Package 'mixR'

June 7, 2018

Type Package

Title Finite Mixture Modeling for Raw and Binned Data

Version 0.1.1

Date 2018-06-05

Maintainer Youjiao Yu <jiaoisjiao@gmail.com>

Description Performs maximum likelihood estimation for finite mixture models for families including Normal, Weibull, Gamma and Lognormal by using EM algorithm, together with Newton-Raphson algorithm or bisection method when necessary. It also conducts mixture model selection by using information criteria or bootstrap likelihood ratio test. The data used for mixture model fitting can be raw data or binned data. The model fitting process is accelerated by using R package 'Rcpp'.

License GPL (>= 2)

LazyData TRUE

Imports ggplot2 (>= 2.1.0), graphics, Rcpp (>= 0.12.8), stats

LinkingTo Rcpp

RoxygenNote 6.0.1

NeedsCompilation yes

Author Youjiao Yu [aut, cre]

Repository CRAN

Date/Publication 2018-06-07 13:50:35 UTC

R topics documented:

`mixR-package` *Finite Mixture Modeling for Raw and Binned Data*

Description

The package `mixR` performs maximum likelihood estimation for finite mixture models for families including Normal, Weibull, Gamma and Lognormal via EM algorithm. It also conducts model selection by using information criteria or bootstrap likelihood ratio test. The data used for mixture model fitting can be raw data or binned data. The model fitting is accelerated by using R package Rcpp.

Details

Finite mixture models can be represented by

$$f(x;\Phi) = \sum_{j=1}^{g} \pi_j f_j(x;\theta_j)$$

where $f(x;\Phi)$ is the probability density function (p.d.f.) or probability mass function (p.m.f.) of the mixture model, $f_j(x;\theta_j)$ is the p.d.f. or p.m.f. of the jth component of the mixture model, π_j is the proportion of the jth component and θ_j is the parameter of the jth component, which can be a scalar or a vector, Φ is a vector of all the parameters of the mixture model. The maximum likelihood estimate of the parameter vector Φ can be obtained by using the EM algorithm (Dempster *et al*, 1977). The binned data is present sometimes instead of the raw data, for the reason of storage convenience or necessity. The binned data is recorded in the form of (a_i, b_i, n_i) where a_i is the lower bound of the ith bin, b_i is the upper bound of the ith bin, and n_i is the number of observations that fall in the ith bin, for $i = 1, \ldots, r$, and r is the total number of bins.

To obtain maximum likelihood estimate of the finite mixture model for binned data, we can introduce two types of latent variables x and z, where$x$ represents the value of the unknown raw data, and z is a vector of zeros and one indicating the component that x belongs to. To use the EM algorithm we first write the complete-data log-likelihood

$$Q(\Phi; \Phi^{(p)}) = \sum_{j=1}^{g} \sum_{i=1}^{r} n_i z^{(p)} [\log f(x^{(p)}; \theta_j) + \log \pi_j]$$

where $z^{(p)}$ is the expected value of z given the estimated value of Φ and expected value $x^{(p)}$ at pth iteration. The estimated value of Φ can be updated iteratively via the E-step, in which we estimate Φ by maximizing the complete-data loglikelihood, and M-step, in which we calculate the expected value of the latent variables x and z. The EM algorithm is terminated by using a stopping rule. The M-step of the EM algorithm may or may not have closed-form solution (e.g. the Weibull mixture model or Gamma mixture model). If not, an iterative approach like Newton's algorithm or bisection method may be used.

For a given data set, when we have no prior information about the number of components g, its value should be estimated from the data. Because mixture models don't satisfy the regularity condition for the likelihood ratio test (which requires that the true parameter under the null hypothesis should be in the interior of the parameter space of the full model under the alternative hypothesis), a bootstrap approach is usually used in the literature (see McLachlan (1987, 2004), Feng and McCulloch (1996)). The general step of bootstrap likelihood ratio test is as follows.

1. For the given data x, estimate Φ under both the null and the alternative hypothesis to get $\hat{\Phi}_0$ and $\hat{\Phi}_1$. Calculate the observed log-likelihood $\ell(x; \hat{\Phi}_0)$ and $\ell(x; \hat{\Phi}_1)$. The likelihood ratio test statistic is defined as
$$w_0 = -2(\ell(x; \hat{\Phi}_0) - \ell(x; \hat{\Phi}_1)).$$
2. Generate random data of the same size as the original data x from the model under the null hypothesis using estimated parameter $\hat{\Phi}_0$, then repeat step 1 using the simulated data. Repeat this process for B times to get a vector of the simulated likelihood ratio test statistics $w_1^1, \ldots, w_1^B$.
3. Calculate the empirical p-value
$$p = \frac{1}{B} \sum_{i=1}^{B} I(w_1^{(i)} > w_0)$$
where I is the indicator function.

