Package 'mixtools'

February 7, 2020

Version 1.2.0

Date 2020-02-05

Title Tools for Analyzing Finite Mixture Models

Depends R (>= 3.5.0)

Imports kernlab, MASS, segmented, stats, survival

Description Analyzes finite mixture models for various parametric and semiparametric settings. This includes mixtures of parametric distributions (normal, multivariate normal, multinomial, gamma), various Reliability Mixture Models (RMMs), mixtures-of-regressions settings (linear regression, logistic regression, Poisson regression, linear regression with changepoints, predictor-dependent mixing proportions, random effects regressions, hierarchical mixtures-of-experts), and tools for selecting the number of components (bootstrapping the likelihood ratio test statistic, mixturegrams, and model selection criteria). Bayesian estimation of mixtures-of-linear-regressions models is available as well as a novel data depth method for obtaining credible bands. This package is based upon work supported by the National Science Foundation under Grant No. SES-0518772.

License GPL (>= 2)

NeedsCompilation yes

Author Derek Young [aut, cre] (<https://orcid.org/0000-0002-3048-3803>),
Tatiana Benaglia [aut],
Didier Chauveau [aut],
David Hunter [aut],
Ryan Elmore [ctb],
Thomas Hettmansperger [ctb],
Hoben Thomas [ctb],
Fengjuan Xuan [ctb]

Maintainer Derek Young <derek.young@uky.edu>

Repository CRAN

Date/Publication 2020-02-07 12:20:02 UTC

R topics documented:

`boot.comp` *Performs Parametric Bootstrap for Sequentially Testing the Number of Components in Various Mixture Models*

Description

Performs a parametric bootstrap by producing B bootstrap realizations of the likelihood ratio statistic for testing the null hypothesis of a k-component fit versus the alternative hypothesis of a (k+1)-component fit to various mixture models. This is performed for up to a specified number of maximum components, k. A p-value is calculated for each test and once the p-value is above a specified significance level, the testing terminates. An optional histogram showing the distribution of the likelihood ratio statistic along with the observed statistic can also be produced.

Usage

```
boot.comp(y, x = NULL, N = NULL, max.comp = 2, B = 100,
          sig = 0.05, arbmean = TRUE, arbvar = TRUE,
          mix.type = c("logisregmix", "multmix", "mvnormalmix",
          "normalmix", "poisregmix", "regmix", "regmix.mixed",
          "repnormmix"), hist = TRUE, ...)
```

Arguments

y	The raw data for multmix, mvnormalmix, normalmix, and repnormmix and the response values for logisregmix, poisregmix, and regmix. See the documentation concerning their respective EM algorithms for specific structure of the raw data.
x	The predictor values required only for the regression mixtures logisregmix, poisregmix, and regmix. A column of 1s for the intercept term must not be included! See the documentation concerning their respective EM algorithms for specific structure of the predictor values.
N	An n-vector of number of trials for the logistic regression type logisregmix. If NULL, then N is an n-vector of 1s for binary logistic regression.
max.comp	The maximum number of components to test for. The default is 2. This function will perform a test of k-components versus (k+1)-components sequentially until we fail to reject the null hypothesis. This decision rule is governed by the calculated p-value and sig.
B	The number of bootstrap realizations of the likelihood ratio statistic to produce. The default is 100, but ideally, values of 1000 or more would be more acceptable.
sig	The significance level for which to compare the p-value against when performing the test of k-components versus (k+1)-components.
arbmean	If FALSE, then a scale mixture analysis can be performed for mvnormalmix, normalmix, regmix, or repnormmix. The default is TRUE.
arbvar	If FALSE, then a location mixture analysis can be performed for mvnormalmix, normalmix, regmix, or repnormmix. The default is TRUE.
mix.type	The type of mixture analysis you wish to perform. The data inputted for y and x depend on which type of mixture is selected. logisregmix corresponds to a mixture of logistic regressions. multmix corresponds to a mixture of multinomials with data determined by the cut-point method. mvnormalmix corresponds to a mixture of multivariate normals. normalmix corresponds to a mixture of univariate normals. poisregmix corresponds to a mixture of Poisson regressions. regmix corresponds to a mixture of regressions with normal components. regmix.mixed corresponds to a mixture of regressions with random or mixed effects. repnormmix corresponds to a mixture of normals with repeated measurements.
hist	An argument to provide a matrix plot of histograms for the boostrapped likelihood ratio statistic.
...	Additional arguments passed to the various EM algorithms for the mixture of interest.

Value

boot.comp returns a list with items:

p.values	The p-values for each test of k-components versus (k+1)-components.
log.lik	The B bootstrap realizations of the likelihood ratio statistic.
obs.log.lik	The observed likelihood ratio statistic for each test which is used in determining the p-values.

References

McLachlan, G. J. and Peel, D. (2000) *Finite Mixture Models*, John Wiley \& Sons, Inc.

See Also

logisregmixEM, multmixEM, mvnormalmixEM, normalmixEM, poisregmixEM, regmixEM, regmixEM.mixed, repnormmixEM

Examples

```
## Bootstrapping to test the number of components on the RTdata.

data(RTdata)
set.seed(100)
x <- as.matrix(RTdata[, 1:3])
y <- makemultdata(x, cuts = quantile(x, (1:9)/10))$y
a <- boot.comp(y = y, max.comp = 1, B = 5, mix.type = "multmix",
               epsilon = 1e-3)
a$p.values
```

---

boot.se *Performs Parametric Bootstrap for Standard Error Approximation*

Description

Performs a parametric bootstrap by producing B bootstrap samples for the parameters in the specified mixture model.

Usage

```
boot.se(em.fit, B = 100, arbmean = TRUE, arbvar = TRUE,
        N = NULL, ...)
```

Arguments

em.fit	An object of class mixEM. The estimates produced in em.fit will be used as the parameters for the distribution from which we generate the bootstrap data.
B	The number of bootstrap samples to produce. The default is 100, but ideally, values of 1000 or more would be more acceptable.
arbmean	If FALSE, then a scale mixture analysis can be performed for mvnormalmix, normalmix, regmix, or repnormmix. The default is TRUE.
arbvar	If FALSE, then a location mixture analysis can be performed for mvnormalmix, normalmix, regmix, or repnormmix. The default is TRUE.
N	An n-vector of number of trials for the logistic regression type logisregmix. If NULL, then N is an n-vector of 1s for binary logistic regression.
...	Additional arguments passed to the various EM algorithms for the mixture of interest.

Value

boot.se returns a list with the bootstrap samples and standard errors for the mixture of interest.

References

McLachlan, G. J. and Peel, D. (2000) *Finite Mixture Models*, John Wiley \& Sons, Inc.

Examples

```
## Bootstrapping standard errors for a regression mixture case.

data(NOdata)
attach(NOdata)
set.seed(100)
em.out <- regmixEM(Equivalence, NO, arbvar = FALSE)
out.bs <- boot.se(em.out, B = 10, arbvar = FALSE)
out.bs
```

CO2data *GNP and CO2 Data Set*

Description

This data set gives the gross national product (GNP) per capita in 1996 for various countries as well as their estimated carbon dioxide (CO2) emission per capita for the same year.

Usage

```
data(CO2data)
```

Format

This data frame consists of 28 countries and the following columns:

- GNPThe gross national product per capita in 1996.
- CO2The estimated carbon dioxide emission per capita in 1996.
- countryAn abbreviation pertaining to the country measured (e.g., "GRC" = Greece and "CH" = Switzerland).

References

Hurn, M., Justel, A. and Robert, C. P. (2003) Estimating Mixtures of Regressions, *Journal of Computational and Graphical Statistics* **12(1)**, 55–79.

---

compCDF *Plot the Component CDF*

Description

Plot the components' CDF via the posterior probabilities.

Usage

```
compCDF(data, weights,
        x=seq(min(data, na.rm=TRUE), max(data, na.rm=TRUE), len=250),
        comp=1:NCOL(weights), makeplot=TRUE, ...)
```

Arguments

data	A matrix containing the raw data. Rows are subjects and columns are repeated measurements.
weights	The weights to compute the empirical CDF; however, most of time they are the posterior probabilities.
x	The points at which the CDFs are to be evaluated.
comp	The mixture components for which CDFs are desired.
makeplot	Logical: Should a plot be produced as a side effect?
...	Additional arguments (other than lty and type, which are already used) to be passed directly to plot and lines functions.

Details

When makeplot is TRUE, a line plot is produced of the CDFs evaluated at x. The plot is not a step function plot; the points $(x, CDF(x))$ are simply joined by line segments.

Value

A matrix with length(comp) rows and length(x) columns in which each row gives the CDF evaluated at each point of x.

References

McLachlan, G. J. and Peel, D. (2000) *Finite Mixture Models*, John Wiley \& Sons, Inc.

Elmore, R. T., Hettmansperger, T. P. and Xuan, F. (2004) The Sign Statistic, One-Way Layouts and Mixture Models, *Statistical Science* **19(4)**, 579–587.

See Also

`makemultdata`, `multmixmodel.sel`, `multmixEM`.

Examples

```
## The sulfur content of the coal seams in Texas

set.seed(100)

A <- c(1.51, 1.92, 1.08, 2.04, 2.14, 1.76, 1.17)
B <- c(1.69, 0.64, .9, 1.41, 1.01, .84, 1.28, 1.59)
C <- c(1.56, 1.22, 1.32, 1.39, 1.33, 1.54, 1.04, 2.25, 1.49)
D <- c(1.3, .75, 1.26, .69, .62, .9, 1.2, .32)
E <- c(.73, .8, .9, 1.24, .82, .72, .57, 1.18, .54, 1.3)

dis.coal <- makemultdata(A, B, C, D, E,
                         cuts = median(c(A, B, C, D, E)))
temp <- multmixEM(dis.coal)

## Now plot the components' CDF via the posterior probabilities

compCDF(dis.coal$x, temp$posterior, xlab="Sulfur", ylab="", main="empirical CDFs")
```

`density.npEM` *Normal kernel density estimate for nonparametric EM output*

Description

Takes an object of class `npEM` and returns an object of class `density` giving the kernel density estimate for the selected component and, if applicable, the selected block.

Usage

```
## S3 method for class 'npEM'
density(x, u=NULL, component=1, block=1, scale=FALSE, ...)
```

Arguments

`x`	An object of class `npEM` such as the output of the `npEM` or `spEMsymloc` functions.
`u`	Vector of points at which the density is to be evaluated
`component`	Mixture component number; should be an integer from 1 to the number of columns of `x$posteriors`.

block	Block of repeated measures. Only applicable in repeated measures case, for which x$blockid exists; should be an integer from 1 to max(x$blockid).
scale	Logical: If TRUE, multiply the density values by the corresponding mixing proportions found in x$lambdahat
...	Additional arguments; not used by this method.

Details

The bandwidth is taken to be the same as that used to produce the npEM object, which is given by x$bandwidth.

Value

density.npEM returns a list of type "density". See density for details. In particular, the output of density.npEM may be used directly by functions such as plot or lines.

See Also

npEM, spEMsymloc, plot.npEM

Examples

```
## Look at histogram of Old Faithful waiting times
data(faithful)
Minutes <- faithful$waiting
hist(Minutes, freq=FALSE)

## Superimpose equal-variance normal mixture fit:
set.seed(100)
nm <- normalmixEM(Minutes, mu=c(50,80), sigma=5, arbvar=FALSE, fast=TRUE)
x <- seq(min(Minutes), max(Minutes), len=200)
for (j in 1:2)
  lines(x, nm$lambda[j]*dnorm(x, mean=nm$mu[j], sd=nm$sigma), lwd=3, lty=2)

## Superimpose several semiparametric fits with different bandwidths:
bw <- c(1, 3, 5)
for (i in 1:3) {
  sp <- spEMsymloc(Minutes, c(50,80), bw=bw[i], eps=1e-3)
  for (j in 1:2)
    lines(density(sp, component=j, scale=TRUE), col=1+i, lwd=2)
}
legend("topleft", legend=paste("Bandwidth =",bw), fill=2:4)
```

`density.spEM` *Normal kernel density estimate for semiparametric EM output*

Description

Takes an object of class `spEM` and returns an object of class `density` giving the kernel density estimate.

Usage

```
## S3 method for class 'spEM'
density(x, u=NULL, component=1, block=1, scale=FALSE, ...)
```

Arguments

`x`	An object of class `npEM` such as the output of the `npEM` or `spEMsymloc` functions.
`u`	Vector of points at which the density is to be evaluated
`component`	Mixture component number; should be an integer from 1 to the number of columns of `x$posteriors`.
`block`	Block of repeated measures. Only applicable in repeated measures case, for which `x$blockid` exists; should be an integer from 1 to `max(x$blockid)`.
`scale`	Logical: If TRUE, multiply the density values by the corresponding mixing proportions found in `x$lambdahat`
`...`	Additional arguments; not used by this method.

Details

The bandwidth is taken to be the same as that used to produce the `npEM` object, which is given by `x$bandwidth`.

Value

`density.spEM` returns a list of type `"density"`. See `density` for details. In particular, the output of `density.spEM` may be used directly by functions such as `plot` or `lines`.

See Also

`spEM`, `spEMsymloc`, `plot.spEM`

Examples

```
set.seed(100)
mu <- matrix(c(0, 15), 2, 3)
sigma <- matrix(c(1, 5), 2, 3)
x <- rmvnormmix(300, lambda = c(.4,.6), mu = mu, sigma = sigma)
```

```
d <- spEM(x, mu0 = 2, blockid = rep(1,3), constbw = TRUE)
plot(d, xlim=c(-10, 40), ylim = c(0, .16), xlab = "", breaks = 30,
     cex.lab=1.5, cex.axis=1.5) # plot.spEM calls density.spEM here
```

---

depth *Elliptical and Spherical Depth*

Description

Computation of spherical or elliptical depth.

Usage

```
depth(pts, x, Cx = var(x))
```

Arguments

pts	A kxd matrix containing the k points that one wants to compute the depth. Each row is a point.
x	A nxd matrix containing the reference data. Each row is an observation.
Cx	A dxd scatter matrix for the data x where the default is var(x). When Cx = I(d), it returns the sphercial depth.

Value

depth returns a k-vector where each entry is the elliptical depth of a point in pts.

Note

depth is used in regcr.

References

Elmore, R. T., Hettmansperger, T. P. and Xuan, F. (2000) Spherical Data Depth and a Multivariate Median, *Proceedings of Data Depth: Robust Multivariate Statistical Analysis, Computational Geometry and Applications*.

See Also

regcr

Examples

```
set.seed(100)
x <- matrix(rnorm(200),nc = 2)
depth(x[1:3, ], x)
```

dmvnorm *The Multivariate Normal Density*

Description

Density and log-density for the multivariate normal distribution with mean equal to `mu` and variance matrix equal to `sigma`.

Usage

```
dmvnorm(y, mu=NULL, sigma=NULL)
logdmvnorm(y, mu=NULL, sigma=NULL)
```

Arguments

`y`	Either a d - vector or an $n \times d$ matrix, where d is the dimension of the normal distribution and n is the number of points at which the density is to be evaluated.
`mu`	d - vector: Mean of the normal distribution (or NULL uses the origin as default)
`sigma`	This $d \times d$ matrix is the variance matrix of the normal distribution (or NULL uses the identity matrix as default)

Details

This code is written to be efficient, using the qr-decomposition of the covariance matrix (and using it only once, rather than recalculating it for both the determinant and the inverse of `sigma`).

Value

`dmvnorm` gives the densities, while `logdmvnorm` gives the logarithm of the densities.

See Also

`qr`, `qr.solve`, `dnorm`, `rmvnorm`

ellipse *Draw Two-Dimensional Ellipse Based on Mean and Covariance*

Description

Draw a two-dimensional ellipse that traces a bivariate normal density contour for a given mean vector, covariance matrix, and probability content.

Usage

```
ellipse(mu, sigma, alpha = .05, npoints = 250, newplot = FALSE,
        draw = TRUE, ...)
```

Arguments

mu	A 2-vector giving the mean.
sigma	A 2x2 matrix giving the covariance matrix.
alpha	Probability to be excluded from the ellipse. The default value is alpha = .05, which results in a 95% ellipse.
npoints	Number of points comprising the border of the ellipse.
newplot	If newplot = TRUE and draw = TRUE, plot the ellipse on a new plot. If newplot = FALSE and draw = TRUE, add the ellipse to an existing plot.
draw	If TRUE, draw the ellipse.
...	Graphical parameters passed to lines or plot command.

Value

ellipse returns an npointsx2 matrix of the points forming the border of the ellipse.

References

Johnson, R. A. and Wichern, D. W. (2002) *Applied Multivariate Statistical Analysis, Fifth Edition*, Prentice Hall.

See Also

regcr

Examples

```
## Produce a 95% ellipse with the specified mean and covariance structure.

mu <- c(1, 3)
sigma <- matrix(c(1, .3, .3, 1.5), 2, 2)

ellipse(mu, sigma, npoints = 200, newplot = TRUE)
```

---

expRMM_EM *EM algorithm for Reliability Mixture Models (RMM) with right Censoring*

Description

Parametric EM algorithm for univariate finite mixture of exponentials distributions with randomly right censored data.

Usage

```
expRMM_EM(x, d=NULL, lambda = NULL, rate = NULL, k = 2,
  complete = "tdz", epsilon = 1e-08, maxit = 1000, verb = FALSE)
```

Arguments

x	A vector of n real positive lifetime (possibly censored) durations. If d is not NULL then a vector of random censoring times c occurred, so that $x = min(x, c)$ and $d = I(x <= c)$.
d	The vector of censoring indication, where 1 means observed lifetime data, and 0 means censored lifetime data.
lambda	Initial value of mixing proportions. If NULL, then lambda is set to rep(1/k,k).
rate	Initial value of component exponential rates, all set to 1 if NULL.
k	Number of components of the mixture.
complete	Nature of complete data involved within the EM machinery, can be "tdz" for (t,d,z) (the default), or "xz" for (x,z) (see Bordes L. and Chauveau D. (2016) reference below).
epsilon	Tolerance limit for declaring algorithm convergence based on the change between two consecutive iterations.
maxit	The maximum number of iterations allowed, convergence may be declared before maxit iterations (see epsilon above).
verb	If TRUE, print updates for every iteration of the algorithm as it runs

Value

expRMM_EM returns a list of class "mixEM" with the following items:

x	The input data.
d	The input censoring indicator.
lambda	The estimates for the mixing proportions.
rate	The estimates for the component rates.
loglik	The log-likelihood value at convergence of the algorithm.
posterior	An $n \times k$ matrix of posterior probabilities for observation, after convergence of the algorithm.
all.loglik	The sequence of log-likelihoods over iterations.
all.lambda	The sequence of mixing proportions over iterations.
all.rate	The sequence of component rates over iterations.
ft	A character vector giving the name of the function.

Author(s)

Didier Chauveau

References

- Bordes, L., and Chauveau, D. (2016), Stochastic EM algorithms for parametric and semiparametric mixture models for right-censored lifetime data, Computational Statistics, Volume 31, Issue 4, pages 1513-1538. http://link.springer.com/article/10.1007/s00180-016-0661-7

See Also

Related functions: plotexpRMM, summary.mixEM.

Other models and algorithms for censored lifetime data: weibullRMM_SEM, spRMM_SEM.

Examples

```
n <- 300 # sample size
m <- 2   # number of mixture components
lambda <- c(1/3,1-1/3); rate <- c(1,1/10) # mixture parameters
set.seed(1234)
x <- rexpmix(n, lambda, rate) # iid ~ exponential mixture
cs <- runif(n,0,max(x)) # Censoring (uniform) and incomplete data
t <- apply(cbind(x,cs),1,min) # observed or censored data
d <- 1*(x <= cs)              # censoring indicator

###### EM for RMM, exponential lifetimes
l0 <- rep(1/m,m); r0 <- c(1, 0.5) # "arbitrary" initial values
a <- expRMM_EM(t, d, lambda = l0, rate = r0, k = m)
summary(a)                # EM estimates etc
plotexpRMM(a, lwd=2) # default plot of EM sequences
plot(a, which=2) # or equivalently, S3 method for "mixEM" object
```

`flaremixEM` *EM Algorithm for Mixtures of Regressions with Flare*

Description

Returns output for 2-component mixture of regressions with flaring using an EM algorithm with one step of Newton-Raphson requiring an adaptive barrier for maximization of the objective function. A mixture of regressions with flare occurs when there appears to be a common regression relationship for the data, but the error terms have a mixture structure of one normal component and one exponential component.

Usage

```
flaremixEM(y, x, lambda = NULL, beta = NULL, sigma = NULL,
           alpha = NULL, nu = NULL, epsilon = 1e-04,
           maxit = 10000, verb = FALSE, restart = 50)
```

Arguments

y	An n-vector of response values.
x	An n-vector of predictor values. An intercept term will be added by default.
lambda	Initial value of mixing proportions. Entries should sum to 1.

beta	Initial value of beta parameters. Should be a 2x2 matrix where the columns correspond to the component.
sigma	A vector of standard deviations.
alpha	A scalar for the exponential component's rate.
nu	A vector specifying the barrier constants to use. The first barrier constant where the algorithm converges is used.
epsilon	The convergence criterion.
maxit	The maximum number of iterations.
verb	If TRUE, then various updates are printed during each iteration of the algorithm.
restart	The number of times to restart the algorithm in case convergence is not attained. The default is 50.

Value

flaremixEM returns a list of class mixEM with items:

x	The set of predictors (which includes a column of 1's).
y	The response values.
posterior	An nx2 matrix of posterior probabilities for observations.
lambda	The final mixing proportions.
beta	The final regression coefficients.
sigma	The final standard deviations.
alpha	The final exponential rate.
loglik	The final log-likelihood.
all.loglik	A vector of each iteration's log-likelihood.
ft	A character vector giving the name of the function.

See Also

regmixEM

Examples

```
## Simulation output.

set.seed(100)
j=1
while(j == 1){
    x1 <- runif(30, 0, 10)
    x2 <- runif(20, 10, 20)
    x3 <- runif(30, 20, 30)
    y1 <- 3+4*x1+rnorm(30, sd = 1)
    y2 <- 3+4*x2+rexp(20, rate = .05)
    y3 <- 3+4*x3+rnorm(30, sd = 1)
    x <- c(x1, x2, x3)
    y <- c(y1, y2, y3)
```

```
    nu <- (1:30)/2

    out <- try(flaremixEM(y, x, beta = c(3, 4), nu = nu,
               lambda = c(.75, .25), sigma = 1), silent = TRUE)
    if(any(class(out) == "try-error")){
        j <- 1
    } else j <- 2
}

out[4:7]
plot(x, y, pch = 19)
abline(out$beta)
```

---

gammamixEM *EM Algorithm for Mixtures of Gamma Distributions*

Description

Return EM algorithm output for mixtures of gamma distributions.

Usage

```
gammamixEM(x, lambda = NULL, alpha = NULL, beta = NULL, k = 2,
           mom.start = TRUE, fix.alpha = FALSE, epsilon = 1e-08,
           maxit = 1000, maxrestarts = 20, verb = FALSE)
```

Arguments

x	A vector of length n consisting of the data.
lambda	Initial value of mixing proportions. If NULL, then lambda is random from a uniform Dirichlet distribution (i.e., its entries are uniform random and then it is normalized to sum to 1).
alpha	Starting value of vector of component shape parameters. If non-NULL, alpha must be of length k if allowing different component shape parameters, or a single value if fix.alpha = TRUE. If NULL, then the initial value is estimated by partitioning the data into k regions (with lambda determining the proportion of values in each region) and then calculating the method of moments estimates.
beta	Starting value of vector of component scale parameters. If non-NULL and a vector, k is set to length(beta). If NULL, then the initial value is estimated the same method described for alpha.
k	Number of components. Initial value ignored unless alpha and beta are both NULL.
mom.start	Logical to indicate if a method of moments starting value strategy should be implemented. If TRUE, then only unspecified starting values will be generated according to this strategy.

epsilon	The convergence criterion. Convergence is declared when the change in the observed data log-likelihood increases by less than epsilon.
fix.alpha	Logical to indicate if the components should have a common shape parameter alpha estimated. The default is FALSE.
maxit	The maximum number of iterations.
maxrestarts	The maximum number of restarts allowed in case of a problem with the particular starting values chosen (each restart uses randomly chosen starting values).
verb	If TRUE, then various updates are printed during each iteration of the algorithm.

Value

gammamixEM returns a list of class mixEM with items:

x	The raw data.
lambda	The final mixing proportions.
gamma.pars	A 2xk matrix where each column provides the component estimates of alpha and beta.
loglik	The final log-likelihood.
posterior	An nxk matrix of posterior probabilities for observations.
all.loglik	A vector of each iteration's log-likelihood. This vector includes both the initial and the final values; thus, the number of iterations is one less than its length.
ft	A character vector giving the name of the function.

