Package: mclustAddonsEHA (via r-universe)

October 18, 2024

Version 0.7.2

Date 2022-12-24

Title FORK of original mclustAddons. Addons for the 'mclust' Package

Description Extend the functionality of the 'mclust' package for Gaussian finite mixture modeling by including: density estimation for data with bounded support (Scrucca, 2019 <doi:10.1002/bimj.201800174>); modal clustering using MEM (Modal EM) algorithm for Gaussian mixtures (Scrucca, 2021 <doi:10.1002/sam.11527>); entropy estimation via Gaussian mixture modeling (Robin & Scrucca, 2023 <doi:10.1016/j.csda.2022.107582>).

Depends R (>= 4.0), mclust (>= 5.4)

Imports stats, graphics, grDevices, methods, foreach, iterators, utils, Rcpp (>= 1.0)

LinkingTo Rcpp, RcppArmadillo (>= 0.10)

Suggests parallel, doParallel, doRNG (>= 1.6), cli, crayon, knitr (>= 1.12), rmarkdown (>= 0.9)

License GPL (>= 2)

URL https://github.com/ecohealthalliance/mclustAddonsEHA

VignetteBuilder knitr

Encoding UTF-8

Repository https://ecohealthalliance.r-universe.dev

RemoteUrl https://github.com/ecohealthalliance/mclustAddonsEHA

RemoteRef working

RemoteSha 5e94a1f8c108cb3bff7e95b2748a50f4ded31892

Contents

`mclustAddons-package` *Addons for the* **mclust** *package*

Description

Extend the functionality of the **mclust** package for Gaussian finite mixture modeling by including:

- density estimation for data with bounded support (Scrucca, 2019)
- modal clustering using MEM algorithm for Gaussian mixtures (Scrucca, 2021)
- entropy estimation via Gaussian mixture modeling (Robin & Scrucca, 2023)

Details

For a quick introduction to **mclustAddons** see the vignette A quick tour of mclustAddons.

See also:

- `densityMclustBounded()` for density estimation of bounded data;
- `MclustMEM()` for modal clustering;
- `EntropyGMM()` for entropy estimation.

Author(s)

Luca Scrucca.

Maintainer: Luca Scrucca <luca.scrucca@unipg.it>

References

Scrucca L. (2019) A transformation-based approach to Gaussian mixture density estimation for bounded data. *Biometrical Journal*, 61:4, 873–888. https://doi.org/10.1002/bimj.201800174

Scrucca L. (2021) A fast and efficient Modal EM algorithm for Gaussian mixtures. *Statistical Analysis and Data Mining*, 14:4, 305–314. https://doi.org/10.1002/sam.11527

Robin S. and Scrucca L. (2023) Mixture-based estimation of entropy. *Computational Statistics & Data Analysis*, 177, 107582. https://doi.org/10.1016/j.csda.2022.107582

`cdfDensityBounded` *Cumulative distribution and quantiles of univariate model-based mixture density estimation for bounded data*

Description

Compute the cumulative density function (cdf) or quantiles of a one-dimensional density for bounded data estimated via transformation-based approach for Gaussian mixtures using `densityMclustBounded`.

Usage

```
cdfDensityBounded(object, data, ngrid = 100, ...)
quantileDensityBounded(object, p, ...)
```

Arguments

`object`	a `densityMclustBounded` model object.
`data`	a numeric vector of evaluation points.
`ngrid`	the number of points in a regular grid to be used as evaluation points if no `data` are provided.
`p`	a numeric vector of probabilities.
`...`	further arguments passed to or from other methods.

Details

The cdf is evaluated at points given by the optional argument `data`. If not provided, a regular grid of length `ngrid` for the evaluation points is used.

The quantiles are computed using bisection linear search algorithm.

Value

`cdfDensityBounded` returns a list of `x` and `y` values providing, respectively, the evaluation points and the estimated cdf.

`quantileDensityBounded` returns a vector of quantiles.

Author(s)

Luca Scrucca

See Also

`densityMclustBounded`, `plot.densityMclustBounded`.

