Package 'otrimle'

July 22, 2025

Type Package

Title Robust Model-Based Clustering

Description Performs robust cluster analysis allowing for outliers and noise that cannot be fitted by any cluster. The data are modelled by a mixture of Gaussian distributions and a noise component, which is an improper uniform distribution covering the whole Euclidean space. Parameters are estimated by (pseudo) maximum likelihood. This is fitted by a EM-type algorithm. See Coretto and Hennig (2016) <doi:10.1080/01621459.2015.1100996>, and Coretto and Hennig (2017) <https://jmlr.org/papers/v18/16-382.html>.

Version 2.0

Date 2021-05-28

Imports stats, utils, graphics, grDevices, mvtnorm, parallel, foreach, doParallel, robustbase, mclust

License GPL (>= 2)

LazyData TRUE

NeedsCompilation no

Author Pietro Coretto [aut, cre] (Homepage: <https://pietro-coretto.github.io>), Christian Hennig [aut] (Homepage: <https://www.unibo.it/sitoweb/christian.hennig/en>)

Maintainer Pietro Coretto <pcoretto@unisa.it>

Repository CRAN

Date/Publication 2021-05-29 06:30:02 UTC

Contents

`banknote` *Swiss Banknotes Data*

Description

Data from Tables 1.1 and 1.2 (pp. 5-8) of Flury and Riedwyl (1988). There are six measurements made on 200 Swiss banknotes (the old-Swiss 1000-franc). The banknotes belong to two classes of equal size: *genuine* and *counterfeit*.

Usage

```
data(banknote)
```

Format

A `data.frame` of dimension `200x7` with the following variables:

Class a `factor` with classes: `genuine`, `counterfeit`

Length Length of bill (mm)

Left Width of left edge (mm)

Right Width of right edge (mm)

Bottom Bottom margin width (mm)

Top Top margin width (mm)

Diagonal Length of diagonal (mm)

Source

Flury, B. and Riedwyl, H. (1988). *Multivariate Statistics: A practical approach.* London: Chapman & Hall.

`generator.otrimle` *Generates random data from OTRIMLE output model*

Description

This uses data and the output of `otrimle` or `rimle` to generate a new artificial dataset of the size of the original data using noise and cluster proportions from the clustering output. The clusters are then generated from multivariate normal distributions with the parameters estimated by `otrimle`, the noise is generated resampling from what is estimated as moise component with weights given by posterior probabilities of all observations to be noise. See Hennig and Coretto (2021).

Usage

```
generator.otrimle(data, fit)
```

Arguments

`data`	something that can be coerced into a matrix. Dataset.
`fit`	output object of `otrimle` or `rimle`.

Value

A list with components `data, clustering`.

`data`	matrix of generated data.
`cs`	vector of integers. Clustering indicator.

Author(s)

Christian Hennig <christian.hennig@unibo.it> https://www.unibo.it/sitoweb/christian.hennig/en/

References

Hennig, C. and P.Coretto (2021). An adequacy approach for deciding the number of clusters for OTRIMLE robust Gaussian mixture based clustering. To appear in *Australian and New Zealand Journal of Statistics*, https://arxiv.org/abs/2009.00921.

See Also

`kerndensp`, `kerndensmeasure`, `otrimle`, `rimle`

Examples

```
data(banknote)
selectdata <- c(1:30,101:110,117:136,160:161)
set.seed(555566)
x <- banknote[selectdata,5:7]
ox <- otrimle(x, G=2 , ncores = 1)
str(generator.otrimle(x, ox))
```

`InitClust` *Robust Initialization for Model-based Clustering Methods*

Description

Computes the initial cluster assignment based on a combination of nearest neighbor based noise detection, and agglomerative hierarchical clustering based on maximum likelihood criteria for Gaussian mixture models.

Usage

```
InitClust(data , G , k = 3 , knnd.trim = 0.5 , modelName='VVV')
```

Arguments

`data`	A numeric vector, matrix, or data frame of observations. Rows correspond to observations and columns correspond to variables. Categorical variables and `NA` values are not allowed.
`G`	An integer specifying the number of clusters.
`k`	An integer specifying the number of considered nearest neighbors per point used for the denoising step (see *Details*).
`knnd.trim`	A number in [0,1) which defines the proportion of points initialized as noise. Tipically `knnd.trim<=0.5` (see *Details*).
`modelName`	A character string indicating the covariance model to be used. Possible models are: `"E"`: equal variance (one-dimensional) `"V"` : spherical, variable variance (one-dimensional) `"EII"`: spherical, equal volume `"VII"`: spherical, unequal volume `"EEE"`: ellipsoidal, equal volume, shape, and orientation `"VVV"`: ellipsoidal, varying volume, shape, and orientation (default). See *Details*.

Details

The initialization is based on Coretto and Hennig (2017). First, wwo steps are performed:

Step 1 (denoising step): for each data point compute its kth-nearest neighbors distance (k-NND). All points with k-NND larger than the (1-knnd.trim)-quantile of the k-NND are initialized as noise. Intepretaion of k is that: (k-1), but not k, points close together may still be interpreted as noise or outliers

Step 2 (clustering step): perform the model-based hierarchical clustering (MBHC) proposed in Fraley (1998). This step is performed using hc. The input argument modelName is passed to hc. See *Details* of hc for more details.

If the previous *Step 2* fails to provide G clusters each containing at least 2 distinct data points, it is replaced with classical hirararchical clustering implemented in hclust. Finally, if hclust fails to provide a valid partition, up to ten random partitions are tried.

Value

An integer vector specifying the initial cluster assignment with 0 denoting noise/outliers.

References

Fraley, C. (1998). Algorithms for model-based Gaussian hierarchical clustering. *SIAM Journal on Scientific Computing* 20:270-281.

P. Coretto and C. Hennig (2017). Consistency, breakdown robustness, and algorithms for robust improper maximum likelihood clustering. *Journal of Machine Learning Research*, Vol. 18(142), pp. 1-39. https://jmlr.org/papers/v18/16-382.html

Author(s)

Pietro Coretto <pcoretto@unisa.it> https://pietro-coretto.github.io

See Also

hc

Examples

```
## Load  Swiss banknotes data
data(banknote)
x <- banknote[,-1]

## Initial clusters with default arguments
init <- InitClust(data = x, G = 2)
print(init)

## Perform otrimle
a <- otrimle(data = x, G = 2, initial = init,
             logicd = c(-Inf, -50, -10), ncores = 1)
plot(a, what="clustering", data=x)
```

kerndenscluster *Aggregated distance to elliptical unimodal density over clusters*

Description

This calls kerndensp for computing and aggregating density- and principal components-based distances between multivariate data and a unimodal elliptical distribution about the data mean for all clusters in a mixture-based clustering as generated by otrimle or rimle. For use in otrimleg.