References

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977) Maximum Likelihood From Incomplete Data Via the EM Algorithm, *Journal of the Royal Statistical Society, Series B*, **39(1)**, 1–38.

Young, D. S., Chen, X., Hewage, D., and Nilo-Poyanco, R. (2019) Finite Mixture-of-Gamma Distributions: Estimation, Inference, and Model-Based Clustering, *Advances in Data Analysis and Classification*, **13(4)**, 1053–1082.

Examples

```
##Analyzing a 3-component mixture of gammas.

set.seed(100)
x <- c(rgamma(200, shape = 0.2, scale = 14), rgamma(200,
     shape = 32, scale = 10), rgamma(200, shape = 5, scale = 6))
out <- gammamixEM(x, lambda = c(1, 1, 1)/3, verb = TRUE)
out[2:4]
```

Habituationdata *Infant habituation data*

Description

From Thomas et al (2011):

"Habituation is a standard method of studying infant behaviors. Indeed, much of what is known about infant memory and perception rests on habituation methods. Six-month infants (n = 51) were habituated to a checker-board pattern on two occasions, one week apart. On each occasion, the infant was presented with the checkerboard pattern and the length of time the infant viewed the pattern before disengaging was recorded; this denoted the end of a trial. After disengagement, another trial was presented. The procedure was implemented for eleven trials. The conventional index of habituation performance is the summed observed fixation to the checkerboard pattern over the eleven trials. Thus, an index of reliability focuses on how these fixation times, in seconds, on the two assessment occasions correlate: $r = .29$."

Usage

```
data(Habituationdata)
```

Format

A data frame with two variables, `m1` and `m2`, and 51 cases. The two variables are the summed observations times for the two occasions described above.

Author(s)

Hoben Thomas

Source

Original source: Thomas et al. (2011). See references section.

References

Thomas, H., Lohaus, A., and Domsch, H. (2011), Extensions of Reliability Theory, in Nonparametric Statistics and Mixture Models: A Festschrift in Honor of Thomas Hettmansperger (Singapore: World Scientific), pp. 309-316.

hmeEM *EM Algorithm for Mixtures-of-Experts*

Description

Returns EM algorithm output for a mixture-of-experts model. Currently, this code only handles a 2-component mixture-of-experts, but will be extended to the general k-component hierarchical mixture-of-experts.

Usage

```
hmeEM(y, x, lambda = NULL, beta = NULL, sigma = NULL, w = NULL,
      k = 2, addintercept = TRUE, epsilon = 1e-08,
      maxit = 10000, verb = FALSE)
```

Arguments

y	An n-vector of response values.
x	An nxp matrix of predictors. See addintercept below.
lambda	Initial value of mixing proportions, which are modeled as an inverse logit function of the predictors. Entries should sum to 1. If NULL, then lambda is taken as 1/k for each x.
beta	Initial value of beta parameters. Should be a pxk matrix, where p is the number of columns of x and k is number of components. If NULL, then beta has standard normal entries according to a binning method done on the data.
sigma	A vector of standard deviations. If NULL, then $1/\texttt{sigma}^2$ has random standard exponential entries according to a binning method done on the data.
w	A p-vector of coefficients for the way the mixing proportions are modeled. See lambda.
k	Number of components. Currently, only k=2 is accepted.
addintercept	If TRUE, a column of ones is appended to the x matrix before the value of p is calculated.
epsilon	The convergence criterion.
maxit	The maximum number of iterations.
verb	If TRUE, then various updates are printed during each iteration of the algorithm.

Value

hmeEM returns a list of class mixEM with items:

x	The set of predictors (which includes a column of 1's if addintercept = TRUE).
y	The response values.
w	The final coefficients for the functional form of the mixing proportions.

lambda	An nxk matrix of the final mixing proportions.
beta	The final regression coefficients.
sigma	The final standard deviations. If arbmean = FALSE, then only the smallest standard deviation is returned. See scale below.
loglik	The final log-likelihood.
posterior	An nxk matrix of posterior probabilities for observations.
all.loglik	A vector of each iteration's log-likelihood.
restarts	The number of times the algorithm restarted due to unacceptable choice of initial values.
ft	A character vector giving the name of the function.

References

Jacobs, R. A., Jordan, M. I., Nowlan, S. J. and Hinton, G. E. (1991) Adaptive Mixtures of Local Experts, *Neural Computation* **3(1)**, 79–87.

McLachlan, G. J. and Peel, D. (2000) *Finite Mixture Models*, John Wiley \& Sons, Inc.

See Also

regmixEM

Examples

```
## EM output for NOdata.

data(NOdata)
attach(NOdata)
set.seed(100)
em.out <- regmixEM(Equivalence, NO)
hme.out <- hmeEM(Equivalence, NO, beta = em.out$beta)
hme.out[3:7]
```

---

ise.npEM *Integrated Squared Error for a selected density from npEM output*

Description

Computes the integrated squared error for a selected estimated density from npEM output (selected by specifying the component and block number), relative to a true pdf that must be specified by the user. The range for the numerical integration must be specified. This function also returns (by default) a plot of the true and estimated densities.

Usage

```
ise.npEM(npEMout, component=1, block=1, truepdf, lower=-Inf,
         upper=Inf, plots = TRUE, ...)
```

Arguments

npEMout	An object of class npEM such as the output of the npEM function
component, block	Component and block of particular density to analyze from npEMout.
truepdf	an R function taking a numeric first argument and returning a numeric vector of the same length. Returning a non-finite element will generate an error.
lower, upper	the limits of integration. Can be infinite.
plots	logical: Should plots be produced?
...	additional arguments to be passed to truepdf (and that may be mandatory like, e.g., the df = argument of dt). Remember to use argument names not matching those of ise.npRM.

Details

This function calls the wkde (weighted kernel density estimate) function.

Value

Just as for the integrate function, a list of class "integrate" with components

value	the final estimate of the integral.
abs.error	estimate of the modulus of the absolute error.
subdivisions	the number of subintervals produced in the subdivision process.
message	"OK" or a character string giving the error message.
call	the matched call.

References

- Benaglia, T., Chauveau, D., and Hunter, D. R. (2009), An EM-like algorithm for semi- and non-parametric estimation in multivariate mixtures, Journal of Computational and Graphical Statistics, 18, 505-526.
- Benaglia, T., Chauveau, D., Hunter, D. R., and Young, D. (2009), mixtools: An R package for analyzing finite mixture models. Journal of Statistical Software, 32(6):1-29.

See Also

npEM, wkde, integrate

Examples

```
# Mixture with mv gaussian model
set.seed(100)
m <- 2 # no. of components
r <- 3 # no. of repeated measures (coordinates)
lambda <- c(0.4, 0.6)
# Note:  Need first 2 coordinates conditionally iid due to block structure
mu <- matrix(c(0, 0, 0, 3, 3, 5), m, r, byrow=TRUE) # means
```

```
sigma <- matrix(rep(1, 6), m, r, byrow=TRUE) # stdevs
blockid = c(1,1,2) # block structure of coordinates
n <- 200
x <- rmvnormmix(n, lambda, mu, sigma) # simulated data

# fit the model with "arbitrary" initial centers
centers <- matrix(c(0, 0, 0, 4, 4, 4), 2, 3, byrow=TRUE)
a <- npEM(x, centers, blockid, eps=1e-8, verb=FALSE)

# Calculate integrated squared error for j=2, b=1:
j <- 2 # component
b <- 1 # block
coords <- a$blockid == b
ise.npEM(a, j, b, dnorm, lower=0, upper=10, plots=TRUE,
         mean=mu[j,coords][1], sd=sigma[j, coords][1])


# The following (lengthy) example recreates the normal multivariate
# mixture model simulation from Benaglia et al (2009).
mu <- matrix(c(0, 0, 0, 3, 4, 5), m, r, byrow=TRUE)
nbrep <- 5  # Benaglia et al use 300 replications

# matrix for storing sums of Integrated Squared Errors
ISE <- matrix(0,m,r,dimnames=list(Components=1:m, Blocks=1:r))

nblabsw <- 0 # no. of label switches
for (mc in 1:nbrep) {
  print(paste("REPETITION", mc))
x <- rmvnormmix(n,lambda,mu,sigma) # simulated data
  a <- npEM(x, centers, verb=FALSE) #default:
if (a$lambda[1] > a$lambda[2]) nblabsw <- nblabsw + 1
for (j in 1:m) {  # for each component
for (k in 1:r) { # for each coordinate; not assuming iid!
      # dnorm with correct mean, sd is the true density:
      ISE[j,k] <- ISE[j,k] + ise.npEM(a, j, k, dnorm, lower=mu[j,k]-5,
               upper=mu[j,k]+5, plots=FALSE, mean=mu[j,k],
               sd=sigma[j,k])$value
    }
  }
MISE <- ISE/nbrep # Mean ISE
sqMISE <- sqrt(MISE) # root-mean-integrated-squared error
}
sqMISE
```

---

logisregmixEM *EM Algorithm for Mixtures of Logistic Regressions*

Description

Returns EM algorithm output for mixtures of logistic regressions with arbitrarily many components.

Usage

```
logisregmixEM(y, x, N = NULL, lambda = NULL, beta = NULL, k = 2,
              addintercept = TRUE, epsilon = 1e-08,
              maxit = 10000, verb = FALSE)
```

Arguments

y	An n-vector of successes out of N trials.
x	An nxp matrix of predictors. See addintercept below.
N	An n-vector of number of trials for the logistic regression. If NULL, then N is an n-vector of 1s for binary logistic regression.
lambda	Initial value of mixing proportions. Entries should sum to 1. This determines number of components. If NULL, then lambda is random from uniform Dirichlet and number of components is determined by beta.
beta	Initial value of beta parameters. Should be a pxk matrix, where p is the number of columns of x and k is number of components. If NULL, then beta is generated by binning the data into k bins and using glm on the values in each of the bins. If both lambda and beta are NULL, then number of components is determined by k.
k	Number of components. Ignored unless lambda and beta are both NULL.
addintercept	If TRUE, a column of ones is appended to the x matrix before the value of p is calculated.
epsilon	The convergence criterion.
maxit	The maximum number of iterations.
verb	If TRUE, then various updates are printed during each iteration of the algorithm.

Value

logisregmixEM returns a list of class mixEM with items:

x	The predictor values.
y	The response values.
lambda	The final mixing proportions.
beta	The final logistic regression coefficients.
loglik	The final log-likelihood.
posterior	An nxk matrix of posterior probabilities for observations.
all.loglik	A vector of each iteration's log-likelihood.
restarts	The number of times the algorithm restarted due to unacceptable choice of initial values.
ft	A character vector giving the name of the function.

References

McLachlan, G. J. and Peel, D. (2000) *Finite Mixture Models*, John Wiley \& Sons, Inc.

See Also

poisregmixEM

Examples

```
## EM output for data generated from a 2-component logistic regression model.

set.seed(100)
beta <- matrix(c(1, .5, 2, -.8), 2, 2)
x <- runif(50, 0, 10)
x1 <- cbind(1, x)
xbeta <- x1%*%beta
N <- ceiling(runif(50, 50, 75))
w <- rbinom(50, 1, .3)
y <- w*rbinom(50, size = N, prob = (1/(1+exp(-xbeta[, 1]))))+
              (1-w)*rbinom(50, size = N, prob =
              (1/(1+exp(-xbeta[, 2]))))
out.1 <- logisregmixEM(y, x, N, verb = TRUE, epsilon = 1e-01)
out.1

## EM output for data generated from a 2-component binary logistic regression model.

beta <- matrix(c(-10, .1, 20, -.1), 2, 2)
x <- runif(500, 50, 250)
x1 <- cbind(1, x)
xbeta <- x1%*%beta
w <- rbinom(500, 1, .3)
y <- w*rbinom(500, size = 1, prob = (1/(1+exp(-xbeta[, 1]))))+
              (1-w)*rbinom(500, size = 1, prob =
              (1/(1+exp(-xbeta[, 2]))))
out.2 <- logisregmixEM(y, x, beta = beta, lambda = c(.3, .7),
                       verb = TRUE, epsilon = 1e-01)
out.2
```

---

makemultdata *Produce Cutpoint Multinomial Data*

Description

Change data into a matrix of multinomial counts using the cutpoint method and generate EM algorithm starting values for a k-component mixture of multinomials.

Usage

```
makemultdata(..., cuts)
```

Arguments

...	Either vectors (possibly of different lengths) of raw data or an nxm matrix (or data frame) of data. If ... are vectors of varying length, then makemultdata will create a matrix of size nxm where n is the sample size and m is the length of the vector with maximum length. Those vectors with length less than m will have NAs to make the corresponding row in the matrix of length m. If ... is a matrix (or data frame), then the rows must correspond to the sample and the columns the repeated measures.
cuts	A vector of cutpoints. This vector is sorted by the algorithm.

Details

The (i, j)th entry of the matrix y (for j < p) is equal to the number of entries in the ith column of x that are less than or equal to cuts[j]. The (i, p)th entry is equal to the number of entries greater than cuts[j].

Value

makemultdata returns an object which is a list with components:

x	An nxm matrix of the raw data.
y	An nxp matrix of the discretized data where p is one more than the number of cutpoints. Each row is a multinomial vector of counts. In particular, each row should sum to the number of repeated measures for that sample.

References

Elmore, R. T., Hettmansperger, T. P. and Xuan, F. (2004) The Sign Statistic, One-Way Layouts and Mixture Models, *Statistical Science* **19(4)**, 579–587.

See Also

compCDF, multmixmodel.sel, multmixEM

Examples

```
## Randomly generated data.

set.seed(100)
y <- matrix(rpois(70, 6), 10, 7)
cuts <- c(2, 5, 7)
out1 <- makemultdata(y, cuts = cuts)
out1

## The sulfur content of the coal seams in Texas.

A <- c(1.51, 1.92, 1.08, 2.04, 2.14, 1.76, 1.17)
B <- c(1.69, 0.64, .9, 1.41, 1.01, .84, 1.28, 1.59)
C <- c(1.56, 1.22, 1.32, 1.39, 1.33, 1.54, 1.04, 2.25, 1.49)
D <- c(1.3, .75, 1.26, .69, .62, .9, 1.2, .32)
```

```
E <- c(.73, .8, .9, 1.24, .82, .72, .57, 1.18, .54, 1.3)

out2 <- makemultdata(A, B, C, D, E,
                     cuts = median(c(A, B, C, D, E)))
out2

## The reaction time data.

data(RTdata)
out3 <- makemultdata(RTdata, cuts =
                     100*c(5, 10, 12, 14, 16, 20, 25, 30, 40, 50))
dim(out3$y)
out3$y[1:10,]
```

`mixturegram` *Mixturegrams*

Description

Construct a mixturegram for determining an apporpriate number of components.

Usage

```
mixturegram(data, pmbs, method = c("pca", "kpca", "lda"), all.n = FALSE,
id.con = NULL, score = 1, iter.max = 50, nstart = 25, ...)
```

Arguments

`data`	The data, which must either be a vector or a matrix. If a matrix, then the rows correspond to the observations.
`pmbs`	A list of length (K-1) such that each element is an nxk matrix of the posterior membership probabilities. These are obtained from each of the "best" estimated k-component mixture models, k = 2,...,K.
`method`	The dimension reduction method used. `method = "pca"` implements principal components analysis. `method = "kpca"` implements kernel principal components analysis. `method = "lda"` implements reduced rank linear discriminant analysis.
`all.n`	A logical specifying whether the mixturegram should plot the profiles of all observations (`TRUE`) or just the K-profile summaries (`FALSE`). The default is `FALSE`.
`id.con`	An argument that allows one to impose some sort of (meaningful) identifiability constraint so that the mixture components are in some sort of comparable order between mixture models with different numbers of components. If `NULL`, then the components are ordered by the component means for univariate data or ordered by the first dimension of the component means for multivariate data.

score	The value for the specified dimension reduction technique's score, which is used for constructing the mixturegram. By default, this value is 1, which is the value that will typically be used. Larger values will result in more variability displayed on the mixturegram. Note that the largest value that can be calculated at each value of k>1 on the mixturegram is p+k-1, where p is the number of columns of data.
iter.max	The maximum number of iterations allowed for the k-means clustering algorithm, which is passed to the kmeans function. The default is 50.
nstart	The number of random sets chosen based on k centers, which is passed to the kmeans function. The default is 25.
...	Additional arguments that can be passed to the underlying plot function.

Value

mixturegram returns a mixturegram where the profiles are plotted over component values of k = 1,...,K.

References

Young, D. S., Ke, C., and Zeng, X. (2018) The Mixturegram: A Visualization Tool for Assessing the Number of Components in Finite Mixture Models, *Journal of Computational and Graphical Statistics*, **27(3)**, 564–575.

See Also

boot.comp

Examples

```
##Data generated from a 2-component mixture of normals.


set.seed(100)
n <- 100
w <- rmultinom(n,1,c(.3,.7))
y <- sapply(1:n,function(i) w[1,i]*rnorm(1,-6,1) +
            w[2,i]*rnorm(1,0,1))

selection <- function(i,data,rep=30){
  out <- replicate(rep,normalmixEM(data,epsilon=1e-06,
      k=i,maxit=5000),simplify=FALSE)
  counts <- lapply(1:rep,function(j)
                  table(apply(out[[j]]$posterior,1,
                  which.max)))
  counts.length <- sapply(counts, length)
  counts.min <- sapply(counts, min)
  counts.test <- (counts.length != i)|(counts.min < 5)
  if(sum(counts.test) > 0 & sum(counts.test) < rep)
   out <- out[!counts.test]
  l <- unlist(lapply(out, function(x) x$loglik))
```

```
  tmp <- out[[which.max(l)]]
}

all.out <- lapply(2:5, selection, data = y, rep = 2)
pmbs <- lapply(1:length(all.out), function(i)
   all.out[[i]]$post)
mixturegram(y, pmbs, method = "pca", all.n = FALSE,
    id.con = NULL, score = 1,
    main = "Mixturegram (Well-Separated Data)")
```

`multmixEM` *EM Algorithm for Mixtures of Multinomials*

Description

Return EM algorithm output for mixtures of multinomial distributions.

Usage

```
multmixEM(y, lambda = NULL, theta = NULL, k = 2,
          maxit = 10000, epsilon = 1e-08, verb = FALSE)
```

Arguments

`y`	Either An nxp matrix of data (multinomial counts), where n is the sample size and p is the number of multinomial bins, or the output of the `makemultdata` function. It is not necessary that all of the rows contain the same number of multinomial trials (i.e., the row sums of y need not be identical).
`lambda`	Initial value of mixing proportions. Entries should sum to 1. This determines number of components. If NULL, then `lambda` is random from uniform Dirichlet and number of components is determined by `theta`.
`theta`	Initial value of `theta` parameters. Should be a kxp matrix, where p is the number of columns of y and k is number of components. Each row of `theta` should sum to 1. If NULL, then each row is random from uniform Dirichlet. If both `lambda` and `theta` are NULL, then number of components is determined by k.
`k`	Number of components. Ignored unless `lambda` and `theta` are NULL.
`epsilon`	The convergence criterion.
`maxit`	The maximum number of iterations.
`verb`	If TRUE, then various updates are printed during each iteration of the algorithm.

Value

`multmixEM` returns a list of class `mixEM` with items:

`y`	The raw data.
`lambda`	The final mixing proportions.

theta	The final multinomial parameters.
loglik	The final log-likelihood.
posterior	An nxk matrix of posterior probabilities for observations.
all.loglik	A vector of each iteration's log-likelihood.
restarts	The number of times the algorithm restarted due to unacceptable choice of initial values.
ft	A character vector giving the name of the function.

References

- McLachlan, G. J. and Peel, D. (2000) *Finite Mixture Models*, John Wiley \& Sons, Inc.
- Elmore, R. T., Hettmansperger, T. P. and Xuan, F. (2004) The Sign Statistic, One-Way Layouts and Mixture Models, *Statistical Science* **19(4)**, 579–587.

See Also

compCDF, makemultdata, multmixmodel.sel

Examples

```
## The sulfur content of the coal seams in Texas

set.seed(100)
A <- c(1.51, 1.92, 1.08, 2.04, 2.14, 1.76, 1.17)
B <- c(1.69, 0.64, .9, 1.41, 1.01, .84, 1.28, 1.59)
C <- c(1.56, 1.22, 1.32, 1.39, 1.33, 1.54, 1.04, 2.25, 1.49)
D <- c(1.3, .75, 1.26, .69, .62, .9, 1.2, .32)
E <- c(.73, .8, .9, 1.24, .82, .72, .57, 1.18, .54, 1.3)

dis.coal <- makemultdata(A, B, C, D, E,
                         cuts = median(c(A, B, C, D, E)))
em.out <- multmixEM(dis.coal)
em.out[1:4]
```

---

multmixmodel.sel *Model Selection Mixtures of Multinomials*

Description

Assess the number of components in a mixture of multinomials model using the Akaike's information criterion (AIC), Schwartz's Bayesian information criterion (BIC), Bozdogan's consistent AIC (CAIC), and Integrated Completed Likelihood (ICL).

Usage

```
multmixmodel.sel(y, comps = NULL, ...)
```

Arguments

y	Either An nxp matrix of data (multinomial counts), where n is the sample size and p is the number of multinomial bins, or the output of the makemultdata function. It is not necessary that all of the rows contain the same number of multinomial trials (i.e., the row sums of y need not be identical).
comps	Vector containing the numbers of components to consider. If NULL, this is set to be 1:(max possible), where (max possible) is floor((m+1)/2) and m is the minimum row sum of y.
...	Arguments passed to multmixEM that control convergence of the underlying EM algorithm.

Value

multmixmodel.sel returns a table summarizing the AIC, BIC, CAIC, ICL, and log-likelihood values along with the winner (the number with the lowest aforementioned values).

See Also

compCDF, makemultdata, multmixEM

Examples

```
##Data generated using the multinomial cutpoint method.

set.seed(100)
x <- matrix(rpois(70, 6), 10, 7)
x.new <- makemultdata(x, cuts = 5)
multmixmodel.sel(x.new$y, comps = c(1,2), epsilon = 1e-03)
```

---

mvnormalmixEM *EM Algorithm for Mixtures of Multivariate Normals*

Description

Return EM algorithm output for mixtures of multivariate normal distributions.

Usage

```
mvnormalmixEM(x, lambda = NULL, mu = NULL, sigma = NULL, k = 2,
              arbmean = TRUE, arbvar = TRUE, epsilon = 1e-08,
              maxit = 10000, verb = FALSE)
```

Arguments

x	A matrix of size nxp consisting of the data.
lambda	Initial value of mixing proportions. Entries should sum to 1. This determines number of components. If NULL, then lambda is random from uniform Dirichlet and number of components is determined by mu.
mu	A list of size k consisting of initial values for the p-vector mean parameters. If NULL, then the vectors are generated from a normal distribution with mean and standard deviation according to a binning method done on the data. If both lambda and mu are NULL, then number of components is determined by sigma.
sigma	A list of size k consisting of initial values for the pxp variance-covariance matrices. If NULL, then sigma is generated using the data. If lambda, mu, and sigma are NULL, then number of components is determined by k.
k	Number of components. Ignored unless lambda, mu, and sigma are all NULL.
arbmean	If TRUE, then the component densities are allowed to have different mus. If FALSE, then a scale mixture will be fit.
arbvar	If TRUE, then the component densities are allowed to have different sigmas. If FALSE, then a location mixture will be fit.
epsilon	The convergence criterion.
maxit	The maximum number of iterations.
verb	If TRUE, then various updates are printed during each iteration of the algorithm.

Value

normalmixEM returns a list of class mixEM with items:

x	The raw data.
lambda	The final mixing proportions.
mu	A list of with the final mean vectors.
sigma	A list with the final variance-covariance matrices.
loglik	The final log-likelihood.
posterior	An nxk matrix of posterior probabilities for observations.
all.loglik	A vector of each iteration's log-likelihood.
restarts	The number of times the algorithm restarted due to unacceptable choice of initial values.
ft	A character vector giving the name of the function.

References

McLachlan, G. J. and Peel, D. (2000) *Finite Mixture Models*, John Wiley \& Sons, Inc.

