Package 'mclustAddons'

May 8, 2026

Title Addons for the 'mclust' Package

Version 0.10

Date 2025-12-03

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>); Gaussian mixtures modeling of financial log-returns (Scrucca, 2024 <doi:10.3390/e26110907>).

License GPL (>= 2)

URL https://mclust-org.github.io/mclustAddons/

Depends mclust (>= 6.1.1), R (>= 4.3)

Imports cli, doParallel, doRNG (>= 1.6), foreach, graphics, grDevices, iterators, knitr (>= 1.12), parallel, Rcpp (>= 1.0), rmarkdown (>= 0.9), stats, utils

LinkingTo Rcpp (>= 1.0), RcppArmadillo (>= 14.4)

VignetteBuilder knitr

ByteCompile true

Encoding UTF-8

NeedsCompilation yes

Repository CRAN

RoxygenNote 7.3.3

Author Luca Scrucca [aut, cre, cph] (ORCID: <https://orcid.org/0000-0003-3826-0484>)

Maintainer Luca Scrucca <luca.scrucca@unibo.it>

Date/Publication 2025-12-03 12:10:02 UTC

Contents

`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 the transformation-based approach for Gaussian mixtures in `densityMclustBounded()`.

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

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

Usage

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

quantileDensityBounded(object, p, ...)

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.
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.
...	Additional arguments.
p	A numeric vector of probabilities corresponding to quantiles.
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.

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.

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()`.

`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)
```

```
# 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")
```

---

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,
  criterion = c("BIC", "ICL"),
  lbound = NULL,
  ubound = NULL,
  lambda = c(-3, 3),
  prior = NULL,
  initialization = NULL,
  nstart = 25,
  parallel = FALSE,
  seed = NULL,
  ...
)

## S3 method for class 'densityMclustBounded'
summary(object, parameters = 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 mclust::mclustModelNames() for a descripton of available models.
criterion	A character string specifying the information criterion for model selection. Possible values are BIC (default) or ICL.
lbound	Numeric vector proving lower bounds for variables.
ubound	Numeric vector proving upper bounds for variables.
lambda	A numeric vector providing the range (min and max) of searched values for the transformation parameter(s). If a matrix is provided, then for each variable a row should be provided containing the range of lambda values for the transformation parameter. If a variable must have a fixed lambda value, the provided min and max values should be equal. See examples below.
prior	A function specifying a prior for Bayesian regularization of Gaussian mixtures. See mclust::priorControl() for details.
initialization	A list containing one or more of the following components: • noise A logical or numeric vector indicating an initial guess as to which observations are noise in the data. If numeric the entries should correspond to row indexes of the data. If logical an automatic entropy-based guess of noisy observations is made. When supplied, a noise term will be added to the model in the estimation. • Vinv When a noise component is included in the model, this is a numerical optional argument providing the reciprocal of the volume of the data. By default, the mclust::hypvol() is used on the transformed data from a preliminary model.
nstart	An integer value specifying the number of replications of k-means clustering to be used for initializing the EM algorithm. See kmeans().
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 parallel::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 `parallel::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.
`...`	Further arguments passed to or from other methods.
`object`	An object of class `'densityMclustBounded'`.
`parameters`	A logical, if `TRUE` the estimated parameters of mixture components are printed.

Details

For more details see `vignette("mclustAddons")`

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. doi: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")
# specify different ranges for the lambda values of each variable
dens1 <- densityMclustBounded(x, lbound = c(0,0),
                              lambda = matrix(c(-2,2,0,1), 2, 2, byrow=TRUE))
# set lambda = 0 fixed for the second variable
dens2 <- densityMclustBounded(x, lbound = c(0,0),
                              lambda = matrix(c(0,1,0,0), 2, 2, byrow=TRUE))

dens[c("lambdaRange", "lambda", "loglik", "df")]
dens1[c("lambdaRange", "lambda", "loglik", "df")]
dens2[c("lambdaRange", "lambda", "loglik", "df")]
```

`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 'densityMclustBounded'
```

```
EntropyGMM(object, ...)