This package does the following three things.

1. Fitting finite mixture models for both raw data and binned data by using EM algorithm, together with Newton-Raphson algorithm and bisection method.
2. Do parametric bootstrap likelihood ratio test for two candidate models.
3. Do model selection by Bayesian information criterion.

To speed up computation, the EM algorithm is fulfilled in C++ by using Rcpp (Eddelbuettel and Francois (2011)).

Author(s)

Maintainer: Youjiao Yu <jiaoisjiao@gmail.com>

References

Dempster, A. P., Laird, N. M., and Rubin, D. B. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1-38, 1977.

Dirk Eddelbuettel and Romain Francois (2011). Rcpp: Seamless R and C++ Integration. *Journal of Statistical Software*, 40(8), 1-18. URL http://www.jstatsoft.org/v40/i08/.

Efron, B. Bootstrap methods: Another look at the jackknife. *Ann. Statist.*, 7(1):1-26, 01 1979.

Feng, Z. D. and McCulloch, C. E. Using bootstrap likelihood ratios in finite mixture models. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 609-617, 1996.

Lo, Y., Mendell, N. R., and Rubin, D. B. Testing the number of components in a normal mixture. *Biometrika*, 88(3):767-778, 2001.

McLachlan, G. J. On bootstrapping the likelihood ratio test statistic for the number of components in a normal mixture. *Applied statistics*, pages 318-324, 1987.

McLachlan, G. and Jones, P. Fitting mixture models to grouped and truncated data via the EM algorithm. *Biometrics*, pages 571-578, 1988.

McLachlan, G. and Peel, D. *Finite mixture models*. John Wiley & Sons, 2004.

`bin` *Binning the Raw Data*

Description

This function creates a binned data from a numeric vector

Usage

```
bin(x, brks)
```

Arguments

`x`	a numeric vector
`brks`	a numeric vector in increasing order, representing the value of each bin

Details

Given a numeric vector, the function `bin` creates binned data with bin value provided by `brks`. Fitting mixture models with a large dataset may be slow, especially when we want to fit a mixture model outside of normal family. Binning the data with a relatively small binwidth speeds up the computation of EM algorithm while at the same time keeps the precision of the estimation result.

Value

The function bin returns a matrix with three columns, representing the value of the left bin, the value of the right bin and the number of observations in x that falls in each bin.

See Also

reinstate

Examples

```
set.seed(99)
x <- rmixnormal(200, c(0.5, 0.5), c(2, 5), c(1, 1))
data <- bin(x, seq(-2, 10, 0.1))
fit1 <- mixfit(x, ncomp = 2)
fit2 <- mixfit(data, ncomp = 2)
```

---

bs.test *Bootstrap Likelihood Ratio Test for Finite Mixture Models*

Description

This function performs likelihood ratio test by parametric bootstrapping for mixture models with two different number of components.

Usage

```
bs.test(x, ncomp = c(1, 2), family = c("normal", "weibull", "gamma",
  "lnorm"), B = 100, ev = FALSE, mstep.method = c("bisection", "newton"),
  init.method = c("kmeans", "hclust"), tol = 1e-06, max_iter = 500)
```

Arguments

x	a numeric vector for the raw data or a three-column matrix for the binned data.
ncomp	a vector of two positive integers specifying the number of components of the mixture model under the null and alternative hypothesis. The first integer should be less than the second one. The default value is c(1, 2).
family	a character string specifying the family of the mixture model, which can be one of normal, weibull, gamma, or lnorm (default normal).
B	the number of bootstrap iterations (default 100).
ev	a logical value indicating whether we constrain the variances of each component to be equal or not when testing normal mixture models (default FALSE). ev is ignored when other family members are used.
mstep.method	the method used in M-step of EM algorithm when using weibull or gamma family. It is ignored when using normal or lnorm family, which has closed-form solution in the M-step. The default value is bisection.

init.method	a character string specifying the method used for providing initial values for the parameters for EM algorithm. It can be one of kmeans or hclust. The default is kmeans
tol	the tolerance for the stopping rule of EM algorithm. It is the value to stop EM algorithm when the two consecutive iterations produces log-likelihood with difference less than tol. The default value is 1e-6.
max_iter	the maximum number of iterations for the EM algorithm (default 500).

