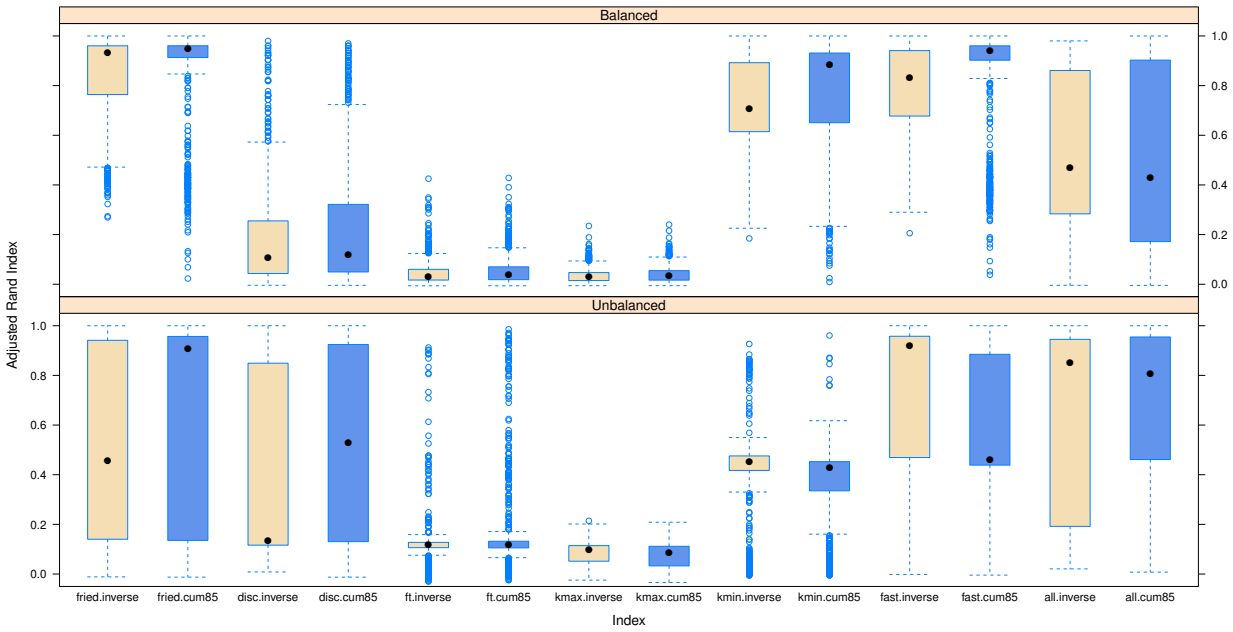

Figure 3: Adjusted Rand index for the different combinations of the found projections based on 1000 repetitions in both settings (balanced at the top and unbalanced at the bottom).

directions selected, and in general, it seems that cum85 chooses more directions than `inverse`.

At the end of this Section, we put EPP in context to competing methods. We consider applying k-means to the raw data using all ten dimensions as well as principal component analysis, where we use the first PC, the first two PCs and the first l PCs which explain 80% of the variation. We also compare EPP to invariant coordinate selection (ICS) (Tyler et al. 2009) using the default as implemented in `ICS` (Nordhausen et al. 2008) and choose the first two and last two components. The corresponding adjusted Rand index values are compared to our EPP approach in Figure 5. From this comparison, one can conclude that using PCA in this context is not beneficial at all. It seems even worse than applying k-means on the raw data. ICS is a clear improvement but still not as good as EPP. While for the balanced setting the Friedman index is performing best when aggregated using 85% in the cumulative approach, for the unbalanced setting, the combination of Friedmann and minimizing kurtosis using the inverse aggregation approach seems best. In general, we believe that the best index and aggregating method is data dependent. However, combining several projections using the inverse aggregation seems to be a safe choice.