Examples

```
# univariate case with lower bound
x <- rchisq(200, 3)
dens <- densityMclustBounded(x, lbound = 0)

xgrid <- seq(-2, max(x), length=1000)
cdf <- cdfDensityBounded(dens, xgrid)
str(cdf)
plot(xgrid, pchisq(xgrid, df = 3), type = "l", xlab = "x", ylab = "CDF")
lines(cdf, col = 4, lwd = 2)

q <- quantileDensityBounded(dens, p = c(0.01, 0.1, 0.5, 0.9, 0.99))
cbind(quantile = q, cdf = cdfDensityBounded(dens, q)$y)
plot(cdf, type = "l", col = 4, xlab = "x", ylab = "CDF")
points(q, cdfDensityBounded(dens, q)$y, pch = 19, col = 4)

# univariate case with lower & upper bounds
x <- rbeta(200, 5, 1.5)
dens <- densityMclustBounded(x, lbound = 0, ubound = 1)

xgrid <- seq(-0.1, 1.1, length=1000)
cdf <- cdfDensityBounded(dens, xgrid)
str(cdf)
plot(xgrid, pbeta(xgrid, 5, 1.5), type = "l", xlab = "x", ylab = "CDF")
lines(cdf, col = 4, lwd = 2)

q <- quantileDensityBounded(dens, p = c(0.01, 0.1, 0.5, 0.9, 0.99))
cbind(quantile = q, cdf = cdfDensityBounded(dens, q)$y)
plot(cdf, type = "l", col = 4, xlab = "x", ylab = "CDF")
points(q, cdfDensityBounded(dens, q)$y, pch = 19, col = 4)
```

`densityMclustBounded` *Model-based mixture density estimation for bounded data*

Description

Density estimation for bounded data via transformation-based approach for Gaussian mixtures.

Usage

```
densityMclustBounded(data,
                     G = NULL, modelNames = NULL,
                     lbound = NULL,
                     ubound = NULL,
                     lambda = c(-3, 3),
                     prior = NULL,
                     parallel = FALSE,
                     seed = NULL,
```

```
                    ...)

## S3 method for class 'densityMclustBounded'
print(x, digits = getOption("digits"), ...)

## S3 method for class 'densityMclustBounded'
summary(object, parameters = FALSE, classification = FALSE, ...)
```

Arguments

data	A numeric vector, matrix, or data frame of observations. If a matrix or data frame, rows correspond to observations and columns correspond to variables.
G	An integer vector specifying the numbers of mixture components. By default G=1:3.
modelNames	A vector of character strings indicating the Gaussian mixture models to be fitted on the transformed-data space. See mclustModelNames for a descripton of available models.
lbound	Numeric vector proving lower bounds for variables.
ubound	Numeric vector proving upper bounds for variables.
lambda	A numeric vector providing the range of searched values for the transformation parameter(s).
prior	A list containing the prior probabilities of the mixture components and the parameters of the prior distributions for the mixture components. If NULL, the default priors are used. See Mclust for details.
parallel	An optional argument which allows to specify if the search over all possible models should be run sequentially (default) or in parallel. For a single machine with multiple cores, possible values are: • a logical value specifying if parallel computing should be used (TRUE) or not (FALSE, default) for evaluating the fitness function; • a numerical value which gives the number of cores to employ. By default, this is obtained from the function detectCores; • a character string specifying the type of parallelisation to use. This depends on system OS: on Windows OS only "snow" type functionality is available, while on Unix/Linux/Mac OSX both "snow" and "multicore" (default) functionalities are available. In all the cases described above, at the end of the search the cluster is automatically stopped by shutting down the workers. If a cluster of multiple machines is available, evaluation of the fitness function can be executed in parallel using all, or a subset of, the cores available to the machines belonging to the cluster. However, this option requires more work from the user, who needs to set up and register a parallel back end. In this case the cluster must be explicitely stopped with stopCluster.
seed	An integer value containing the random number generator state. This argument can be used to replicate the result of k-means initialisation strategy. Note that if parallel computing is required, the **doRNG** package must be installed.

x, object	An object of class "densityMclustBounded".
digits	The number of significant digits to use for printing.
parameters	Logical; if TRUE, the parameters of mixture components are printed.
classification	Logical; if TRUE, the MAP classification/clustering of observations is printed.
...	Further arguments passed to or from other methods.