Details

For the given data x and the specified family, the function bs.test conducts a bootstrap likelihood ratio test for two mixture models with the number of components under the null and the alternative hypothesis specified in ncomp.

Value

The function bs.test returns an object of class bootEM which contains the following three items.

pvalue	The p-value of the bootstrap likelihood ratio test
w0	the observed likelihood ratio test statistic
w1	a vector of simulated likelihood ratio test statistics

See Also

plot.bootEM, mixfit, select

Examples

```
## testing normal mixture models with 2 and 3 components
set.seed(100)
x <- rmixnormal(200, c(0.5, 0.5), c(2, 5), c(1, 0.7))
ret <- bs.test(x, ncomp = c(2, 3), B = 30)
ret

## (not run) testing Weibull mixture models with 2 and 3 components
## set.seed(101)
## x <- rmixweibull(200, c(0.3, 0.4, 0.3), c(2, 5, 8), c(1, 0.6, 0.8))
## ret <- bs.test(x, ncomp = c(2, 3), family = "weibull", B = 30)
## ret

## (not run) testing Gamma mixture models with 1 and 2 components
## set.seed(102)
## x <- rgamma(200, 2, 1)
## ret <- bs.test(x, ncomp = c(1, 2), family = "gamma", B = 30)
## ret
```

`density.mixfitEM` *The Density of Finite Mixture Models*

Description

This function calculates the probability density of a finite mixture model.

Usage

```
## S3 method for class 'mixfitEM'
density(x, smoothness = 512, from = NULL, to = NULL,
  cut = 3.5, ...)
```

Arguments

`x`	an object of class `mixfitEM`
`smoothness`	a positive integer controlling the smoothness of the density curve (default 512). The higher this value is, the more locations of the mixture model the density is calculated.
`from`	the starting location the density is going to be calculated
`to`	the ending location the density is going to be computed
`cut`	the number of standard deviations away the density is to be computed (default 3.5)
`...`	other arguments passed to `density`

Details

The function `density.mixfitEM` is the method of the generic function `density` for the class `mixfitEM`.

Value

This function returns a list of class `density`, which contains the following items.

`x`	a numeric vector of locations where density is calculated.
`y`	the density of the mixture model at the corresponding locations in `x`

See Also

`mixfit`

Examples

```
set.seed(102)
x <- rmixnormal(200, c(0.5, 0.5), c(2, 5), c(1, 0.7))
fit1 <- mixfit(x, ncomp = 2)
fit2 <- mixfit(x, ncomp = 2, ev = TRUE)
plot(fit1, detail = FALSE, breaks = 20)
lines(density(fit2), col = "red")
```

---

initz *Initialization of EM Algorithm*

Description

This function returns the mean and standard deviation of each component by using K-means clustering method or hierarchical clustering method.

Usage

```
initz(x, ncomp, init.method = c("kmeans", "hclust"))
```

Arguments

x	a numeric vector for raw data or a three-column matrix for binned data
ncomp	a positive integer specifying the number of components for a mixture model
init.method	the method used for providing initial values, which can be one of kmeans or hclust.

Details

The function initz returns the mean and standard deviation of each component of a mixture model by using K-means clustering algorithm, or hierarchical clustering method. It is used for automatically selecting initial values for the EM algorithm, so as to enable mixture model selection by bootstrapping likelihood ratio test or using information criteria.

Value

initz returns a list with three items

pi	a numeric vector of component proportions
mu	a numeric vector of component means
sd	a numeric vector of component standard deviations

Examples

```
x <- rmixnormal(500, c(0.5, 0.5), c(2, 5), c(1, 0.7))
data <- bin(x, seq(-2, 8, 0.25))
par1 <- initz(x, 2)
par2 <- initz(data, 2)
```

`mixfit` *Finite Mixture Modeling for Raw Data and Binned Data*

Description

This function is used to perform maximum likelihood estimation for a variety of finite mixture models for both raw data and binned data, by using the EM algorithm, combining Newton-Raphson algorithm or bisection method when necessary.