## S3 method for class 'Mclust'
EntropyGMM(object, ...)

## S3 method for class 'data.frame'
EntropyGMM(object, ...)

## S3 method for class 'matrix'
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::Mclust(), 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).
...	Further arguments passed to or from other methods.
sigma	A symmetric covariance matrix.
x	A vector of values.

Details

For more details see vignette("mclustAddons")

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. doi:10.1016/j.csda.2022.107582

See Also

`mclust::Mclust()`, `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-05, maxiter = 1000, 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 hypervolume 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. doi: 10.1002/sam.11527

See Also

`MclustMEM()`.

`GMMlogreturn` *Modeling log-returns distribution via Gaussian Mixture Models*

Description

Gaussian mixtures for modeling the distribution of financial log-returns.

Usage

```
GMMlogreturn(y, ...)

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

Arguments

`y`	A numeric vector providing the log-returns of a financial stock.
`...`	Further arguments passed to `mclust::densityMclust()`. For a full description of available arguments see the corresponding help page.
`object`	An object of class `'GMMlogreturn'`.

Details

Let P_t be the price of a financial stock for the current time frame (day for instance), and P_{t-1} the price of the previous time frame. The log-return at time t is defined as:

$$y_t = \log(\frac{P_t}{P_{t-1}})$$

By default, a univariate heteroscedastic GMM using Bayesian regularization (as described in `mclust::priorControl()`) is fitted to the observed log-returns. The number of mixture components is automatically selected by BIC, unless specified with the optional `G` argument.

Value

Returns an object of class `'GMMlogreturn'`.

Author(s)

Luca Scrucca

References

Scrucca L. (2024) Entropy-based volatility analysis of financial log-returns using Gaussian mixture models. *Entropy*, 26(11), 907. doi:10.3390/e26110907

See Also

`VaR.GMMlogreturn()`, `ES.GMMlogreturn()`.

Examples

```
data(gold)
head(gold)
mod = GMMlogreturn(gold$log.returns)
summary(mod)
plot(mod, what = "density", data = gold$log.returns,
     xlab = "log-returns", col = 4, lwd = 2)
```

`gold` *Gold price log-returns*

Description

Gold price log-returns for the year 2023 obtained from Yahoo Finance using the `quantmod` R package. Code used to download, format, and save the data:

```
gold = quantmod::getSymbols("GC=F", src = "yahoo", auto.assign = FALSE)
gold = quantmod::dailyReturn(gold, type = "log")
gold = data.frame("date" = as.Date(zoo::index(gold)),
                  "log.returns" = as.vector(gold$daily.returns),
                  row.names = NULL)
```

Format

A data frame with the following variables:

date Date (format: yyyy-mmm-dd).

log.returns Daily log-return.

`hypcube_lhs` *Latin Hypercube Sampling*

Description

Generates a Latin Hypercube Sampling (LHS) design matrix over the hypercube.

Usage

```
hypcube_lhs(n, d, lbound = NA_real_, ubound = NA_real_)
```

Arguments

n	Integer. The number of samples points of the hypercube.
d	Integer. The the dimension of the hypercube.
lbound	Numeric vector of length d specifying the lower bounds for each dimension of the d-dimensional hypercube.
ubound	Numeric vector of length d specifying the ubound bounds for each dimension of the d-dimensional hypercube.

Value

A n x d matrix containing the LHS design. Each element is scaled to the range defined by lbound and ubound.

References

McKay, M.D., Beckman, R.J., Conover, W.J. (1979) A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. Technometrics. 21(2), 239—245 (reprinted in 2000: Technometrics 42(1), 55–61).

Owen, A. B. (1992b) A central limit theorem for Latin hypercube sampling. JRSS Series B 54, 541-551.

Stein, M. (1987) Large sample properties of simulations using Latin hypercube sampling. Technometrics 29, 143-151.

Examples

```
x = hypcube_lhs(100, 2)
plot(x, xlim = c(0,1), ylim = c(0,1))
rug(x[,1]); rug(x[,2], side = 2)
x = hypcube_lhs(100, 2, lbound = c(-5,1), ubound = c(10,3))
plot(x, xlim = c(-5,10), ylim = c(1,3))
rug(x[,1]); rug(x[,2], side = 2)
```

---

hypcube_smc *Simple Monte Carlo Sampling*

Description

This function generates a Simple Monte Carlo (SMC) random design matrix over the hypercube.

Usage