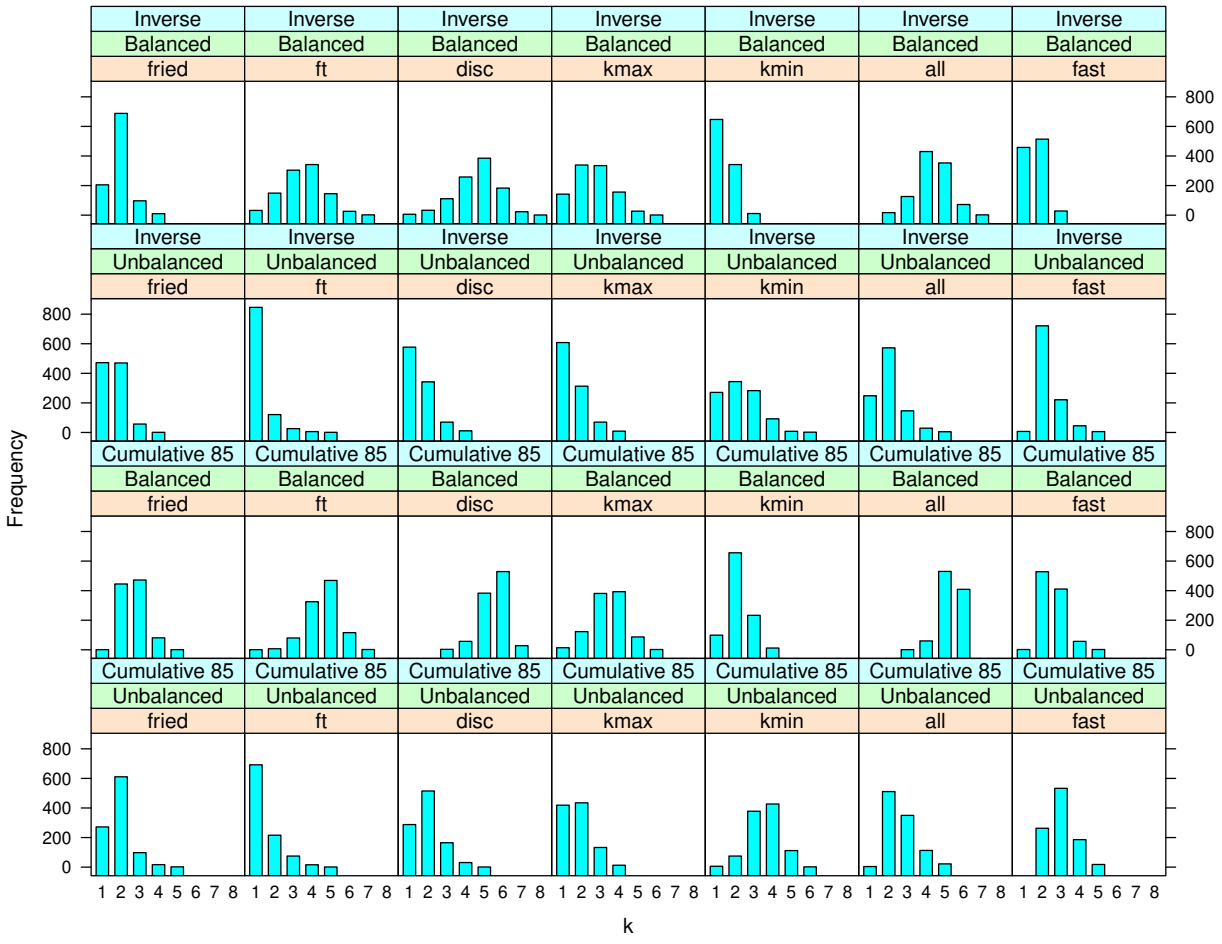

Figure 4: Number of directions chosen by the different aggregation methods for the two cluster settings.

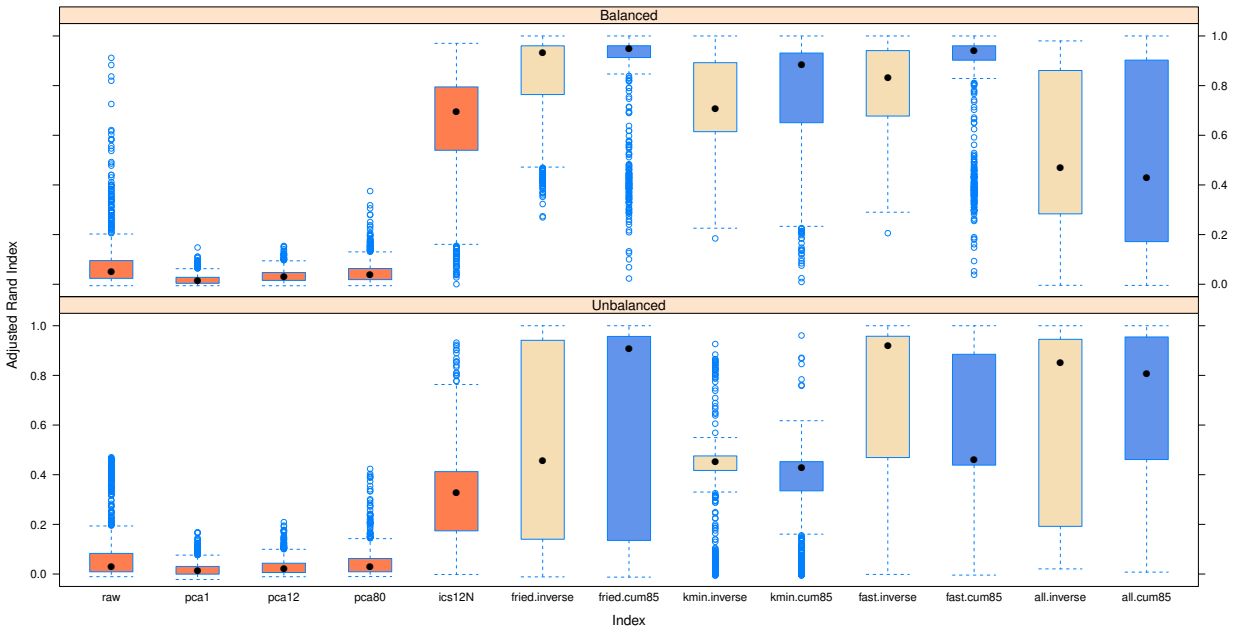

Figure 5: Adjusted Rand index for k-means underlying different component selection methods for the two cluster settings.

4 Examples

The following examples will show how to use the `REPPlab` package for exploratory data analysis to find clusters and outliers in multivariate data.

4.1 Detecting clusters

First, we consider how to detect clusters in multivariate data using the `REPPlab` package. For this demonstration purpose, we use the so-called Lubishew data which is available in the package `amap`. The data set is known to contain three clusters, and the goal is to find all of them assuming that the number of clusters is also unknown. It is a very simple example helpful to understand the different steps of the strategy that can be implemented using `REPPlab`.

We first load the necessary packages and the data and also fix the random seed.

```
> library("REPPlab"); library("amap")
> set.seed(4567)
> data("lubisch")
```