See Also

normalmixEM

Examples

```
##Fitting randomly generated data with a 2-component location mixture of bivariate normals.

set.seed(100)
x.1 <- rmvnorm(40, c(0, 0))
x.2 <- rmvnorm(60, c(3, 4))
X.1 <- rbind(x.1, x.2)
mu <- list(c(0, 0), c(3, 4))

out.1 <- mvnormalmixEM(X.1, arbvar = FALSE, mu = mu,
                       epsilon = 1e-02)
out.1[2:5]

##Fitting randomly generated data with a 2-component scale mixture of bivariate normals.

x.3 <- rmvnorm(40, c(0, 0), sigma =
               matrix(c(200, 1, 1, 150), 2, 2))
x.4 <- rmvnorm(60, c(0, 0))
X.2 <- rbind(x.3, x.4)
lambda <- c(0.40, 0.60)
sigma <- list(diag(1, 2), matrix(c(200, 1, 1, 150), 2, 2))

out.2 <- mvnormalmixEM(X.2, arbmean = FALSE,
                       sigma = sigma, lambda = lambda,
                       epsilon = 1e-02)
out.2[2:5]
```

`mvnpEM` *EM-like Algorithm for Nonparametric Mixture Models with Conditionally Independent Multivariate Component Densities*

Description

An extension of the original npEM algorithm, for mixtures of multivariate data where the coordinates of a row (case) in the data matrix are assumed to be made of independent but multivariate blocks (instead of just coordinates), conditional on the mixture component (subpopulation) from which they are drawn (Chauveau and Hoang 2015).

Usage

```
mvnpEM(x, mu0, blockid = 1:ncol(x), samebw = TRUE,
       bwdefault = apply(x,2,bw.nrd0), init = NULL,
       eps = 1e-8, maxiter = 500, verb = TRUE)
```

Arguments

x	An $n \times r$ matrix of data. Each of the n rows is a case, and each case has r repeated measurements. These measurements are assumed to be conditionally independent, conditional on the mixture component (subpopulation) from which the case is drawn.

mu0	Either an $m \times r$ matrix specifying the initial centers for the kmeans function, or an integer m specifying the number of initial centers, which are then chosen randomly in kmeans
blockid	A vector of length r identifying coordinates (columns of x) that are in the same block. The default has all distinct elements, indicating that the model has r blocks of dimension 1, in which case the model is handled directly by the npEM algorithm. See example below for actual multivariate blocks example.
samebw	Logical: If TRUE, use the same bandwidth per coordinate for all iteration and all components. If FALSE, use a separate bandwidth for each component and coordinate, and update this bandwidth at each iteration of the algorithm using a suitably modified bw.nrd0 method as described in Benaglia et al (2011) and Chauveau and Hoang (2015).
bwdefault	Bandwidth default for density estimation,a simplistic application of the default bw.nrd0 for each coordinate (column) of the data.
init	Initialization method, based on an initial $n \times m$ matrix for the posterior probabilities. If NULL, a kmeans clustering with mu0 initial centers is applied to the data and the initial matrix of posteriors is built from the result.
eps	Tolerance limit for declaring algorithm convergence. Convergence is declared whenever the maximum change in any coordinate of the lambda vector (of mixing proportion estimates) does not exceed eps.
maxiter	The maximum number of iterations allowed; convergence may be declared before maxiter iterations (see eps above).
verb	Verbose mode; if TRUE, print updates for every iteration of the algorithm as it runs

Value

mvnpEM returns a list of class mvnpEM with the following items:

data	The raw data (an $n \times r$ matrix).
posteriors	An $n \times m$ matrix of posterior probabilities for each observation (row).
lambda	The sequence of mixing proportions over iterations.
blockid	The blockid input argument. Needed by any method that produces density estimates from the output, like plot.mvnpEM.
samebw	The samebw input argument. Needed by any method that produces density estimates from the output, like plot.mvnpEM.
bandwidth	The final bandwidth matrix after convergence of the algorithm. Its shape depends on the samebw input argument. If samebw = TRUE, a vectors with the bandwidth value for each of the r coordinates (same for all components and iterations). If samebw = FALSE, a $m \times r$ matrix, where each row is associated to one component and gives the r bandwidth values, one for each coordinate. Needed by any method that produces density estimates from the output, like plot.mvnpEM.
lambdahat	The final mixing proportions.
loglik	The sequence of pseudo log-likelihood values over iterations.

References

- Benaglia, T., Chauveau, D., and Hunter, D. R. (2009), An EM-like algorithm for semi- and non-parametric estimation in multivariate mixtures, Journal of Computational and Graphical Statistics, 18, 505-526.
- Benaglia, T., Chauveau, D. and Hunter, D.R. (2011), Bandwidth Selection in an EM-like algorithm for nonparametric multivariate mixtures. Nonparametric Statistics and Mixture Models: A Festschrift in Honor of Thomas P. Hettmansperger. World Scientific Publishing Co., pages 15-27.
- Chauveau, D., and Hoang, V. T. L. (2015), Nonparametric mixture models with conditionally independent multivariate component densities, Preprint under revision. https://hal.archives-ouvertes.fr/hal-01094837

See Also

plot.mvnpEM, npEM

Examples

```
# Example as in Chauveau and Hoang (2015) with 6 coordinates
## Not run:
m=2; r=6; blockid <-c(1,1,2,2,3,3) # 3 bivariate blocks
# generate some data x ...
a <- mvnpEM(x, mu0=2, blockid, samebw=F) # adaptive bandwidth
plot(a) # this S3 method produces 6 plots of univariate marginals
summary(a)
## End(Not run)
```

NOdata *Ethanol Fuel Data Set*

Description

This data set gives the equivalence ratios and peak nitrogen oxide emissions in a study using pure ethanol as a spark-ignition engine fuel.

Usage

```
data(NOdata)
```

Format

This data frame consists of:

- NOThe peak nitrogen oxide emission levels.
- EquivalenceThe equivalence ratios for the engine at compression ratios from 7.5 to 18.

Source

Brinkman, N. D. (1981) Ethanol Fuel – A Single-Cylinder Engine Study of Efficiency and Exhaust Emissions, *S.A.E. Transactions*, 68.

References

Hurn, M., Justel, A. and Robert, C. P. (2003) Estimating Mixtures of Regressions, *Journal of Computational and Graphical Statistics* **12(1)**, 55–79.

`normalmixEM` *EM Algorithm for Mixtures of Univariate Normals*

Description

Return EM algorithm output for mixtures of normal distributions.

Usage

```
normalmixEM(x, lambda = NULL, mu = NULL, sigma = NULL, k = 2,
            mean.constr = NULL, sd.constr = NULL,
            epsilon = 1e-08, maxit = 1000, maxrestarts=20,
            verb = FALSE, fast=FALSE, ECM = FALSE,
            arbmean = TRUE, arbvar = TRUE)
```

Arguments

`x`	A vector of length n consisting of the data.
`lambda`	Initial value of mixing proportions. Automatically repeated as necessary to produce a vector of length `k`, then normalized to sum to 1. If `NULL`, then `lambda` is random from a uniform Dirichlet distribution (i.e., its entries are uniform random and then it is normalized to sum to 1).
`mu`	Starting value of vector of component means. If non-NULL and a scalar, `arbmean` is set to `FALSE`. If non-NULL and a vector, `k` is set to `length(mu)`. If NULL, then the initial value is randomly generated from a normal distribution with center(s) determined by binning the data.
`sigma`	Starting value of vector of component standard deviations for algorithm. If non-NULL and a scalar, `arbvar` is set to `FALSE`. If non-NULL and a vector, `arbvar` is set to `TRUE` and `k` is set to `length(sigma)`. If NULL, then the initial value is the reciprocal of the square root of a vector of random exponential-distribution values whose means are determined according to a binning method done on the data.
`k`	Number of components. Initial value ignored unless `mu` and `sigma` are both NULL.

`mean.constr` Equality constraints on the mean parameters, given as a vector of length `k`. Each vector entry helps specify the constraints, if any, on the corresponding mean parameter: If `NA`, the corresponding parameter is unconstrained. If numeric, the corresponding parameter is fixed at that value. If a character string consisting of a single character preceded by a coefficient, such as `"0.5a"` or `"-b"`, all parameters using the same single character in their constraints will fix these parameters equal to the coefficient times some the same free parameter. For instance, if `mean.constr = c(NA,0,"a","-a")`, then the first mean parameter is unconstrained, the second is fixed at zero, and the third and forth are constrained to be equal and opposite in sign.

`sd.constr` Equality constraints on the standard deviation parameters. See `mean.constr`.

`epsilon` The convergence criterion. Convergence is declared when the change in the observed data log-likelihood increases by less than epsilon.

`maxit` The maximum number of iterations.

`maxrestarts` The maximum number of restarts allowed in case of a problem with the particular starting values chosen due to one of the variance estimates getting too small (each restart uses randomly chosen starting values). It is well-known that when each component of a normal mixture may have its own mean and variance, the likelihood has no maximizer; in such cases, we hope to find a "nice" local maximum with this algorithm instead, but occasionally the algorithm finds a "not nice" solution and one of the variances goes to zero, driving the likelihood to infinity.

`verb` If TRUE, then various updates are printed during each iteration of the algorithm.

`fast` If TRUE and k==2 and arbmean==TRUE, then use `normalmixEM2comp`, which is a much faster version of the EM algorithm for this case. This version is less protected against certain kinds of underflow that can cause numerical problems and it does not permit any restarts. If k>2, `fast` is ignored.

`ECM` logical: Should this algorithm be an ECM algorithm in the sense of Meng and Rubin (1993)? If FALSE, the algorithm is a true EM algorithm; if TRUE, then every half-iteration alternately updates the means conditional on the variances or the variances conditional on the means, with an extra E-step in between these updates.

`arbmean` If TRUE, then the component densities are allowed to have different `mus`. If FALSE, then a scale mixture will be fit. Initial value ignored unless `mu` is NULL.

`arbvar` If TRUE, then the component densities are allowed to have different `sigmas`. If FALSE, then a location mixture will be fit. Initial value ignored unless `sigma` is NULL.

Details

This is the standard EM algorithm for normal mixtures that maximizes the conditional expected complete-data log-likelihood at each M-step of the algorithm. If desired, the EM algorithm may be replaced by an ECM algorithm (see `ECM` argument) that alternates between maximizing with respect to the `mu` and `lambda` while holding `sigma` fixed, and maximizing with respect to `sigma` and `lambda` while holding `mu` fixed. In the case where `arbmean` is `FALSE` and `arbvar` is `TRUE`, there is no closed-form EM algorithm, so the ECM option is forced in this case.

Value

normalmixEM returns a list of class mixEM with items:

x	The raw data.
lambda	The final mixing proportions.
mu	The final mean parameters.
sigma	The final standard deviations. If arbmean = FALSE, then only the smallest standard deviation is returned. See scale below.
scale	If arbmean = FALSE, then the scale factor for the component standard deviations is returned. Otherwise, this is omitted from the output.
loglik	The final log-likelihood.
posterior	An nxk matrix of posterior probabilities for observations.
all.loglik	A vector of each iteration's log-likelihood. This vector includes both the initial and the final values; thus, the number of iterations is one less than its length.
restarts	The number of times the algorithm restarted due to unacceptable choice of initial values.
ft	A character vector giving the name of the function.

References

- McLachlan, G. J. and Peel, D. (2000) *Finite Mixture Models*, John Wiley \& Sons, Inc.
- Meng, X.-L. and Rubin, D. B. (1993) Maximum Likelihood Estimation Via the ECM Algorithm: A General Framework, *Biometrika* 80(2): 267-278.
- Benaglia, T., Chauveau, D., Hunter, D. R., and Young, D. mixtools: An R package for analyzing finite mixture models. Journal of Statistical Software, 32(6):1-29, 2009.

See Also

mvnormalmixEM, normalmixEM2comp, normalmixMMlc, spEMsymloc

Examples

```
##Analyzing the Old Faithful geyser data with a 2-component mixture of normals.

data(faithful)
attach(faithful)
set.seed(100)
system.time(out<-normalmixEM(waiting, arbvar = FALSE, epsilon = 1e-03))
out
system.time(out2<-normalmixEM(waiting, arbvar = FALSE, epsilon = 1e-03, fast=TRUE))
out2 # same thing but much faster
```

normalmixEM2comp *Fast EM Algorithm for 2-Component Mixtures of Univariate Normals*

Description

Return EM algorithm output for mixtures of univariate normal distributions for the special case of 2 components, exploiting the simple structure of the problem to speed up the code.

Usage

```
normalmixEM2comp(x, lambda, mu, sigsqrd, eps= 1e-8, maxit = 1000, verb=FALSE)
```

Arguments

x	A vector of length n consisting of the data.
lambda	Initial value of first-component mixing proportion.
mu	A 2-vector of initial values for the mean parameters.
sigsqrd	Either a scalar or a 2-vector with initial value(s) for the variance parameters. If a scalar, the algorithm assumes that the two components have equal variances; if a 2-vector, it assumes that the two components do not have equal variances.
eps	The convergence criterion. Convergence is declared when the change in the observed data log-likelihood increases by less than epsilon.
maxit	The maximum possible number of iterations.
verb	If TRUE, then various updates are printed during each iteration of the algorithm.

Details

This code is written to be very fast, sometimes more than an order of magnitude faster than normalmixEM for the same problem. It is less numerically stable that normalmixEM in the sense that it does not safeguard against underflow as carefully.

Note that when the two components are assumed to have unequal variances, the loglikelihood is unbounded. However, in practice this is rarely a problem and quite often the algorithm converges to a "nice" local maximum.

Value

normalmixEM2comp returns a list of class mixEM with items:

x	The raw data.
lambda	The final mixing proportions (lambda and 1-lambda).
mu	The final two mean parameters.
sigma	The final one or two standard deviations.
loglik	The final log-likelihood.
posterior	An nx2 matrix of posterior probabilities for observations.

`all.loglik`	A vector of each iteration's log-likelihood. This vector includes both the initial and the final values; thus, the number of iterations is one less than its length.
`restarts`	The number of times the algorithm restarted due to unacceptable choice of initial values (always zero).
`ft`	A character vector giving the name of the function.

References

McLachlan, G. J. and Peel, D. (2000) *Finite Mixture Models*, John Wiley \& Sons, Inc.

See Also

`mvnormalmixEM`, `normalmixEM`

Examples

```
##Analyzing the Old Faithful geyser data with a 2-component mixture of normals.

data(faithful)
attach(faithful)
set.seed(100)
system.time(out <- normalmixEM2comp(waiting, lambda=.5,
            mu=c(50,80), sigsqrd=100))
out$all.loglik # Note:  must be monotone increasing

# Compare elapsed time with more general version
system.time(out2 <- normalmixEM(waiting, lambda=c(.5,.5),
            mu=c(50,80), sigma=c(10,10), arbvar=FALSE))
out2$all.loglik # Values should be identical to above
```

---

normalmixMMlc *EC-MM Algorithm for Mixtures of Univariate Normals with linear constraints*

Description

Return EC-MM (see below) algorithm output for mixtures of normal distributions with linear constraints on the means and variances parameters, as in Chauveau and Hunter (2013). The linear constraint for the means is of the form $\mu = M\beta + C$, where M and C are matrix and vector specified as parameters. The linear constraints for the variances are actually specified on the inverse variances, by $\pi = A\gamma$, where π is the vector of inverse variances, and A is a matrix specified as a parameter (see below).

Usage

```
normalmixMMlc(x, lambda = NULL, mu = NULL, sigma = NULL, k = 2,
              mean.constr = NULL, mean.lincstr = NULL,
              mean.constant = NULL, var.lincstr = NULL,
              gparam = NULL, epsilon = 1e-08, maxit = 1000,
              maxrestarts=20, verb = FALSE)
```

Arguments

x	A vector of length n consisting of the data.
lambda	Initial value of mixing proportions. Automatically repeated as necessary to produce a vector of length k, then normalized to sum to 1. If NULL, then lambda is random from a uniform Dirichlet distribution (i.e., its entries are uniform random and then it is normalized to sum to 1).
mu	Starting value of vector of component means. If non-NULL and a vector, k is set to length(mu). If NULL, then the initial value is randomly generated from a normal distribution with center(s) determined by binning the data.
sigma	Starting value of vector of component standard deviations for algorithm. Obsolete for linear constraints on the inverse variances; use gparam instead to specify a starting value.
k	Number of components. Initial value ignored unless mu and sigma are both NULL.
mean.constr	First, simplest way to define equality constraints on the mean parameters, given as a vector of length k, as in normalmixEM. Each vector entry specifies the constraints, if any, on the corresponding mean parameter: If NA, the corresponding parameter is unconstrained. If numeric, the corresponding parameter is fixed at that value. If a character string consisting of a single character preceded by a coefficient, such as "0.5a" or "-b", all parameters using the same single character in their constraints will fix these parameters equal to the coefficient times some the same free parameter. For instance, if mean.constr = c(NA,0,"a","-a"), then the first mean parameter is unconstrained, the second is fixed at zero, and the third and forth are constrained to be equal and opposite in sign. Note: if there are no linear constraints for the means, it is more efficient to use directly normalmixEM.
mean.lincstr	Matrix M (k, p) in the linear constraint for the means equation $\mu = M\beta + C$, with $p \leq k$.
mean.constant	Vector of k constants C in the linear constraint for the means equation $\mu = M\beta + C$.
var.lincstr	Matrix A (k, q) in the linear constraint for the inverse variances equation $\pi = A\gamma$, with $q \leq k$.
gparam	Vector of q starting values for the γ parameter in the linear constraint for the inverse variances; see var.lincstr. If NULL, a vector of randomly generated standard exponential variables is used.
epsilon	The convergence criterion. Convergence is declared when the change in the observed data log-likelihood increases by less than epsilon.
maxit	The maximum allowed number of iterations.
maxrestarts	The maximum number of restarts allowed in case of a problem with the particular starting values chosen due to one of the variance estimates getting too small (each restart uses randomly chosen starting values). It is well-known that when each component of a normal mixture may have its own mean and variance, the likelihood has no maximizer; in such cases, we hope to find a "nice" local maximum with this algorithm instead, but occasionally the algorithm finds a "not

nice" solution and one of the variances goes to zero, driving the likelihood to infinity.

`verb` If TRUE, then various updates are printed during each iteration of the algorithm.

Details

This is a specific "EC-MM" algorithm for normal mixtures with linear constraints on the means and variances parameters. EC-MM here means that this algorithm is similar to an ECM algorithm as in Meng and Rubin (1993), except that it uses conditional MM (Minorization-Maximization)-steps instead of simple M-steps. Conditional means that it alternates between maximizing with respect to the `mu` and `lambda` while holding `sigma` fixed, and maximizing with respect to `sigma` and `lambda` while holding `mu` fixed. This ECM generalization of EM is forced in the case of linear constraints because there is no closed-form EM algorithm.

Value

`normalmixMMlc` returns a list of class `mixEM` with items:

`x`	The raw data.
`lambda`	The final mixing proportions.
`mu`	The final mean parameters.
`sigma`	The final standard deviation(s)
`scale`	Scale factor for the component standard deviations, if applicable.
`loglik`	The final log-likelihood.
`posterior`	An nxk matrix of posterior probabilities for observations.
`all.loglik`	A vector of each iteration's log-likelihood. This vector includes both the initial and the final values; thus, the number of iterations is one less than its length.
`restarts`	The number of times the algorithm restarted due to unacceptable choice of initial values.
`beta`	The final β parameter estimate.
`gamma`	The final γ parameter estimate.
`ft`	A character vector giving the name of the function.

Author(s)

Didier Chauveau

References

- McLachlan, G. J. and Peel, D. (2000) *Finite Mixture Models*, John Wiley & Sons, Inc.
- Meng, X.-L. and Rubin, D. B. (1993) Maximum Likelihood Estimation Via the ECM Algorithm: A General Framework, *Biometrika* 80(2): 267-278.
- Chauveau, D. and Hunter, D.R. (2013) ECM and MM algorithms for mixtures with constrained parameters, *preprint* http://hal.archives-ouvertes.fr/hal-00625285.
- Thomas, H., Lohaus, A., and Domsch, H. (2011) Stable Unstable Reliability Theory, *British Journal of Mathematical and Statistical Psychology* 65(2): 201-221.

See Also

normalmixEM, mvnormalmixEM, normalmixEM2comp, tauequivnormalmixEM

Examples

```
## Analyzing synthetic data as in the tau equivalent model
## From Thomas et al (2011), see also Chauveau and Hunter (2013)
## a 3-component mixture of normals with linear constraints.
lbd <- c(0.6,0.3,0.1); m <- length(lbd)
sigma <- sig0 <- sqrt(c(1,9,9))
# means constaints mu = M beta
M <- matrix(c(1,1,1,0,-1,1), 3, 2)
beta <- c(1,5) # unknown constrained mean
mu0 <- mu <- as.vector(M %*% beta)
# linear constraint on the inverse variances pi = A.g
A <- matrix(c(1,1,1,0,1,0), m, 2, byrow=TRUE)
iv0 <- 1/(sig0^2)
g0 <- c(iv0[2],iv0[1] - iv0[2]) # gamma^0 init

# simulation and EM fits
set.seed(50); n=100; x <- rnormmix(n,lbd,mu,sigma)
s <- normalmixEM(x,mu=mu0,sigma=sig0,maxit=2000) # plain EM
# EM with var and mean linear constraints
sc <- normalmixMMlc(x, lambda=lbd, mu=mu0, sigma=sig0,
mean.lincstr=M, var.lincstr=A, gparam=g0)
# plot and compare both estimates
dnormmixt <- function(t, lam, mu, sig){
m <- length(lam); f <- 0
for (j in 1:m) f <- f + lam[j]*dnorm(t,mean=mu[j],sd=sig[j])
f}
t <- seq(min(x)-2, max(x)+2, len=200)
hist(x, freq=FALSE, col="lightgrey",
ylim=c(0,0.3), ylab="density",main="")
lines(t, dnormmixt(t, lbd, mu, sigma), col="darkgrey", lwd=2) # true
lines(t, dnormmixt(t, s$lambda, s$mu, s$sigma), lty=2)
lines(t, dnormmixt(t, sc$lambda, sc$mu, sc$sigma), col=1, lty=3)
legend("topleft", c("true","plain EM","constr EM"),
col=c("darkgrey",1,1), lty=c(1,2,3), lwd=c(2,1,1))
```

---

npEM — *Nonparametric EM-like Algorithm for Mixtures of Independent Repeated Measurements*

Description

Returns nonparametric EM algorithm output (Benaglia et al, 2009) for mixtures of multivariate (repeated measures) data where the coordinates of a row (case) in the data matrix are assumed to be independent, conditional on the mixture component (subpopulation) from which they are drawn.

Usage

```
npEM(x, mu0, blockid = 1:ncol(x),
     bw = bw.nrd0(as.vector(as.matrix(x))), samebw = TRUE,
     h = bw, eps = 1e-8,
     maxiter = 500, stochastic = FALSE, verb = TRUE)
```

Arguments

x	An $n \times r$ matrix of data. Each of the n rows is a case, and each case has r repeated measurements. These measurements are assumed to be conditionally independent, conditional on the mixture component (subpopulation) from which the case is drawn.
mu0	Either an $m \times r$ matrix specifying the initial centers for the kmeans function, or an integer m specifying the number of initial centers, which are then choosen randomly in kmeans
blockid	A vector of length r identifying coordinates (columns of x) that are assumed to be identically distributed (i.e., in the same block). For instance, the default has all distinct elements, indicating that no two coordinates are assumed identically distributed and thus a separate set of m density estimates is produced for each column of x. On the other hand, if blockid=rep(1,ncol(x)), then the coordinates in each row are assumed conditionally i.i.d.
bw	Bandwidth for density estimation, equal to the standard deviation of the kernel density. By default, a simplistic application of the default bw.nrd0 bandwidth used by density to the entire dataset.
samebw	Logical: If TRUE, use the same bandwidth for each iteration and for each component and block. If FALSE, use a separate bandwidth for each component and block, and update this bandwidth at each iteration of the algorithm using a suitably modified bw.nrd0 method as described in Benaglia et al (2011).
h	Alternative way to specify the bandwidth, to provide backward compatibility.
eps	Tolerance limit for declaring algorithm convergence. Convergence is declared whenever the maximum change in any coordinate of the lambda vector (of mixing proportion estimates) does not exceed eps.
maxiter	The maximum number of iterations allowed, for both stochastic and non-stochastic versions; for non-stochastic algorithms (stochastic = FALSE), convergence may be declared before maxiter iterations (see eps above).
stochastic	Flag, if FALSE (the default), runs the non-stochastic version of the npEM algorithm, as in Benaglia et al (2009). Set to TRUE to run a stochastic version which simulates the posteriors at each iteration, and runs for maxiter iterations.
verb	If TRUE, print updates for every iteration of the algorithm as it runs

Value

npEM returns a list of class npEM with the following items:

data	The raw data (an $n \times r$ matrix).

posteriors	An $n \times m$ matrix of posterior probabilities for observation. If stochastic = TRUE, this matrix is computed from an average over the maxiter iterations.
bandwidth	If samebw==TRUE, same as the bw input argument; otherwise, value of bw matrix at final iteration. This information is needed by any method that produces density estimates from the output.
blockid	Same as the blockid input argument, but recoded to have positive integer values. Also needed by any method that produces density estimates from the output.
lambda	The sequence of mixing proportions over iterations.
lambdahat	The final mixing proportions if stochastic = FALSE, or the average mixing proportions if stochastic = TRUE.
loglik	The sequence of log-likelihoods over iterations.