```
hypcube_smc(n, d, lbound = NA_real_, ubound = NA_real_)
```

Arguments

n	Integer. The number of samples points of the hypercube.
d	Integer. The the dimension of the hypercube.
lbound	Numeric vector of length d specifying the lower bounds for each dimension of the d-dimensional hypercube.
ubound	Numeric vector of length d specifying the ubound bounds for each dimension of the d-dimensional hypercube.

Value

A n x d matrix containing the SMC design. Each element is scaled to the range defined by lbound and ubound.

Examples

```
x = hypcube_smc(100, 2)
plot(x, xlim = c(0,1), ylim = c(0,1))
rug(x[,1]); rug(x[,2], side = 2)
x = hypcube_smc(100, 2, lbound = c(-5,1), ubound = c(10,3))
plot(x, xlim = c(-5,10), ylim = c(1,3))
rug(x[,1]); rug(x[,2], side = 2)
```

---

hypvolgmm *Gaussian mixture-based hypervolume estimation of multivariate data*

Description

Estimates the hypervolume of a multivariate dataset by fitting a Gaussian Mixture Model (GMM).

Usage

```
hypvolgmm(
  data,
  method = c("lhs", "smc", "is", "none"),
  G = 1:9,
  prior = TRUE,
  nsim = 1e+06,
  hdlevel = NULL,
  GMM = NULL,
  ...
)
```

Arguments

data	A numeric vector, matrix, or data frame. If a matrix or data frame, rows correspond to observations and columns correspond to variables. Categorical variables and missing values are not allowed.
method	A character specifying the sampling scheme to be used for the estimation. Available options are: • "lhs" = Latin Hypercube Sampling (default); • "smc" = Simple Monte Carlo sampling; • "is" = Importance Sampling.
G	An integer vector specifying the numbers of mixture components to fit, then model selection if performed via BIC.
prior	A logical specifying if a prior (on the covariance) matrix should be used for regularization.
nsim	Integer specifying the number of simulations used in MC sampling.
hdlevel	A numerical value in the range (0,1] specifying the level of the highest density region (HDR) to be used for defining the GMM hull. If not provided, a default data-driven value is computed.
GMM	Optional Mclust or densityMclust object to be used. If provided, the data argument is ignored.
...	Further arguments passed to or from other methods.

Value

Returns a list containing the following elements:

- GMM An object of class densityMclust containing the fitted Gaussian mixture.
- method The sampling method employed.
- hdlevel The hdlevel used for definying the GMM hull.
- nsim The number of simulations used in MC sampling.
- ESS The effective sample size.
- h The density level of GMM hull.
- gamma The estimated parameter used for adjustment
- logvol The estimated log hypervolume.

Author(s)

Luca Scrucca

Examples

```
x = matrix(rnorm(100), ncol = 2)
mod = hypvolgmm(x)
summary(mod$GMM)
mod$logvol
```

`hypvolgmm_hdlevel` *Default HDR level defining the GMM hull*

Description

Compute default HDR level used to define the GMM hull. This is obtained by fitting a two-segment piecewise linear regression model (i.e., single change point model) to a grid of (α, h_α) values with α in $[0.9, 1]$.

Usage

```
hypvolgmm_hdlevel(dens, ...)
```

Arguments

`dens`	A vector of density estimates for the n observed data points.
`...`	Further arguments passed to or from other methods.

Value

Returns a list containing the following elements:

- `alpha` A vector of alpha values specifying the HDR levels.
- `h_alpha` A vector of HDR density values corresponding to `alpha`.
- `pcwsreg` The two-segment piecewise linear regression model.
- `hdl` The optimal HDR level selected as the change-point value.
- `h` The HDR density values corresponding to the optimal HDR level.

Author(s)

Luca Scrucca

Examples