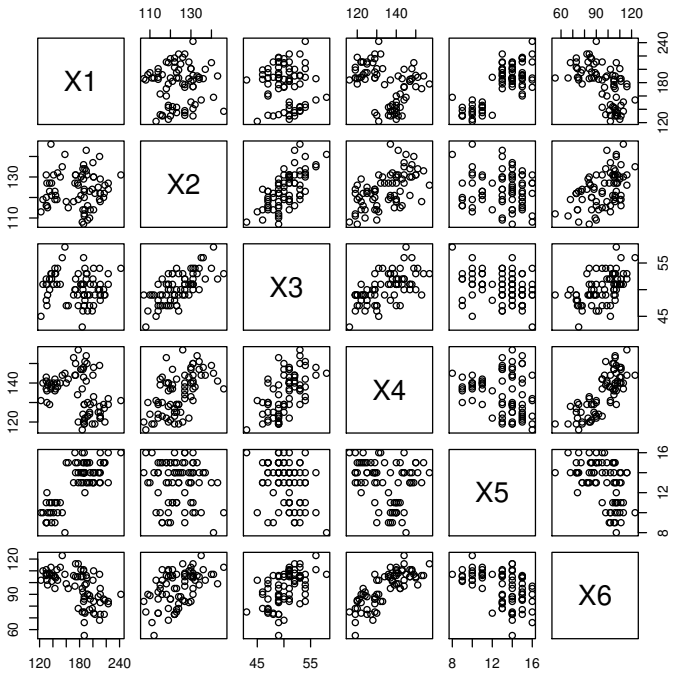

Figure 6: Scatterplot matrix of the Lubishew data.

```
> X <- lubisch[, 2:7]
> Class <- lubisch[, 8]
```

For convenience, we denoted the variables which will be used to identify the clusters as `X` and stored the correct class labels in the vector `Class`. Looking at the scatterplot matrix in Figure 6 does indicate that there are clusters but not so clear that there are three clusters.

```
> pairs(X)
```

We use now the function `EPPlab` to obtain 100 directions using the Friedman index for the sphered data using the Tribes algorithm.

```
> res.Fried.Tribe <- EPPlab(X, PPalg = "Tribe", PPindex = "Friedman",
+ n.simu = 100, maxiter = 200, sphere = TRUE)
> summary(res.Fried.Tribe, digits=4)

REPPlab Summary
---------------
```

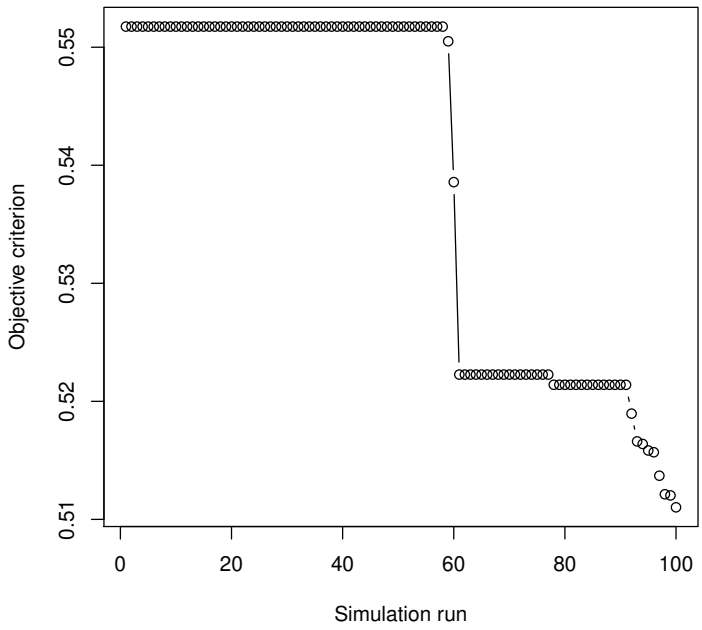

Figure 7: Screeplot of the criterion values of the Friedman index for the Lubishew data.

```
Index name       : Friedman
Index values     : 0.5517 0.5517 0.5517 0.5517 0.5517 0.5517
0.5517 0.5517 0.5517 0.5517
Algorithm used   : Tribe
Sphered          : TRUE
Iterations       : 48 52 48 53 47 55 42 53 58 46
```

The `summary` is here not too informative but tells us which index and algorithm were used as well as if the data was sphered or not. It also shows the index value of the first ten directions and how many iterations they needed. For the first ten directions the criterion value is always the same, but to see if different directions are found, plots are more informative.

First, we look at the screeplot of the `res.Fried.Tribe` object

```
> screeplot(res.Fried.Tribe, which = 1:100)
```

which, as Figure 7 reveals, gives the same criterion value for around the first 60 directions. Then

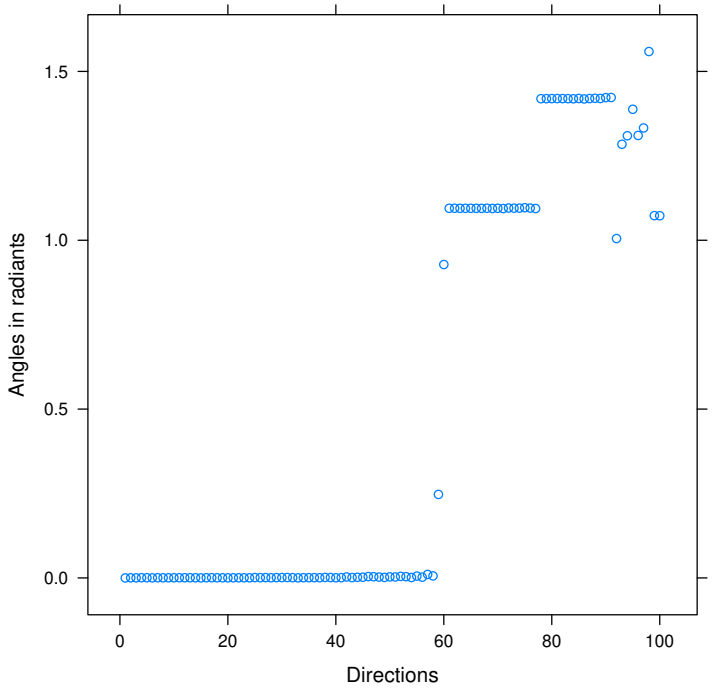

Figure 8: Angles between the first direction and all other directions for the Lubishew data.