References

- Benaglia, T., Chauveau, D., and Hunter, D. R. (2009), An EM-like algorithm for semi- and non-parametric estimation in multivariate mixtures, Journal of Computational and Graphical Statistics, 18, 505-526.
- Benaglia, T., Chauveau, D., Hunter, D. R., and Young, D. (2009), mixtools: An R package for analyzing finite mixture models. Journal of Statistical Software, 32(6):1-29.
- Benaglia, T., Chauveau, D. and Hunter, D.R. (2011), Bandwidth Selection in an EM-like algorithm for nonparametric multivariate mixtures. Nonparametric Statistics and Mixture Models: A Festschrift in Honor of Thomas P. Hettmansperger. World Scientific Publishing Co., pages 15-27.
- Bordes, L., Chauveau, D., and Vandekerkhove, P. (2007), An EM algorithm for a semiparametric mixture model, Computational Statistics and Data Analysis, 51: 5429-5443.

See Also

plot.npEM, normmixrm.sim, spEMsymloc, spEM, plotseq.npEM

Examples

```
## Examine and plot water-level task data set.

## First, try a 3-component solution where no two coordinates are
## assumed i.d.
data(Waterdata)
set.seed(100)
## Not run:
a <- npEM(Waterdata[,3:10], mu0=3, bw=4) # Assume indep but not iid
plot(a) # This produces 8 plots, one for each coordinate

## End(Not run)

## Next, same thing but pairing clock angles that are directly opposite one
## another (1:00 with 7:00, 2:00 with 8:00, etc.)
## Not run:
```

```
b <- npEM(Waterdata[,3:10], mu0=3, blockid=c(4,3,2,1,3,4,1,2), bw=4) # iid in pairs
plot(b) # Now only 4 plots, one for each block

## End(Not run)
```

---

npMSL *Nonparametric EM-like Algorithm for Mixtures of Independent Repeated Measurements - Maximum Smoothed Likelihood version*

Description

Returns nonparametric Smoothed Likelihood algorithm output (Levine et al, 2011) for mixtures of multivariate (repeated measures) data where the coordinates of a row (case) in the data matrix are assumed to be independent, conditional on the mixture component (subpopulation) from which they are drawn.

Usage

```
npMSL(x, mu0, blockid = 1:ncol(x),
      bw = bw.nrd0(as.vector(as.matrix(x))), samebw = TRUE,
      bwmethod = "S", h = bw, eps = 1e-8,
      maxiter=500, bwiter = maxiter, nbfold = NULL,
      ngrid=200, post=NULL, verb = TRUE)
```

Arguments

x	An $n \times r$ matrix of data. Each of the n rows is a case, and each case has r repeated measurements. These measurements are assumed to be conditionally independent, conditional on the mixture component (subpopulation) from which the case is drawn.
mu0	Either an $m \times r$ matrix specifying the initial centers for the kmeans function, or an integer m specifying the number of initial centers, which are then choosen randomly in kmeans
blockid	A vector of length r identifying coordinates (columns of x) that are assumed to be identically distributed (i.e., in the same block). For instance, the default has all distinct elements, indicating that no two coordinates are assumed identically distributed and thus a separate set of m density estimates is produced for each column of x. On the other hand, if blockid=rep(1,ncol(x)), then the coordinates in each row are assumed conditionally i.i.d.
bw	Bandwidth for density estimation, equal to the standard deviation of the kernel density. By default, a simplistic application of the default bw.nrd0 bandwidth used by density to the entire dataset.
samebw	Logical: If TRUE, use the same bandwidth for each iteration and for each component and block. If FALSE, use a separate bandwidth for each component and block, and update this bandwidth at each iteration of the algorithm until bwiter is reached (see below). Two adaptation methods are provided, see bwmethod below.

bwmethod	Define the adaptive bandwidth strategy when samebw = FALSE, in which case the bandwidth depends on each component, block, and iteration of the algorithm. If set to "S" (the default), adaptation is done using a suitably modified bw.nrd0 method as described in Benaglia et al (2011). If set to "CV", an adaptive k-fold Cross Validation method is applied, as described in Chauveau et al (2014), where nbfold is the number of subsamples. This corresponds to a Leave-$[n/nbfold]$-Out CV.
h	Alternative way to specify the bandwidth, to provide backward compatibility.
eps	Tolerance limit for declaring algorithm convergence. Convergence is declared whenever the maximum change in any coordinate of the lambda vector (of mixing proportion estimates) does not exceed eps.
maxiter	The maximum number of iterations allowed, convergence may be declared before maxiter iterations (see eps above).
bwiter	The maximum number of iterations allowed for adaptive bandwidth stage, when samebw = FALSE. If set to 0, then the initial bandwidth matrix is used without adaptation.
nbfold	A parameter passed to the internal function wbs.kCV, which controls the weighted bandwidth selection by k-fold cross-validation.
ngrid	Number of points in the discretization of the intervals over which are approximated the (univariate) integrals for non linear smoothing of the log-densities, as required in the E step of the npMSL algorithm, see Levine et al (2011).
post	If non-NULL, an $n \times m$ matrix specifying the initial posterior probability vectors for each of the observations, i.e., the initial values to start the EM-like algorithm.
verb	If TRUE, print updates for every iteration of the algorithm as it runs

Value

npMSL returns a list of class npEM with the following items:

data	The raw data (an $n \times r$ matrix).
posteriors	An $n \times m$ matrix of posterior probabilities for observation.
bandwidth	If samebw==TRUE, same as the bw input argument; otherwise, value of bw matrix at final iteration. This information is needed by any method that produces density estimates from the output.
blockid	Same as the blockid input argument, but recoded to have positive integer values. Also needed by any method that produces density estimates from the output.
lambda	The sequence of mixing proportions over iterations.
lambdahat	The final mixing proportions.
loglik	The sequence of log-likelihoods over iterations.
f	An array of size $ngrid \times m \times l$, returning last values of density for component j and block k over grid points.
meanNaN	Average number of NaN that occured over iterations (for internal testing and control purpose).
meanUdfl	Average number of "underflow" that occured over iterations (for internal testing and control purpose).

References

- Benaglia, T., Chauveau, D., and Hunter, D. R. (2009), An EM-like algorithm for semi- and non-parametric estimation in multivariate mixtures, Journal of Computational and Graphical Statistics, 18, 505-526.
- Benaglia, T., Chauveau, D. and Hunter, D.R. (2011), Bandwidth Selection in an EM-like algorithm for nonparametric multivariate mixtures. Nonparametric Statistics and Mixture Models: A Festschrift in Honor of Thomas P. Hettmansperger. World Scientific Publishing Co., pages 15-27.
- Chauveau D., Hunter D. R. and Levine M. (2014), Semi-Parametric Estimation for Conditional Independence Multivariate Finite Mixture Models. Preprint (under revision).
- Levine, M., Hunter, D. and Chauveau, D. (2011), Maximum Smoothed Likelihood for Multivariate Mixtures, Biometrika 98(2): 403-416.

See Also

npEM, plot.npEM, normmixrm.sim, spEMsymloc, spEM, plotseq.npEM

Examples

```
## Examine and plot water-level task data set.
## Block structure pairing clock angles that are directly opposite one
## another (1:00 with 7:00, 2:00 with 8:00, etc.)
set.seed(111) # Ensure that results are exactly reproducible
data(Waterdata)
blockid <- c(4,3,2,1,3,4,1,2) # see Benaglia et al (2009a)

## Not run:
a <- npEM(Waterdata[,3:10], mu0=3, blockid=blockid, bw=4)  # npEM solution
b <- npMSL(Waterdata[,3:10], mu0=3, blockid=blockid, bw=4) # smoothed version

# Comparisons on the 4 default plots, one for each block
par(mfrow=c(2,2))
for (l in 1:4){
plot(a, blocks=l, breaks=5*(0:37)-92.5,
xlim=c(-90,90), xaxt="n",ylim=c(0,.035), xlab="")
plot(b, blocks=l, hist=FALSE, newplot=FALSE, addlegend=FALSE, lty=2,
dens.col=1)
axis(1, at=30*(1:7)-120, cex.axis=1)
legend("topleft",c("npMSL"),lty=2, lwd=2)}

## End(Not run)
```

---

plot.mixEM *Various Plots Pertaining to Mixture Models*

Description

Takes an object of class mixEM and returns various graphical output for select mixture models.

Usage

```
## S3 method for class 'mixEM'
plot(x, whichplots = 1,
     loglik = 1 %in% whichplots,
     density = 2 %in% whichplots,
     xlab1="Iteration", ylab1="Log-Likelihood",
     main1="Observed Data Log-Likelihood", col1=1, lwd1=2,
     xlab2=NULL, ylab2=NULL, main2=NULL, col2=NULL,
     lwd2=2, alpha = 0.05, marginal = FALSE, ...)
```

Arguments

x	An object of class mixEM.
whichplots	vector telling which plots to produce: 1 = loglikelihood plot, 2 = density plot. Irrelevant if loglik and density are specified.
loglik	If TRUE, a plot of the log-likelihood versus the EM iterations is given.
density	Graphics pertaining to certain mixture models. The details are given below.
xlab1, ylab1, main1, col1, lwd1	Graphical parameters xlab, ..., lwd to be passed to the loglikelihood plot. Trying to change these parameters using xlab, ..., lwd will result in an error, but all other graphical parameters are passed directly to the plotting functions via ...
xlab2, ylab2, main2, col2, lwd2	Same as xlab1 etc. but for the density plot
alpha	A vector of significance levels when constructing confidence ellipses and confidence bands for the mixture of multivariate normals and mixture of regressions cases, respectively. The default is 0.05.
marginal	For the mixture of bivariate normals, should optional marginal histograms be included?
...	Graphical parameters passed to plot command.

Value

plot.mixEM returns a plot of the log-likelihood versus the EM iterations by default for all objects of class mixEM. In addition, other plots may be produced for the following k-component mixture model functions:

normalmixEM	A histogram of the raw data is produced along with k density curves determined by normalmixEM.
repnormmixEM	A histogram of the raw data produced in a similar manner as for normalmixEM.
mvnormalmixEM	A 2-dimensional plot with each point color-coded to denote its most probable component membership. In addition, the estimated component means are plotted along with (1 - alpha)% bivariate normal density contours. These ellipses

are constructed by assigning each value to their component of most probable membership and then using normal theory. Optional marginal histograms may also be produced.

`regmixEM` A plot of the response versus the predictor with each point color-coded to denote its most probable component membership. In addition, the estimated component regression lines are plotted along with (1 - `alpha`)% Working-Hotelling confidence bands. These bands are constructed by assigning each value to their component of most probable membership and then performing least squares estimation.

`logisregmixEM` A plot of the binary response versus the predictor with each point color-coded to denote its most probable compopnent membership. In addition, the estimate component logistic regression lines are plotted.

`regmixEM.mixed` Provides a 2x2 matrix of plots summarizing the posterior slope and posterior intercept terms from a mixture of random effects regression. See `post.beta` for a more detailed description.

See Also

`post.beta`

Examples

```
##Analyzing the Old Faithful geyser data with a 2-component mixture of normals.

data(faithful)
attach(faithful)
set.seed(100)
out <- normalmixEM(waiting, arbvar = FALSE, verb = TRUE,
                   epsilon = 1e-04)
plot(out, density = TRUE, w = 1.1)

##Fitting randomly generated data with a 2-component location mixture of bivariate normals.

x.1 <- rmvnorm(40, c(0, 0))
x.2 <- rmvnorm(60, c(3, 4))
X.1 <- rbind(x.1, x.2)

out.1 <- mvnormalmixEM(X.1, arbvar = FALSE, verb = TRUE,
                       epsilon = 1e-03)
plot(out.1, density = TRUE, alpha = c(0.01, 0.05, 0.10),
     marginal = TRUE)
```

`plot.mixMCMC` *Various Plots Pertaining to Mixture Model Output Using MCMC Methods*

Description

Takes an object of class mixMCMC and returns various graphical output for select mixture models.

Usage

```
## S3 method for class 'mixMCMC'
plot(x, trace.plots = TRUE,
     summary.plots = FALSE, burnin = 2000, ...)
```

Arguments

x	An object of class mixMCMC.
trace.plots	If TRUE, trace plots of the various parameters estimated by the MCMC methods is given.
summary.plots	Graphics pertaining to certain mixture models. The details are given below.
burnin	The values 1 to burnin are dropped when producing the plots in summary.plots.
...	Graphical parameters passed to regcr function.

Value

plot.mixMCMC returns trace plots of the various parameters estimated by the MCMC methods for all objects of class mixMCMC. In addition, other plots may be produced for the following k-component mixture model functions:

regmixMH	Credible bands for the regression lines in a mixture of linear regressions. See regcr for more details.

See Also

regcr

Examples

```
## M-H algorithm for NOdata with acceptance rate about 40%.

data(NOdata)
attach(NOdata)
set.seed(100)
beta <- matrix(c(1.3, -0.1, 0.6, 0.1), 2, 2)
sigma <- c(.02, .05)
MH.out <- regmixMH(Equivalence, NO, beta = beta, s = sigma,
                   sampsize = 2500, omega = .0013)
plot(MH.out, summary.plots = TRUE, burnin = 2450,
     alpha = 0.01)
```

plot.mvnpEM *Plots of Marginal Density Estimates from the mvnpEM Algorithm Output*

Description

Takes an object of class mvnpEM, as the one returned by the mvnpEM algorithm, and returns a set of plots of the density estimates for each coordinate within each multivariate block. All the components are displayed on each plot so it is possible to see the mixture structure for each coordinate and block. The final bandwidth values are also displayed, in a format depending on the bandwidth strategy .

Usage

```
## S3 method for class 'mvnpEM'
plot(x, truenorm = FALSE, lambda = NULL, mu = NULL, v = NULL,
            lgdcex = 1, ...)
```

Arguments

x	An object of class mvnpEM such as the output of the mvnpEM function
truenorm	Mostly for checking purpose, if the nonparametric model is to be compared with a multivariate Gaussian mixture as the true model.
lambda	true weight parameters, for Gaussian models only (see above)
mu	true mean parameters, for Gaussian models only (see above)
v	true covariance matrices, for Gaussian models only (see above)
lgdcex	Character expansion factor for legend.
...	Any remaining arguments are passed to hist.

Value

plot.mvnpEM currently just plots the figure.

See Also

mvnpEM, npEM, density.npEM

Examples

```
# example as in Chauveau and Hoang (2015) with 6 coordinates
## Not run:
m=2; r=6; blockid <-c(1,1,2,2,3,3) # 3 bivariate blocks
# generate some data x ...
a <- mvnpEM(x, mu0=2, blockid, samebw=F) # adaptive bandwidth
plot(a) # this S3 method produces 6 plots of univariate marginals
summary(a)
## End(Not run)
```

plot.npEM *Plot Nonparametric or Semiparametric EM Output*

Description

Takes an object of class npEM and returns a set of plots of the density estimates for each block and each component. There is one plot per block, with all the components displayed on each block so it is possible to see the mixture structure for each block.

Usage

```
## S3 method for class 'npEM'
plot(x, blocks = NULL, hist=TRUE, addlegend = TRUE,
      scale=TRUE, title=NULL, breaks="Sturges", ylim=NULL, dens.col,
      newplot = TRUE, pos.legend = "topright", cex.legend = 1, ...)
## S3 method for class 'spEM'
plot(x, blocks = NULL, hist=TRUE, addlegend = TRUE,
      scale=TRUE, title=NULL, breaks="Sturges", ylim=NULL, dens.col,
      newplot = TRUE, pos.legend = "topright", cex.legend = 1, ...)
```

Arguments

x	An object of class npEM such as the output of the npEM function
blocks	Blocks (of repeated measures coordinates) to plot; not relevant for univariate case. Default is to plot all blocks.
hist	If TRUE, superimpose density estimate plots on a histogram of the data
addlegend	If TRUE, adds legend to the plot.
scale	If TRUE, scale each density estimate by its corresponding estimated mixing proportion, so that the total area under all densities equals 1 and the densities plotted may be added to produce an estimate of the mixture density. When FALSE, each density curve has area 1 in the plot.
title	Alternative vector of main titles for plots (recycled as many times as needed)
breaks	Passed directly to the hist function
ylim	ylim parameter to use for all plots, if desired. If not given, each plot uses its own ylim that ensures that no part of the plot will go past the top of the plotting area.
dens.col	Color values to use for the individual component density functions, repeated as necessary. Default value is 2:(m+1).
newplot	If TRUE, creates a new plot.
pos.legend	Single argument specifying the position of the legend. See 'Details' section of legend.
cex.legend	Character expansion factor for legend.
...	Any remaining arguments are passed to the hist and lines functions.

Value

`plot.npEM` returns a list with two elements:

`x`	List of matrices. The jth column of the ith matrix is the vector of x-values for the jth density in the ith plot.
`y`	y-values, given in the same form as the x-values.

See Also

`npEM`, `density.npEM`, `spEMsymloc`, `plotseq.npEM`

Examples

```
## Examine and plot water-level task data set.

## First, try a 3-component solution where no two coordinates are
## assumed i.d.
data(Waterdata)
set.seed(100)
## Not run:
a <- npEM(Waterdata[,3:10], 3, bw=4)
par(mfrow=c(2,4))
plot(a) # This produces 8 plots, one for each coordinate

## End(Not run)

## Not run:
## Next, same thing but pairing clock angles that are directly opposite one
## another (1:00 with 7:00, 2:00 with 8:00, etc.)
b <- npEM(Waterdata[,3:10], 3, blockid=c(4,3,2,1,3,4,1,2), bw=4)
par(mfrow=c(2,2))
plot(b) # Now only 4 plots, one for each block

## End(Not run)
```

---

plot.spEMN01 — *Plot mixture pdf for the semiparametric mixture model output by spEMsymlocN01*

Description

Plot mixture density for the semiparametric mixture model output by spEMsymlocN01, with one component known and set to normal(0,1), and a symmetric nonparametric density with location parameter.

Usage

```
## S3 method for class 'spEMN01'
plot(x, bw = x$bandwidth, knownpdf = dnorm, add.plot = FALSE, ...)
```

Arguments

x	An object of class "spEMN01" as returned by spEMsymlocN01
bw	Bandwidth for weighted kernel density estimation.
knownpdf	The known density of component 1, default to dnorm.
add.plot	Set to TRUE to add to an existing plot.
...	further arguments passed to plot if add.plot = FALSE, and to lines if add.plot = TRUE.

Value

A plot of the density of the mixture

Author(s)

Didier Chauveau

References

- Chauveau, D., Saby, N., Orton, T. G., Lemercier B., Walter, C. and Arrouys, D. Large-scale simultaneous hypothesis testing in soil monitoring: A semi-parametric mixture approach, preprint (2013).

See Also

spEMsymlocN01

---

plotexpRMM *Plot sequences from the EM algorithm for censored mixture of exponentials*

Description

Function for plotting sequences of estimates along iterations, from an object returned by the expRMM_EM, an EM algorithm for mixture of exponential distributions with randomly right censored data (see reference below).

Usage

```
plotexpRMM(a, title=NULL, rowstyle=TRUE, subtitle=NULL, ...)
```

Arguments

a	An object returned by expRMM_EM.
title	The title of the plot, set to some default value if NULL.
rowstyle	Window organization, for plots in rows (the default) or columns.
subtitle	A subtitle for the plot, set to some default value if NULL.
...	Other parameters (such as lwd) passed to plot, lines, and legend commands.

Value

The plot returned

Author(s)

Didier Chauveau

References

- Bordes, L., and Chauveau, D. (2016), Stochastic EM algorithms for parametric and semiparametric mixture models for right-censored lifetime data, Computational Statistics, Volume 31, Issue 4, pages 1513-1538. http://link.springer.com/article/10.1007/s00180-016-0661-7

See Also

Related functions: expRMM_EM, summary.mixEM, plot.mixEM.

Other models and algorithms for censored lifetime data (name convention is model_algorithm): weibullRMM_SEM, spRMM_SEM.

Examples

```
n=300 # sample size
m=2   # number of mixture components
lambda <- c(1/3,1-1/3); rate <- c(1,1/10) # mixture parameters
set.seed(1234)
x <- rexpmix(n, lambda, rate) # iid ~ exponential mixture
cs=runif(n,0,max(x)) # Censoring (uniform) and incomplete data
t <- apply(cbind(x,cs),1,min) # observed or censored data
d <- 1*(x <= cs)              # censoring indicator

###### EM for RMM, exponential lifetimes
l0 <- rep(1/m,m); r0 <- c(1, 0.5) # "arbitrary" initial values
a <- expRMM_EM(t, d, lambda=l0, rate=r0, k = m)
summary(a)             # EM estimates etc
plotexpRMM(a, lwd=2) # plot of EM sequences
```

plotFDR *Plot False Discovery Rate (FDR) estimates from output by EM-like strategies*

Description

Plot FDR(p_i) estimates against index of sorted p-values from, e.g., normalmixEM or the semiparametric mixture model posterior probabilities output by spEMsymlocN01, or any EM-algorithm like normalmixEM which returns posterior probabilities. The function can simultaneously plot FDR estimates from two strategies for comparison. Plot of the true FDR can be added if complete data are available (typically in simulation studies).

Usage

```
plotFDR(post1, post2 = NULL, lg1 = "FDR 1", lg2 = NULL, title = NULL,
        compH0 = 1, alpha = 0.1, complete.data = NULL, pctfdr = 0.3)
```

Arguments

post1	The matrix of posterior probabilities from objects such as the output from spEMsymlocN01. The rows need to be sorted by increasing pvalues.
post2	A second object like post1 if comparison is desired, also sorted by increasing pvalues.
lg1	Text describing the FDR estimate in post1.
lg2	Text describing the FDR estimate in post2 if provided.
title	Plot title, a default is provided if NULL.
compH0	The component indicator associated to the null hypothesis H0, normally 1 since it is defined in this way in spEMsymlocN01, but in case of label switching in other algorithms it can be set to 2.
alpha	The target FDR level; the index at which the FDR estimate crosses the horizontal line for level alpha gives the maximum number of cases to reject.
complete.data	An array with n lines and 2 columns, with the component indicator in column 1 and the p-values in column 2, sorted by p-values.
pctfdr	The level up to which the FDR is plotted, i.e. the scale of the vertical axis.

Value

A plot of one or two FDR estimates, with the true FDR if available

Author(s)

Didier Chauveau

References

- Chauveau, D., Saby, N., Orton, T. G., Lemercier B., Walter, C. and Arrouys, D. Large-scale simultaneous hypothesis testing in monitoring carbon content from French soil database – A semi-parametric mixture approach, Geoderma 219-220 (2014), 117-124.

See Also

spEMsymlocN01

`plotseq.npEM` *Plotting sequences of estimates from non- or semiparametric EM-like Algorithm*

Description

Returns plots of the sequences of scalar parameter estimates along iterations from an object of class `npEM`.

Usage

```
  ## S3 method for class 'npEM'
plotseq(x, ...)
```

Arguments

`x`	an object of class `npEM`, as output by `npEM` or `spEMsymloc`
`...`	further parameters that are passed to `plot`

Details

`plotseq.npEM` returns a figure with one plot for each component proportion, and, in the case of `spEMsymloc`, one plot for each component mean.

References

- Benaglia, T., Chauveau, D., and Hunter, D. R. (2009), An EM-like algorithm for semi- and non-parametric estimation in multivariate mixtures, Journal of Computational and Graphical Statistics (to appear).
- Bordes, L., Chauveau, D., and Vandekerkhove, P. (2007), An EM algorithm for a semiparametric mixture model, Computational Statistics and Data Analysis, 51: 5429-5443.

See Also

`plot.npEM`, `rnormmix`, `npEM`, `spEMsymloc`

Examples

```
## Example from a normal location mixture
n <- 200
set.seed(100)
lambda <- c(1/3,2/3)
mu <- c(0, 4); sigma<-rep(1, 2)
x <- rnormmix(n, lambda, mu, sigma)
b <- spEMsymloc(x, mu0=c(-1, 2), stochastic=FALSE)
plotseq(b)
bst <- spEMsymloc(x, mu0=c(-1, 2), stochastic=TRUE)
plotseq(bst)
```

`plotspRMM` *Plot output from Stochastic EM algorithm for semiparametric scaled mixture of censored data*

Description

Function for plotting various results from an object returned by `spRMM_SEM`, a Stochastic EM algorithm for semiparametric scaled mixture of randomly right censored lifetime data. Four plots of sequences of estimates along iterations, survival and density estimates (see reference below).

Usage

```
plotspRMM(sem, tmax = NULL)
```

Arguments

`sem`	An object returned by `spRMM_SEM`.
`tmax`	The max time for x axis, set to some default value if `NULL`.