```
x = subset(faithful, eruptions > 3)
mod = densityMclust(x, plot = FALSE)
out = hypvolgmm_hdlevel(mod$density)
with(out,
{
  plot(alpha, h_alpha, type = "b", pch = 20)
  lines(alpha[1:pcwsreg$c],
        pcwsreg$a1 * alpha[1:pcwsreg$c] + pcwsreg$b1,
        lty = 2, lwd = 2, col = "red2")
  lines(alpha[pcwsreg$c:length(alpha)],
        pcwsreg$a2 * alpha[pcwsreg$c:length(alpha)] + pcwsreg$b2,
        lty = 2, lwd = 2, col = "red2")
  abline(v = hdl, lty = 3, col = "red2", lwd = 2)
```

```
  abline(h = h, lty = 3, col = "red2", lwd = 2)
})

plot(x, xlim = c(2,6), ylim = c(60,100))
surfacePlot(data = mod$data, parameters = mod$parameters,
            what = "density", type = "contour",
            levels = c(out$h, min(mod$density)),
            xlim = c(2,6), ylim = c(60,100), lty = c(1,2),
            add = TRUE, drawlabels = FALSE)
```

`hypvoltmvnorm` *Approximate hypervolume for multivariate data*

Description

Approximate the hypervolume of a multivariate dataset by assuming a truncated multivariate normal (TMVN) distribution within given radius.

Usage

```
hypvoltmvnorm(data, radius = NULL, ...)
```

Arguments

`data`	A numeric vector, matrix, or data frame. If a matrix or data frame, rows correspond to observations and columns correspond to variables. Categorical variables and missing values are not allowed.
`radius`	A numerical value specifying the radius to be used for truncating the multivariate Gaussian distribution. If not provided is set to $sqrt(d + 1)$, where d is the dimensionality of the data, i.e. number of columns of `data`.
`...`	Further arguments passed to or from other methods.

Value

Returns a list with the following components:

- `sigma` The estimated covariance matrix.
- `radius` The radius used for computation.
- `logvol` The log of hypervolume.
- `logdens` The smallest log-density for observed data points within the radius.

Author(s)

Luca Scrucca

Examples

```
x1 = rnorm(1000)
x2 = 0.8*x1 + rnorm(1000)
x = cbind(x1, x2)
hypvoltmvnorm(x, radius = 1)
```

---

hypvolunif *Approximate hypervolume for multivariate data*

Description

Simple approximations for the hypervolume of a multivariate dataset by assuming a uniform distribution.

Usage

```
hypvolunif(data, pc = FALSE, logarithm = TRUE, ...)
```

Arguments

data	A numeric vector, matrix, or data frame. If a matrix or data frame, rows correspond to observations and columns correspond to variables. Categorical variables and missing values are not allowed.
pc	A logical value indicating whether the volume of hyperectangle enclosing the data (FALSE) or the volume of hyperectangle aligned with the principal axes enclosing the data should be computed.
logarithm	A logical value indicating whether or not the logarithm of the hypervolume should be returned.
...	Further arguments passed to or from other methods.

Value

Returns the (log) hypervolume from a uniform distribution aligned with the original axes or along the principal components.

Author(s)

Luca Scrucca

Examples

```
x1 = rnorm(1000)
x2 = 0.8*x1 + rnorm(1000)
x = cbind(x1, x2)
hypvolunif(x)
hypvolunif(x, pc = TRUE)
```

mclustAddons-internal *Internal* **mclustAddons** *functions*

Description

Internal functions not intended to be called directly by users.

Usage

```
as.MclustBounded(x, ...)

## Default S3 method:
as.MclustBounded(x, ...)

## S3 method for class 'densityMclustBounded'
as.MclustBounded(x, ...)

as.densityMclustBounded(x, ...)

## Default S3 method:
as.densityMclustBounded(x, ...)

## S3 method for class 'MclustBounded'
as.densityMclustBounded(x, ...)
```

Arguments

x	An object of class specific for the method.
...	Further arguments passed to or from other methods.

MclustBounded *Model-based clustering for bounded data*

Description

Clustering of bounded data via transformation-based approach for Gaussian mixtures.

Usage

```
MclustBounded(data, ...)

