Package 'otrimle'

November 30, 2016

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 (2015) <https://arxiv.org/abs/1406.0808>, and Coretto and Hennig (2016) <https://arxiv.org/abs/1309.6895>.

Version 0.4

Date 2016-11-30

Author Pietro Coretto [aut, cre], Christian Hennig [aut]

Maintainer Pietro Coretto <pcoretto@unisa.it>

Imports stats, graphics, grDevices, mclust, prabclus

License GPL (>= 2)

LazyData TRUE

NeedsCompilation no

Repository CRAN

Date/Publication 2016-11-30 14:55:24

R topics documented:

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.

InitClust *Robust Initilization for Model-based Clustering Methods*

Description

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

Usage

```
InitClust(data, G, cpr.min={ncol(data)+1}/nrow(data),
          K=5, nstart.km=50, modelName="VVV", monitor=FALSE)
```

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.
cpr.min	The minimum cluster proportion allowed in the initial clustering.
K	An integer specifying the number of considered nearest neighbors per point used for the denoising step (see *Details*).
nstart.km	An integer specifying the number of random starts for the k-means step.
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*.
monitor	A logical value; TRUE means that tracing messages will be produced.

Details

The initialization is described in the supplementary material of Coretto and Hennig (2015). Noise/outliers are removed based on nearest neighbor based clutter/noise detection (NNC) of Byers and Raftery (1998). This step is performed with NNclean. The input argument K is passed as k to NNclean. Based on this step a denoised version of data is obtained. The initial clustering is then obtained based on the following steps. Note that these steps are reported in the code element of the output list (see *Value*).

Clustering steps:

Step 1: 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.

Step 2: if too small clusters (cluster proportions <cpr.min) are found in the previous step, assign small clusters to noise and perform MBHC again on the denoised data.

Step 3: if too small clusters are found in the previous step, assign small clusters to noise and perform k-means on the denoised data.

Step 4: if too small clusters are found in the previous step, then a completely random partition that satisfies cpr.min is returned.

Value

A list with the following components:

code	An integer indicating the step at which the initial clustering has been found (see *Details*).
cluster	A vector of integers denoting cluster assignments for each observation. `cluster=0` for observations assigned to noise/outliers.

References

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

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

Coretto, P. and C. Hennig (2015). Robust improper maximum likelihood: tuning, computation, and a comparison with other methods for robust Gaussian clustering. To appear on the *Journal of the American Statistical Association*. arXiv preprint at arXiv:1406.0808 with (supplement).

See Also

NNclean, hc

Examples

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

## Initial clusters with default arguments
init1 <- InitClust(data=x, G=2)
print(init1)

## Perform otrimle
a <- otrimle(data=x, initial=init1$cluster)
plot(a, what="clustering", data=x)
```

---

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=50, det.min=.Machine$double.eps, beta=0,
        opt.selector=FALSE, cmstep=TRUE,
        iter.max=100, tol=1e-4, em.iter.max=500, em.tol=1e-6, monitor=1)
```

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 c(log(icd.min), log(icd.max)), where icd.min and icd.max are the lower and the upper bound of the improper constant density (icd) with .Machine$double.xmin <= icd.min < icd.max < Inf. If logicd=NULL (default) the bounds are computed based on the data. A pure Gaussian Mixture Model fit (obtained when log(icd)=-Inf) is always included in the search path.
npr.max	A number in (0,1) specifying the maximum proportion of noise/outliers. This defines the *noise proportion constraint*.
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.
det.min	Lower bound for the minimum determinant of covariance matrices. This is only active if cmstep=FALSE (see *Details*).
beta	A number >=0 that specifies the penalty parameter for the noise level. When beta=0 (default) the penalty term is inactive.
opt.selector	A logical value. When set to TRUE solutions on the border of the parameter space are only explored if an interior solution is not available (see *Details*).
cmstep	A logical value. When set to TRUE the *eigenratio constraint* is enforced at each M-step of the underlying EM-algorithm (see *Details*).
iter.max	An integer value specifying the maximum number of iterations for the OTRIMLE criterion optimization.
tol	Stopping criterion for the OTRIMLE criterion optimization. The optimization stops when the difference between two successive values of log(icd) is smaller than tol.
em.iter.max	An integer value specifying the maximum number of iterations allowed in the underlying EM-algorithm.
em.tol	Stopping criterion for the the underlying EM-algorithm. An EM iteration stops if two successive improper log-likelihood values are within em.tol.
monitor	Set the verbosity level of tracing messages. Possible values are monitor=0 (no messages), monitor=1 and monitor=2 for increased verbosity.

Details

The otrimle function allows to approximate the OTRIMLE solution with two different versions of the underlying EM-type algorithm.

***ECM-algorithm*:** `opt.selector=FALSE, cmstep=TRUE` The OTRIMLE search computes the RIMLE (see `rimle`) based on the *ECM-algorithm* proposed in Coretto and Hennig (2016). In this case both the *eigenratio constraint*, and the *noise proportion constraint* are enforced in each conditional M-step of the algorithm.