Value

The four plots returned

Author(s)

Didier Chauveau

References

- Bordes, L., and Chauveau, D. (2016), Stochastic EM algorithms for parametric and semiparametric mixture models for right-censored lifetime data, Computational Statistics, Volume 31, Issue 4, pages 1513-1538. http://link.springer.com/article/10.1007/s00180-016-0661-7

See Also

Related functions: spRMM_SEM.

Other models and algorithms for censored lifetime data (name convention is model_algorithm): expRMM_EM, weibullRMM_SEM.

Examples

```
# See example(spRMM_SEM)
```

`plotweibullRMM` *Plot sequences from the Stochastic EM algorithm for mixture of Weibull*

Description

Function for plotting sequences of estimates along iterations, from an object returned by weibullRMM_SEM, a Stochastic EM algorithm for mixture of Weibull distributions with randomly right censored data (see reference below).

Usage

```
plotweibullRMM(a, title = NULL, rowstyle = TRUE, subtitle = NULL, ...)
```

Arguments

a	An object returned by weibullRMM_SEM.
title	The title of the plot, set to some default value if NULL.
rowstyle	Window organization, for plots in rows (the default) or columns.
subtitle	A subtitle for the plot, set to some default value if NULL.
...	Other parameters (such as lwd) passed to plot, lines, and legend commands.

Value

The plot returned

Author(s)

Didier Chauveau

References

- Bordes, L., and Chauveau, D. (2016), Stochastic EM algorithms for parametric and semiparametric mixture models for right-censored lifetime data, Computational Statistics, Volume 31, Issue 4, pages 1513-1538. http://link.springer.com/article/10.1007/s00180-016-0661-7

See Also

Related functions: weibullRMM_SEM, summary.mixEM.

Other models and algorithms for censored lifetime data (name convention is model_algorithm): expRMM_EM, spRMM_SEM .

Examples

```
n = 500 # sample size
m = 2 # nb components
lambda=c(0.4, 0.6)
shape <- c(0.5,5); scale <- c(1,20) # model parameters
set.seed(321)
x <- rweibullmix(n, lambda, shape, scale) # iid ~ weibull mixture
cs=runif(n,0,max(x)+10) # iid censoring times
t <- apply(cbind(x,cs),1,min) # censored observations
d <- 1*(x <= cs)              # censoring indicator

## set arbitrary or "reasonable" (e.g., data-driven) initial values
l0 <- rep(1/m,m); sh0 <- c(1, 2); sc0 <- c(2,10)
# Stochastic EM algorithm
a <- weibullRMM_SEM(t, d, lambda = l0, shape = sh0, scale = sc0, maxit = 200)

summary(a) # Parameters estimates etc
plotweibullRMM(a) # default plot of St-EM sequences
```

---

poisregmixEM *EM Algorithm for Mixtures of Poisson Regressions*

Description

Returns EM algorithm output for mixtures of Poisson regressions with arbitrarily many components.

Usage

```
poisregmixEM(y, x, lambda = NULL, beta = NULL, k = 2,
             addintercept = TRUE, epsilon = 1e-08,
             maxit = 10000, verb = FALSE)
```

Arguments

y	An n-vector of response values.
x	An nxp matrix of predictors. See addintercept below.
lambda	Initial value of mixing proportions. Entries should sum to 1. This determines number of components. If NULL, then lambda is random from uniform Dirichlet and number of components is determined by beta.

beta	Initial value of beta parameters. Should be a pxk matrix, where p is the number of columns of x and k is number of components. If NULL, then beta is generated by binning the data into k bins and using glm on the values in each of the bins. If both lambda and beta are NULL, then number of components is determined by k.
k	Number of components. Ignored unless lambda and beta are both NULL.
addintercept	If TRUE, a column of ones is appended to the x matrix before the value of p is calculated.
epsilon	The convergence criterion.
maxit	The maximum number of iterations.
verb	If TRUE, then various updates are printed during each iteration of the algorithm.

Value

poisregmixEM returns a list of class mixEM with items:

x	The predictor values.
y	The response values.
lambda	The final mixing proportions.
beta	The final Poisson regression coefficients.
loglik	The final log-likelihood.
posterior	An nxk matrix of posterior probabilities for observations.
all.loglik	A vector of each iteration's log-likelihood.
restarts	The number of times the algorithm restarted due to unacceptable choice of initial values.
ft	A character vector giving the name of the function.

References

McLachlan, G. J. and Peel, D. (2000) *Finite Mixture Models*, John Wiley \& Sons, Inc.

Wang, P., Puterman, M. L., Cockburn, I. and Le, N. (1996) Mixed Poisson Regression Models with Covariate Dependent Rates, *Biometrics*, **52(2)**, 381–400.

See Also

logisregmixEM

Examples

```
## EM output for data generated from a 2-component model.

set.seed(100)
beta <- matrix(c(1, .5, .7, -.8), 2, 2)
x <- runif(50, 0, 10)
xbeta <- cbind(1, x)%*%beta
w <- rbinom(50, 1, .5)
```

```
y <- w*rpois(50, exp(xbeta[, 1]))+(1-w)*rpois(50, exp(xbeta[, 2]))
out <- poisregmixEM(y, x, verb = TRUE,  epsilon = 1e-03)
out
```

---

print.mvnpEM *Printing of Results from the mvnpEM Algorithm Output*

Description

print method for class mvnpEM.

Usage

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

Arguments

x	an object of class mvnpEM such as a result of a call to mvnpEM
...	Additional arguments to print

Details

print.mvnpEM prints the elements of an mvnpEM object without printing the data or the posterior probabilities. (These may still be accessed as x$data and x$posteriors.)

Value

print.mvnpEM returns (invisibly) the full value of x itself, including the data and posteriors elements.

See Also

mvnpEM, plot.mvnpEM summary.mvnpEM

Examples

```
# Example as in Chauveau and Hoang (2015) with 6 coordinates
## Not run:
m=2; r=6; blockid <-c(1,1,2,2,3,3) # 3 bivariate blocks
# generate some data x ...
a <- mvnpEM(x, mu0=2, blockid, samebw=F) # adaptive bandwidth
print(a)
## End(Not run)
```

print.npEM *Printing non- and semi-parametric multivariate mixture model fits*

Description

print method for class npEM.

Usage

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

Arguments

x	an object of class npEM such as a result of a call to npEM
...	Additional arguments to print

Details

print.npEM prints the elements of an npEM object without printing the data or the posterior probabilities. (These may still be accessed as x$data and x$posteriors.)

Value

print.npEM returns (invisibly) the full value of x itself, including the data and posteriors elements.

See Also

npEM, plot.npEM summary.npEM

Examples

```
data(Waterdata)
set.seed(100)
## Not run: npEM(Waterdata[,3:10], 3, bw=4, verb=FALSE) # Assume indep but not iid
```

RanEffdata — *Simulated Data from 2-Component Mixture of Regressions with Random Effects*

Description

This data set was generated from a 2-component mixture of regressions with random effects.

Usage

```
data(RanEffdata)
```

Format

This data set consists of a list with 100 25x3 matrices. The first column is the response variable, the second column is a column of 1's and the last column is the predictor variable.

See Also

regmixEM.mixed

regcr — *Add a Confidence Region or Bayesian Credible Region for Regression Lines to a Scatterplot*

Description

Produce a confidence or credible region for regression lines based on a sample of bootstrap beta values or posterior beta values. The beta parameters are the intercept and slope from a simple linear regression.

Usage

```
regcr(beta, x, em.beta = NULL, em.sigma = NULL, alpha = .05,
      nonparametric = FALSE, plot = FALSE, xyaxes = TRUE, ...)
```

Arguments

beta	An nx2 matrix of regression parameters. The first column gives the intercepts and the second column gives the slopes.
x	An n-vector of the predictor variable which is necessary when nonparametric = TRUE.
em.beta	The estimates for beta required when obtaining confidence regions. This is required for performing the standardization necessary when obtaining nonparametric confidence regions.

em.sigma	The estimates for the regression standard deviation required when obtaining confidence regions. This is required for performing the standardization necessary when obtaining nonparametric confidence regions.
alpha	The proportion of the beta sample to remove. In other words, 1-alpha is the level of the credible region.
nonparametric	If nonparametric = TRUE, then the region is based on the convex hull of the remaining beta after trimming, which is accomplished using a data depth technique. If nonparametric = FALSE, then the region is based on the asymptotic normal approximation.
plot	If plot = TRUE, lines are added to the existing plot. The type of plot created depends on the value of xyaxes.
xyaxes	If xyaxes = TRUE and plot = TRUE, then a confidence or credible region for the regression lines is plotted on the x-y axes, presumably overlaid on a scatterplot of the data. If xyaxes = FALSE and plot = TRUE, the (convex) credible region for the regression line is plotted on the beta, or intercept-slope, axes, presumably overlaid on a scatterplot of beta.
...	Graphical parameters passed to lines or plot command.

Value

regcr returns a list containing the following items:

boundary	A matrix of points in beta, or intercept-slope, space arrayed along the boundary of the confidence or credible region.
upper	A matrix of points in x-y space arrayed along the upper confidence or credible limit for the regression line.
lower	A matrix of points in x-y space arrayed along the lower confidence or credible limit for the regression line.

See Also

regmixEM, regmixMH

Examples

```
## Nonparametric credible regions fit to NOdata.

data(NOdata)
attach(NOdata)
set.seed(100)
beta <- matrix(c(1.3, -0.1, 0.6, 0.1), 2, 2)
sigma <- c(.02, .05)
MH.out <- regmixMH(Equivalence, NO, beta = beta, s = sigma,
                   sampsize = 2500, omega = .0013)
attach(data.frame(MH.out$theta))
beta.c1 <- cbind(beta0.1[2400:2499], beta1.1[2400:2499])
beta.c2 <- cbind(beta0.2[2400:2499], beta1.2[2400:2499])
plot(NO, Equivalence)
regcr(beta.c1, x = NO, nonparametric = TRUE, plot = TRUE,
```

```
      col = 2)
regcr(beta.c2, x = NO, nonparametric = TRUE, plot = TRUE,
      col = 3)
```

---

regmixEM *EM Algorithm for Mixtures of Regressions*

Description

Returns EM algorithm output for mixtures of multiple regressions with arbitrarily many components.

Usage

```
regmixEM(y, x, lambda = NULL, beta = NULL, sigma = NULL, k = 2,
         addintercept = TRUE, arbmean = TRUE, arbvar = TRUE,
         epsilon = 1e-08, maxit = 10000, verb = FALSE)
```

Arguments

y	An n-vector of response values.
x	An nxp matrix of predictors. See addintercept below.
lambda	Initial value of mixing proportions. Entries should sum to 1. This determines number of components. If NULL, then lambda is random from uniform Dirichlet and number of components is determined by beta.
beta	Initial value of beta parameters. Should be a pxk matrix, where p is the number of columns of x and k is number of components. If NULL, then beta has standard normal entries according to a binning method done on the data. If both lambda and beta are NULL, then number of components is determined by sigma.
sigma	A vector of standard deviations. If NULL, then 1/sigma^2 has random standard exponential entries according to a binning method done on the data. If lambda, beta, and sigma are NULL, then number of components is determined by k.
k	Number of components. Ignored unless all of lambda, beta, and sigma are NULL.
addintercept	If TRUE, a column of ones is appended to the x matrix before the value of p is calculated.
arbmean	If TRUE, each mixture component is assumed to have a different set of regression coefficients (i.e., the betas).
arbvar	If TRUE, each mixture component is assumed to have a different sigma.
epsilon	The convergence criterion.
maxit	The maximum number of iterations.
verb	If TRUE, then various updates are printed during each iteration of the algorithm.

Value

regmixEM returns a list of class mixEM with items:

x	The set of predictors (which includes a column of 1's if addintercept = TRUE).
y	The response values.
lambda	The final mixing proportions.
beta	The final regression coefficients.
sigma	The final standard deviations. If arbmean = FALSE, then only the smallest standard deviation is returned. See scale below.
scale	If arbmean = FALSE, then the scale factor for the component standard deviations is returned. Otherwise, this is omitted from the output.
loglik	The final log-likelihood.
posterior	An nxk matrix of posterior probabilities for observations.
all.loglik	A vector of each iteration's log-likelihood.
restarts	The number of times the algorithm restarted due to unacceptable choice of initial values.
ft	A character vector giving the name of the function.

References

de Veaux, R. D. (1989), Mixtures of Linear Regressions, *Computational Statistics and Data Analysis* 8, 227-245.

Hurn, M., Justel, A. and Robert, C. P. (2003) Estimating Mixtures of Regressions, *Journal of Computational and Graphical Statistics* **12(1)**, 55–79.

McLachlan, G. J. and Peel, D. (2000) *Finite Mixture Models*, John Wiley \& Sons, Inc.

See Also

regcr, regmixMH

Examples

```
## EM output for NOdata.

data(NOdata)
attach(NOdata)
set.seed(100)
em.out <- regmixEM(Equivalence, NO, verb = TRUE, epsilon = 1e-04)
em.out[3:6]
```

regmixEM.lambda *EM Algorithm for Mixtures of Regressions with Local Lambda Estimates*

Description

Returns output for one step of an EM algorithm output for mixtures of multiple regressions where the mixing proportions are estimated locally.

Usage

```
regmixEM.lambda(y, x, lambda = NULL, beta = NULL, sigma = NULL,
                k = 2, addintercept = TRUE, arbmean = TRUE,
                arbvar = TRUE, epsilon = 1e-8, maxit = 10000,
                verb = FALSE)
```

Arguments

y	An n-vector of response values.
x	An nxp matrix of predictors. See addintercept below.
lambda	An nxk matrix of initial local values of mixing proportions. Entries should sum to 1. This determines number of components. If NULL, then lambda is simply one over the number of components.
beta	Initial value of beta parameters. Should be a pxk matrix, where p is the number of columns of x and k is number of components. If NULL, then beta has uniform standard normal entries. If both lambda and beta are NULL, then number of components is determined by sigma.
sigma	k-vector of initial global values of standard deviations. If NULL, then $1/\texttt{sigma}^2$ has random standard exponential entries. If lambda, beta, and sigma are NULL, then number of components is determined by k.
k	The number of components. Ignored unless all of lambda, beta, and sigma are NULL.
addintercept	If TRUE, a column of ones is appended to the x matrix before the value of p is calculated.
arbmean	If TRUE, each mixture component is assumed to have a different set of regression coefficients (i.e., the betas).
arbvar	If TRUE, each mixture component is assumed to have a different sigma.
epsilon	The convergence criterion.
maxit	The maximum number of iterations.
verb	If TRUE, then various updates are printed during each iteration of the algorithm.

Details

Primarily used within regmixEM.loc.

Value

regmixEM.lambda returns a list of class mixEM with items:

x	The set of predictors (which includes a column of 1's if addintercept = TRUE).
y	The response values.
lambda	The inputted mixing proportions.
beta	The final regression coefficients.
sigma	The final standard deviations. If arbmean = FALSE, then only the smallest standard deviation is returned. See scale below.
scale	If arbmean = FALSE, then the scale factor for the component standard deviations is returned. Otherwise, this is omitted from the output.
loglik	The final log-likelihood.
posterior	An nxk matrix of posterior probabilities for observations.
all.loglik	A vector of each iteration's log-likelihood.
restarts	The number of times the algorithm restarted due to unacceptable choice of initial values.
ft	A character vector giving the name of the function.

See Also

regmixEM.loc

Examples

```
## Compare a 2-component and 3-component fit to NOdata.

data(NOdata)
attach(NOdata)
set.seed(100)
out1 <- regmixEM.lambda(Equivalence, NO)
out2 <- regmixEM.lambda(Equivalence, NO, k = 3)
c(out1$loglik, out2$loglik)
```

---

regmixEM.loc *Iterative Algorithm Using EM Algorithm for Mixtures of Regressions with Local Lambda Estimates*

Description

Iterative algorithm returning EM algorithm output for mixtures of multiple regressions where the mixing proportions are estimated locally.

Usage

```
regmixEM.loc(y, x, lambda = NULL, beta = NULL, sigma = NULL,
             k = 2, addintercept = TRUE, kern.l = c("Gaussian",
             "Beta", "Triangle", "Cosinus", "Optcosinus"),
             epsilon = 1e-08, maxit = 10000, kernl.g = 0,
             kernl.h = 1, verb = FALSE)
```

Arguments

y	An n-vector of response values.
x	An nxp matrix of predictors. See addintercept below.
lambda	An nxk matrix of initial local values of mixing proportions. Entries should sum to 1. This determines number of components. If NULL, then lambda is simply one over the number of components.
beta	Initial global values of beta parameters. Should be a pxk matrix, where p is the number of columns of x and k is number of components. If NULL, then beta has uniform standard normal entries. If both lambda and beta are NULL, then number of components is determined by sigma.
sigma	A k-vector of initial global values of standard deviations. If NULL, then $1/\texttt{sigma}^2$ has random standard exponential entries. If lambda, beta, and sigma are NULL, then number of components determined by k.
k	Number of components. Ignored unless all of lambda, beta, and sigma are NULL.
addintercept	If TRUE, a column of ones is appended to the x matrix before the value of p is calculated.
kern.l	The type of kernel to use in the local estimation of lambda.
epsilon	The convergence criterion.
maxit	The maximum number of iterations.
kernl.g	A shape parameter required for the symmetric beta kernel for local estimation of lambda. The default is g = 0 which yields the uniform kernel. Some common values are g = 1 for the Epanechnikov kernel, g = 2 for the biweight kernel, and g = 3 for the triweight kernel.
kernl.h	The bandwidth controlling the size of the window used in the local estimation of lambda around x.
verb	If TRUE, then various updates are printed during each iteration of the algorithm.

Value

regmixEM.loc returns a list of class mixEM with items:

x	The set of predictors (which includes a column of 1's if addintercept = TRUE).
y	The response values.
lambda.x	The final local mixing proportions.
beta	The final global regression coefficients.

sigma	The final global standard deviations.
loglik	The final log-likelihood.
posterior	An nxk matrix of posterior probabilities for observations.
all.loglik	A vector of each iteration's log-likelihood.
restarts	The number of times the algorithm restarted due to unacceptable choice of initial values.
ft	A character vector giving the name of the function.

See Also

regmixEM.lambda

Examples

```
## Compare a 2-component and 3-component fit to NOdata.

data(NOdata)
attach(NOdata)
set.seed(100)
out1 <- regmixEM.loc(Equivalence, NO, kernl.h = 2,
                     epsilon = 1e-02, verb = TRUE)
out2 <- regmixEM.loc(Equivalence, NO, kernl.h = 2, k = 3,
                     epsilon = 1e-02, verb = TRUE)
c(out1$loglik, out2$loglik)
```

---

regmixEM.mixed *EM Algorithm for Mixtures of Regressions with Random Effects*

Description

Returns EM algorithm output for mixtures of multiple regressions with random effects and an option to incorporate fixed effects and/or AR(1) errors.

Usage

```
regmixEM.mixed(y, x, w = NULL, sigma = NULL, arb.sigma = TRUE,
               alpha = NULL, lambda = NULL, mu = NULL,
               rho = NULL, R = NULL, arb.R = TRUE, k = 2,
               ar.1 = FALSE, addintercept.fixed = FALSE,
               addintercept.random = TRUE, epsilon = 1e-08,
               maxit = 10000, verb = FALSE)
```

Arguments

y	A list of N response trajectories with (possibly) varying dimensions of length n_i.
x	A list of N design matrices of dimensions $(n_i) \times p$. Each trajectory in y has its own design matrix.
w	A list of N known explanatory variables having dimensions $(n_i) \times q$. If mixed = FALSE, then w is replaced by a list of N zeros.
sigma	A vector of standard deviations. If NULL, then $1/s^2$ has random standard exponential entries according to a binning method done on the data.
arb.sigma	If TRUE, then sigma is k-dimensional. Else a common standard deviation is assumed.
alpha	A q-vector of unknown regression parameters for the fixed effects. If NULL and mixed = TRUE, then alpha is random from a normal distribution with mean and variance according to a binning method done on the data. If mixed = FALSE, then alpha = 0.
lambda	Initial value of mixing proportions for the assumed mixture structure on the regression coefficients. Entries should sum to 1. This determines number of components. If NULL, then lambda is random from uniform Dirichlet and the number of components is determined by mu.
mu	A pxk matrix of the mean for the mixture components of the random regression coefficients. If NULL, then the columns of mu are random from a multivariate normal distribution with mean and variance determined by a binning method done on the data.
rho	An Nxk matrix giving initial values for the correlation term in an AR(1) process. If NULL, then these values are simulated from a uniform distribution on the interval (-1, 1).
R	A list of N pxp covariance matrices for the mixture components of the random regression coefficients. If NULL, then each matrix is random from a standard Wishart distribution according to a binning method done on the data.
arb.R	If TRUE, then R is a list of N pxp covariance matrices. Else, one common covariance matrix is assumed.
k	Number of components. Ignored unless lambda is NULL.
ar.1	If TRUE, then an AR(1) process on the error terms is included. The default is FALSE.
addintercept.fixed	If TRUE, a column of ones is appended to the matrices in w.
addintercept.random	If TRUE, a column of ones is appended to the matrices in x before p is calculated.
epsilon	The convergence criterion.
maxit	The maximum number of iterations.
verb	If TRUE, then various updates are printed during each iteration of the algorithm.

Value

regmixEM returns a list of class mixEM with items:

x	The predictor values corresponding to the random effects.
y	The response values.
w	The predictor values corresponding to the (optional) fixed effects.
lambda	The final mixing proportions.
mu	The final mean vectors.
R	The final covariance matrices.
sigma	The final component error standard deviations.
alpha	The final regression coefficients for the fixed effects.
rho	The final error correlation values if an AR(1) process is included.
loglik	The final log-likelihood.
posterior.z	An Nxk matrix of posterior membership probabilities.
posterior.beta	A list of N pxk matrices giving the posterior regression coefficient values.
all.loglik	A vector of each iteration's log-likelihood.
restarts	The number of times the algorithm restarted due to unacceptable choice of initial values.
ft	A character vector giving the name of the function.

References

Xu, W. and Hedeker, D. (2001) A Random-Effects Mixture Model for Classifying Treatment Response in Longitudinal Clinical Trials, *Journal of Biopharmaceutical Statistics*, **11(4)**, 253–273.

Young, D. S. and Hunter, D. R. (2015) Random Effects Regression Mixtures for Analyzing Infant Habituation, *Journal of Applied Statistics*, **42(7)**, 1421–1441.

See Also

regmixEM, post.beta

Examples

```
## EM output for simulated data from 2-component mixture of random effects.

data(RanEffdata)
set.seed(100)
x <- lapply(1:length(RanEffdata), function(i)
            matrix(RanEffdata[[i]][, 2:3], ncol = 2))
x <- x[1:20]
y <- lapply(1:length(RanEffdata), function(i)
            matrix(RanEffdata[[i]][, 1], ncol = 1))
y <- y[1:20]
lambda <- c(0.45, 0.55)
mu <- matrix(c(0, 4, 100, 12), 2, 2)
```

```
sigma <- 2
R <- list(diag(1, 2), diag(1, 2))
em.out <- regmixEM.mixed(y, x, sigma = sigma, arb.sigma = FALSE,
                         lambda = lambda, mu = mu, R = R,
                         addintercept.random = FALSE,
                         epsilon = 1e-02, verb = TRUE)
em.out[4:10]
```

---

regmixMH *Metropolis-Hastings Algorithm for Mixtures of Regressions*

Description

Return Metropolis-Hastings (M-H) algorithm output for mixtures of multiple regressions with arbitrarily many components.

Usage

```
regmixMH(y, x, lambda = NULL, beta = NULL, s = NULL, k = 2,
         addintercept = TRUE, mu = NULL, sig = NULL, lam.hyp = NULL,
         sampsize = 1000, omega = 0.01, thin = 1)
```

Arguments

y	An n-vector of response values.
x	An nxp matrix of predictors. See addintercept below.
lambda	Initial value of mixing proportions. Entries should sum to 1. This determines number of components. If NULL, then lambda is random from uniform Dirichlet and number of components is determined by beta.
beta	Initial value of beta parameters. Should be a pxk matrix, where p is the number of columns of x and k is number of components. If NULL, then beta has uniform standard normal entries. If both lambda and beta are NULL, then number of components is determined by s.
s	k-vector of standard deviations. If NULL, then $1/s^2$ has random standard exponential entries. If lambda, beta, and s are NULL, then number of components determined by k.
k	Number of components. Ignored unless all of lambda, beta, and s are NULL.
addintercept	If TRUE, a column of ones is appended to the x matrix before the value of p is calculated.
mu	The prior hyperparameter of same size as beta; the means of beta components. If NULL, these are set to zero.
sig	The prior hyperparameter of same size as beta; the standard deviations of beta components. If NULL, these are all set to five times the overall standard deviation of y.

lam.hyp	The prior hyperparameter of length k for the mixing proportions (i.e., these are hyperparameters for the Dirichlet distribution). If NULL, these are generated from a standard uniform distribution and then scaled to sum to 1.
sampsize	Size of posterior sample returned.
omega	Multiplier of step size to control M-H acceptance rate. Values closer to zero result in higher acceptance rates, generally.
thin	Lag between parameter vectors that will be kept.