Usage

```
kerndenscluster(x,fit,maxq=qnorm(0.9995),kernn=100)
```

Arguments

x	something that can be coerced into a matrix. Dataset.
fit	output object of otrimle or rimle.
maxq	positive numeric. One-dimensional densities are evaluated between mean(x)-maxq and mean(x)+maxq.
kernn	integer. Number of points at which the one-dimensional density is evaluated, input parameter n of density. This should be even.

Details

See Hennig and Coretto (2021), Sec. 4.2. kerndenscluster calls kerndensp for all clusters and aggregates the resulting measures as root sum of squares.

Value

A list with components ddpi, ddpm, measure.

ddpi	list of outputs of kerndensp for all clusters.
ddpm	vector of measure-components of ddpi.
measure	Final aggregation result.

Author(s)

Christian Hennig <christian.hennig@unibo.it> https://www.unibo.it/sitoweb/christian.hennig/en/

References

Hennig, C. and P.Coretto (2021). An adequacy approach for deciding the number of clusters for OTRIMLE robust Gaussian mixture based clustering. To appear in *Australian and New Zealand Journal of Statistics*, https://arxiv.org/abs/2009.00921.

See Also

kerndensp, kerndensmeasure, otrimle, rimle

Examples

```
data(banknote)
selectdata <- c(1:30,101:110,117:136,160:161)
set.seed(555566)
x <- banknote[selectdata,5:7]
ox <- otrimle(x, G=2, ncores=1)
kerndenscluster(x,ox)$measure
```

---

kerndensmeasure *Statistic measuring closeness to symmetric unimodal distribution*

Description

Density-based distance between one-dimensional data and a unimodal symmetric distribution about the data mean based on Pons (2013, p.79), adapted by Hennig and Coretto (2021), see details.

Usage

```
kerndensmeasure(x,weights=rep(1,nrow(as.matrix(x))),maxq=qnorm(0.9995),
                kernn=100)
```

Arguments

x	vector. One-dimensional dataset.
weights	non-negative vector. Relative weights of observations (will be standardised to sup up to one internally).
maxq	densities are evaluated between mean(x)-maxq and mean(x)+maxq.
kernn	integer. Number of points at which the density is evaluated, input parameter n of density. This should be even.

Details

Function density is used in order to compute a kernel density estimator from the data. The kernn values of the density are then ordered from the pargest to the smallest. Beginning from the largest to the smallest, pairs of two values are formed (largest and largest biggest, third and fourth largest, and so on). Each pair is replaced by two copies of the average of the two values. Then on each side of the mean one of each copy is placed from the biggest to the smallest, and this produces a symmetric density about the mean. The the root mean squared difference between this and the original density is computed.

Value

A list with components cp, cpx, measure.

cp	vector of generated symmetric density values from largest to smallest (just one copy, sp kernn/2 values).
cpx	y-component of density-output.
measure	root mean squared difference between the densities.

Author(s)

Christian Hennig <christian.hennig@unibo.it> https://www.unibo.it/sitoweb/christian.hennig/en/

References

Hennig, C. and P.Coretto (2021). An adequacy approach for deciding the number of clusters for OTRIMLE robust Gaussian mixture based clustering. To appear in *Australian and New Zealand Journal of Statistics*, https://arxiv.org/abs/2009.00921.

Pons, O. (2013). *Statistical Tests of Nonparametric Hypotheses: Asymptotic Theory*. World Scientific, Singapore.

See Also

kerndensp

Examples

```
set.seed(124578)
x <- runif(20)
str(kerndensmeasure(x))
```

---

kerndensp *Closeness of multivariate distribution to elliptical unimodal distribution*

Description

Density- and and principal components-based distance between multivariate data and a unimodal elliptical distribution about the data mean, see Hennig and Coretto (2021). For use in kerndenscluster.

Usage

```
kerndensp(x,weights=rep(1,nrow(as.matrix(x))), siglist,maxq=qnorm(0.9995),
          kernn=100)
```

Arguments

x	something that can be coerced into a matrix. Dataset.
weights	non-negative vector. Relative weights of observations (will be standardised to sup up to one internally).
siglist	list with components cov (covariance matrix), center (mean) and n.obs (number of observations).
maxq	positive numeric. One-dimensional densities are evaluated between mean(x)-maxq and mean(x)+maxq.
kernn	integer. Number of points at which the one-dimensional density is evaluated, input parameter n of density. This should be even.

Details

See Hennig and Coretto (2021), Sec. 4.2. kerndensmeasure is run on the principal components of x. The resulting measures are standardised by kmeanfun and ksdfun and then aggregated as mean square of the positive values, see Hennig and Coretto (2021). The PCS is computed by princomp and will always use siglist rather than statistics computed from x.

Value

A list with components cml, cm, pca, stanmeasure, measure.

cml	List of outputs of kerndensmeasure for the principal components.
cm	vector of measure components of kerndensmeasure for the principal components.
stanmeasure	vector of standardised measure components of kerndensmeasure for the principal components.
pca	output of princomp.
measure	Final aggregation result.

Author(s)

Christian Hennig <christian.hennig@unibo.it> https://www.unibo.it/sitoweb/christian.hennig/en/

References

Hennig, C. and P.Coretto (2021). An adequacy approach for deciding the number of clusters for OTRIMLE robust Gaussian mixture based clustering. To appear in *Australian and New Zealand Journal of Statistics*, https://arxiv.org/abs/2009.00921.

See Also

kerndensmeasure, kerndenscluster

Examples

```
set.seed(124578)
x <- cbind(runif(20),runif(20))
siglist <- list(cov=cov(x),center=colMeans(x),n.obs=20)
kerndensp(x,siglist=siglist)$measure
```

---

kmeanfun *Mean and standard deviation of unimodality statistic*

Description

These functions approximate the mean and standard deviation of the unimodality statistic computed by kerndensmeasure assuming standard Gaussian data dependent on the number of observationsn. They have been chosen based on a simulation involving 74 different values of n. Used for standardisation in kerndensp.