there is a sharp drop and the next around 20 directions have the same criterion value before a final small drop occurs for around ten directions. Finally, the last six directions are associated with some dispersed index values. Naturally, two identical criterion values do not exclude different directions, and the last small drop might also not be a new interesting direction. This can be further investigated by looking at the angles between the directions.

```
> plot(res.Fried.Tribe, type = "angles", which = 1:100)
```

Figure 8 shows that the directions in each of the three criterion groups are identical and supports the hypothesis that the three different criterion groups correspond to different directions of interest. This can be verified for example by looking at candidates from all these groups. We choose for this purpose one direction from each of the three groups using Figure 8, namely directions 1, 70 and 90 which have the criterion values

```
> res.Fried.Tribe$PPindexVal[c(1, 70, 90)]
```

```
[1] 0.5517 0.5223 0.5214
```

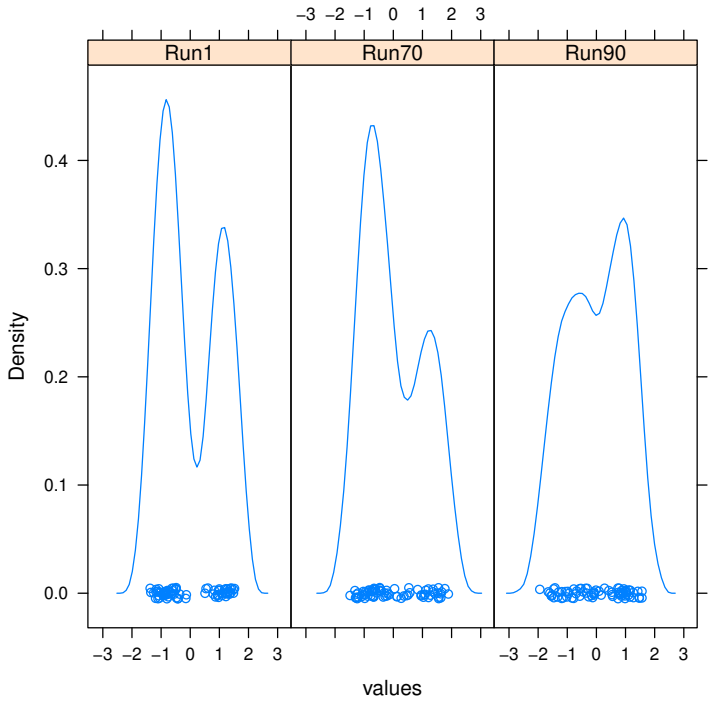

Figure 9: Marginal density plots for direction 1, 70 and 90 for the Lubishew data.

The code

```
> plot(res.Fried.Tribe, which = c(1, 70, 90), layout = c(3, 1))
```

```
> pairs(res.Fried.Tribe, which = c(1, 70, 90))
```

gives the different marginal kernel density estimates as shown in Figure 9 and reveals that, at least for direction 1 and 70, there seem to be two clusters. Looking therefore at the scatter plot matrix of these three directions shows in Figure 10 that the directions 1 and 70 combined reveal three clusters whereas direction 90 seems not really to add anything of interest.

This can be verified as shown in Figure 11 using the class information labels and the code

```
> plot(fitted(res.Fried.Tribe, which = c(1, 70)), col = Class, pch = 16)
```

The analysis above is fast to do but focuses on one index only and is difficult to automatize. Following the recently suggested methodology of (Liski et al. 2016) to summarize many projections,

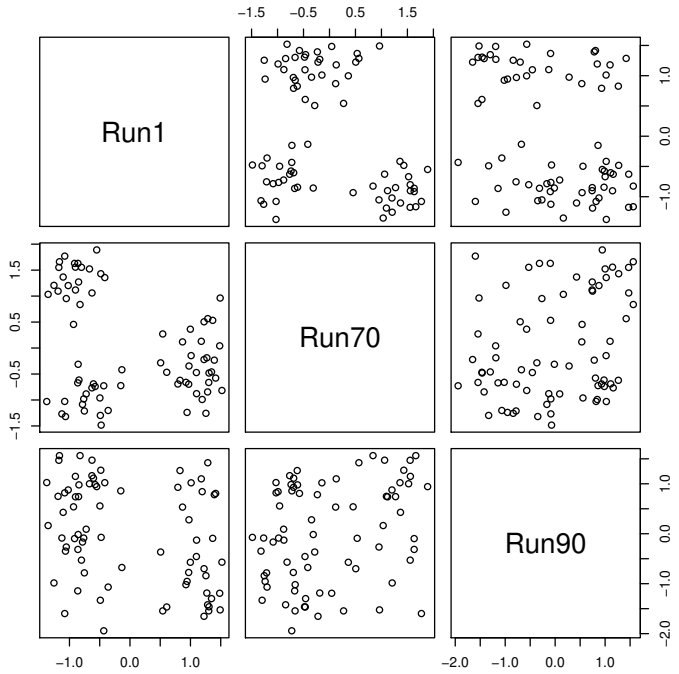

Figure 10: Scatter plot matrix of directions 1, 70 and 90 for the Lubishew data.

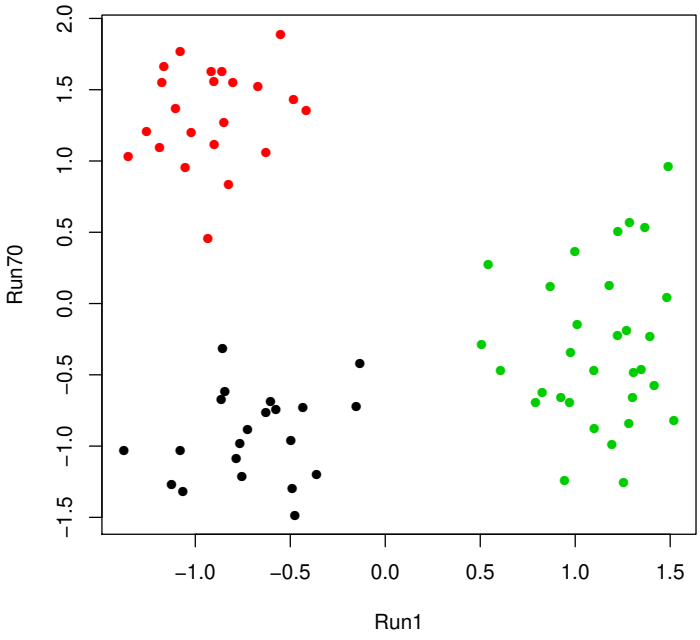

Figure 11: Scatter plot of directions 1, 70 for the Lubishew data with colored groups.