Value

regmixMH returns a list of class mixMCMC with items:

x	A nxp matrix of the predictors.
y	A vector of the responses.
theta	A (sampsize/thin) x q matrix of MCMC-sampled q-vectors, where q is the total number of parameters in beta, s, and lambda.
k	The number of components.

References

Hurn, M., Justel, A. and Robert, C. P. (2003) Estimating Mixtures of Regressions, *Journal of Computational and Graphical Statistics* **12(1)**, 55–79.

See Also

regcr

Examples

```
## M-H algorithm for NOdata with acceptance rate about 40%.

data(NOdata)
attach(NOdata)
set.seed(100)
beta <- matrix(c(1.3, -0.1, 0.6, 0.1), 2, 2)
sigma <- c(.02, .05)
MH.out <- regmixMH(Equivalence, NO, beta = beta, s = sigma,
                   sampsize = 2500, omega = .0013)
MH.out$theta[2400:2499,]
```

`regmixmodel.sel` *Model Selection in Mixtures of Regressions*

Description

Assess the number of components in a mixture of regressions model using the Akaike's information criterion (AIC), Schwartz's Bayesian information criterion (BIC), Bozdogan's consistent AIC (CAIC), and Integrated Completed Likelihood (ICL).

Usage

```
regmixmodel.sel(x, y, w = NULL, k = 2, type = c("fixed",
                "random", "mixed"), ...)
```

Arguments

`x`	An nxp matrix (or list) of predictors. If an intercept is required, then `x` must NOT include a column of 1's! Requiring an intercept may be controlled through arguments specified in `....`
`y`	An n-vector (or list) of response values.
`w`	An optional list of fixed effects predictors for type "mixed" or "random".
`k`	The maximum number of components to assess.
`type`	The type of regression mixture to use. If "fixed", then a mixture of regressions with fixed effects will be used. If "random", then a mixture of regressions where the random effects regression coefficients are assumed to come from a mixture will be used. If "mixed", the mixture structure used is the same as "random", except a coefficient of fixed effects is also assumed.
`...`	Additional arguments passed to the EM algorithm used for calculating the type of regression mixture specified in `type`.

Value

`regmixmodel.sel` returns a matrix of the AIC, BIC, CAIC, and ICL values along with the winner (i.e., the highest value given by the model selection criterion) for various types of regression mixtures.

References

Biernacki, C., Celeux, G. and Govaert, G. (2000) Assessing a Mixture Model for Clustering with the Integrated Completed Likelihood, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **22(7)**, 719–725.

Bozdogan, H. (1987) Model Selection and Akaike's Information Criterion (AIC): The General Theory and its Analytical Extensions, *Psychometrika* **52**, 345–370.

See Also

regmixEM, regmixEM.mixed

Examples

```
## Assessing the number of components for NOdata.

data(NOdata)
attach(NOdata)
set.seed(100)
regmixmodel.sel(x = NO, y = Equivalence, k = 3, type = "fixed")
```

---

repnormmixEM *EM Algorithm for Mixtures of Normals with Repeated Measurements*

Description

Returns EM algorithm output for mixtures of normals with repeated measurements and arbitrarily many components.

Usage

```
repnormmixEM(x, lambda = NULL, mu = NULL, sigma = NULL, k = 2,
             arbmean = TRUE, arbvar = TRUE, epsilon = 1e-08,
             maxit = 10000, verb = FALSE)
```

Arguments

x	An mxn matrix of data. The columns correspond to the subjects and the rows correspond to the repeated measurements.
lambda	Initial value of mixing proportions. Entries should sum to 1. This determines number of components. If NULL, then lambda is random from uniform Dirichlet and number of components is determined by mu.
mu	A k-vector of component means. If NULL, then mu is determined by a normal distribution according to a binning method done on the data. If both lambda and mu are NULL, then number of components is determined by sigma.
sigma	A vector of standard deviations. If NULL, then $1/\texttt{sigma}^2$ has random standard exponential entries according to a binning method done on the data. If lambda, mu, and sigma are NULL, then number of components is determined by k.
k	Number of components. Ignored unless all of lambda, mu, and sigma are NULL.
arbmean	If TRUE, then the component densities are allowed to have different mus. If FALSE, then a scale mixture will be fit.
arbvar	If TRUE, then the component densities are allowed to have different sigmas. If FALSE, then a location mixture will be fit.
epsilon	The convergence criterion.
maxit	The maximum number of iterations.
verb	If TRUE, then various updates are printed during each iteration of the algorithm.

Value

repnormmixEM returns a list of class mixEM with items:

x	The raw data.
lambda	The final mixing proportions.
mu	The final mean parameters.
sigma	The final standard deviations. If arbmean = FALSE, then only the smallest standard deviation is returned. See scale below.
scale	If arbmean = FALSE, then the scale factor for the component standard deviations is returned. Otherwise, this is omitted from the output.
loglik	The final log-likelihood.
posterior	An nxk matrix of posterior probabilities for observations.
all.loglik	A vector of each iteration's log-likelihood.
restarts	The number of times the algorithm restarted due to unacceptable choice of initial values.
ft	A character vector giving the name of the function.

References

Hettmansperger, T. P. and Thomas, H. (2000) Almost Nonparametric Inference for Repeated Measures in Mixture Models, *Journal of the Royals Statistical Society, Series B* **62(4)** 811–825.

See Also

normalmixEM

Examples

```
## EM output for the water-level task data set.

data(Waterdata)
set.seed(100)
water <- t(as.matrix(Waterdata[,3:10]))
em.out <- repnormmixEM(water, k = 2, verb = TRUE, epsilon = 1e-03)
em.out
```

---

repnormmixmodel.sel *Model Selection in Mixtures of Normals with Repeated Measures*

Description

Assess the number of components in a mixture model with normal components and repeated measures using the Akaike's information criterion (AIC), Schwartz's Bayesian information criterion (BIC), Bozdogan's consistent AIC (CAIC), and Integrated Completed Likelihood (ICL).

Usage

```
repnormmixmodel.sel(x, k = 2, ...)
```

Arguments

x	An mxn matrix of observations. The rows correspond to the repeated measures and the columns correspond to the subject.
k	The maximum number of components to assess.
...	Additional arguments passed to repnormmixEM.

Value

repnormmixmodel.sel returns a matrix of the AIC, BIC, CAIC, and ICL values along with the winner (i.e., the highest value given by the model selection criterion) for a mixture of normals with repeated measures.

References

Biernacki, C., Celeux, G., and Govaert, G. (2000). Assessing a Mixture Model for Clustering with the Integrated Completed Likelihood. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(7):719-725.

Bozdogan, H. (1987). Model Selection and Akaike's Information Criterion (AIC): The General Theory and its Analytical Extensions. *Psychometrika*, 52:345-370.

See Also

repnormmixEM

Examples

```
## Assessing the number of components for the water-level task data set.

data(Waterdata)
water<-t(as.matrix(Waterdata[,3:10]))
set.seed(100)
out <- repnormmixmodel.sel(water, k = 3, epsilon = 5e-01)
out
```

---

rexpmix *Simulate from Mixtures of Exponentials*

Description

Simulate from a mixture of univariate exponential distributions.

Usage

```
rexpmix(n, lambda = 1, rate = 1)
```

Arguments

n	Number of cases to simulate.
lambda	Vector of mixture probabilities, with length equal to m, the desired number of components (subpopulations). This is assumed to sum to 1.
rate	Vector of component rates.

Value

rexpmix returns an n-vector sampled from an m-component mixture of univariate exponential distributions.

See Also

rnormmix, rmvnormmix for Gaussian mixtures, rweibullmix for mixture of Weibull distributions.

Examples

```
## Generate data from a 2-component mixture of exponentials.
n=300 # sample size
m=2 # nb components
lambda=c(1/3, 2/3); rate = c(1,1/10) # parameters
set.seed(1234)
x <- rexpmix(n, lambda, rate) # iid ~ exp mixture

## histogram of the simulated data.
hist(x, col=8)
```

---

rmvnorm *Simulate from a Multivariate Normal Distribution*

Description

Simulate from a multiviate normal distribution

Usage

```
rmvnorm(n, mu=NULL, sigma=NULL)
```

Arguments

n	Number of vectors to simulate
mu	mean vector
sigma	covariance matrix, assumed symmetric and nonnegative definite

Details

This function uses an eigen decomposition assuming sigma is symmetric. In particular, the upper triangle of sigma is ignored.

Value

An $n \times d$ matrix in which each row is an independently generated realization from the desired multivariate normal distribution

See Also

eigen, dnorm, dmvnorm

---

rmvnormmix *Simulate from Multivariate (repeated measures) Mixtures of Normals*

Description

Simulate from a mixture of multivariate zero-correlation normal distributions

Usage

```
rmvnormmix(n, lambda=1, mu=0, sigma=1)
```

Arguments

n	Number of cases to simulate.
lambda	Vector of mixture probabilities with length equal to m, the desired number of components. This is assumed to sum to 1; if not, it is normalized.
mu	Matrix of means of dimensions $m \times r$, where m is the number of components (subpopulations) and r is the number of coordinates (repeated measurements) per case. Note: mu is automatically coerced to a matrix with m rows even if it is not given in this form, which can lead to unexpected behavior in some cases.
sigma	Matrix of standard deviations, same dimensions as mu. The coordinates within a case are independent, conditional on the mixture component. (There is marginal correlation among the coordinates, but this is due to the mixture structure only.) Note: sigma is automatically coerced to a matrix with m rows even if it is not given in this form, which can lead to unexpected behavior in some cases.

Details

It is possible to generate univariate standard normal random variables using the default values (but why bother?). The case of conditionally iid coordinates is covered by the situation in which all columns in mu and sigma are identical.

Value

rmvnormmix returns an $n \times r$ matrix in which each row is a sample from one of the components of a mixture of zero-correlation multivariate normals. The mixture structure induces nonzero correlations among the coordinates.

See Also

rnormmix

Examples

```
##Generate data from a 2-component mixture of trivariate normals.

set.seed(100)
n <- 200
lambda <- rep(1, 2)/2
mu <- matrix(2*(1:6), 2, 3)
sigma <- matrix(1,2,3)
mydata<-rmvnormmix(n, lambda, mu, sigma)

## Now check to see if we can estimate mixture densities well:
title <- paste("Does this resemble N(", mu[1,], ",1) and N(", mu[2,],",1)?",
               sep="")
plot(npEM(mydata, 2), title=title)
```

---

rnormmix *Simulate from Mixtures of Normals*

Description

Simulate from a mixture of univariate normal distributions.

Usage

```
rnormmix(n, lambda=1, mu=0, sigma=1)
```

Arguments

n	Number of cases to simulate.
lambda	Vector of mixture probabilities, with length equal to m, the desired number of components (subpopulations). This is assumed to sum to 1; if not, it is normalized.
mu	Vector of means.
sigma	Vector of standard deviations.

Details

This function simply calls rmvnormmix.

Value

rnormmix returns an n-vector sampled from an m-component mixture of univariate normal distributions.

See Also

makemultdata, rmvnormmix

Examples

```
##Generate data from a 2-component mixture of normals.

set.seed(100)
n <- 500
lambda <- rep(1, 2)/2
mu <- c(0, 5)
sigma <- rep(1, 2)
mixnorm.data <- rnormmix(n, lambda, mu, sigma)

##A histogram of the simulated data.

hist(mixnorm.data)
```

RodFramedata *Rod and Frame Task Data Set*

Description

This data set involves assessing children longitudinally at 6 age points from ages 4 through 18 years for the rod and frame task. This task sits the child in a darkened room in front of a luminous square frame tilted at 28 degrees on its axis to the left or right. Centered inside the frame was a luminous rod also tilted 28 degrees to the left or right. The child's task was to adjust the rod to the vertical position and the absolute deviation from the vertical (in degrees) was the measured response.

Usage

```
data(RodFramedata)
```

Format

This data frame consists of 140 children (the rows). Column 1 is the subject number and column 2 is the sex (0=MALE and 1=FEMALE). Columns 3 through 26 give the 8 responses at each of the ages 4, 5, and 7. Columns 27 through 56 give the 10 responses at each of the ages 11, 14, and 18. A value of 99 denotes missing data.

Source

Thomas, H. and Dahlin, M. P. (2005) Individual Development and Latent Groups: Analytical Tools for Interpreting Heterogeneity, *Developmental Review* **25(2)**, 133–154.

RTdata *Reaction Time (RT) Data Set*

Description

This data set involves normally developing children 9 years of age presented with two visual simuli on a computer monitor. The left image is the target stimuli and on the right is either an exact copy or a mirror image of the target stimuli. The child must press one key if it is a copy or another key if it is a mirror image. The data consists of the reaction times (RT) of the 197 children who provided correct responses for all 6 task trials.

Usage

```
data(RTdata)
```

Format

This data frame consists of 197 children (the rows) and their 6 responses (the columns) to the stimulus presented. The response (RT) is recorded in milliseconds.

References

Cruz-Medina, I. R., Hettmansperger, T. P. and Thomas, H. (2004) Semiparametric Mixture Models and Repeated Measures: The Multinomial Cut Point Model, *Applied Statistics* **53(3)**, 463–474.

Miller, C. A., Kail, R., Leonard, L. B. and Tomblin, J. B. (2001) Speed of Processing in Children with Specific Language Impairment, *Journal of Speech, Language, and Hearing Research* **44(2)**, 416–433.

See Also

RTdata2

RTdata2 *Reaction Time (RT) Data Set \# 2*

Description

This data set involves normally developing children 9 years of age presented visual simuli on a computer monitor. There are three different experimental conditions, according to the length of the delay after which the stimulus was displayed on the screen. Each subject experienced each condition eight times, and these 24 trials were given in random order. These data give the 82 children for whom there are complete measurements among over 200 total subjects.

Usage

```
data(RTdata2)
```

Format

This data frame consists of 82 children (the rows) and their 24 responses (the columns) to the stimulus presented. The response is recorded in milliseconds. The columns are not in the order in which the stimuli were presented to the children; rather, they are arranged into three blocks of eight columns each so that each eight-column block contains only trials from one of the three conditions.

References

Miller, C. A., Kail, R., Leonard, L. B. and Tomblin, J. B. (2001) Speed of Processing in Children with Specific Language Impairment, *Journal of Speech, Language, and Hearing Research* **44(2)**, 416–433.

See Also

RTdata

`rweibullmix` *Simulate from Mixtures of Weibull distributions*

Description

Simulate from a mixture of univariate Weibull distributions.

Usage

```
rweibullmix(n, lambda = 1, shape = 1, scale = 1)
```

Arguments

`n`	Number of cases to simulate.
`lambda`	Vector of mixture probabilities, with length equal to m, the desired number of components (subpopulations). This is assumed to sum to 1.
`shape`	Vector of component shapes.
`scale`	Vector of component scales.

Value

`rexpmix` returns an n-vector sampled from an m-component mixture of univariate Weibull distributions.

See Also

`rnormmix` and `rmvnormmix` for Gaussian mixtures, `rexpmix` for mixture of exponentials.

Examples

```
n = 500 # sample size
m = 2 # nb components
lambda=c(0.4, 0.6)
shape <- c(0.5,5); scale <- c(1,20) # model parameters
set.seed(321)
x <- rweibullmix(n, lambda, shape, scale) # iid ~ weibull mixture

## histogram of the simulated data.
hist(x, col=8)
```

`segregmixEM` *ECM Algorithm for Mixtures of Regressions with Changepoints*

Description

Returns ECM algorithm output for mixtures of multiple regressions with changepoints and arbitrarily many components.

Usage

```
segregmixEM(y, x, lambda = NULL, beta = NULL, sigma = NULL,
            k = 2, seg.Z, psi, psi.locs = NULL, delta = NULL,
            epsilon = 1e-08, maxit = 10000, verb = FALSE,
            max.restarts = 15)
```

Arguments

`y`	An n-vector of response values.
`x`	An nxp matrix of predictors. Note that this model assumes the presence of an intercept.
`lambda`	Initial value of mixing proportions. Entries should sum to 1. This determines number of components. If NULL, then `lambda` is random from uniform Dirichlet and the number of components is determined by `beta`.
`beta`	Initial value of `beta` parameters. This is a list of length `k` such that each element must contain a vector having length consistent with the defined changepoint structure. See `seg.Z`, `psi`, and `psi.loc` below. If NULL, then `beta` has standard normal entries according to a binning method done on the data. If both `lambda` and `beta` are NULL, then number of components is determined by `sigma`.
`sigma`	A vector of standard deviations. If NULL, then 1/`sigma`^2 has random standard exponential entries according to a binning method done on the data. If `lambda`, `beta`, and `sigma` are NULL, then number of components is determined by `k`.
`k`	Number of components. Ignored unless all of `lambda`, `beta`, and `sigma` are NULL.

seg.Z	A list of length k whose elements are right-hand side formulas, which are additive linear models of the predictors that have changepoints in their respective components. See below for more details.
psi	A kxp matrix specifying the number of changepoints for each predictor in each component. See below for more details.
psi.locs	A list of length k that has initial estimates for the changepoint locations. Each element of the list must have length equal to the number of chanegpoints specified in the corresponding row of the psi matrix. For components with no changepoints, simply set that element equal to NULL. See below for more details.
delta	An optional list of values quantifying the amount of separation at each changepoint if assuming discontinuities at the changepoints. This has the same dimensions as psi.locs.
epsilon	The convergence criterion.
maxit	The maximum number of iterations.
verb	If TRUE, then various updates are printed during each iteration of the algorithm.
max.restarts	The number of times to try restarting the ECM algorithm if estimation problems occur - such as choice of poor initial values or a poorly chosen changepoint structure.

Details

seg.Z is defined as a list of right-hand side linear model formulas that are used to identify which predictors have changepoints in each component. For example, suppose you have a dataframe with three predictors: V1, V2, V3. Suppose now that you wish to model a 3-component mixture of regressions with changepoints structure such that the first component has changepoints in V1 and V2, the second component has changepoints in V3, and the third component has no changepoints. Then you would define seg.Z = list(~V1+V2,~V3,NULL). Note that you MUST place the variables in order with respect to how they appear in the predictor matrix x.

psi is a kxp matrix specifying the number of changepoints for each predictor in each component. For the example given above, suppose there are three changepoints for V1, two changepoints for V2, and four changepoints for V3. Then you would define psi = rbind(c(3,2,0),c(0,0,4),c(0,0,0)).

psi.locs is a list of length k whose elements give the initial locations of the changepoints for each component. Each element of the list must have length equal to the total number of changepoints for that component's regression equation. For the example given above, in component 1, assume that the three changepoints for V1 are at 3, 7, and 10 and the two changepoints for V1 are at 5, 20, and 30. In component 2, assume that the four changepoints for V3 are at 2, 4, 6, and 8. Then you would define psi.locs = list(c(3,7,10,5,20,30),c(2,4,6,8),NULL). Note that the order of the changepoints is determined by first sorting the predictors by how they appear in the formulas in seg.Z and then sorting in increasing order within each predictor.

Value

segregmixEM returns a list of class segregmixEM with items:

x	The set of predictors.
y	The response values.

lambda	The final mixing proportions.
beta	The final regression coefficients.
sigma	The final standard deviations.
seg.Z	The list of right-hand side formulas as defined by the user.
psi.locs	A list of length k with the final estimates for the changepoint locations.
delta	A list of the delta values that were optionally specified by the user.
loglik	The final log-likelihood.
posterior	An nxk matrix of posterior probabilities for observations.
all.loglik	A vector of each iteration's log-likelihood.
restarts	The number of times the algorithm restarted due to unacceptable choice of initial values.
ft	A character vector giving the name of the function.

Note

As of version 0.4.6, this more general function has replaced the now defunct `regmixEM.chgpt` and associated internal functions.

References

Young, D. S. (2014) Mixtures of Regressions with Changepoints, *Statistics and Computing*, **24(2)**, 265–281.

See Also

regmixEM

Examples

```
## Not run:
## Simulated example.

set.seed(100)
x <- 1:20
y1 <- 3 + x + rnorm(20)
y2 <- 3 - x - 5*(x - 15)*(x > 15) + rnorm(20)
y <- c(y1, y2)
x <- c(x, x)

set.seed(100)
be <- list(c(3, -1, -5), c(3, 1))
s <- c(1, 1)
psi.locs <- list(comp.1 = list(x = 15), comp.2 = NULL)
out <- segregmixEM(y, cbind(1,x), verb = TRUE, k = 2,
                   beta = be, sigma = s, lambda = c(1, 1)/2,
                   seg.Z = list(~x, NULL), psi = rbind(1, 0),
                   psi.locs = psi.locs, epsilon = 0.9)
```

```
z <- seq(0, 21, len = 40)
plot(x, y, col = apply(out$post, 1, which.max) + 1, pch = 19,
   cex.lab = 1.4, cex = 1.4)
b <- out$beta
d <- out$psi.locs
lines(z, b[[1]][1] + b[[1]][2] * z + b[[1]][3] *
      (z - d[[1]][[1]]) * (z > d[[1]][[1]]) , col = 2, lwd = 2)
lines(z, b[[2]][1] + b[[2]][2] * z, col = 3, lwd = 2)
abline(v = out$psi.locs[[1]][1], col = 2, lty = 2)

## End(Not run)

## Not run:
## Example using the NOdata.

data(NOdata)
attach(NOdata)

set.seed(100)
be <- list(c(1.30, -0.13, 0.08), c(0.56, 0.09))
s <- c(0.02, 0.04)
psi.locs <- list(comp.1 = list(NO = 1.57), comp.2 = NULL)
out <- segregmixEM(Equivalence, cbind(NO), verb = TRUE, k = 2,
                   beta = be, sigma = s, lambda = c(1, 1)/2,
                   seg.Z = list(~NO, NULL), psi = rbind(1, 0),
                   psi.locs = psi.locs, epsilon = 0.1)

z <- seq(0, 5, len = 1000)
plot(NOdata, col = apply(out$post, 1, which.max) + 1, pch = 19,
   cex.lab = 1.4, cex = 1.4, ylab = "Equivalence Ratio")
b <- out$beta
d <- out$psi.locs
lines(z, b[[1]][1] + b[[1]][2] * z + b[[1]][3] *
      (z - d[[1]][[1]]) * (z > d[[1]][[1]]) , col = 2, lwd = 2)
lines(z, b[[2]][1] + b[[2]][2] * z, col = 3, lwd = 2)
abline(v = out$psi.locs[[1]][1], col = 2, lty = 2)

detach(NOdata)

## End(Not run)
```

---

spEM *Semiparametric EM-like Algorithm for Mixtures of Independent Repeated Measurements*

Description

Returns semiparametric EM algorithm output (Benaglia et al, 2009) for mixtures of multivariate (repeated measures) data where the coordinates of a row (case) in the data matrix are assumed to be independent, conditional on the mixture component (subpopulation) from which they are drawn.

For now, this algorithm only implements model (4.7) in Benaglia et al, in which each component and block has exactly the same (nonparametric) shape and they differ only by location and scale.