Usage

```
kmeanfun(n)
ksdfun(n)
```

Arguments

n	integer. Number of observations.

Value

The resulting mean (kmeanfun) or standard deviation (ksdfun).

Author(s)

Christian Hennig <christian.hennig@unibo.it> https://www.unibo.it/sitoweb/christian.hennig/en/

See Also

kerndensp

Examples

```
kmeanfun(50)
ksdfun(50)
```

otrimle *Optimally Tuned Robust Improper Maximum Likelihood Clustering*

Description

otrimle searches for G approximately Gaussian-shaped clusters with/without noise/outliers. The method's tuning controlling the noise level is adaptively chosen based on the data.

Usage

```
otrimle(data, G, initial = NULL, logicd = NULL, npr.max = 0.5, erc = 20,
beta = 0, iter.max = 500, tol = 1e-06, ncores = NULL, monitor = TRUE)
```

Arguments

data	A numeric vector, matrix, or data frame of observations. Rows correspond to observations and columns correspond to variables. Categorical variables and NA values are not allowed.
G	An integer specifying the number of clusters.
initial	An integer vector specifying the initial cluster assignment with 0 denoting noise/outliers. If NULL (default) initialization is performed using InitClust.
logicd	A vector defining a grid of *log(icd)* values, where *icd* denotes the improper constant density. If logicd=NULL a default grid is considered. A pure Gaussian Mixture Model fit (obtained when log(icd)=-Inf) is included in the default search path.
npr.max	A number in [0,1) specifying the maximum proportion of noise/outliers. This defines the *noise proportion constraint*. If npr.max=0 a single solution without noise component is computed (corresponding to logicd = -Inf.
erc	A number >=1 specifying the maximum allowed ratio between within-cluster covariance matrix eigenvalues. This defines the *eigenratio constraint*. erc=1 enforces spherical clusters with equal covariance matrices. A large erc allows for large between-cluster covariance discrepancies. In order to facilitate the setting of erc, it is suggested to scale the columns of data (see scale) whenever measurement units of the different variables are grossly incompatible.
beta	A non-negative constant. This is the *beta* penalty coefficient introduced in Coretto and Hennig (2016).
iter.max	An integer value specifying the maximum number of iterations allowed in the underlying ECM-algorithm.
tol	Stopping criterion for the underlying ECM-algorithm. An ECM iteration stops if two successive improper log-likelihood values are within tol.
ncores	an integer value defining the number of cores used for parallel computing. When ncores=NULL (default), the number r of available cores is detected, and (r-1) of them are used (See *Details*). If ncores=1 no parallel backend is started.
monitor	logical. If TRUE progress messages are printed on screen.

Details

The otrimle function computes the OTRIMLE solution based on the ECM-algorithm (expectation conditional maximization algorithm) proposed in Coretto and Hennig (2017).

The otrimle criterion is minimized over the logicd grid of log(icd) values using parallel computing based on the foreach. Note that, depending on the BLAS/LAPACK setting, increasing ncores may not produce the desired reduction in computing time. The latter happens when optimized linear algebra routines are in use (e.g. OpenBLAS, Intel Math Kernel Library (MKL), etc.). These optimized shared libraries already implement multithreading. Therefore, in this case increasing ncores may only reduce the computing time marginally.

Occasionally, there may be datasets for which the function does not provide a solution based on default arguments. This corresponds to code=0 and flag=1 or flag=2 in the output (see *Value*-section below). This usually happens when some (or all) of the following circumstances occur: (i) erc is too large; (ii) npr.max is too large; (iii) choice of the initial partition. Regarding (i) and (ii) it is not possible to give numeric references because whether these numbers are too large/small strongly depends on the sample size and the dimensionality of the data. References given below explain the relationship between these quantities.

It is suggested to try the following whenever a code=0 non-solution occurs. Set the logicd range wide enough (e.g. logicd=seq(-500,-5, length=50)), choose erc=1, and a low choice of npr.max (e.g. npr.max=0.02). Monitor the solution with the criterion profiling plot (plot.otrimle). According to the criterion profiling plot change logicd, and increase erc and npr.max up to the point when a "clear" minimum in the criterion profiling plot is obtained. If this strategy does not work it is suggested to experiment with a different initial partitions (see initial above).

TBA: Christian may add something about the beta here.

The pi object returned by the rimle function (see *Value*) corresponds to the vector of pi parameters in the underlying pseudo-model (1) defined in Coretto and Hennig (2017). With logicd = -Inf the rimle function approximates the MLE for the *plain* Gaussian mixture model with eigenratio covariance regularization, in this case the the first element of the pi vector is set to zero because the noise component is not considered. In general, for iid sampling from finite mixture models context, these *pi* parameters define expected clusters' proportions. Because of the noise proportion constraint in the RIMLE, there are situations where this connection may not happen in practice. The latter is likely to happen when both logicd and npr.max are large. Therefore, estimated expected clusters' proportions are reported in the exproportion object of the rimle output, and these are computed based on the improper posterior probabilities given in tau. See Coretto and Hennig (2017) for more discussion on this.

An earlier approximate version of the algorithm was originally proposed in Coretto and Hennig (2016). Software for the original version of the algorithm can be found in the supplementary materials of Coretto and Hennig (2016).

Value

An S3 object of class 'otrimle' providing the optimal (according to the OTRIMLE criterion) clustering. Output components are as follows:

code — An integer indicator for the convergence. code=0 if no solution is found (see *Details*); code=1 if at the optimal icd value the corresponding EM-algorithm did not converge within em.iter.max; code=2 convergence is fully achieved.