as implemented in the function `EPPlabAgg`, provides therefore a way to combine several indices as already demonstrated in Section 3.
For this data, we will then compute, three more indices, each with 100 runs.

```
> set.seed(1234)
> res.Dis.Tribe <- EPPlab(X, PPalg = "Tribe",PPindex = "Discriminant",
+ n.simu = 100, maxiter = 200, sphere = TRUE)
> res.Kmin.Tribe <- EPPlab(X, PPalg = "Tribe", PPindex = "KurtosisMin",
+ n.simu = 100, maxiter = 200, sphere = TRUE)
> res.FT.Tribe <- EPPlab(X, PPalg = "Tribe", PPindex = "FriedmanTukey",
+ n.simu = 100, maxiter = 200, sphere = TRUE)
```

Note from the previous warning that the convergence criterion was not reached for one simulation run but this has no impact on the final result given the large number of runs.

```
> res.ALL <- list(res.Fried.Tribe, res.Dis.Tribe, res.Kmin.Tribe, res.FT.Tribe)
```

The object `res.ALL` is then a combination of all the different indices and to summarize the results we use the function `EPPlabAgg` with the option `method = "inverse"` which automatically chooses the rank of the final projection matrices as recommended in (Liski et al. 2016).

```
> ALL.agg.inverse <- EPPlabAgg(res.ALL, method = "inverse")
> ALL.agg.inverse$k

[1] 3
```

Hence we can see that, by combining all the 400 one-dimensional projections, three interesting directions are suggested by this method. Looking at the scatterplot based on these three directions,

```
> pairs( as.matrix(X) %*% ALL.agg.inverse$O, col = Class, pch=16)
```

reveals in Figure 12 also the three clusters of interest. appropriately.
However, actually, for the separation here, the second direction is not useful at all moreover, also, with the first and second directions, the separation is not as clear as in Figure 11 with the two

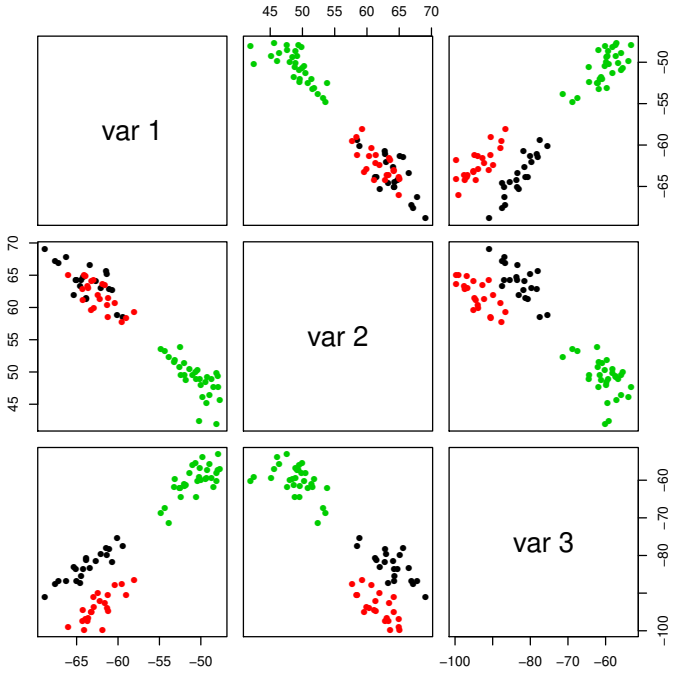

Figure 12: Scatter plot matrix of three directions derived from `AOP` for the Lubishew data, where the different colors correspond to the different classes.

directions 1 and 70. However, the second analysis was more automated and can combine results from different indices. Note that it is also possible to combine projection matrices obtained using different optimization algorithms.

4.2 Outlier detection example

In this example the goal is to detect outliers. For the demonstration we use the `ReliabilityData` which is made available in `REPPlab`. The data provides 55 measurements made on 520 units during one production process. For the producer, it is of interest to find those produced units which might be faulty in order to check them before selling them.

Looking first at all marginal boxplots as shown in

```
> library("REPPlab")
> data("ReliabilityData")
```

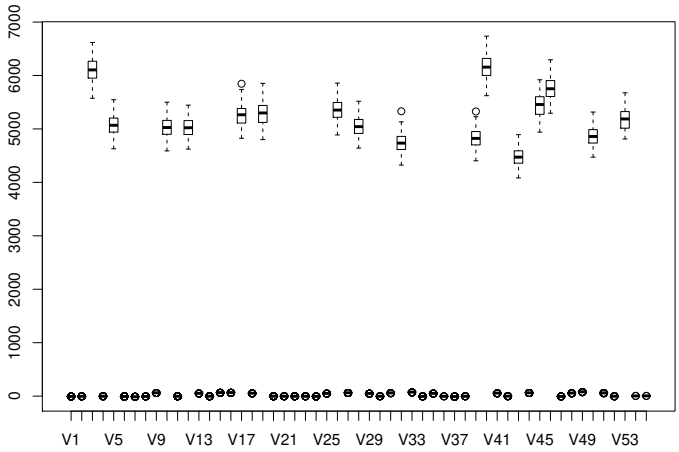

Figure 13: Boxplots for all variables in the data set `ReliabilityData`.

```
> boxplot(ReliabilityData)
```

reveals, that the scales of the variables differ considerably and shows marginal outliers. The exact number of marginal outliers can be obtained for example using the code

```
> id.out <- function(x)
+     { ind <- 1:length(x)
+       BD <- boxplot.stats(x, do.conf = FALSE, do.out = FALSE)
+       ind[x < BD$stats[1] | x > BD$stats[5]]}
> OUT.IDS <- apply(ReliabilityData,2,id.out)
> OUT.IDS.unique <- sort(unique(unlist(OUT.IDS)))
> (N.out <- length(OUT.IDS.unique))
```

```
[1] 239
```