***Approximate EM-algorithm*:** `opt.selector=TRUE, cmstep=FALSE` This corresponds to the algorithm proposed in Coretto and Hennig (2015). In this case covariance matrices are regularized in each step based on `det.min`, and the *eigenratio constraint* is applied only at the end of the EM iteration.

The *ECM-algorithm* is the default choice. The *Approximate EM-algorithm* is often slower than the *ECM-algorithm* by a factor of two. Furthermore the *Approximate EM-algorithm* is more prone to lead to problems indicated by `code=0` (see *Value*-section below) because of numerical degeneracies connected to a low value of `min.det`.

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 2 or 3 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 `logicd` wide enough (e.g. `logicd=c(-500,-5)`), 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).

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 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 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.
npr	Estimated expected noise proportion.
cpr	Vector of estimated expected cluster proportions (notice that sum(cpr)=1-npr).
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.
optimization	A data.frame with the OTRIMLE optimization profiling. For each value of log(icd) explored by the algorithm the data.frame stores criterion, iloglik, code and flag.

References

Coretto, P. and C. Hennig (2015). Robust improper maximum likelihood: tuning, computation, and a comparison with other methods for robust Gaussian clustering. To appear on the *Journal of the American Statistical Association*. arXiv preprint at arXiv:1406.0808 with (supplement).

Coretto, P. and C. Hennig (2016). Consistency, breakdown robustness, and algorithms for robust improper maximum likelihood clustering. arXiv preprint at arXiv:1309.6895.

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)
print(a)

## 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\code{-}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)
```

`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 (2015). Robust improper maximum likelihood: tuning, computation, and a comparison with other methods for robust Gaussian clustering. To appear on the *Journal of the American Statistical Association*. arXiv preprint at arXiv:1406.0808 with (supplement).

Coretto, P. and C. Hennig (2016). Consistency, breakdown robustness, and algorithms for robust improper maximum likelihood clustering. arXiv preprint at arXiv:1309.6895.

See Also

`plot.otrimle`

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)
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)
```

`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 (2015). Robust improper maximum likelihood: tuning, computation, and a comparison with other methods for robust Gaussian clustering. To appear on the *Journal of the American Statistical Association*. arXiv preprint at arXiv:1406.0808 with (supplement).

Coretto, P. and C. Hennig (2016). Consistency, breakdown robustness, and algorithms for robust improper maximum likelihood clustering. arXiv preprint at arXiv:1309.6895.

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=50, det.min=.Machine$double.eps, cmstep=TRUE,
      em.iter.max=500, em.tol=1e-6, monitor=1)
```

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=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 `log(icd)=-Inf`.
`npr.max`	A number in `(0,1)` specifying the maximum proportion of noise/outliers. This defines the *noise proportion constraint*.
`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.
`det.min`	Lower bound for the minimum determinant of covariance matrices. This is only active if `cmstep=FALSE` (see *Details*).
`cmstep`	A logical value. When set to `TRUE` the *eigenratio constraint* is enforced at each M-step of the underlying EM-algorithm (see *Details*).
`em.iter.max`	An integer value specifying the maximum number of iterations allowed in the underlying EM-algorithm.
`em.tol`	Stopping criterion for the the underlying EM-algorithm. An EM iteration stops if two successive improper log-likelihood values are within `em.tol`.
`monitor`	Set the verbosity level of tracing messages. Possible values are `monitor=0` (no messages), `monitor=1` and `monitor=2` for increased verbosity.

Details

The `rimle` function allows to approximate the RIMLE solution with two different versions of the underlying EM-type algorithm.

***ECM-algorithm*:** `cmstep=TRUE` The RIMLE solution is obtained based on the *ECM-algorithm* proposed in Coretto and Hennig (2016). In this case both the *eigenratio constraint* and the *noise proportion constraint* are enforced in each conditional M-step of the algorithm.

***Approximate EM-algorithm*:** `cmstep=FALSE` This corresponds to the algorithm proposed in Coretto and Hennig (2015). In this case covariance matrices are regularized in each step based on `det.min`, and the *eigenratio constraint* is applied only at the end of the EM iteration.

The *ECM-algorithm* is the default choice. The *Approximate EM-algorithm* is often slower than the *ECM-algorithm* by a factor of two. Furthermore the *Approximate EM-algorithm* is more prone to lead to problems indicated by `code=0` (see *Value*-section below) because of numerical degeneracies connected to a low value of `min.det`.

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 2 or 3 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.

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.
`npr`	Estimated expected noise proportion.
`cpr`	Vector of estimated expected cluster proportions (notice that `sum(cpr)=1-npr`).
`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.

References

Coretto, P. and C. Hennig (2015). Robust improper maximum likelihood: tuning, computation, and a comparison with other methods for robust Gaussian clustering. To appear on the *Journal of the American Statistical Association*. arXiv preprint at arXiv:1406.0808 with (supplement).

Coretto, P. and C. Hennig (2016). Consistency, breakdown robustness, and algorithms for robust improper maximum likelihood clustering. arXiv preprint at arXiv:1309.6895.

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