`flag`	A character string containing one or more flags related to the EM iteration at the optimal icd. `flag=1` if it was not possible to prevent the numerical degeneracy of improper posterior probabilities (`tau` value below). `flag=2` if enforcement of the *noise proportion constraint* failed for numerical reasons. `flag=3` if the *noise proportion constraint* has been successfully applied at least once. `flag=4` if the *eigenratio constraint* has been successfully applied at least once.
`iter`	Number of iterations performed in the underlying EM-algorithm at the optimal `icd`.
`logicd`	Resulting value of the optimal `log(icd)`.
`iloglik`	Resulting value of the improper likelihood.
`criterion`	Resulting value of the OTRIMLE criterion.
`pi`	Estimated vector of the `pi` parameters of the underlying pseudo-model (see *Details*).
`mean`	A matrix of dimension `ncol(data) x G` containing the mean parameters of each cluster (column-wise).
`cov`	An array of size `ncol(data) x ncol(data) x G` containing the covariance matrices of each cluster.
`tau`	A matrix of dimension `nrow(data) x {1+G}` where `tau[i, 1+j]` is the estimated (improper) posterior probability that the *i*th observation belongs to the *j*th cluster. `tau[i,1]` is the estimated (improper) posterior probability that *i*th observation belongs to the noise component.
`smd`	A matrix of dimension `nrow(data) x G` where `smd[i,j]` is the squared Mahalanobis distance of `data[i,]` from `mean[,j]` according to `cov[,,j]`.
`cluster`	A vector of integers denoting cluster assignments for each observation. It's `0` for observations assigned to noise/outliers.
`size`	A vector of integers with sizes (counts) of each cluster.
`exproportion`	A vector of estimated expected clusters' proportions (see *Details*).
`optimization`	A data.frame with the OTRIMLE optimization profiling. For each value of `log(icd)` explored by the algorithm the data.frame stores `logicd`, `criterion`, `iloglik`, `code`, `flag` (defined above), and `enpr` that denotes the expected noise proportion.

References

Coretto, P. and C. Hennig (2016). Robust improper maximum likelihood: tuning, computation, and a comparison with other methods for robust Gaussian clustering. *Journal of the American Statistical Association*, Vol. 111(516), pp. 1648-1659. doi: 10.1080/01621459.2015.1100996

P. Coretto and C. Hennig (2017). Consistency, breakdown robustness, and algorithms for robust improper maximum likelihood clustering. *Journal of Machine Learning Research*, Vol. 18(142), pp. 1-39. https://jmlr.org/papers/v18/16-382.html

Author(s)

Pietro Coretto <pcoretto@unisa.it> https://pietro-coretto.github.io

See Also

plot.otrimle, InitClust, rimle,

Examples

```
## Load  Swiss banknotes data
data(banknote)
x <- banknote[,-1]

## Perform otrimle clustering with default arguments
set.seed(1)
a <- otrimle(data=x, G=2, logicd=c(-Inf, -50, -10), ncores=1)

## Plot clustering
plot(a, data=x, what="clustering")

## Plot OTRIMLE criterion profiling
plot(a, what="criterion")

## Plot Improper log-likelihood profiling
plot(a, what="iloglik")

## P-P plot of the clusterwise empirical weighted squared Mahalanobis
## distances against the target distribution pchisq(, df=ncol(data))
plot(a, what="fit")
plot(a, what="fit", cluster=1)



## Perform the same otrimle as before with non-zero penalty
set.seed(1)
b <- otrimle(data=x, G=2, beta = 0.5, logicd=c(-Inf, -50, -10), ncores=1)

## Plot clustering
plot(b, data=x, what="clustering")

## Plot OTRIMLE criterion profiling
plot(b, what="criterion")

## Plot Improper log-likelihood profiling
plot(b, what="iloglik")

## P-P plot of the clusterwise empirical weighted squared Mahalanobis
## distances against the target distribution pchisq(, df=ncol(data))
plot(b, what="fit")
plot(b, what="fit", cluster=1)
```

```
## Not run:
## Perform the same example using the finer default grid of logicd
## values using multiple cores
##
a <- otrimle(data = x, G = 2)

## Inspect the otrimle criterion-vs-logicd
plot(a, what = 'criterion')

## The minimum occurs at  a$logicd=-9, and exploring a$optimization it
## cane be seen that the interval [-12.5, -4] brackets the optimal
## solution. We search with a finer grid located around the minimum
##
b <- otrimle(data = x, G = 2, logicd = seq(-12.5, -4, length.out = 25))

## Inspect the otrimle criterion-vs-logicd
plot(b, what = 'criterion')

## Check the difference between the two clusterings
table(A = a$cluster, B = b$cluster)

## Check differences in estimated parameters
##
colSums(abs(a$mean - b$mean))               ## L1 distance for mean vectors
apply({a$cov-b$cov}, 3, norm, type = "F")   ## Frobenius distance for covariances
c(Noise=abs(a$npr-b$npr), abs(a$cpr-b$cpr)) ## Absolute difference for proportions

## End(Not run)
```

`otrimleg` *OTRIMLE for a range of numbers of clusters with density-based cluster quality statistic*

Description

Computes Optimally Tuned Robust Improper Maximum Likelihood Clustering (OTRIMLE), see `otrimle`, together with the density-based cluster quality statistics Q (Hennig and Coretto 2021) for a range of values of the number of clusters.

Usage

```
otrimleg(dataset, G=1:6, multicore=TRUE, ncores=detectCores(logical=FALSE)-1,
   erc=20, beta0=0, fixlogicd=NULL, monitor=1, dmaxq=qnorm(0.9995))
```

Arguments

`dataset`	something that can be coerced into an observations times variables matrix. The dataset.
`G`	vector of integers (normally starting from 1). Numbers of clusters to be considered.

multicore	logical. If TRUE, parallel computing is used through the function mclapply from package parallel; read warnings there if you intend to use this; it won't work on Windows.
ncores	integer. Number of cores for parallelisation.
erc	A number larger or equal than one specifying the maximum allowed ratio between within-cluster covariance matrix eigenvalues. See otrimle.
beta0	A non-negative constant, penalty term for noise, to be passed as beta to otrimle, see documentation there.
fixlogicd	numeric of NULL. Value for the logarithm of the improper constant density logicd, see rimle, which is run instead of otrimle if this is not NULL. NULL means that otrimle determines it from the data.
monitor	0 or 1. If 1, progress messages are printed on screen.
dmaxq	numeric. Passed as maxq to kerndensmeasure. The interval considered for the one-dimensional density estimator is (-maxq,maxq).

Details

For estimating the number of clusters this is meant to be called by otrimlesimg. The output of otrimleg is not meant to be used directly for estimating the number of clusters, see Hennig and Coretto (2021).