However, according to the manufacturer all marginal measurements are in the acceptable range and such a vast number of outliers seems unrealistic. Therefore rather multivariate methods should be applied. For many multivariate methods, it is, however, a problem that several of these variables have almost no variation

```
> round(head(sort(apply(ReliabilityData, 2, sd))), 4)
```

```
V1    V42    V21    V38    V24    V52
0.0066 0.0095 0.0104 0.0108 0.0121 0.0122

> round(head(sort(apply(ReliabilityData, 2, mad))), 4)

V24    V22     V1    V11    V20    V14
0.0000 0.0033 0.0040 0.0060 0.0068 0.0086
```

For example, methods based on the MCD (Minimum Covariance Determinant), a popular estimate for robust scatter, cannot be computed here for the whole data set due to these variables with almost equal measurements. EPP, on the other hand, does not have such problems as we will demonstrate now. The recommended projection index for outlier detection is `KurtosisMax` for which we will compute 100 runs for this data.

```
> set.seed(4567)
> res.KurtM.Tribe <- EPPlab(ReliabilityData, PPalg = "Tribe", n.simu = 100,
+ maxiter = 200, sphere = TRUE)
```

To perform outlier detection, we call the function `EPPlabOutlier` on the object. To define an outlier in this context, we use first the rule that the observation must deviate by more than five standard deviations from the mean.

```
> OUTms <- EPPlabOutlier(res.KurtM.Tribe, k = 5, location = mean, scale = sd)
> summary(OUTms)

REPPlab Outlier Summary
-----------------------
Index name       : KurtosisMax
Algorithm used   : Tribe
Location used    : mean
Scale used       : sd
k value used     : 5
-----------------------
```

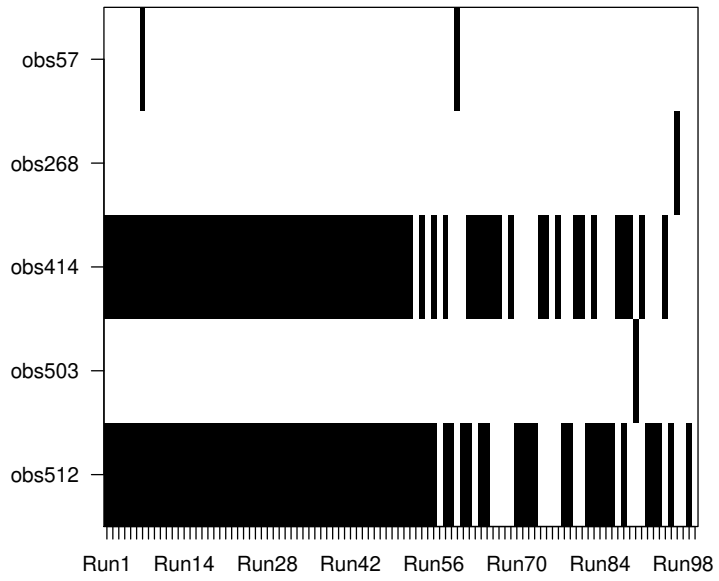

Figure 14: Visual representation of the observations considered outliers for the `ReliabilityData` when using mean and standard deviation.

```
Number of outliers detected: 5

Observations considered outliers:

OutlierID:  obs57 obs268 obs414 obs503 obs512

Frequency:  2     1      73     1      79

Percentage: 2     1      73     1      79
```

The summary of this function reveals that five observations are considered outliers in this context. Observations 414 and 512 were extreme in 73 and 79 runs out of the 100 runs while the other three observations were only rarely considered outliers. A graphical display (see Figure 14) of this result is obtained as

```
> plot(OUTms, las = 1)
```

In case it is costly to check for production errors, one may require that observation is an outlier only when it is detected so in at least four directions.

```
> totms <- apply(OUTms$outlier, 1, sum)
> totms[totms > 3]

obs414 obs512
73     79
```

This condition leaves two candidates.

It is however well known that both mean and standard deviation suffer considerably from the presence of outliers. Therefore also robust measures can be used to categorize outliers. Common choices are for example to replace the mean and the standard deviation with the median and the median absolute deviation respectively.

```
> OUTmm <- EPPlabOutlier(res.KurtM.Tribe, k = 5, location =  median, scale = mad)
> summary(OUTmm)

REPPlab Outlier Summary
-----------------------
Index name       : KurtosisMax
Algorithm used   : Tribe
Location used    : median
Scale used       : mad
k value used     : 5
-----------------------
Number of outliers detected: 32
Observations considered outliers:
OutlierID:  obs34 obs48 obs57 obs74 obs129 obs135 obs156  obs164 obs173 obs174
Frequency:  2     2     33    2     2      2      1       2      2      2
Percentage: 2     2     33    2     2      2      1       2      2      2
OutlierID:  obs178 obs183 obs196 obs225 obs259 obs268 obs319 obs326 obs344
Frequency:  2      3      2      1      1      1      3      1      2
Percentage: 2      3      2      1      1      1      3      1      2
OutlierID:  obs355 obs363 obs367 obs372 obs375 obs414  obs429 obs460 obs461
```