Usage

```
mixfit(x, ncomp = NULL, family = c("normal", "weibull", "gamma", "lnorm"),
  pi = NULL, mu = NULL, sd = NULL, ev = FALSE,
  mstep.method = c("bisection", "newton"), init.method = c("kmeans",
  "hclust"), tol = 1e-06, max_iter = 500)
```

Arguments

`x`	a numeric vector for row data or a three-column matrix for the binned data
`ncomp`	a positive integer specifying the number of components of the mixture model
`family`	a character string specifying the family of the mixture model. It can only be one element from `normal`, `weibull`, `gamma` or `lnorm`.
`pi`	a vector of the initial value for the proportion
`mu`	a vector of the initial value for the mean
`sd`	a vector of the initial value for the standard deviation
`ev`	a logical value controlling whether each component has the same variance when fitting normal mixture models. It is ignored when fitting other mixture models. The default is `FALSE`.
`mstep.method`	a character string specifying the method used in M-step of the EM algorithm when fitting weibull or gamma mixture models. It can be either `bisection` or `newton`. The default is `bisection`.
`init.method`	a character string specifying the method used for providing initial values for the parameters for EM algorithm. It can be one of `kmeans` or `hclust`. The default is `kmeans`
`tol`	the tolerance for the stopping rule of EM algorithm. It is the value to stop EM algorithm when the two consecutive iterations produces loglikelihood with difference less than `tol`. The default value is 1e-6.
`max_iter`	the maximum number of iterations for the EM algorithm (default 500).

Details

The function `mixfit` is the core function in this package. It is used to perform the maximum likelihood estimation for finite mixture models from the families of normal, weibull, gamma or lognormal by using the EM algorithm. When the family is `weibull` or `gamma`, the M-step of the EM algorithm has no closed-form solution and we can use Newton algorithm by specifying `method = "newton"` or use bisection method by specifying `method = "bisection"`.

The initial values of the EM algorithm can be provided by specifying the proportion of each component `pi`, the mean of each component `mu` and the standard deviation of each component `sd`. If one or more of these initial values are not provided, then their values are estimated by using K-means clustering method or hierarchical clustering method. If all of `pi`, `mu`, and `sd` are not provided, then `ncomp` should be provided so initial values are automatically generated. For the normal mixture models, we can control whether each component has the same variance or not.

Value

the function `mixfit` return an object of class `mixfitEM`, which contains a list of different number of items when fitting different mixture models. The common items include

`pi`	a numeric vector representing the estimated proportion of each component
`mu`	a numeric vector representing the estimated mean of each component
`sd`	a numeric vector representing the estimated standard deviation of each component
`iter`	a positive integer recording the number of EM iteration performed
`loglik`	the loglikelihood of the estimated mixture model for the data `x`
`aic`	the value of AIC of the estimated model for the data `x`
`bic`	the value of BIC of the estimated model for the data `x`
`data`	the data `x`
`comp.prob`	the probability that `x` belongs to each component
`family`	the family the mixture model belongs to

For the Weibull mixture model, the following extra items are returned.

`k`	a numeric vector representing the estimated shape parameter of each component
`lambda`	a numeric vector representing the estimated scale parameter of each component

For the Gamma mixture model, the following extra items are returned.

`alpha`	a numeric vector representing the estimated shape parameter of each component
`lambda`	a numeric vector representing the estimated rate parameter of each component

For the lognormal mixture model, the following extra items are returned.

`mulog`	a numeric vector representing the estimated logarithm mean of each component
`sdlog`	a numeric vector representing the estimated logarithm standard deviation of each component

See Also

plot.mixfitEM, density.mixfitEM, select, bs.test

Examples

```
## fitting the normal mixture models
set.seed(103)
x <- rmixnormal(200, c(0.3, 0.7), c(2, 5), c(1, 1))
data <- bin(x, seq(-1, 8, 0.25))
fit1 <- mixfit(x, ncomp = 2)  # raw data
fit2 <- mixfit(data, ncomp = 2)  # binned data
fit3 <- mixfit(x, pi = c(0.5, 0.5), mu = c(1, 4), sd = c(1, 1))  # providing the initial values
fit4 <- mixfit(x, ncomp = 2, ev = TRUE)  # setting the same variance

## (not run) fitting the weibull mixture models
## x <- rmixweibull(200, c(0.3, 0.7), c(2, 5), c(1, 1))
## data <- bin(x, seq(0, 8, 0.25))
## fit5 <- mixfit(x, ncomp = 2, family = "weibull")  # raw data
## fit6 <- mixfit(data, ncomp = 2, family = "weibull")  # binned data

## (not run) fitting the Gamma mixture models
## x <- rmixgamma(200, c(0.3, 0.7), c(2, 5), c(1, 1))
## data <- bin(x, seq(0, 8, 0.25))
## fit7 <- mixfit(x, ncomp = 2, family = "gamma")  # raw data
## fit8 <- mixfit(data, ncomp = 2, family = "gamma")  # binned data

## (not run) fitting the lognormal mixture models
## x <- rmixlnorm(200, c(0.3, 0.7), c(2, 5), c(1, 1))
## data <- bin(x, seq(0, 8, 0.25))
## fit9 <- mixfit(x, ncomp = 2, family = "lnorm")  # raw data
## fit10 <- mixfit(data, ncomp = 2, family = "lnorm")  # binned data
```

---

plot.bootEM *Plot Bootstrap Likelihood Ratio Test*

Description

This function is the plot method for the class bootEM.