Value

Returns an object of class "densityMclustBounded".

Author(s)

Luca Scrucca

References

Scrucca L. (2019) A transformation-based approach to Gaussian mixture density estimation for bounded data. *Biometrical Journal*, 61:4, 873–888. https://doi.org/10.1002/bimj.201800174

See Also

predict.densityMclustBounded, plot.densityMclustBounded.

Examples

```
# univariate case with lower bound
x <- rchisq(200, 3)
xgrid <- seq(-2, max(x), length=1000)
f <- dchisq(xgrid, 3)  # true density
dens <- densityMclustBounded(x, lbound = 0)
summary(dens)
summary(dens, parameters = TRUE)
plot(dens, what = "BIC")
plot(dens, what = "density")
lines(xgrid, f, lty = 2)
plot(dens, what = "density", data = x, breaks = 15)

# univariate case with lower & upper bounds
x <- rbeta(200, 5, 1.5)
xgrid <- seq(-0.1, 1.1, length=1000)
f <- dbeta(xgrid, 5, 1.5)  # true density
dens <- densityMclustBounded(x, lbound = 0, ubound = 1)
summary(dens)
plot(dens, what = "BIC")
plot(dens, what = "density")
plot(dens, what = "density", data = x, breaks = 9)

# bivariate case with lower bounds
x1 <- rchisq(200, 3)
x2 <- 0.5*x1 + sqrt(1-0.5^2)*rchisq(200, 5)
x <- cbind(x1, x2)
```

```
plot(x)
dens <- densityMclustBounded(x, lbound = c(0,0))
summary(dens, parameters = TRUE)
plot(dens, what = "BIC")
plot(dens, what = "density")
plot(dens, what = "density", type = "hdr")
plot(dens, what = "density", type = "persp")
```

`densityMclustBounded.diagnostic` *Diagnostic plots for* `mclustDensityBounded` *estimation*

Description

Diagnostic plots for density estimation of bounded data via transformation-based approach of Gaussian mixtures. Only available for the one-dimensional case.

Usage

```
densityMclustBounded.diagnostic(object, type = c("cdf", "qq"),
                                col = c("black", "black"),
                                lwd = c(2,1), lty = c(1,1),
                                legend = TRUE, grid = TRUE,
                                ...)
```

Arguments

`object`	An object of class `'mclustDensityBounded'` obtained from a call to `densityMclustBounded` function.
`type`	The type of graph requested: `"cdf"` = a plot of the estimated CDF versus the empirical distribution function. `"qq"` = a Q-Q plot of sample quantiles versus the quantiles obtained from the inverse of the estimated cdf.
`col`	A pair of values for the color to be used for plotting, respectively, the estimated CDF and the empirical cdf.
`lwd`	A pair of values for the line width to be used for plotting, respectively, the estimated CDF and the empirical cdf.
`lty`	A pair of values for the line type to be used for plotting, respectively, the estimated CDF and the empirical cdf.
`legend`	A logical indicating if a legend must be added to the plot of fitted CDF vs the empirical CDF.
`grid`	A logical indicating if a `grid` should be added to the plot.
`...`	Additional arguments.

Details

The two diagnostic plots for density estimation in the one-dimensional case are discussed in Loader (1999, pp- 87-90).

Value

No return value, called for side effects.

Author(s)

Luca Scrucca

References

Loader C. (1999), Local Regression and Likelihood. New York, Springer.

See Also

densityMclustBounded, plot.densityMclustBounded.

Examples

```
# univariate case with lower bound
x <- rchisq(200, 3)
dens <- densityMclustBounded(x, lbound = 0)
plot(dens, x, what = "diagnostic")
# or
densityMclustBounded.diagnostic(dens, type = "cdf")
densityMclustBounded.diagnostic(dens, type = "qq")

# univariate case with lower & upper bounds
x <- rbeta(200, 5, 1.5)
dens <- densityMclustBounded(x, lbound = 0, ubound = 1)
plot(dens, x, what = "diagnostic")
# or
densityMclustBounded.diagnostic(dens, type = "cdf")
densityMclustBounded.diagnostic(dens, type = "qq")
```

EntropyGMM *Gaussian mixture-based estimation of entropy*

Description

Compute an estimate of the (differential) entropy from a Gaussian Mixture Model (GMM) fitted using the *mclust* package.