Value

otrimleg returns a list containing the components solution, iloglik, ibic, criterion, logicd, noiseprob, denscrit, ddpm. All of these are lists or vectors of which the component number is the number of clusters.

solution	list of output objects of otrimle or rimle.
iloglik	vector of improper likelihood values from otrimle.
ibic	vector of improper BIC-values (small is good) computed from iloglik and the numbers of parameters. Note that the behaviour of the improper likelihood is not compatible with the standard use of the BIC, so this is experimental and should not be trusted for choosing the number of clusters.
criterion	vector of values of OTRIMLE criterion, see otrimle.
noiseprob	vector of estimated noise proportions, exproportion[1] from otrimle.
denscrit	vector of density-based cluster quality statistics Q (Hennig and Coretto 2021) as provided by the measure-component of kerndensmeasure.
ddpm	list of the vector of cluster-wise density-based cluster quality measures as provided by the ddpm-component of kerndensmeasure.

Author(s)

Christian Hennig <christian.hennig@unibo.it> https://www.unibo.it/sitoweb/christian.hennig/en/

References

Coretto, P. and C. Hennig (2016). Robust improper maximum likelihood: tuning, computation, and a comparison with other methods for robust Gaussian clustering. *Journal of the American Statistical Association*, Vol. 111(516), pp. 1648-1659. doi: 10.1080/01621459.2015.1100996

P. Coretto and C. Hennig (2017). Consistency, breakdown robustness, and algorithms for robust improper maximum likelihood clustering. *Journal of Machine Learning Research*, Vol. 18(142), pp. 1-39. https://jmlr.org/papers/v18/16-382.html

Hennig, C. and P.Coretto (2021). An adequacy approach for deciding the number of clusters for OTRIMLE robust Gaussian mixture based clustering. To appear in *Australian and New Zealand Journal of Statistics*, https://arxiv.org/abs/2009.00921.

See Also

otrimle, rimle, otrimlesimg, kerndensmeasure

Examples

```
data(banknote)
selectdata <- c(1:30,101:110,117:136,160:161)
x <- banknote[selectdata,5:7]
obanknote <- otrimleg(x,G=1:2,multicore=FALSE)
```

---

otrimlesimg *Adequacy approach for number of clusters for OTRIMLE*

Description

otrimlesimg computes Optimally Tuned Robust Improper Maximum Likelihood Clustering (OTRIMLE), see otrimle for a range of values of the number of clusters, and also for artificial datasets simulated from the model parameters estimated on the original data. The summary-methods present and evaluate the results so that a smallest adequate number of clusters can be found as the smallest one for which the value of the density-based cluster quality statistics Q on the original data is compatible with its distribution on the artificial datasets with the same number of clusters, see Hennig and Coretto 2021 for details.

Usage

```
otrimlesimg(dataset, G=1:6, multicore=TRUE,
ncores=detectCores(logical=FALSE)-1, erc=20, beta0=0, simruns=20,
sim.est.logicd=FALSE,
monitor=1)

## S3 method for class 'otrimlesimgdens'
summary(object, noisepenalty=0.05 , sdcutoff=2
, ...)
```

```
## S3 method for class 'summary.otrimlesimgdens'
print(x, ...)

## S3 method for class 'summary.otrimlesimgdens'
plot(x , plot="criterion", penx=NULL,
peny=NULL, pencex=1, cutoff=TRUE, ylim=NULL, ...)
```

Arguments

dataset	something that can be coerced into an observations times variables matrix. The dataset.
G	vector of integers (normally starting from 1). Numbers of clusters to be considered.
multicore	logical. If TRUE, parallel computing is used through the function mclapply from package parallel; read warnings there if you intend to use this; it won't work on Windows.
ncores	integer. Number of cores for parallelisation.
erc	A number larger or equal than one specifying the maximum allowed ratio between within-cluster covariance matrix eigenvalues. See otrimle.
beta0	A non-negative constant, penalty term for noise, to be passed as beta to otrimle, see documentation there.
simruns	integer. Number of replicate artificial datasets drawn from each model.
sim.est.logicd	logical. If TRUE, the logarithm of the improper constant density logicd, see otrimle, is re-estimated when running otrimle on the artificial datasets. Otherwise the value estimated on the original data is taken as fixed. TRUE requires much longer computation time, but can be seen as generating more realistic variation.
monitor	0 or 1. If 1, progress messages are printed on screen.
noisepenalty	number between 0 and 1. p_0 in Hennig and Coretto (2021); normally small. The method prefers to treat a proportion of <=noisepenalty of points as outliers to adding a cluster.
sdcutoff	numerical. c in formula (7) in Hennig and Coretto (2021). A clustering is treated as adequate if its value of the density-based cluster quality measure Q calibrated (i.e., mean/sd-standardised) by the values on the artificial datasets generated from the estimated model is <=sdcutoff.
plot	"criterion" or "noise", see details.
penx	FALSE, NULL, or numerical. x-coordinate from where the simplicity ordering of clustering is given (as test in the plot). If FALSE, this is not added to the plot. If NULL a default guess is made for a good position (which doesn't always work well).
peny	NULL, or numerical. x-coordinate from where the simplicity ordering of clustering is given (as test in the plot). If NULL, a default guess is made for a good position (which doesn't always work well).

pencex	numeric. Magnification factor (parameter cex to be passed on to legend) for simplicity ordering, see parameter penx.
cutoff	logical. If TRUE, the "criterion"-plot shows the cutoff value below which numbers of clusters are adequate, see details.
ylim	vector of two numericals, range of the y-axis to be passed on to plot. If NULL, the range is chosen automatically (but can be different from the plot default).
object	an object of class 'otrimlesimgdens' obtained from calling otrimlesimg
x	an object of class 'summary.otrimlesimgdens' obtained from calling summary function over an object of class 'otrimlesimgdens' obtained from calling otrimlesimg.
...	optional parameters to be passed on to plot.

Details

The method is fully described in Hennig and Coretto (2021). The required tuning constants for choosing an optimal number of clusters, the smallest percentage of additional noise that the user is willing to trade in for adding another cluster (p_0 in the paper, noisepenalty here) and the critical value (c in the paper, sdcutoff here) for adequacy of the standardised density based quality measure Q are provided to the summary function, which is required to choose the best (simplest adequate) number of clusters.

The plot function plot.summary.otrimlesimgdens can produce two plots. If plot="criterion", the standardised density-based cluster quality measure Q is plotted against the number of clusters. The values for the simulated artificial datasets are points, the values for the original dataset are given as line type. If cutoff="TRUE", the critical values (see above) are added as red crosses; a number of clusters is adequate if the value of the original data is below the critical value, i.e., Q is not significantly larger than for the artificial datasets generated from the fitted model. Using penx, the ordered numbers of clusters from the simplest to the least simple can also be indicated in the plot, where simplicitly is defined as the number of clusters plus the estimated noise proportion divided by noisepenalty, see above. The chosen number of clusters is the simplest adequate one, meaning that a low number of clusters and a low noise proportion are preferred.