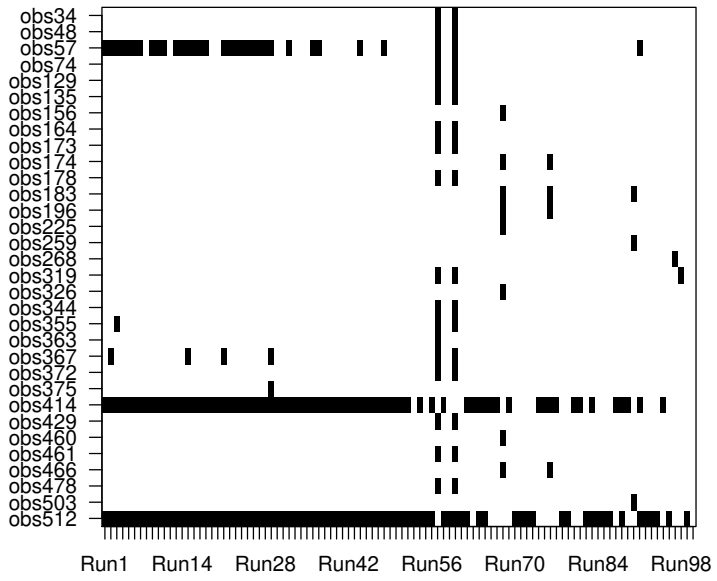

Figure 15: Visual representation of the observations considered outliers for the `ReliabilityData` when using median and median absolute deviation.

```
Frequency:  3      1      6      2      1      74     2      1      2

Percentage: 3      1      6      2      1      74     2      1      2

OutlierID:  obs466 obs478 obs503 obs512

Frequency:  2      2      1      81

Percentage: 2      2      1      81
```

In that case, 32 observations are considered outliers. The most extreme ones are, as for the non-robust call, observations 414 and 512. However, also observation 57 is now considered quite often as an outlier. Again we can also visualize this (Figure 15) via

```
> plot(OUTmm, las = 1)
```

Selecting now only those observations which were considered in at least 4 directions as outlier, gives the following observations

```
> totmm <- apply(OUTmm$outlier, 1, sum)
> totmm[totmm > 3]
```

```
obs57 obs367 obs414 obs512
33      6     74     81
```

Hence, next to the same two observations from the previous call, we have to add two more suspects. A scatter plot of the first and second direction (Figure 16) will then be used to see how extreme the observations are in these directions.

```
> ProjDir1 <- fitted(res.KurtM.Tribe)
> ProjDir2 <- fitted(res.KurtM.Tribe, which = 2)
> range(ProjDir1)

[1] -10.22  15.17

> range(ProjDir2)

[1] -12.98  13.64

> plot(ProjDir1[-c(57, 367, 414, 512)], ProjDir2[-c(57, 367, 414, 512)],  ylim =
+ = c(-16, 16), xlim = c(-16, 16), ylab = "Projection 1", xlab = "Projection 2")
> points(ProjDir1[c(57, 367, 414, 512)], ProjDir2[c(57, 367, 414, 512)],
+ col = 2:5, pch = 15)
> text(ProjDir1[c(57, 367, 414, 512)], ProjDir2[c(57, 367,  414, 512)], pos = 2,
+  label = row.names(ReliabilityData)[c(57, 367, 414, 512)])
```

As can be seen in this last figure, the two directions lead to a similar representation and all four observations can be considered as quite extreme with observations 414 and 512 being more extreme than 57 and 367.

5 Conclusion

EPP is a useful and interesting preliminary step in data analysis that may reveal non-gaussian hidden structures such as clusters or outliers in multivariate numerical data sets. However, very few EPP tools are available in the standard statistical software. The package `REPPlab` is an excellent opportunity for R users to access several projection indices and optimization algorithms that are

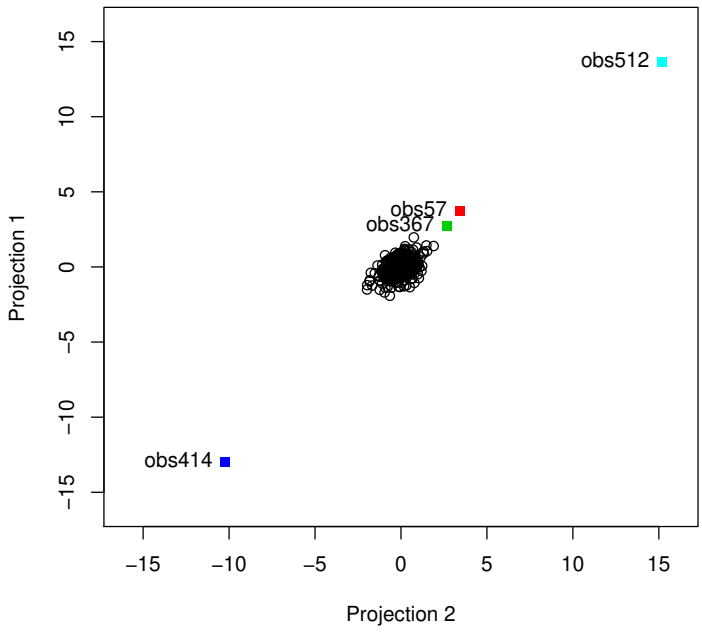

Figure 16: Visual inspection of the four outlier candidates in the `ReliabilityData` for the first two projections.

already available in the Java program `EPP-lab`. `REPPlab` also offers some extra functionality to explore and summarize the obtained projections. Some of the functionalities, such as the exploration of the projection index values and the cosines between projection directions and also some outlier detection tools, were already present in `EPP-lab`. Nevertheless, the novel approach of combining several projections from many different indices or/and algorithms is original and seems promising given the shown simulation results and examples. The implementation of other PP indices like the Stahel-Donoho index used for instance in (Maronna and Yohai 1995) for outlier detection is one of the perspectives of the present work.