Usage

```
EntropyGMM(object, ...)

## S3 method for class 'densityMclust'
EntropyGMM(object, ...)
## S3 method for class 'Mclust'
EntropyGMM(object, ...)
## S3 method for class 'densityMclustBounded'
EntropyGMM(object, ...)
## S3 method for class 'matrix'
EntropyGMM(object, ...)
## S3 method for class 'data.frame'
EntropyGMM(object, ...)

EntropyGauss(sigma)

nats2bits(x)
bits2nats(x)
```

Arguments

object	An object of class 'Mclust', 'densityMclust', or 'densityMclustBounded', obtained by fitting a Gaussian mixture via, respectively, Mclust(), densityMclust(), and densityMclustBounded(). If a 'matrix' or 'data.frame' is provided as input, a GMM using the provided data is estimated preliminary to computing the entropy. In this case further arguments can be provided to control the fitted model (e.g. number of mixture components and/or covariances decomposition).
sigma	A symmetric covariance matrix.
x	A vector of values.
...	Further arguments passed to or from other methods.

Value

EntropyGMM() returns an estimate of the entropy based on a estimated Gaussian mixture model (GMM) fitted using the *mclust* package. If a matrix of data values is provided, a GMM is preliminary fitted to the data and then the entropy computed.

EntropyGauss() returns the entropy for a multivariate Gaussian distribution with covariance matrix sigma.

nats2bits() and bits2nats() convert input values in nats to bits, and viceversa. Information-theoretic quantities have different units depending on the base of the logarithm used: nats are expressed in base-2 logarithms, whereas bits in natural logarithms.

Author(s)

Luca Scrucca

References

Robin S. and Scrucca L. (2023) Mixture-based estimation of entropy. *Computational Statistics & Data Analysis*, 177, 107582. https://doi.org/10.1016/j.csda.2022.107582

See Also

`Mclust`, `densityMclust`.

Examples

```
X = iris[,1:4]
mod = densityMclust(X, plot = FALSE)
h = EntropyGMM(mod)
h
bits2nats(h)
EntropyGMM(X)
```

`GaussianMixtureMEM` *Modal EM algorithm for Gaussian Mixtures*

Description

A function implementing a fast and efficient Modal EM algorithm for Gaussian mixtures.

Usage

```
GaussianMixtureMEM(data, pro, mu, sigma,
                   control = list(eps = 1e-5,
                                  maxiter = 1e3,
                                  stepsize = function(t) 1-exp(-0.1*t),
                                  denoise = TRUE,
                                  alpha = 0.01,
                                  keep.path = FALSE),
                   ...)
```

Arguments

`data`	A numeric vector, matrix, or data frame of observations. Categorical variables are not allowed. If a matrix or data frame, rows correspond to observations (n) and columns correspond to variables (d).
`pro`	A $(G \times 1)$ vector of mixing probabilities for a Gaussian mixture of G components.
`mu`	A $(d \times G)$ matrix of component means for a d-variate Gaussian mixture of G components.
`sigma`	A $(d \times d \times G)$ array of component covariance matrices for a d-variate Gaussian mixture of G components.

`control` A list of control parameters:

`eps, maxiter` numerical values setting the tolerance and the maximum number of iterations of the MEM algorithm;

`stepsize` a function controlling the step size of the MEM algorithm;

`denoise` a logical, if `TRUE` a denoising procedure is used when $d > 1$ to discard all modes whose density is negligible;

`alpha` a numerical value used when `denoise = TRUE` for computing the hyper-volume of central $(1 - \alpha)100$ region of a multivariate Gaussian:

`keep.path` a logical controlling whether or not the full paths to modes must be returned.

`...` Further arguments passed to or from other methods.

Value

Returns a list containing the following elements:

`n`	The number of input data points.
`d`	The number of variables/features.
`parameters`	The Gaussian mixture parameters.
`iter`	The number of iterations of MEM algorithm.
`nmodes`	The number of modes estimated by the MEM algorithm.
`modes`	The coordinates of modes estimated by MEM algorithm.
`path`	If requested, the coordinates of full paths to modes for each data point.
`logdens`	The log-density at the estimated modes.
`logvol`	The log-volume used for denoising (if requested).
`classification`	The modal clustering classification of input data points.