## S3 method for class 'MclustBounded'
summary(object, classification = TRUE, parameters = 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.
`...`	Further arguments passed to `densityMclustBounded()`. For a full description of available arguments see the corresponding help page.
`object`	An object of class `'MclustBounded'`.
`classification`	A logical, if `TRUE` a table of MAP classification/clustering of observations is printed.
`parameters`	A logical, if `TRUE` the estimated parameters of mixture components are printed.

Details

For more details see `vignette("mclustAddons")`

Value

Returns an object of class `'MclustBounded'`.

Author(s)

Luca Scrucca

References

Scrucca L. (2024) A model-based clustering approach for bounded data using transformation-based Gaussian mixture models. *Under review*. arXiv pre-print available at http://arxiv.org/abs/2412.13572

See Also

`densityMclustBounded()`, `predict.MclustBounded()`, `plot.MclustBounded()`.

`MclustBoundedParameters` *Recover parameters in the original scale*

Description

Given a GMM for bounded data, computes the means and variances in the original scale from the estimated mixture components parameters dataset using simulations.

Usage

```
MclustBoundedParameters(object, nsim = 1e+06, ...)
```

Arguments

object	An object of class 'MclustBounded' or 'densityMclustBounded'.
nsim	An integer specifying the number of simulations to employ.
...	Further arguments passed to or from other methods.

Examples

```
x = rlnorm(1000, 0, 1)
mod = densityMclustBounded(x, lbound = 0, lambda = 0)
summary(mod, parameters = TRUE)
plot(mod, what = "density")
# transformed parameters (from log-normal distribution)
# mean
with(mod$parameters,
     exp(mean + 0.5*variance$sigmasq))
# var
with(mod$parameters,
     (exp(variance$sigmasq) - 1)*exp(2*mean + variance$sigmasq))
# using simulations
MclustBoundedParameters(mod)
```

---

mclustMarginalParams *Marginal parameters from fitted GMMs via mclust*

Description

Function to compute the marginal parameters from a fitted Gaussian mixture models.

Usage

```
mclustMarginalParams(object, ...)

gmm2margParams(pro, mu, sigma, ...)
```

Arguments

object	An object of class Mclust or densityMclust.
...	Further arguments passed to or from other methods.
pro	A vector of mixing proportions for each mixture component.
mu	A matrix of mean vectors for each mixture component. For a d-variate dataset on G components, the matrix has dimension $(d \times G)$.
sigma	An array of covariance matrices for each mixture component. For a d-variate dataset on G components, the array has dimension $(d \times d \times G)$.

Details

Given a G-component GMM with estimated mixture weight π_k, mean vector μ_k, and covariance matrix Σ_k, for mixture component $k = 1, \ldots, G$, then the marginal distribution has:

- mean vector
$$\mu = \sum_{k=1}^{G} \pi_k \mu_k$$
- covariance matrix
$$\Sigma = \sum_{k=1}^{G} \pi_k \Sigma_k + \pi_k (\mu_k - \mu)'(\mu_k - \mu)$$

Value

Returns a list of two components for the mean and covariance of the marginal distribution.

Author(s)

Luca Scrucca

References

Frühwirth-Schnatter S. (2006) *Finite Mixture and Markov Switching Models*, Springer, Sec. 6.1.1

See Also

`mclust::Mclust()`, `mclust::densityMclust()`.

Examples

```
x = iris[,1:4]
mod = Mclust(x, G = 3)
mod$parameters$pro
mod$parameters$mean
mod$parameters$variance$sigma
mclustMarginalParams(mod)
```

`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(object, data = NULL, ...)

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

Arguments

`object`	An object of class `'Mclust'` or `'densityMclust'` obtained by fitting a Gaussian mixture via, respectively, `mclust::Mclust()` and `mclust::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.
`...`	Further arguments passed to or from other methods.