Usage

```
spEM(x, mu0, blockid = 1:ncol(x),
     bw = bw.nrd0(as.vector(as.matrix(x))), constbw = TRUE,
     h = bw, eps = 1e-8,
     maxiter = 500, stochastic = FALSE, verb = TRUE)
```

Arguments

x	An $n \times r$ matrix of data. Each of the n rows is a case, and each case has r repeated measurements. These measurements are assumed to be conditionally independent, conditional on the mixture component (subpopulation) from which the case is drawn.
mu0	Either an $m \times r$ matrix specifying the initial centers for the kmeans function, or an integer m specifying the number of initial centers, which are then choosen randomly in kmeans
blockid	A vector of length r identifying coordinates (columns of x) that are assumed to be identically distributed (i.e., in the same block). For instance, the default has all distinct elements, indicating that no two coordinates are assumed identically distributed and thus a separate set of m density estimates is produced for each column of x. On the other hand, if blockid=rep(1,ncol(x)), then the coordinates in each row are assumed conditionally i.i.d.
bw	Bandwidth for density estimation, equal to the standard deviation of the kernel density. By default, a simplistic application of the default bw.nrd0 bandwidth used by density to the entire dataset.
constbw	Logical: If TRUE, use the same bandwidth for each iteration and for each component and block. If FALSE, use a separate bandwidth for each component and block, and update this bandwidth at each iteration of the algorithm using a suitably modified bw.nrd0 method as described in Benaglia et al (2011).
h	Alternative way to specify the bandwidth, to provide backward compatibility.
eps	Tolerance limit for declaring algorithm convergence. Convergence is declared whenever the maximum change in any coordinate of the lambda vector (of mixing proportion estimates) does not exceed eps.
maxiter	The maximum number of iterations allowed, for both stochastic and non-stochastic versions; for non-stochastic algorithms (stochastic = FALSE), convergence may be declared before maxiter iterations (see eps above).
stochastic	Flag, if FALSE (the default), runs the non-stochastic version of the npEM algorithm, as in Benaglia et al (2009). Set to TRUE to run a stochastic version which simulates the posteriors at each iteration, and runs for maxiter iterations.
verb	If TRUE, print updates for every iteration of the algorithm as it runs

Value

spEM returns a list of class spEM with the following items:

data	The raw data (an $n \times r$ matrix).
posteriors	An $n \times m$ matrix of posterior probabilities for observation. If stochastic = TRUE, this matrix is computed from an average over the maxiter iterations.
bandwidth	If constbw==TRUE, same as the bw input argument; otherwise, value of bw matrix at final iteration (since for now this algorithm only implements model (4.7) in Benaglia et al, the bandwidth matrix is reduced to a single bandwith scalar). This information is needed by any method that produces density estimates from the output.
blockid	Same as the blockid input argument, but recoded to have positive integer values. Also needed by any method that produces density estimates from the output.
lambda	The sequence of mixing proportions over iterations.
lambdahat	The final mixing proportions if stochastic = FALSE, or the average mixing proportions if stochastic = TRUE.
mu	The sequence of location parameters over iterations.
muhat	The final location parameters if stochastic = FALSE, or the average location parameters if stochastic = TRUE.
sigma	The sequence of scale parameters over iterations.
sigmahat	The final scale parameters if stochastic = FALSE, or the average scale parameters if stochastic = TRUE.
loglik	The sequence of log-likelihoods over iterations.

References

- Benaglia, T., Chauveau, D., and Hunter, D. R., An EM-like algorithm for semi- and nonparametric estimation in multivariate mixtures, Journal of Computational and Graphical Statistics, 18, 505-526, 2009.
- Benaglia, T., Chauveau, D. and Hunter, D.R. Bandwidth Selection in an EM-like algorithm for nonparametric multivariate mixtures. Nonparametric Statistics and Mixture Models: A Festschrift in Honor of Thomas P. Hettmansperger. World Scientific Publishing Co., pages 15-27, 2011.
- Bordes, L., Chauveau, D., and Vandekerkhove, P., An EM algorithm for a semiparametric mixture model, Computational Statistics and Data Analysis, 51: 5429-5443, 2007.

See Also

plot.spEM, normmixrm.sim, spEMsymloc, npEM, plotseq.npEM

Examples

```
## Not run:
## simulate a 2-component gaussian mixture with 3 iid repeated measures
set.seed(100)
mu <- matrix(c(0, 15), 2, 3)
sigma <- matrix(c(1, 5), 2, 3)
x <- rmvnormmix(300, lambda = c(.4,.6), mu = mu, sigma = sigma)

## apply spEM with or without an iterative bandwidth selection
d <- spEM(x, mu0 = 2, blockid = rep(1,3), constbw = FALSE)
d2 <- spEM(x, mu0 = 2, blockid = rep(1,3), constbw = TRUE)
plot(d, xlim=c(-10, 40), ylim = c(0, .16), xlab = "", breaks = 30,
     cex.lab=1.5, cex.axis=1.5, addlegend=FALSE)
plot(d2, newplot=FALSE, addlegend=FALSE, lty=2)
## End(Not run)
```

`spEMsymloc` *Semiparametric EM-like Algorithm for univariate symmetric location mixture*

Description

Returns semiparametric EM algorithm output (Bordes et al, 2007, and Benaglia et al, 2009) for location mixtures of univariate data and symmetric component density.

Usage

```
spEMsymloc(x, mu0, bw = bw.nrd0(x), h=bw, eps = 1e-8, maxiter = 100,
           stochastic = FALSE, verbose = FALSE)
```

Arguments

`x`	A vector of length n consisting of the data.
`mu0`	Either a vector specifying the initial centers for the kmeans function, and from which the number of component is obtained, or an integer m specifying the number of initial centers, which are then choosen randomly in kmeans.
`bw`	Bandwidth for density estimation, equal to the standard deviation of the kernel density.
`h`	Alternative way to specify the bandwidth, to provide backward compatibility.
`eps`	Tolerance limit for declaring algorithm convergence. Convergence is declared before `maxiter` iterations whenever the maximum change in any coordinate of the `lambda` (mixing proportion estimates) and `mu` (means) vector does not exceed `eps`.
`maxiter`	The maximum number of iterations allowed, for both stochastic and non-stochastic versions; for non-stochastic algorithms (`stochastic = FALSE`), convergence may be declared before `maxiter` iterations (see `eps` above).

stochastic	Flag, if FALSE (the default), runs the non-stochastic version of the algorithm, as in Benaglia et al (2009). Set to TRUE to run a stochastic version which simulates the posteriors at each iteration (as in Bordes et al, 2007), and runs for maxiter iterations.
verbose	If TRUE, print updates for every iteration of the algorithm as it runs

Value

spEMsymloc returns a list of class npEM with the following items:

data	The raw data (an $n \times r$ matrix).
posteriors	An $n \times m$ matrix of posterior probabilities for observations. If stochastic = TRUE, this matrix is computed from an average over the maxiter iterations.
bandwidth	Same as the bw input argument, returned because this information is needed by any method that produces density estimates from the output.
lambda	The sequence of mixing proportions over iterations.
lambdahat	The final estimate for mixing proportions if stochastic = FALSE, the average over the sequence if stochastic = TRUE.
mu	the sequence of component means over iterations.
muhat	the final estimate of component means if stochastic = FALSE, the average over the sequence if stochastic = TRUE.
symmetric	Flag indicating that the kernel density estimate is using a symmetry assumption.

References

- Benaglia, T., Chauveau, D., and Hunter, D. R., An EM-like algorithm for semi- and non-parametric estimation in multivariate mixtures, Journal of Computational and Graphical Statistics, 18, 505-526, 2009.
- Benaglia, T., Chauveau, D., Hunter, D. R., and Young, D. mixtools: An R package for analyzing finite mixture models. Journal of Statistical Software, 32(6):1-29, 2009.
- Bordes, L., Chauveau, D., and Vandekerkhove, P. (2007), An EM algorithm for a semiparametric mixture model, Computational Statistics and Data Analysis, 51: 5429-5443.

See Also

plot.npEM, rnormmix, npEM, spEMsymlocN01, plotseq.npEM

Examples

```
## Example from a normal location mixture
set.seed(100)
n <- 200
lambda <- c(1/3,2/3)
mu <- c(0, 4); sigma<-rep(1, 2)
x <- rnormmix(n, lambda, mu, sigma)
out.stoc <- spEMsymloc(x, mu0=c(-1, 2), stochastic=TRUE)
out.nonstoc <- spEMsymloc(x, mu0=c(-1, 2))
```

spEMsymlocN01 *semiparametric EM-like algorithm for univariate mixture in False Discovery Rate (FDR) estimation*

Description

Return semiparametric EM-like algorithm output for a 2-components mixture model with one component set to Normal(0,1), and the other component being a unspecified but symmetric density with a location parameter. This model is tailored to FDR estimation on probit transform (qnorm) of p-values arising from multiple testing.

Usage

```
spEMsymlocN01(x, mu0 = 2, bw = bw.nrd0(x), h=bw, eps = 1e-8,
              maxiter = 100, verbose = FALSE, plotf = FALSE)
```

Arguments

x	A vector of length n consisting of the data, probit transform of pvalues, preferably sorted.
mu0	Starting value of vector of component means. If not set then the initial value is randomly generated by a kmeans of the data in two bins. Since component 1 is theoretically normal(0,1), mu[1] must be 0 and mu[2] some negative value (see details).
bw	Bandwidth for weighted kernel density estimation.
h	Alternative way to specify the bandwidth, to provide backward compatibility.
eps	Tolerance limit for declaring algorithm convergence. Convergence is declared before maxiter iterations whenever the maximum change in any coordinate of the lambda (mixing proportion estimates) and mu (mean of the semiparametric component) vector does not exceed eps
maxiter	The maximum number of iterations allowed; convergence may be declared before maxiter iterations (see eps above).
verbose	If TRUE, print updates for every iteration of the algorithm as it runs.
plotf	If TRUE, plots successive updates of the nonparametric density estimate over iterations. Mostly for testing purpose.

Details

This algorithm is a specific version of semiparametric EM-like algorithm similar in spirit to spEMsymloc, but specialized for FDR estimation on probit transform (qnorm) of p-values in multiple testing framework. In this model, component 1 corresponds to the individuals under the null hypothesis, i.e. theoretically normal(0,1) distributed, whereas component 2 corresponds to individuals in the alternative hypothesis, with typically very small p-values and consequently negative values for probit(p) data. This model only assumes that these individuals come from an unspecified but symmetric density with a location parameter, as in Bordes and Vandekerkhove (2010) and Chauveau et al. (2014).

Value

spEMsymlocN01 returns a list of class spEMN01 with the following items:

data	The raw data (an $n \times r$ matrix).
posteriors	An $n \times 2$ matrix of posterior probabilities for observations. This can be used in, e.g., plotFDR to plot False Discovery Rate estimates.
bandwidth	Same as the bw input argument, returned because this information is needed by any method that produces density estimates from the output.
lambda	The sequence of mixing proportions over iterations.
lambdahat	The final estimate for mixing proportions.
mu	the sequence of second component mean over iterations.
muhat	the final estimate of second component mean.
symmetric	Flag indicating that the kernel density estimate is using a symmetry assumption.

Author(s)

Didier Chauveau

References

- Bordes, L. and Vandekerkhove, P. (2010). Semiparametric two-component mixture model with a known component: an asymptotically normal estimator. Mathematical Methods of Statistics, 19(1):22-41
- Chauveau, D., Saby, N., Orton, T. G., Lemercier B., Walter, C. and Arrouys, D. (2014) Large-scale simultaneous hypothesis testing in monitoring carbon content from french soil database: A semi-parametric mixture approach. Geoderma 219-220 (2014): 117-124.

See Also

spEMsymloc, normalmixEM, npEM, plot.spEMN01, plotFDR

Examples

```
## Probit transform of p-values
## from a Beta-Uniform mixture model
## comparion of parametric and semiparametric EM fit
## Note: in actual situations n=thousands
set.seed(50)
n=300 # nb of multiple tests
m=2 # 2 mixture components
a=c(1,0.1); b=c(1,1); lambda=c(0.6,0.4) # parameters
z=sample(1:m, n, rep=TRUE, prob = lambda)
p <- rbeta(n, shape1 = a[z], shape2 = b[z]) # p-values
o <- order(p)
cpd <- cbind(z,p)[o,] # sorted complete data, z=1 if H0, 2 if H1
p <- cpd[,2] # sorted p-values

y <- qnorm(p) # probit transform of the pvalues
```

```
# gaussian EM fit with component 1 constrained to N(0,1)
s1 <- normalmixEM(y, mu=c(0,-4),
mean.constr = c(0,NA), sd.constr = c(1,NA))
s2 <- spEMsymlocN01(y, mu0 = c(0,-3)) # spEM with N(0,1) fit
hist(y, freq = FALSE, col = 8, main = "histogram of probit(pvalues)")
plot(s2, add.plot = TRUE, lwd = 2)

# Exemples of plot capabilities
# Note: posteriors must be ordered by p for plot.FDR
# plotFDR(s1$post) # when true complete data not observed
# plotFDR(s1$post, s2$post) # comparing 2 strategies
plotFDR(s1$post, s2$post, lg1 = "normalmixEM", lg2 = "spEMsymlocN01",
complete.data = cpd) # with true FDR computed from z
```

---

spregmix *EM-like Algorithm for Semiparametric Mixtures of Regressions*

Description

Returns parameter estimates for finite mixtures of linear regressions with unspecified error structure. Based on Hunter and Young (2012).

Usage

```
spregmix(lmformula, bw = NULL, constbw = FALSE,
         bwmult = 0.9, z.hat = NULL, symm = TRUE, betamethod = "LS",
         m = ifelse(is.null(z.hat), 2, ncol(z.hat)),
         epsilon = 1e-04, maxit = 1000, verbose = FALSE,
         ...)
```

Arguments

lmformula	Formula for a linear model, in the same format used by lm. Additional parameters may be passed to lm via the ... argument.
bw	Initial bandwidth value. If NULL, this will be chosen automatically by the algorithm.
constbw	Logical: If TRUE, the bandwidth is held constant throughout the algorithm; if FALSE, it adapts at each iteration according to the rules given in Hunter and Young (2012).
bwmult	Whenever it is updated automatically, the bandwidth is equal to bwmult divided by the fifth root of n times the smaller of s and IQR/1.34, where s and IQR are estimates of the standard deviation and interquartile range of the residuals, as explained in Hunter and Young (2012). The value of 0.9 gives the rule of Silverman (1986) and the value of 1.06 gives the rule of Scott (1992). Larger values lead to greater smoothing, whereas smaller values lead to less smoothing.

z.hat	Initial nxm matrix of posterior probabilities. If NULL, this is initialized randomly. As long as a parametric estimation method like least squares is used to estimate beta in each M-step, the z.hat values are the only values necessary to begin the EM iterations.
symm	Logical: If TRUE, the error density is assumed symmetric about zero. If FALSE, it is not. WARNING: If FALSE, the intercept parameter is not uniquely identifiable if it is included in the linear model.
betamethod	Method of calculating beta coefficients in the M-step. Current possible values are "LS" for least-squares; "L1" for least absolute deviation; "NP" for fully nonparametric; and "transition" for a transition from least squares to fully nonparametric. If something other than these four possibilities is used, then "NP" is assumed. For details of these methods, see Hunter and Young (2012).
m	Number of components in the mixture.
epsilon	Convergence is declared if the largest change in any lambda or beta coordinate is smaller than epsilon.
maxit	The maximum number of iterations; if convergence is never declared based on comparison with epsilon, then the algorithm stops after maxit iterations.
verbose	Logical: If TRUE, then various updates are printed during each iteration of the algorithm.
...	Additional parameters passed to the model.frame and model.matrix functions, which are used to obtain the response and predictor of the regression.

Value

regmixEM returns a list of class npEM with items:

x	The set of predictors (which includes a column of 1's if addintercept = TRUE).
y	The response values.
lambda	The mixing proportions for every iteration in the form of a matrix with m columns and (#iterations) rows
beta	The final regression coefficients.
posterior	An nxm matrix of posterior probabilities for observations.
np.stdev	Nonparametric estimate of the standard deviation, as given in Hunter and Young (2012)
bandwidth	Final value of the bandwidth
density.x	Points at which the error density is estimated
density.y	Values of the error density at the points density.x
symmetric	Logical: Was the error density assumed symmetric?
loglik	A quantity similar to a log-likelihood, computed just like a standard loglikelihood would be, conditional on the component density functions being equal to the final density estimates.
ft	A character vector giving the name of the function.

References

Hunter, D. R. and Young, D. S. (2012) Semi-parametric Mixtures of Regressions, Journal of Nonparametric Statistics 24(1): 19-38.

Scott, D. W. (1992) *Multivariate Density Estimation*, John Wiley & Sons Inc., New York.

Silverman, B. W. (1986). *Density Estimation for Statistics and Data Analysis*, Chapman & Hall, London.

See Also

regmixEM, spEMsymloc, lm

Examples

```
data(tonedata)
## By default, the bandwidth will adapt and the error density is assumed symmetric
set.seed(100)
a=spregmix(tuned~stretchratio, bw=.2, data=tonedata, verb=TRUE)

## Look at the sp mixreg solution:
plot(tonedata)
abline(a=a$beta[1,1],b=a$beta[2,1], col=2)
abline(a=a$beta[1,2],b=a$beta[2,2], col=3)

## Look at the nonparametric KD-based estimate of the error density,
## constrained to be zero-symmetric:
plot(xx<-a$density.x, yy<-a$density.y, type="l")
## Compare to a normal density with mean 0 and NP-estimated stdev:
z <- seq(min(xx), max(xx), len=200)
lines(z, dnorm(z, sd=sqrt((a$np.stdev)^2+a$bandwidth^2)), col=2, lty=2)
# Add bandwidth^2 to variance estimate to get estimated var of KDE

## Now add the sp mixreg estimate without assuming symmetric errors:
b=spregmix(tuned~stretchratio, bw=.2, , symm=FALSE, data=tonedata, verb=TRUE)
lines(b$density.x, b$density.y, col=3)
```

---

spRMM_SEM *Stochastic EM algorithm for semiparametric scaled mixture of censored data*

Description

Stochastic EM algorithm for semiparametric scaled mixture for randomly right censored data.

Usage

```
spRMM_SEM(t, d = NULL, lambda = NULL, scaling = NULL,
        centers = 2, kernelft = triang_wkde,
        bw = rep(bw.nrd0(t),length(t)), averaged = TRUE,
        epsilon = 1e-08, maxit = 100, batchsize = 1, verb = FALSE)
```

Arguments

t	A vector of n real positive lifetime (possibly censored) durations. If d is not NULL then a vector of random censoring times c occurred, so that $x = min(x, c)$ and $d = I(x <= c)$.
d	The vector of censoring indication, where 1 means observed lifetime data, and 0 means censored lifetime data.
lambda	Initial value of mixing proportions. If NULL, then lambda is set to rep(1/k,k).
scaling	Initial value of scaling between components, set to 1 if NULL.
centers	initial centers for initial call to kmeans for initialization.
kernelft	.
bw	Bandwidth in the kernel hazard estimates.
averaged	averaged.
epsilon	Tolerance limit.
maxit	The number of iterations allowed.
batchsize	The batchsize (see reference below).
verb	If TRUE, print updates for every iteration of the algorithm as it runs

Value

spRMM_SEM returns a list of class "spRMM" with the following items:

t	The input data.
d	The input censoring indicator.
lambda	The estimates for the mixing proportions.
scaling	The estimates for the components scaling.
posterior	An $n \times k$ matrix of posterior probabilities for observation, after convergence of the algorithm.
loglik	The (pseudo) log-likelihood value at convergence of the algorithm.
all.loglik	The sequence of log-likelihood values over iterations.
all.lambda	The sequence of mixing proportions over iterations.
all.scaling	The sequence of scaling parameter over iterations.
meanpost	Posterior probabilities averaged over iterations.
survival	Kaplan-Meier last iteration estimate (a stepfun object).
hazard	Hazard rate last iteration estimate evaluated at final.t.
final.t	Last iteration unscaled sample (see reference).
s.hat	Kaplan-Meier average estimate.
t.hat	Ordered unscaled sample, for testing purpose.
avg.od	For testing purpose only.
hazard.hat	Hazard rate average estimate on t.hat.
batch.t	Batch sample (not ordered), see reference.
batch.d	Associated event indicators just rep(d,batchsize), for testing purpose.
sumNaNs	Internal control of numerical stability.
ft	A character vector giving the name of the function.

Author(s)

Didier Chauveau

References

- Bordes, L., and Chauveau, D. (2016), Stochastic EM algorithms for parametric and semiparametric mixture models for right-censored lifetime data, Computational Statistics, Volume 31, Issue 4, pages 1513-1538. http://link.springer.com/article/10.1007/s00180-016-0661-7

See Also

Related functions: plotspRMM, summary.spRMM.

Other models and algorithms for censored lifetime data (name convention is model_algorithm): expRMM_EM, weibullRMM_SEM.

Examples

```
## Not run:
n=500 # sample size
m=2 # nb components
lambda=c(0.4, 0.6) # parameters
meanlog=3; sdlog=0.5; scale=0.1
set.seed(12)
# simulate a scaled mixture of lognormals
x <- rlnormscalemix(n, lambda, meanlog, sdlog, scale)
cs=runif(n,20,max(x)+400) # Censoring (uniform) and incomplete data
t <- apply(cbind(x,cs),1,min)
d <- 1*(x <= cs)
tauxc <- 100*round( 1-mean(d),3)
cat(tauxc, "percents of data censored.\n")

c0 <- c(25, 180) # data-driven initial centers (visible modes)
sc0 <- 25/180    # and scaling
s <- spRMM_SEM(t, d, scaling = sc0, centers = c0, bw = 15, maxit = 100)

plotspRMM(s) # default
summary(s)   # S3 method for class "spRMM"

## End(Not run)
```

---

summary.mixEM *Summarizing EM mixture model fits*

Description

summary method for class mixEM.

Usage

```
## S3 method for class 'mixEM'
summary(object, digits=6, ...)
```

Arguments

object	an object of class mixEM such as a result of a call to normalmixEM
digits	Significant digits for printing values
...	further arguments passed to print method.

Details

summary.mixEM prints parameter estimates for each component of a fitted mixture model. The estimates printed vary with the type of model.

Value

The function summary.mixEM prints the final loglikelihood value at the solution as well as a matrix of values for each component that could include:

lambda	The estimated mixing weights
mu	The estimated mean parameters
sigma	The estimated standard deviations
theta	The estimated multinomial parameters
beta	The estimated regression parameters

See Also

normalmixEM, logisregmixEM, multmixEM, mvnormalmixEM, poisregmixEM, regmixEM, regmixEM.lambda, regmixEM.loc, regmixEM.mixed, regmixEM.chgpt, repnormmixEM, expRMM_EM, weibullRMM_SEM

Examples

```
data(faithful)
attach(faithful)
set.seed(100)
out <- normalmixEM(waiting, mu=c(50,80), sigma=c(5,5), lambda=c(.5,.5))
summary(out)
```

summary.mvnpEM *Summarizing Fits for Nonparametric Mixture Models with Conditionally Independent Multivariate Component Densities*

Description

summary method for class mvnpEM.

Usage

```
## S3 method for class 'mvnpEM'
summary(object, ...)
## S3 method for class 'summary.mvnpEM'
print(x, digits=3, ...)
```

Arguments

object,x	an object of class mvnpEM such as a result of a call to mvnpEM
digits	Significant digits for printing values
...	further arguments passed to or from other methods.

Details

summary.mvnpEM prints means and variances of each block for each component. These quantities might not be part of the model, but they are estimated nonparametrically based on the posterior probabilities and the data.

Value

The function summary.mvnpEM returns a list of type summary.mvnpEM with the following components:

n	The number of observations
m	The number of mixture components
B	The number of blocks
blockid	The block ID (from 1 through B) for each of the coordinates of the multivariate observations. The blockid component is of length r, the dimension of each observation.
means	A $B \times m$ matrix giving the estimated mean of each block in each component.
variances	Same as means but giving the estimated variances instead.

References

Benaglia, T., Chauveau, D., and Hunter, D. R. (2009), An EM-like algorithm for semi- and non-parametric estimation in multivariate mixtures, *Journal of Computational and Graphical Statistics*, **18(2)**, 505–526.

Chauveau, D., and Hoang, V. T. L. (2015), Nonparametric mixture models with conditionally independent multivariate component densities, Preprint under revision. https://hal.archives-ouvertes.fr/hal-01094837

See Also

mvnpEM, plot.mvnpEM

Examples

```
# Example as in Chauveau and Hoang (2015) with 6 coordinates
## Not run:
m=2; r=6; blockid <-c(1,1,2,2,3,3) # 3 bivariate blocks
# generate some data x ...
a <- mvnpEM(x, mu0=2, blockid, samebw=F) # adaptive bandwidth
plot(a) # this S3 method produces 6 plots of univariate marginals
summary(a)
## End(Not run)
```

---

summary.npEM *Summarizing non- and semi-parametric multivariate mixture model fits*

Description

summary method for class npEM.

Usage

```
## S3 method for class 'npEM'
summary(object, ...)
## S3 method for class 'summary.npEM'
print(x, digits=3, ...)
```

Arguments

object,x	an object of class npEM such as a result of a call to npEM
digits	Significant digits for printing values
...	further arguments passed to or from other methods.

Details

summary.npEM prints means and variances of each block for each component. These quantities might not be part of the model, but they are estimated nonparametrically based on the posterior probabilities and the data.

Value

The function summary.npEM returns a list of type summary.npEM with the following components:

n	The number of observations
m	The number of mixture components
B	The number of blocks
blockid	The block ID (from 1 through B) for each of the coordinates of the multivariate observations. The blockid component is of length r, the dimension of each observation.
means	A $B \times m$ matrix giving the estimated mean of each block in each component.
variances	Same as means but giving the estimated variances instead.

References

Benaglia, T., Chauveau, D., and Hunter, D. R. (2009), An EM-like algorithm for semi- and non-parametric estimation in multivariate mixtures, *Journal of Computational and Graphical Statistics*, **18(2)**, 505–526.