Author(s)

Luca Scrucca

References

Scrucca L. (2021) A fast and efficient Modal EM algorithm for Gaussian mixtures. *Statistical Analysis and Data Mining*, 14:4, 305–314. https://doi.org/10.1002/sam.11527

See Also

`MclustMEM`.

`logsumexp` *Compute log-sum-exp and softmax functions.*

Description

Efficient implementation (via Rcpp) of log-sum-exp and softmax functions.

Usage

```
logsumexp(x, v = NULL)
softmax(x, v = NULL)
```

Arguments

`x`	a matrix of dimension $(n \times k)$.
`v`	an optional vector of length k. If not provided a vector of zeros is used.

Details

Given the matrix x `logsumexp()` calculates for each row $x_{[i,]}$ the log-sum-exp function computed as

$$m_{[i]} + \log \sum (\exp(x_{[i,]} + v) - m_{[i]})$$

where $m_{[i]} = \max(x_{[i,]} + v)$.

`softmax()` calculates for each row $x_{[i,]}$ the softmax (aka multinomial logistic) function

$$\frac{\exp(x_{[i,c]} + v_{[c]})}{\sum_{j=1}^{k} \exp(x_{[i,j]} + v_{[j]})}$$

Value

`logsumexp()` returns a vector of values of length equal to the number of rows of x. `softmax()` returns a matrix of values of the same dimension as x.

Author(s)

Luca Scrucca

Examples

```
x = matrix(rnorm(15), 5, 3)
v = log(c(0.5, 0.3, 0.2))
logsumexp(x, v)
(z = softmax(x, v))
rowSums(z)
```

`MclustMEM` *Modal EM algorithm for Gaussian Mixtures fitted via* mclust *package*

Description

Modal-clustering estimation by applying the Modal EM algorithm to Gaussian mixtures fitted using the *mclust* package.

Usage

```
MclustMEM(mclustObject, data = NULL, ...)

## S3 method for class 'MclustMEM'
print(x, digits = getOption("digits"), ...)

## S3 method for class 'MclustMEM'
summary(object, ...)
```

Arguments

`mclustObject`	An object of class `'Mclust'` or `'densityMclust'` obtained by fitting a Gaussian mixture via, respectively, `Mclust` and `densityMclust`.
`data`	If provided, a numeric vector, matrix, or data frame of observations. If a matrix or data frame, rows correspond to observations (n) and columns correspond to variables (d). If not provided, the data used for fitting the Gaussian mixture model, and provided with the `object` argument, are used.
`x`, `object`	An object of class `'MclustMEM'`.
`digits`	The number of significant digits to use for printing.
`...`	Further arguments passed to or from other methods.

Value

Returns an object of class `'MclustMEM'`. See also the output returned by `GaussianMixtureMEM`.

Author(s)

Luca Scrucca

References

Scrucca L. (2021) A fast and efficient Modal EM algorithm for Gaussian mixtures. *Statistical Analysis and Data Mining*, 14:4, 305–314. https://doi.org/10.1002/sam.11527

See Also

`GaussianMixtureMEM`, `plot.MclustMEM`.

Examples

```
data(Baudry_etal_2010_JCGS_examples, package = "mclust")
plot(ex4.1)
GMM <- Mclust(ex4.1)
plot(GMM, what = "classification")
MEM <- MclustMEM(GMM)
MEM
summary(MEM)
plot(MEM)

plot(ex4.4.2)
GMM <- Mclust(ex4.4.2)
plot(GMM, what = "classification")
MEM <- MclustMEM(GMM)
MEM
summary(MEM)
plot(MEM, addDensity = FALSE)
```

---

plot.densityMclustBounded

Plotting method for model-based mixture density estimation for bounded data

---

Description

Plots for `mclustDensityBounded` objects.