Details

For more details see `vignette("mclustAddons")`

Value

Returns an object of class `'MclustMEM'` with elements described in `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. doi: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"` for a plot of BIC values for the estimated models versus the number of components. • `"density"` for a plot of estimated density; if `data` is also provided the density is plotted over the given data points. • `"diagnostic"` for 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", "sqr`
	`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,`
	• For $d > 2$-dimensional data:
	`type = c("contour", "hdr"), gap = 0.2, grid = 100, nlevels = 11, levels = NULL, prob =`
	`col = grey(0.6), color.palette = blue2grey.colors, code{points.col = 1, points.cex =`

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. doi: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.MclustBounded *Plotting method for model-based clustering of bounded data*

Description

Plotting method for model-based clustering of bounded data

Usage

```
## S3 method for class 'MclustBounded'
plot(x, what = c("BIC", "classification", "uncertainty"), dimens = NULL, ...)
```

Arguments

x	An object of class 'MclustBounded'.
what	A string specifying the type of graph requested. Available choices are: • "BIC" Plot of BIC values used for choosing the number of clusters. • "classification" Plot showing the clustering. For data in more than two dimensions, a scatterplot of pairwise coordinate projections using the specified dimens is produced. • "uncertainty" Plot of classification uncertainty. For data in more than two dimensions, a scatterplot of pairwise coordinate projections using the specified dimens is produced.
dimens	A vector of integers specifying the dimensions of the coordinate projections.
...	Further arguments passed to or from other methods.

Value

No return value, called for side effects.

---

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. doi: 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

Predict density estimates 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. doi: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")
```

---

predict.MclustBounded *Model-based clustering estimation for bounded data*

Description

Predict clustering for univariate and multivariate bounded data based on Gaussian finite mixture models estimated by `MclustBounded()`.

Usage

```
## S3 method for class 'MclustBounded'
predict(object, newdata, ...)
```

Arguments

`object`	An object of class `'MclustBounded'` resulting from a call to `MclustBounded()`.
`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 `MclustBounded()`.
`...`	Further arguments passed to or from other methods.

Value

Returns a list of with the following components:

- `classification` A factor of predicted cluster labels for newdata.
- `z` A matrix whose $[i, k]$th entry is the probability that ith observation in `newdata` belongs to the kth cluster.

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. doi:10.1002/bimj.201800174

See Also

`MclustBounded()`, `plot.MclustBounded()`.

`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.

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

`rangepowerTransform` *Range–power transformation*

Description

Functions to compute univariate range–power transformation and its back-transform.

Usage

```
rangepowerTransform(x, lbound = NULL, ubound = NULL, lambda = 1)

rangepowerBackTransform(y, lbound = NULL, ubound = NULL, lambda = 1)
```

Arguments

`x`	A numeric vector of data values.
`lbound`	A numerical value of variable lower bound.
`ubound`	A numerical value of variable upper bound.
`lambda`	A numerical value for the power transformation.
`y`	A numeric vector of transformed data values.

Details

The *range-power transformation* can be applied to variables with bounded support.

Lower bound case

Suppose x is a univariate random variable with lower bounded support $\mathcal{S}_{\mathcal{X}} \equiv (l, \infty)$, where $l > -\infty$. Consider a preliminary *range transformation* defined as $x \mapsto (x - l)$, which maps $\mathcal{S}_{\mathcal{X}} \to \mathbb{R}^+$. The *range-power transformation* is a continuous monotonic transformation defined as

$$t(x;\lambda) = \begin{cases} \dfrac{(x-l)^\lambda - 1}{\lambda} & \text{if } \lambda \neq 0 \\ \log(x-l) & \text{if } \lambda = 0 \end{cases}$$

with back-transformation function

$$t^{-1}(y;\lambda) = \begin{cases} (\lambda y + 1)^{1/\lambda} + l & \text{if } \lambda \neq 0 \\ \exp(y) + l & \text{if } \lambda = 0 \end{cases}$$

Lower and upper bound case

Suppose x is a univariate random variable with bounded support $\mathcal{S}_{\mathcal{X}} \equiv (l, u)$, where $-\infty < l < u < +\infty$. Consider a preliminary *range transformation* defined as $x \mapsto (x - l)/(u - x)$, which

maps $\mathcal{S}_{\mathcal{X}} \to \mathbb{R}^+$.
In this case, the *range-power transformation* is a continuous monotonic transformation defined as

$$t(x;\lambda) = \begin{cases} \dfrac{\left(\dfrac{x-l}{u-x}\right)^{\lambda} - 1}{\lambda} & \text{if } \lambda \neq 0 \\ \log\left(\dfrac{x-l}{u-x}\right) & \text{if } \lambda = 0, \end{cases}$$

with back-transformation function

$$t^{-1}(y;\lambda) = \begin{cases} \dfrac{l + u(\lambda y + 1)^{1/\lambda}}{1 + (\lambda y + 1)^{1/\lambda}} & \text{if } \lambda \neq 0 \\ \dfrac{l + u\exp(y)}{1 + \exp(y)} & \text{if } \lambda = 0 \end{cases}$$

Value

Returns a vector of transformed or back-transformed values.

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. doi:10.1002/bimj.201800174

See Also

densityMclustBounded.

Examples