If plot="noise", the noise proportion (black) and the simplicity (red) are plotted against the numnber of clusters.

Value

otrimlesimg returns a list of type "otrimlesimgdens" containing the components result, simresult, simruns.

result	output object of otrimleg (list of results on original data) run with the parameters provided to otrimlesimg.
simresult	list of length simruns of output objects of otrimleg for all the simulated artificial datasets.
simruns	input parameter simruns.

summary.otrimlesimgdens returns a list of type "summary.otrimlesimgdens" with components G, simeval, ssimruns, npr, nprdiff, logicd, denscrit, peng, penorder, bestG, sdcutoff, bestresult, cluster. simruns

G	otrmlesimg input parameter G (numbers of clusters).
simeval	list with components denscrit, meandens, sddens, standens, errors, defined below.
ssimruns	otrmlesimg input parameter simruns.
npr	vector of estimated noise proportions on the original data for all numbers of clusters, exproportion[1] from otrimle.
nprdiff	vector for all numbers of clusters of differences between estimated smallest cluster proportion and noise proportion on the original data.
logicd	vector of logs of improper constant density values on the original data for all numbers of clusters.
denscrit	vector over all numbers of clusters of density-based cluster quality statistics Q on original data as provided by the measure-component of kerndensmeasure.
peng	vector of simplicity values (see Details) over all numbers of clusters.
penorder	simplicity order of number of clusters.
bestG	best (i.e., most simple adequate) number of clusters.
sdcutoff	input parameter sdcutoff.
result	output of otrimle for the best number of clusters bestG.
cluster	clustering vector for the best number of clusters bestG. 0 corresponds to noise/outliers.

Components of summary.otrimlesimgdens output component simeval:

denscritmatrix	maximum number of clusters times simruns matrix of denscrit-vectors for all clusterings on simulated data.
meandens	vector over numbers of clusters of robust estimator of the mean of denscrit over simulated datasets, computed by scaleTau2.
sddens	vector over numbers of clusters of robust estimator of the standard deviation of denscrit over simulated datasets, computed by scaleTau2.
standens	vector over numbers of clusters of denscrit of the original data standardised by meandens and sddens.
errors	vector over numbers of clusters of numbers of times that otrimle led to an error.
	plot.summary.otrimlesimgdens will return the output of par() before anything was changed by the plot function.

Author(s)

Christian Hennig <christian.hennig@unibo.it> https://www.unibo.it/sitoweb/christian.hennig/en/

References

Coretto, P. and C. Hennig (2016). Robust improper maximum likelihood: tuning, computation, and a comparison with other methods for robust Gaussian clustering. *Journal of the American Statistical Association*, Vol. 111(516), pp. 1648-1659. doi: 10.1080/01621459.2015.1100996

P. Coretto and C. Hennig (2017). Consistency, breakdown robustness, and algorithms for robust improper maximum likelihood clustering. *Journal of Machine Learning Research*, Vol. 18(142), pp. 1-39. https://jmlr.org/papers/v18/16-382.html

Hennig, C. and P.Coretto (2021). An adequacy approach for deciding the number of clusters for OTRIMLE robust Gaussian mixture based clustering. To appear in *Australian and New Zealand Journal of Statistics*, https://arxiv.org/abs/2009.00921.

See Also

otrimle, rimle, otrimleg, kerndensmeasure

Examples

```
## otrimlesimg is computer intensive, so only a small data subset
## is used for speed.
data(banknote)
selectdata <- c(1:30,101:110,117:136,160:161)
set.seed(555566)
x <- banknote[selectdata,5:7]

## simruns=2 chosen for speed. This is not recommended in practice.
obanknote <- otrimlesimg(x,G=1:2,multicore=FALSE,simruns=2,monitor=0)
sobanknote <- summary(obanknote)
print(sobanknote)
plot(sobanknote,plot="criterion",penx=1.4)
plot(sobanknote,plot="noise",penx=1.4)
plot(x,col=sobanknote$cluster+1,pch=c("N","1","2")[sobanknote$cluster+1])
```

---

plot.otrimle *Plot Methods for OTRIMLE Objects*

Description

Plot robust model-based clustering results: scatter plot with clustering information, optimization profiling, and cluster fit.

Usage

```
## S3 method for class 'otrimle'
plot(x, what=c("criterion","iloglik", "fit", "clustering"),
     data=NULL, margins=NULL, cluster=NULL, ...)
```

Arguments

x	Output from otrimle
what	The type of graph. It can be one of the following: "criterion" (default), "iloglik", "fit", "clustering". See *Details*.
data	The data vector, matrix or data.frame (or some transformation of them), used for obtaining the 'otrimle' object. This is only relevant if what="clustering".
margins	A vector of integers denoting the variables (numbers of columns of data) to be used for a pairs-plot if what="clustering". When margins=NULL it is set to 1:ncol(data) (default).
cluster	An integer denoting the cluster for which the *fit* plot is returned. This is only relevant if what="fit".
...	further arguments passed to or from other methods.

Value

If what="criterion" A plot with the profiling of the OTRIMLE criterion optimization. Criterion at log(icd)=-Inf is always represented.

If what="iloglik" A plot with the profiling of the improper log-likelihood function along the search path for the OTRIMLE optimization.

If what="fit" The P-P plot (probability-probability plot) of the weighted empirical distribution function of the Mahalanobis distances of observations from clusters' centers against the target distribution. The target distribution is the Chi-square distribution with degrees of freedom equal to ncol(data). The weights are given by the improper posterior probabilities. If cluster=NULL P-P plots are produced for all clusters, otherwise cluster selects a single P-P plot at times.

If what="clustering" A pairwise scatterplot with cluster memberships. Points assigned to the noise/outliers component are denoted by '+'.