Usage

```
## S3 method for class 'densityMclustBounded'
plot(x, what = c("BIC", "density", "diagnostic"),
     data = NULL, ...)
```

Arguments

`x`	An object of class `"densityMclustBounded"` obtained from a call to `densityMclustBounded`.
`what`	The type of graph requested: `"BIC"` = a plot of BIC values for the estimated models versus the number of components. `"density"` = a plot of estimated density; if `data` is also provided the density is plotted over data points. `"diagnostic"` = diagnostic plots (only available for the one-dimensional case).
`data`	Optional data points.
`...`	Further available arguments. • For 1-dimensional data: `hist.col = "lightgrey"`, `hist.border = "white"`, `breaks = "Sturges"`

- For 2-dimensional data:
  `type = c("contour", "hdr", "image", "persp")`
  `transformation = c("none", "log", "sqrt")`
  `grid = 100, nlevels = 11, levels = NULL`
  `prob = c(0.25, 0.5, 0.75)`
  `col = grey(0.6), color.palette = blue2grey.colors`
  `points.col = 1, points.cex = 0.8, points.pch = 1`
- For $d > 2$-dimensional data:
  `type = c("contour", "hdr"), gap = 0.2`
  `grid = 100, nlevels = 11, levels = NULL`
  `prob = c(0.25, 0.5, 0.75)`
  `col = grey(0.6), color.palette = blue2grey.colors`
  `points.col = 1, points.cex = 0.8, points.pch = 1`

Value

No return value, called for side effects.

Author(s)

Luca Scrucca

References

Scrucca L. (2019) A transformation-based approach to Gaussian mixture density estimation for bounded data. *Biometrical Journal*, 61:4, 873–888. https://doi.org/10.1002/bimj.201800174

See Also

`densityMclustBounded`, `predict.densityMclustBounded`.

Examples

```
# univariate case with lower bound
x <- rchisq(200, 3)
dens <- densityMclustBounded(x, lbound = 0)
plot(dens, what = "BIC")
plot(dens, what = "density", data = x, breaks = 15)

# univariate case with lower & upper bound
x <- rbeta(200, 5, 1.5)
dens <- densityMclustBounded(x, lbound = 0, ubound = 1)
plot(dens, what = "BIC")
plot(dens, what = "density", data = x, breaks = 9)

# bivariate case with lower bounds
x1 <- rchisq(200, 3)
x2 <- 0.5*x1 + sqrt(1-0.5^2)*rchisq(200, 5)
x <- cbind(x1, x2)
dens <- densityMclustBounded(x, lbound = c(0,0))
plot(dens, what = "density")
```

```
plot(dens, what = "density", data = x)
plot(dens, what = "density", type = "hdr")
plot(dens, what = "density", type = "persp")
```

`plot.MclustMEM` *Plotting method for modal-clustering based on Gaussian Mixtures*

Description

Plots for `MclustMEM` objects.

Usage

```
## S3 method for class 'MclustMEM'
plot(x, dimens = NULL, addDensity = TRUE, addPoints = TRUE,
     symbols = NULL, colors = NULL, cex = NULL,
     labels = NULL, cex.labels = NULL, gap = 0.2,
    ...)
```

Arguments

`x`	An object of class `"densityMclustBounded"` obtained from a call to `densityMclustBounded`.
`dimens`	A vector of integers specifying the dimensions of the coordinate projections.
`addDensity`	A logical indicating whether or not to add density estimates to the plot.
`addPoints`	A logical indicating whether or not to add data points to the plot.
`symbols`	Either an integer or character vector assigning a plotting symbol to each unique class in `classification`. Elements in `symbols` correspond to classes in order of appearance in the sequence of observations (the order used by the function `unique`). The default is given by `mclust.options("classPlotSymbols")`.
`colors`	Either an integer or character vector assigning a color to each unique class in `classification`. Elements in `colors` correspond to classes in order of appearance in the sequence of observations (the order used by the function `unique`). The default is given by `mclust.options("classPlotColors")`.
`cex`	A vector of numerical values specifying the size of the plotting symbol for each unique class in `classification`. By default `cex = 1` for all classes is used.
`labels`	A vector of character strings for labelling the variables. The default is to use the column dimension names of `data`.
`cex.labels`	A numerical value specifying the size of the text labels.
`gap`	A numerical argument specifying the distance between subplots (see `pairs`).
`...`	Further arguments passed to or from other methods.