See Also

npEM, plot.npEM

Examples

```
data(Waterdata)
set.seed(100)
## Not run:
a <- npEM(Waterdata[,3:10], 3, bw=4) # Assume indep but not iid
summary(a)

b <- npEM(Waterdata[,3:10], 3, bw=4, blockid=rep(1,8)) # Now assume iid
summary(b)

## End(Not run)
```

summary.spRMM *Summarizing fits from Stochastic EM algorithm for semiparametric scaled mixture of censored data*

Description

summary method for class spRMM.

Usage

```
## S3 method for class 'spRMM'
summary(object, digits = 6, ...)
```

Arguments

object	an object of class spRMM such as a result of a call to spRMM_SEM
digits	Significant digits for printing values
...	Additional parameters passed to print.

Details

summary.spRMM prints scalar parameter estimates for a fitted mixture model: each component weight and the scaling factor, see reference below. The functional (nonparametric) estimates of survival and hazard rate funcions can be obtained using plotspRMM.

Value

The function summary.spRMM prints the final loglikelihood value at the solution as well as The estimated mixing weights and the scaling parameter.

Author(s)

Didier Chauveau

References

- Bordes, L., and Chauveau, D. (2016), Stochastic EM algorithms for parametric and semiparametric mixture models for right-censored lifetime data, Computational Statistics, Volume 31, Issue 4, pages 1513-1538. http://link.springer.com/article/10.1007/s00180-016-0661-7

See Also

Function for plotting functional (nonparametric) estimates: plotspRMM.

Other models and algorithms for censored lifetime data (name convention is model_algorithm): expRMM_EM, weibullRMM_SEM.

Examples

```
# See example(spRMM_SEM)
```

---

tauequivnormalmixEM *Special EM Algorithm for three-component tau equivalence model*

Description

Return ECM algorithm output for a specific case of a three-component tau equivalence model

Usage

```
tauequivnormalmixEM (x, lambda = NULL, mu = NULL, sigma = NULL, k = 3,
          mean.constr = NULL, sd.constr = NULL, gparam = NULL,
          epsilon = 1e-08, maxit = 10000, maxrestarts=20,
          verb = FALSE, fast=FALSE, ECM = TRUE,
          arbmean = TRUE, arbvar = TRUE)
```

Arguments

x	A vector of length n consisting of the data, passed directly to normalmixMMlc.
lambda	Initial value of mixing proportions, passed directly to normalmixMMlc. Automatically repeated as necessary to produce a vector of length k, then normalized to sum to 1. If NULL, then lambda is random from a uniform Dirichlet distribution (i.e., its entries are uniform random and then it is normalized to sum to 1).
mu	Starting value of vector of component means for algorithm, passed directly to normalmixMMlc. If non-NULL and a vector, k is set to length(mu). If NULL, then the initial value is randomly generated from a normal distribution with center(s) determined by binning the data.
sigma	Starting value of vector of component standard deviations for algorithm, passed directly to normalmixMMlc. Obsolete for linear constraint on the inverse variances, use gparam instead to specify a starting value. Note: This needs more precision
k	Number of components, passed directly to normalmixMMlc. Initial value ignored unless mu and sigma are both NULL. Also, initial value is ignored if mean.constr is NULL, since in that case we presume k=3.
mean.constr	If non-NULL, this parameter is passed directly to normalmixMMlc and both mean.lincstr and var.lincstr are passed as NULL to normalmixMMlc. If NULL, then it is assumed that k=3 and the means must take the form α, $\alpha - \delta$, and $\alpha + \delta$ for unknown parameters α and δ. Furthermore, the reciprocal variances are assumed to be $\gamma_1 + \gamma_2$, γ_1, and γ_1 for unknown positive parameters γ_1 and γ_2. These constraints are passed to the normalmixMMlc function using the mean.lincstr and var.lincstr arguments as shown in the examples for the normalmixMMlc help file.

sd.constr	Deprecated.
gparam	This argument is passed directly to normalmixMMlc.
epsilon	The convergence criterion. Convergence is declared when the change in the observed data log-likelihood increases by less than epsilon.
maxit	The maximum number of iterations.
maxrestarts	The maximum number of restarts allowed in case of a problem with the particular starting values chosen due to one of the variance estimates getting too small (each restart uses randomly chosen starting values). It is well-known that when each component of a normal mixture may have its own mean and variance, the likelihood has no maximizer; in such cases, we hope to find a "nice" local maximum with this algorithm instead, but occasionally the algorithm finds a "not nice" solution and one of the variances goes to zero, driving the likelihood to infinity.
verb	If TRUE, then various updates are printed during each iteration of the algorithm.
fast	If TRUE and k==2 and arbmean==TRUE, then use normalmixEM2comp, which is a much faster version of the EM algorithm for this case. This version is less protected against certain kinds of underflow that can cause numerical problems and it does not permit any restarts. If k>2, fast is ignored.
ECM	logical: Should this algorithm be an ECM algorithm in the sense of Meng and Rubin (1993)? If FALSE, the algorithm is a true EM algorithm; if TRUE, then every half-iteration alternately updates the means conditional on the variances or the variances conditional on the means, with an extra E-step in between these updates. For tauequivnormalmixEM, it must be TRUE.
arbmean	Deprecated.
arbvar	Deprecated.

Details

The tauequivnormalmixEM function is merely a wrapper for the normalmixMMlc function. # This is the standard EM algorithm for normal mixtures that maximizes # the conditional expected complete-data # log-likelihood at each M-step of the algorithm. # If desired, the # EM algorithm may be replaced by an ECM algorithm (see ECM argument) # that alternates between maximizing with respect to the mu # and lambda while holding sigma fixed, and maximizing with # respect to sigma and lambda while holding mu # fixed. In the case where arbmean is FALSE # and arbvar is TRUE, there is no closed-form EM algorithm, # so the ECM option is forced in this case.

Value

normalmixEM returns a list of class mixEM with items:

x	The raw data.
lambda	The final mixing proportions.
mu	The final mean parameters.
sigma	The final standard deviation(s)
scale	Scale factor for the component standard deviations, if applicable.

loglik	The final log-likelihood.
posterior	An nxk matrix of posterior probabilities for observations.
all.loglik	A vector of each iteration's log-likelihood. This vector includes both the initial and the final values; thus, the number of iterations is one less than its length.
restarts	The number of times the algorithm restarted due to unacceptable choice of initial values.
ft	A character vector giving the name of the function.

References

- Thomas, H., Lohaus, A., and Domsch, H. (2011) Stable Unstable Reliability Theory, *British Journal of Mathematical and Statistical Psychology* 65(2): 201-221.
- Meng, X.-L. and Rubin, D. B. (1993) Maximum Likelihood Estimation Via the ECM Algorithm: A General Framework, *Biometrika* 80(2): 267-278.

See Also

normalmixMMlc, normalmixEM, mvnormalmixEM, normalmixEM2comp

Examples

```
## Analyzing synthetic data as in the tau equivalent model
## From Thomas et al (2011), see also Chauveau and Hunter (2013)
## a 3-component mixture of normals with linear constraints.
lbd <- c(0.6,0.3,0.1); m <- length(lbd)
sigma <- sig0 <- sqrt(c(1,9,9))
# means constaints mu = M beta
M <- matrix(c(1,1,1,0,1,-1), 3, 2)
beta <- c(1,5) # unknown constained mean
mu0 <- mu <- as.vector(M %*% beta)
# linear constraint on the inverse variances pi = A.g
A <- matrix(c(1,1,1,0,1,0), m, 2, byrow=TRUE)
iv0 <- 1/(sig0^2)
g0 <- c(iv0[2],iv0[1] - iv0[2]) # gamma^0 init

# simulation and EM fits
set.seed(40); n=100; x <- rnormmix(n,lbd,mu,sigma)
s <- normalmixEM(x,mu=mu0,sigma=sig0,maxit=2000) # plain EM
# EM with var and mean linear constraints
sc <- normalmixMMlc(x, lambda=lbd, mu=mu0, sigma=sig0,
mean.lincstr=M, var.lincstr=A, gparam=g0)
# Using tauequivnormalmixEM function to call normalmixMMlc
tau <- tauequivnormalmixEM (x, lambda=lbd, mu=mu0, gparam=g0)
# plot and compare both estimates
dnormmixt <- function(t, lam, mu, sig){
m <- length(lam); f <- 0
for (j in 1:m) f <- f + lam[j]*dnorm(t,mean=mu[j],sd=sig[j])
f}
t <- seq(min(x)-2, max(x)+2, len=200)
hist(x, freq=FALSE, col="lightgrey",
```

```
ylim=c(0,0.3), ylab="density",main="")
lines(t, dnormmixt(t, lbd, mu, sigma), col="darkgrey", lwd=2) # true
lines(t, dnormmixt(t, s$lambda, s$mu, s$sigma), lty=2)
lines(t, dnormmixt(t, sc$lambda, sc$mu, sc$sigma), col=1, lty=3)
lines(t, dnormmixt(t, tau$lambda, tau$mu, tau$sigma), col=2, lty=4)
legend("topleft", c("true","plain EM","constr EM", "Tau Equiv"),
col=c("darkgrey",1,1,2), lty=c(1,2,3,4), lwd=c(2,1,1,1))
```

`test.equality` *Performs Chi-Square Tests for Scale and Location Mixtures*

Description

Performs a likelihood ratio test of a location (or scale) normal or regression mixture versus the more general model. For a normal mixture, the alternative hypothesis is that each component has its own mean and variance, whereas the null is that all means (in the case of a scale mixture) or all variances (in the case of a location mixture) are equal. This test is asymptotically chi-square with degrees of freedom equal to k-1, where k is the number of components.

Usage

```
test.equality(y, x = NULL, arbmean = TRUE, arbvar = FALSE,
              mu = NULL, sigma = NULL, beta = NULL,
              lambda = NULL, ...)
```

Arguments

`y`	The responses for `regmixEM` or the data for `normalmixEM`.
`x`	The predictors for `regmixEM`.
`arbmean`	If FALSE, then a scale mixture analysis is performed for `normalmixEM` or `regmixEM`.
`arbvar`	If FALSE, then a location mixture analysis is performed for `normalmixEM` or `regmixEM`.
`mu`	An optional vector for starting values (under the null hypothesis) for `mu` in `normalmixEM`.
`sigma`	An optional vector for starting values (under the null hypothesis) for `sigma` in `normalmixEM` or `regmixEM`.
`beta`	An optional matrix for starting values (under the null hypothesis) for `beta` in `regmixEM`.
`lambda`	An otional vector for starting values (under the null hypothesis) for `lambda` in `normalmixEM` or `regmixEM`.
`...`	Additional arguments passed to the various EM algorithms for the mixture of interest.

Value

`test.equality` returns a list with the following items:

`chi.sq`	The chi-squared test statistic.
`df`	The degrees of freedom for the chi-squared test statistic.
`p.value`	The p-value corresponding to this likelihood ratio test.

See Also

`test.equality.mixed`

Examples

```
## Should a location mixture be used for the Old Faithful data?

data(faithful)
attach(faithful)
set.seed(100)
test.equality(y = waiting, arbmean = FALSE, arbvar = TRUE)
```

---

test.equality.mixed *Performs Chi-Square Test for Mixed Effects Mixtures*

Description

Performs a likelihood ratio test of either common variance terms between the response trajectories in a mixture of random (or mixed) effects regressions or for common variance-covariance matrices for the random effects mixture distribution.

Usage

```
test.equality.mixed(y, x, w=NULL, arb.R = TRUE,
                    arb.sigma = FALSE, lambda = NULL,
                    mu = NULL, sigma = NULL, R = NULL,
                    alpha = NULL, ...)
```

Arguments

`y`	The responses for `regmixEM.mixed`.
`x`	The predictors for the random effects in `regmixEM.mixed`.
`w`	The predictors for the (optional) fixed effects in `regmixEM.mixed`.
`arb.R`	If FALSE, then a test for different variance-covariance matrices for the random effects mixture is performed.
`arb.sigma`	If FALSE, then a test for different variance terms between the response trajectories is performed.

lambda	A vector of mixing proportions (under the null hypothesis) with same purpose as outlined in regmixEM.mixed.
mu	A matrix of the means (under the null hypothesis) with same purpose as outlined in regmixEM.mixed.
sigma	A vector of standard deviations (under the null hypothesis) with same purpose as outlined in regmixEM.mixed.
R	A list of covariance matrices (under the null hypothesis) with same purpose as outlined in regmixEM.mixed.
alpha	An optional vector of fixed effects regression coefficients (under the null hypothesis) with same purpose as outlined in regmixEM.mixed.
...	Additional arguments passed to regmixEM.mixed.

Value

test.equality.mixed returns a list with the following items:

chi.sq	The chi-squared test statistic.
df	The degrees of freedom for the chi-squared test statistic.
p.value	The p-value corresponding to this likelihood ratio test.

See Also

test.equality

Examples

```
##Test of equal variances in the simulated data set.

data(RanEffdata)
set.seed(100)
x<-lapply(1:length(RanEffdata), function(i)
          matrix(RanEffdata[[i]][, 2:3], ncol = 2))
x<-x[1:15]
y<-lapply(1:length(RanEffdata), function(i)
          matrix(RanEffdata[[i]][, 1], ncol = 1))
y<-y[1:15]

out<-test.equality.mixed(y, x, arb.R = TRUE, arb.sigma = FALSE,
                         epsilon = 1e-1,  verb = TRUE,
                         maxit = 50,
                         addintercept.random = FALSE)
out
```

tonedata *Tone perception data*

Description

The tone perception data stem from an experiment of Cohen (1980) and have been analyzed in de Veaux (1989) and Viele and Tong (2002). The dataset and this documentation file were copied from the fpc package by Christian Hennig. A pure fundamental tone was played to a trained musician. Electronically generated overtones were added, determined by a stretching ratio of `stretchratio`. `stretchratio=2.0` corresponds to the harmonic pattern usually heard in traditional definite pitched instruments. The musician was asked to tune an adjustable tone to the octave above the fundamental tone. `tuned` gives the ratio of the adjusted tone to the fundamental, i.e. `tuned=2.0` would be the correct tuning for all `stretchratio`-values. The data analyzed here belong to 150 trials with the same musician. In the original study, there were four further musicians.

Usage

```
data(tonedata)
```

Format

A data frame with 2 variables, `stretchratio` and `tuned`, and 150 cases.

Author(s)

Christian Hennig

Source

Original source: Cohen, E. A. (1980), *Inharmonic tone perception*. Unpublished Ph.D. dissertation, Stanford University

R source: Hennig, Christian (2010), fpc: Flexible procedures for clustering, R package version 2.0-2. https://cran.r-project.org/package=fpc

References

de Veaux, R. D. (1989), Mixtures of Linear Regressions, *Computational Statistics and Data Analysis* 8, 227-245.

Viele, K. and Tong, B. (2002), Modeling with Mixtures of Linear Regressions, *Statistics and Computing* 12, 315-330.

`Waterdata` *Water-Level Task Data Set*

Description

This data set arises from the water-level task proposed by the Swiss psychologist Jean Piaget to assess children's understanding of the physical world. This involves presenting a child with a rectangular vessel with a cap, affixed to a wall, that can be tilted (like the minute hand of a clock) to point in any direction. A separate disk with a water line indicated on it, which can similarly be spun so that the water line may assume any desired angle with the horizontal, is positioned so that by spinning this disk, the child subject may make the hypothetical surface of water inside the vessel assume any desired orientation. For each of eight different orientations of the vessel, corresponding to the clock angles at 1:00, 2:00, 4:00, 5:00, 7:00, 8:00, 10:00, and 11:00, the child subject is asked to position the water level as it would appear in reality if water were in the vessel. The measurement is the acute angle with the horizontal, in degrees, assumed by the water line after it is positioned by the child. A sign is attached to the measurement to indicate whether the line slopes up (positive) or down (negative) from left to right. Thus, each child has 8 repeated measurements, one for each vessel angle, and the range of possible values are from -90 to 90.

The setup of the experiment, along with a photograph of the testing apparatus, is given by Thomas and Jamison (1975). A more detailed analysis using a subset of 405 of the original 579 subjects is given by Thomas and Lohaus (1993); further analyses using the functions in `mixtools` are given by Benaglia et al (2008) and Levine et al (2011), among others.

There are two versions of the dataset included in `mixtools`. The full dataset, called `WaterdataFull`, has 579 individuals. The dataset called `Waterdata` is a subset of 405 individuals, comprising all children aged 11 years or more and omitting any individuals with any observations equal to 100, which in this context indicates a missing value (since all of the degree measurements should be in the range from -90 to +90, 100 is not a possible value).

Usage

```
data(Waterdata)
```

Format

These data frames consist of 405 or 579 rows, one row for each child. There are ten columns: The age (in years) and sex (where 1=male and 0=female) are given for each individual along with the degree of deviation from the horizontal for 8 specified clock-hour orientations (11, 4, 2, 7, 10, 5, 1, and 8 o'clock, in order).

Source

Benaglia, T., Chauveau, D., and Hunter, D.R. (2009), An EM-Like Algorithm for Semi- and Non-Parametric Estimation in Multivariate Mixtures, *Journal of Computational and Graphical Statistics*, 18: 505-526.

Levine, M., Hunter, D.R., and Chauveau, D. (2011), Maximum Smoothed Likelihood for Multivariate Mixtures, *Biometrika*, 98(2): 403-416.

Thomas, H. and Jamison, W. (1975), On the Acquisition of Understanding that Still Water is Horizontal, *Merrill-Palmer Quarterly of Behavior and Development*, 21(1): 31-44.

Thomas, H. and Lohaus, A. (1993), *Modeling Growth and Individual Differences in Spatial Tasks*, University of Chicago Press, Chicago, available on JSTOR.

`weibullRMM_SEM` *St-EM algorithm for Reliability Mixture Models (RMM) of Weibull with right Censoring*

Description

Parametric Stochastic EM (St-EM) algorithm for univariate finite mixture of Weibull distributions with randomly right censored data.

Usage

```
weibullRMM_SEM(x, d = NULL, lambda = NULL, shape = NULL, scale = NULL,
        k = 2, maxit = 200, maxit.survreg = 200, epsilon = 1e-03,
        averaged = TRUE, verb = FALSE)
```

Arguments

`x`	A vector of n real positive lifetime (possibly censored) durations. If `d` is not `NULL` then a vector of random censoring times `c` occured, so that $x = min(x, c)$ and $d = I(x <= c)$.
`d`	The vector of censoring indication, where 1 means observed lifetime data, and 0 means censored lifetime data.
`lambda`	Initial value of mixing proportions. If `NULL`, then `lambda` is set to `rep(1/k,k)`.
`shape`	Initial value of Weibull component shapes, all set to 1 if `NULL`.
`scale`	Initial value of Weibull component scales, all set to 1 if `NULL`.
`k`	Number of components of the mixture.
`maxit`	The number of iterations allowed, since for St-EM algorithms convergence is not based on stabilization, exactly `maxit` iterations are performed (see Bordes L. and Chauveau D. (2016) reference below).
`maxit.survreg`	The number of iterations allowed in the computations of the MLE for censored weibull data from the `survival` package (see Bordes L. and Chauveau D. (2016) reference below).
`epsilon`	Tolerance parameter used in the numerical computations of the MLE for censored weibull data by `survreg` from the `survival` package (see Bordes L. and Chauveau D. (2016) reference below).
`averaged`	The way of updating parameters at each iteration: if `TRUE`, current values of the parameters are obtained by averaging the sequence (see Bordes L. and Chauveau D. (2016) reference below).
`verb`	If TRUE, print updates for every iteration of the algorithm as it runs

Details

This St-EM algorithm calls functions from the `survival` package to compute parametric MLE for censored weibull data.

Value

`weibullRMM_SEM` returns a list of class "mixEM" with the following items:

`x`	The input data.
`d`	The input censoring indicator.
`lambda`	The estimates for the mixing proportions.
`scale`	The estimates for the Weibull component scales.
`shape`	The estimates for the Weibull component shapes.
`loglik`	The log-likelihood value at convergence of the algorithm.
`posterior`	An $n \times k$ matrix of posterior probabilities for observation, after convergence of the algorithm.
`all.loglik`	The sequence of log-likelihoods over iterations.
`all.lambda`	The sequence of mixing proportions over iterations.
`all.scale`	The sequence of component scales over iterations.
`all.shape`	The sequence of component shapes over iterations.
`ft`	A character vector giving the name of the function called.

Author(s)

Didier Chauveau

References

- Bordes, L., and Chauveau, D. (2016), Stochastic EM algorithms for parametric and semiparametric mixture models for right-censored lifetime data, Computational Statistics, Volume 31, Issue 4, pages 1513-1538. http://link.springer.com/article/10.1007/s00180-016-0661-7

See Also

Related functions: `plotweibullRMM`, `summary.mixEM`.

Other models and algorithms for censored lifetime data (name convention is model_algorithm): `expRMM_EM`, `spRMM_SEM`.

Examples

```
n = 500 # sample size
m = 2 # nb components
lambda=c(0.4, 0.6)
shape <- c(0.5,5); scale <- c(1,20) # model parameters
set.seed(321)
x <- rweibullmix(n, lambda, shape, scale) # iid ~ weibull mixture
```

```
cs=runif(n,0,max(x)+10) # iid censoring times
t <- apply(cbind(x,cs),1,min) # censored observations
d <- 1*(x <= cs)              # censoring indicator

## set arbitrary or "reasonable" (e.g., data-driven) initial values
l0 <- rep(1/m,m); sh0 <- c(1, 2); sc0 <- c(2,10)
# Stochastic EM algorithm
a <- weibullRMM_SEM(t, d, lambda = l0, shape = sh0, scale = sc0, maxit = 200)

summary(a) # Parameters estimates etc
plotweibullRMM(a) # plot of St-EM sequences
plot(a, which=2) # or equivalently, S3 method for "mixEM" object
```

---

wkde *Weighted Univariate (Normal) Kernel Density Estimate*

Description

Evaluates a weighted kernel density estimate, using a Gaussian kernel, at a specified vector of points.

Usage

```
wkde(x, u=x, w=rep(1, length(x)), bw=bw.nrd0(as.vector(x)), sym=FALSE)
```

Arguments

x	Data
u	Points at which density is to be estimated
w	Weights (same length as x)
bw	Bandwidth
sym	Logical: Symmetrize about zero?

Value

A vector of the same length as u

References

- Benaglia, T., Chauveau, D., and Hunter, D. R. (2009), An EM-like algorithm for semi- and non-parametric estimation in multivariate mixtures, Journal of Computational and Graphical Statistics, 18, 505-526.
- Benaglia, T., Chauveau, D., Hunter, D. R., and Young, D. (2009), mixtools: An R package for analyzing finite mixture models. Journal of Statistical Software, 32(6):1-29.

See Also

npEM, ise.npEM

Examples

```
# Mixture with mv gaussian model
set.seed(100)
m <- 2 # no. of components
r <- 3 # no. of repeated measures (coordinates)
lambda <- c(0.4, 0.6)
mu <- matrix(c(0, 0, 0, 4, 4, 6), m, r, byrow=TRUE) # means
sigma <- matrix(rep(1, 6), m, r, byrow=TRUE) # stdevs
centers <- matrix(c(0, 0, 0, 4, 4, 4), 2, 3, byrow=TRUE) # initial centers for est

blockid = c(1,1,2) # block structure of coordinates
n = 100
x <- rmvnormmix(n, lambda, mu, sigma) # simulated data
a <- npEM(x, centers, blockid, eps=1e-8, verb=FALSE)

par(mfrow=c(2,2))
u <- seq(min(x), max(x), len=200)
for(j in 1:2) {
  for(b in 1:2) {
    xx <- as.vector(x[,a$blockid==b])
    wts <- rep(a$post[,j], length.out=length(xx))
    bw <- a$bandwidth
    title <- paste("j =", j, "and b =", b)
    plot(u, wkde(xx, u, wts, bw), type="l", main=title)
  }
}
```

---

wquantile *Weighted quantiles*

Description

Functions to compute weighted quantiles and the weighted interquartile range.

Usage

```
wquantile(wt = rep(1,length(x)), x, probs, already.sorted = FALSE,
          already.normalized = FALSE)
wIQR(wt = rep(1,length(x)), x, already.sorted = FALSE,
     already.normalized = FALSE)
```

Arguments

wt	Vector of weights
x	Vector of data, same length as wt
probs	Numeric vector of probabilities with values in [0,1].
already.sorted	If FALSE, sort wt and x in increasing order of x. If TRUE, it is assumed that wt and x are already sorted.
already.normalized	If FALSE, normalize wt by diving each entry by the sum of all entries. If TRUE, it is assumed that sum(wt)==1

Details

wquantile uses the findInterval function. wIQR calls the wquantile function.

Value

Returns the sample quantiles or interquartile range of a discrete distribution with support points x and corresponding probability masses wt

See Also

npEM

Examples

```
IQR(1:10)
wIQR(x=1:10) # Note:  Different algorithm than IQR function
wIQR(1:10,1:10) # Weighted quartiles are now 4 and 8
```

Index