```
# Lower bound case
x = rchisq(1000, 5)
y = rangepowerTransform(x, lbound = 0, lambda = 1/3)
par(mfrow=c(2,2))
hist(x, main = NULL, breaks = 21); rug(x)
hist(y, xlab = "y = t(x)", main = NULL, breaks = 21); rug(y)
xx = rangepowerBackTransform(y, lbound = 0, lambda = 1/3)
hist(xx, xlab = "t^-1(y) = x", main = NULL, breaks = 21); rug(xx)
plot(x, xx, ylab = "t^-1(y)"); abline(0,1)

# Lower and upper bound case
x = rbeta(1000, 2, 1)
y = rangepowerTransform(x, lbound = 0, ubound = 1, lambda = 0)
par(mfrow=c(2,2))
hist(x, main = NULL, breaks = 21); rug(x)
hist(y, xlab = "y = t(x)", main = NULL, breaks = 21); rug(y)
```

```
xx = rangepowerBackTransform(y, lbound = 0, ubound = 1, lambda = 0)
hist(xx, xlab = "t^-1(y) = x", main = NULL, breaks = 21); rug(xx)
plot(x, xx, ylab = "t^-1(y)"); abline(0,1)
```

---

suicide *Suicide data*

Description

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

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.

VaR *Financial risk measures*

Description

Generic functions for computing Value-at-Risk (VaR) and Expected Shortfall (ES).

Usage

```
VaR(object, ...)

ES(object, ...)
```

Arguments

object	An object of class specific for the method.
...	Further arguments passed to or from other methods.

VaR.GMMlogreturn *Risk measures from Gaussian mixtures modeling*

Description

Value-at-Risk (VaR) and Expected Shortfall (ES) from the fit of Gaussian mixtures provided by GMMlogreturn() function.

Usage

```
## S3 method for class 'GMMlogreturn'
VaR(object, alpha, ...)

## S3 method for class 'GMMlogreturn'
ES(object, alpha, ...)
```

Arguments

object	An object of class 'GMMlogreturn'.
alpha	A vector of values in the interval $(0, 1)$ for which the risk measures should be calculated.
...	Further arguments passed to or from other methods.

Details

VaR(α) is the maximum potential loss over a specified time horizon with probability equal to the confidence level $1 - \alpha$.

ES(α) is the expected loss given that the loss exceeds the VaR(α) level.

Value

Returns a numerical value corresponding to VaR or ES at given level(s).

References:

Ruppert Matteson (2015) Statistics and Data Analysis for Financial Engineering, Springer, Chapter 19.

Cizek Hardle Weron (2011) Statistical Tools for Finance and Insurance, 2nd ed., Springer, Chapter 2.

Examples

```
z = sample(1:2, size = 250, replace = TRUE, prob = c(0.8, 0.2))
y = double(length(z))
y[z == 1] = rnorm(sum(z == 1), 0, 1)
y[z == 2] = rnorm(sum(z == 2), -0.5, 2)
GMM = GMMlogreturn(y)
alpha = seq(0.01, 0.1, by = 0.001)
```

```
matplot(alpha, data.frame(VaR = VaR(GMM, alpha),
                          ES = ES(GMM, alpha)),
        type = "l", col = c(2,4), lty = 1, lwd = 2,
        xlab = expression(alpha), ylab = "Loss")
legend("topright", col = c(2,4), lty = 1, lwd = 2,
       legend = c("VaR", "ES"), inset = 0.02)
```

Index