Value

No return value, called for side effects.

Author(s)

Luca Scrucca

References

Scrucca L. (2021) A fast and efficient Modal EM algorithm for Gaussian mixtures. *Statistical Analysis and Data Mining*, 14:4, 305–314. https://doi.org/10.1002/sam.11527

See Also

MclustMEM.

Examples

```
# 1-d example
GMM <- Mclust(iris$Petal.Length)
MEM <- MclustMEM(GMM)
plot(MEM)

# 2-d example
data(Baudry_etal_2010_JCGS_examples)
GMM <- Mclust(ex4.1)
MEM <- MclustMEM(GMM)
plot(MEM)
plot(MEM, addPoints = FALSE)
plot(MEM, addDensity = FALSE)

# 3-d example
GMM <- Mclust(ex4.4.2)
MEM <- MclustMEM(GMM)
plot(MEM)
plot(MEM, addPoints = FALSE)
plot(MEM, addDensity = FALSE)
```

---

predict.densityMclustBounded

Model-based mixture density estimation for bounded data

Description

Compute density estimation for univariate and multivariate bounded data based on Gaussian finite mixture models estimated by densityMclustBounded.

Usage

```
## S3 method for class 'densityMclustBounded'
predict(object, newdata,
        what = c("dens", "cdens", "z"),
        logarithm = FALSE, ...)
```

Arguments

object	An object of class "densityMclustBounded" resulting from a call to densityMclustBounded.
newdata	A numeric vector, matrix, or data frame of observations. If missing the density is computed for the input data obtained from the call to densityMclustBounded.
what	A character string specifying what to retrieve: "dens" returns a vector of values for the mixture density; "cdens" returns a matrix of component densities for each mixture component (along the columns); "z" returns a matrix of component posterior probabilities.
logarithm	A logical value indicating whether or not the logarithm of the densities/probabilities should be returned.
...	Further arguments passed to or from other methods.

Value

Returns a vector or a matrix of values evaluated at newdata depending on the argument what (see above).

Author(s)

Luca Scrucca

References

Scrucca L. (2019) A transformation-based approach to Gaussian mixture density estimation for bounded data. *Biometrical Journal*, 61:4, 873–888. https://doi.org/10.1002/bimj.201800174

See Also

densityMclustBounded, plot.densityMclustBounded.

Examples

```
y <- sample(0:1, size = 200, replace = TRUE, prob = c(0.6, 0.4))
x <- y*rchisq(200, 3) + (1-y)*rchisq(200, 10)
dens <- densityMclustBounded(x, lbound = 0)
summary(dens)
plot(dens, what = "density", data = x, breaks = 11)

xgrid <- seq(0, max(x), length = 201)
densx <- predict(dens, newdata = xgrid, what = "dens")
cdensx <- predict(dens, newdata = xgrid, what = "cdens")
cdensx <- sweep(cdensx, MARGIN = 2, FUN = "*", dens$parameters$pro)
plot(xgrid, densx, type = "l", lwd = 2)
matplot(xgrid, cdensx, type = "l", col = 3:4, lty = 2:3, lwd = 2, add = TRUE)

z <- predict(dens, newdata = xgrid, what = "z")
matplot(xgrid, z, col = 3:4, lty = 2:3, lwd = 2, ylab = "Posterior probabilities")
```

racial *Racial data*

Description

Proportion of white student enrollment in 56 school districts in Nassau County (Long Island, New York), for the 1992-1993 school year.

Usage

```
data(racial)
```

Format

A data frame with the following variables:

District School district.

PropWhite Proportion of white student enrolled.

Source

Simonoff, S.J. (1996) Smoothing Methods in Statistics, Springer-Verlag, New York, p. 52

suicide *Suicide data*

Description

Lengths of treatment spells (in days) of control patients in suicide study.

Usage

```
data(suicide)
```

Format

A vector of containing the lengths (days) of 86 spells of psychiatric treatment undergone by patients used as controls in a study of suicide risks.

Source

Silverman, B. W. (1986) Density Estimation, Chapman & Hall, Tab 2.1.

Index