References

Coretto, P. and C. Hennig (2016). Robust improper maximum likelihood: tuning, computation, and a comparison with other methods for robust Gaussian clustering. *Journal of the American Statistical Association*, Vol. 111(516), pp. 1648-1659. doi: 10.1080/01621459.2015.1100996

P. Coretto and C. Hennig (2017). Consistency, breakdown robustness, and algorithms for robust improper maximum likelihood clustering. *Journal of Machine Learning Research*, Vol. 18(142), pp. 1-39. https://jmlr.org/papers/v18/16-382.html

Author(s)

Pietro Coretto <pcoretto@unisa.it> https://pietro-coretto.github.io

See Also

plot.otrimle

Examples

```
## Load  Swiss banknotes data
data(banknote)
x <- banknote[,-1]

## Perform otrimle clustering on a small grid of logicd values
a <- otrimle(data = x, G = 2, logicd = c(-Inf, -50, -10), ncores = 1)
print(a)

## Plot clustering
plot(a, data = x, what = "clustering")

## Plot clustering on selected margins
plot(a, data = x, what = "clustering", margins = 4:6)

## Plot clustering on the first two principal components
z <- scale(x) %*%   eigen(cor(x), symmetric = TRUE)$vectors
colnames(z) <- paste("PC", 1:ncol(z), sep = "")
plot(a, data = z, what = "clustering", margins = 1:2)

## Plot OTRIMLE criterion profiling
plot(a, what = "criterion")

## Plot Improper log-likelihood profiling
plot(a, what = "iloglik")

## Fit plot for all clusters
plot(a, what = "fit")

## Fit plot for cluster 1
plot(a, what = "fit", cluster = 1)



## Not run:
## Perform the same example using the finer default grid of logicd
## values using multiple cores
##
a <- otrimle(data = x, G = 2)

## Inspect the otrimle criterion-vs-logicd
plot(a, what = 'criterion')

## The minimum occurs at  a$logicd=-9, and exploring a$optimization it
## cane be seen that the interval [-12.5, -4] brackets the optimal
## solution. We search with a finer grid located around the minimum
##
b <- otrimle(data = x, G = 2, logicd = seq(-12.5, -4, length.out = 25))

## Inspect the otrimle criterion-vs-logicd
plot(b, what = 'criterion')
```

```
## Check the difference between the two clusterings
table(A = a$cluster, B = b$cluster)

## Check differences in estimated parameters
##
colSums(abs(a$mean - b$mean))              ## L1 distance for mean vectors
apply({a$cov-b$cov}, 3, norm, type = "F")   ## Frobenius distance for covariances
c(Noise=abs(a$npr-b$npr), abs(a$cpr-b$cpr)) ## Absolute difference for proportions

## End(Not run)
```

`plot.rimle` *Plot Methods for RIMLE Objects*

Description

Plot robust model-based clustering results: scatter plot with clustering information and cluster fit.

Usage

```
## S3 method for class 'rimle'
plot(x, what=c("fit", "clustering"),
     data=NULL, margins=NULL, cluster=NULL, ...)
```

Arguments

`x`	Output from `rimle`
`what`	The type of graph. It can be one of the following: `"fit"` (default), `"clustering"`. See *Details*.
`data`	The data vector, matrix or data.frame (or some transformation of them), used for obtaining the `'rimle'` object. This is only relevant if `what="clustering"`.
`margins`	A vector of integers denoting the variables (numbers of columns of `data`) to be used for a `pairs`-plot if `what="clustering"`. When `margins=NULL` it is set to `1:ncol(data)` (default).
`cluster`	An integer denoting the cluster for which the *fit* plot is returned. This is only relevant if `what="fit"`.
`...`	further arguments passed to or from other methods.

Value

If `what="fit"` The P-P plot (probability-probability plot) of the weighted empirical distribution function of the Mahalanobis distances of observations from clusters' centers against the target distribution. The target distribution is the Chi-square distribution with degrees of freedom equal to `ncol(data)`. The weights are given by the improper posterior probabilities. If `cluster=NULL` P-P plots are produced for all clusters, otherwise `cluster` selects a single P-P plot at times.

If `what="clustering"` A pairwise scatterplot with cluster memberships. Points assigned to the noise/outliers component are denoted by `'+'`.

References

Coretto, P. and C. Hennig (2016). Robust improper maximum likelihood: tuning, computation, and a comparison with other methods for robust Gaussian clustering. *Journal of the American Statistical Association*, Vol. 111(516), pp. 1648-1659. doi: 10.1080/01621459.2015.1100996

P. Coretto and C. Hennig (2017). Consistency, breakdown robustness, and algorithms for robust improper maximum likelihood clustering. *Journal of Machine Learning Research*, Vol. 18(142), pp. 1-39. https://jmlr.org/papers/v18/16-382.html

Author(s)

Pietro Coretto <pcoretto@unisa.it> https://pietro-coretto.github.io

See Also

otrimle

Examples

```
## Load  Swiss banknotes data
data(banknote)
x <- banknote[,-1]

## Perform rimle clustering with default arguments
set.seed(1)
a <- rimle(data = x, G = 2)
print(a)

## Plot clustering
plot(a, data = x, what = "clustering")

## Plot clustering on selected margins
plot(a, data = x, what = "clustering", margins = 4:6)

## Plot clustering on the first two principal components
z <- scale(x) %*%   eigen(cor(x), symmetric = TRUE)$vectors
colnames(z) <- paste("PC", 1:ncol(z), sep = "")
plot(a, data = z, what = "clustering", margins = 1:2)

## Fit plot for all clusters
plot(a, what = "fit")

## Fit plot for cluster 1
plot(a, what = "fit", cluster = 1)
```

---

rimle *Robust Improper Maximum Likelihood Clustering*

Description

rimle searches for G approximately Gaussian-shaped clusters with/without noise/outliers. The method's tuning controlling the noise level is fixed and is to be provided by the user or will be guessed by the function in a rather quick and dirty way (otrimle performs a more sophisticated data-driven choice).

Usage

```
rimle(data, G, initial=NULL, logicd=NULL, npr.max=0.5, erc=20, iter.max=500, tol=1e-6)
```

Arguments

data	A numeric vector, matrix, or data frame of observations. Rows correspond to observations and columns correspond to variables. Categorical variables and NA values are not allowed.
G	An integer specifying the number of clusters.
initial	An integer vector specifying the initial cluster assignment with 0 denoting noise/outliers. If NULL (default) initialization is performed using InitClust.
logicd	A number log(icd), where 0 <= icd < Inf is the value of the improper constant density (icd). This is the RIMLE's tuning for controlling the size of the noise. If logicd=NULL (default), an icd value is guessed based on the data. A pure Gaussian Mixture Model fit is obtained with logicd = -Inf.
npr.max	A number in [0,1) specifying the maximum proportion of noise/outliers. This defines the *noise proportion constraint*. If npr.max=0 a solution without noise component is computed (corresponding to logicd = -Inf.
erc	A number >=1 specifying the maximum allowed ratio between within-cluster covariance matrix eigenvalues. This defines the *eigenratio constraint*. erc=1 enforces spherical clusters with equal covariance matrices. A large erc allows for large between-cluster covariance discrepancies. In order to facilitate the setting of erc, it is suggested to scale the columns of data (see scale) whenever measurement units of the different variables are grossly incompatible.
iter.max	An integer value specifying the maximum number of iterations allowed in the ECM-algorithm (see Details).
tol	Stopping criterion for the underlying ECM-algorithm. An ECM iteration stops if two successive improper log-likelihood values are within tol.