Acknowledgements

The authors wish to acknowledge CSC – IT Center for Science, Finland, for providing computational resources. The article is based upon work from COST Action CRoNoS, supported by COST (European Cooperation in Science and Technology).

References

Berro, A. and Larabi Marie-Sainte, S. (2014). *EPP-lab: A tool for Exploratory Projection Pursuit*.

Berro, A., Larabi Marie-Sainte, S., and Ruiz-Gazen, A. (2010). Genetic algorithms and particle swarm optimization for exploratory projection pursuit. *Annals of Mathematics and Artificial Intelligence*, 60(1-2):153–178.

Caussinus, H. and Ruiz-Gazen, A. (2009). Exploratory projection pursuit. In Govaert, G., editor, *Data Analysis*, pages 67–92. John Wiley & Sons.

Cook, D. and Swayne, D. F. (2007). *Interactive and dynamic graphics for data analysis: with R and GGobi*. Springer Science & Business Media.

Espezua, S., Villanueva, E., Maciel, C. D., and Carvalho, A. (2014). A projection pursuit framework for supervised dimension reduction of high dimensional small sample datasets. *Neurocomputing*.

Filzmoser, P., Fritz, H., and Kalcher, K. (2014). *pcaPP: Robust PCA by Projection Pursuit*. R package version 1.9-60.

Fischer, D., Berro, A., Nordhausen, K., and Ruiz-Gazen, A. (2015). *REPPlab: R Interface to 'EPP-Lab', a Java Program for Exploratory Projection Pursuit*. R package version 0.9.3.

Friedman, J. H. (1987a). Contribution to the discussion Jones and Sibson (1987). *JR Statist. Soc. A*, 150:26–27.

Friedman, J. H. (1987b). Exploratory projection pursuit. *Journal of the American Statistical Association*, 82(397):249–266.

Friedman, J. H. and Tukey, J. W. (1974). A projection pursuit algorithm for exploratory data analysis. *IEEE Transactions on Computers*, C-23:881–889.

Gou, S., Feng, J., and Jiao, L. (2010). Clustering via dimensional reduction method for the projection pursuit based on the icsa. *Journal of Electronics (China)*, 27(4):474–479.

Hou, S. and Wentzell, P. (2011). Fast and simple methods for the optimization of kurtosis used as a projection pursuit index. *Analytica Chimica Acta*, 704(1):1–15.

Huang, B., Cook, D., and Wickham, H. (2012). `tourrGui`: A gwidgets gui for the tour to explore high-dimensional data using low-dimensional projections. *Journal of Statistical Software*, 49:1–12.

Huber, P. J. (1985). Projection pursuit. *The Annals of Statistics*, 13(2):435–475.

Koch, I. (2014). *Analysis of Multivariate and High-Dimensional Data*. Cambridge University Press, New York.

Larabi Marie-Sainte, S. (2016). Detection and visualization of non-linear structures in large datasets using exploratory projection pursuit laboratory (epp-lab) software. *Journal of King Saud University-Computer and Information Sciences*.

Larabi Marie-Sainte, S., Berro, A., and Ruiz-Gazen, A. (2010). An efficient optimization method for revealing local optima of projection pursuit indices. In Dorigo, M. e. a., editor, *Swarm Intelligence*, volume 6234 of *Lecture Notes in Computer Science*, pages 60–71. Springer-Verlag.

Liski, E., Nordhausen, K., Oja, H., and Ruiz-Gazen, A. (2016). Combining linear dimension reduction subspaces. In Agostinelli, C., Basu, A., Filzmoser, P., and Mukherjee, D., editors, *Recent Advances in Robust Statistics: Theory and Applications*. Springer-Verlag.

Malpica, J. A., Rejas, J. G., and Alonso, M. C. (2008). A projection pursuit algorithm for anomaly detection in hyperspectral imagery. *Pattern Recognition*, 41(11):3313–3327.

Maronna, R. A. and Yohai, V. J. (1995). The behavior of the stahel-donoho robust multivariate estimator. *Journal of the American Statistical Association*, 90(429):330–341.

Nordhausen, K., Oja, H., and Tyler, D. E. (2008). Tools for exploring multivariate data: The package ICS. *Journal of Statistical Software*, 28(6):1–31.

Rand, W. M. (1971). Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850.

Rodriguez-Martinez, E., Goulermas, J. Y., Mu, T., and Ralph, J. F. (2010). Automatic induction of projection pursuit indices. *IEEE Transactions on Neural Networks*, 21(8):1281–1295.

Ruiz-Gazen, A., Larabi Marie-Sainte, S., and Berro, A. (2010). Detecting multivariate outliers using projection pursuit with particle swarm optimization. In Lechevallier, Y. and Saporta, G., editors, *Proceedings of COMPSTAT'2010*, pages 89–98. Physica-Verlag HD.

Tu, J., Huang, T., Beveridge, R., and Kirby, M. (2003). Orthogonal projection pursuit using genetic optimization. In *Statistical Signal Processing, 2003 IEEE Workshop on*, pages 266–269. IEEE.

Tyler, D., Critchley, F., Dümbgen, L., and Oja, H. (2009). Invariant coordinate selection. *Journal of Royal Statistical Society B*, 71:549–592.

Villa-Vialaneix, N. and Ruiz-Gazen, A. (2015). Beyond multidimensional data in model visualization: High-dimensional and complex nonnumeric data. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 8(4):232–239.

Wickham, H., Cook, D., and Hofmann, H. (2015a). Authors' response to discussants. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 8(4):242–244.

Wickham, H., Cook, D., and Hofmann, H. (2015b). Visualizing statistical models: Removing the blindfold. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 8(4):203–225.

Wickham, H., Cook, D., Hofmann, H., and Buja, A. (2011). `tourr`: An R package for exploring multivariate data with projections. *Journal of Statistical Software*, 40(2):1–18.