Details

The rimle function computes the RIMLE solution using the ECM-algorithm proposed in Coretto and Hennig (2017).

There may be datasets for which the function does not provide a solution based on default arguments. This corresponds to code=0 and flag=1 or flag=2 in the output (see *Value*-section below). This usually happens when some (or all) of the following circumstances occur: (i) log(icd) is too large; (ii) erc is too large; (iii) npr.max is too large; (iv) choice of the initial partition. In these cases it is suggested to find a suitable interval of icd values by using the otrimle function. The *Details* section of otrimle suggests several actions to take whenever a code=0 non-solution occurs.

The pi object returned by the rimle function (see *Value*) corresponds to the vector of pi parameters in the underlying pseudo-model (1) defined in Coretto and Hennig (2017). With logicd = -Inf the rimle function approximates the MLE for the *plain* Gaussian mixture model with eigenratio covariance regularization, in this case the the first element of the pi vector is set to zero because the noise component is not considered. In general, for iid sampling from finite mixture models context, these *pi* parameters define expected clusters' proportions. Because of the noise proportion constraint in the RIMLE, there are situations where this connection may not happen in practice. The latter is likely to happen when both logicd and npr.max are large. Therefore, estimated expected clusters' proportions are reported in the exproportion object of the rimle output, and these are computed based on the improper posterior probabilities given in tau. See Coretto and Hennig (2017) for more discussion on this.

An earlier approximate version of the algorithm was originally proposed in Coretto and Hennig (2016). Software for the original version of the algorithm can be found in the supplementary materials of Coretto and Hennig (2016).

Value

An S3 object of class 'rimle'. Output components are as follows:

code	An integer indicator for the convergence. code=0 if no solution is found (see *Details*); code=1 if the EM-algorithm did not converge within em.iter.max; code=2 convergence is fully achieved.
flag	A character string containing one or more flags related to the EM iteration at the optimal icd. flag=1 if it was not possible to prevent the numerical degeneracy of improper posterior probabilities (tau value below). flag=2 if enforcement of the *noise proportion constraint* failed for numerical reasons. flag=3 if enforcement of the *eigenratio constraint* failed for numerical reasons. flag=4 if the *noise proportion constraint* has been successfully applied at least once. flag=5 if the *eigenratio constraint* has been successfully applied at least once.
iter	Number of iterations performed in the underlying EM-algorithm.
logicd	Value of the log(icd).
iloglik	Value of the improper likelihood.
criterion	Value of the OTRIMLE criterion.
pi	Estimated vector of the pi parameters of the underlying pseudo-model (see *Details*).
mean	A matrix of dimension ncol(data) x G containing the mean parameters of each cluster (column-wise).
cov	An array of size ncol(data) x ncol(data) x G containing the covariance matrices of each cluster.
tau	A matrix of dimension nrow(data) x {1+G} where tau[i, 1+j] is the estimated (improper) posterior probability that the *i*th observation belongs to the *j*th cluster. tau[i,1] is the estimated (improper) posterior probability that *i*th observation belongs to the noise component.
smd	A matrix of dimension nrow(data) x G where smd[i,j] is the squared Mahalanobis distance of data[i,] from mean[,j] according to cov[,,j].

`cluster`	A vector of integers denoting cluster assignments for each observation. It's `0` for observations assigned to noise/outliers.
`size`	A vector of integers with sizes (counts) of each cluster.
`exproportion`	A vector of estimated expected clusters' proportions (see *Details*).

Author(s)

Pietro Coretto <pcoretto@unisa.it> https://pietro-coretto.github.io

References

Coretto, P. and C. Hennig (2016). Robust improper maximum likelihood: tuning, computation, and a comparison with other methods for robust Gaussian clustering. *Journal of the American Statistical Association*, Vol. 111(516), pp. 1648-1659. doi: 10.1080/01621459.2015.1100996

P. Coretto and C. Hennig (2017). Consistency, breakdown robustness, and algorithms for robust improper maximum likelihood clustering. *Journal of Machine Learning Research*, Vol. 18(142), pp. 1-39. https://jmlr.org/papers/v18/16-382.html

See Also

`plot.rimle`, `InitClust`, `otrimle`,

Examples

```
## Load  Swiss banknotes data
data(banknote)
x <- banknote[,-1]

## ----------------------------------------------------------------------------
## EXAMPLE 1:
## Perform RIMLE with default inputs
## ----------------------------------------------------------------------------
set.seed(1)
a <- rimle(data = x, G = 2)
print(a)

## Plot clustering
plot(a, data = x, what = "clustering")

## P-P plot of the clusterwise empirical weighted squared Mahalanobis
## distances against the target distribution pchisq(, df=ncol(data))
plot(a, what = "fit")
plot(a, what = "fit", cluster = 1)



## ----------------------------------------------------------------------------
## EXAMPLE 2:
## Compare solutions for different choices of logicd
## ----------------------------------------------------------------------------
set.seed(1)
```

```
## Case 1: noiseless solution, that is fit a pure Gaussian Mixture Model
b1 <- rimle(data = x, G = 2, logicd = -Inf)
plot(b1, data=x, what="clustering")
plot(b1, what="fit")

## Case 2: low noise level
b2 <- rimle(data = x, G = 2, logicd = -100)
plot(b2, data=x, what="clustering")
plot(b2, what="fit")

## Case 3: medium noise level
b3 <- rimle(data = x, G = 2, logicd = -10)
plot(b3, data=x, what="clustering")
plot(b3, what="fit")

## Case 3: large noise level
b3 <- rimle(data = x, G = 2, logicd = 5)
plot(b3, data=x, what="clustering")
plot(b3, what="fit")
```

Index