Usage

```
## S3 method for class 'bootEM'
plot(x, ...)
```

Arguments

x	an object of class bootEM, which is the output of the function bs.test.
...	the other parameters passed to the function hist

Details

The histogram of the bootstrap LRT statistics w_1 is plotted, with the observed LRT statistic imposed in a red vertical line.

See Also

bs.test

Examples

```
## plotting the bootstrap LRT result
set.seed(100)
x <- rmixnormal(200, c(0.5, 0.5), c(2, 5), c(1, 0.7))
ret <- bs.test(x, ncomp = c(2, 3), B = 30)
plot(ret)
```

---

plot.mixfitEM *Plotting the Fitted Mixture Models*

Description

This is the plot method for the class mixfitEM. It is used to plot the fitted mixture models by using base R plotting system or using the package ggplot2.

Usage

```
## S3 method for class 'mixfitEM'
plot(x, ps = c("base", "ggplot2"), detail = TRUE,
  smoothness = 512, ...)
```

Arguments

x	an object of class mixfitEM, usually an output from the function mixfit
ps	a character string to select the plotting system, which can be base (default), or ggplot2
detail	a logical value controlling whether to show each component of the fitted mixture model (default TRUE)
smoothness	a positive integer controlling the smoothness of the density curve in the plot. The default value is 512 and increasing this value will produce smoother curve.
...	the other parameters controlling the appearance of the plot, which are the following parameters if we specify family as base: **xlim** a numeric vector of length 2 specifying the range of x-axis of the plot **ylim** a numeric vector of length 2 specifying the range of y-axis of the plot **lty** the line type of the mixture density curve, default 1

lwd the line width of the mixture density curve, default 2

color the line color of the mixture density curve, default "black"

... arguments passed to `hist`

or the following parameters if we specify family as `ggplot2`:

xlim a numeric vector of length 2 specifying the range of x-axis of the plot

ylim a numeric vector of length 2 specifying the range of y-axis of the plot

theme the background of the plot, can be "grey" or "bw" (default "grey")

trans the transparency of the plot, default 0.5

... arguments passed to `geom_path`

Details

The function `plot.mixfitEM` is used for plotting an object of class `mixfitEM`, which is an output of the function `mixfit`. Users can choose base R plotting system or ggplot2 (the package ggplot2 needs to be installed). plotting system. The plot is a density plot of the fitted mixture model imposed on top of a histogram. The parameters that control the appearance of the histogram and the density curve can be changed. The density curve of each component can be shown or hidden.

See Also

`mixfit`

Examples

```
x <- rmixnormal(200, c(0.3, 0.7), c(2, 5), c(1, 0.7))
fit <- mixfit(x, ncomp = 2)
plot(fit)  # base R plotting system
plot(fit, "ggplot2")  # ggplot2 plotting system
```

---

plot.selectEM *Plot Method for Class* `selectEM`

Description

This function plots the result of mixture model selection by BIC.

Usage

```
## S3 method for class 'selectEM'
plot(x, leg.loc = "topright", ...)
```

Arguments

x	an object of class selectEM, which is an output of the function select.
leg.loc	the location of the legend, which is the same as the first argument of the function
...	other arguments passed to plot legend. The default value is "topright". The user can change its location (to "topleft", "bottom right" etc.) if the visual plot conflicts with the legend.

Details

The function plot.selectEM is the plot method for the class selectEM. It plots the number of components against the corresponding value of BIC. It is used to visually display the mixture model selection result by BIC.

See Also

select

Examples

```
x <- rmixnormal(200, c(0.3, 0.7), c(2, 5), c(1, 1))
res <- select(x, ncomp = 1:3)
plot(res)
```

---

print.mixfitEM *Print Method for Class* mixfitEM

Description

This function is the print method for the mixfitEM class.

Usage

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

Arguments

x	an object of class mixfitEM
digits	the digits to print for the values in the print output. The default value is from the global option getOption("digits").
...	other arguments passed to print

Details

print.mixfitEM prints the value of the parameters of a fitted mixture model, together with some other information like the number of iterations of the EM algorithm, the loglikelihood, the value of AIC and BIC.

See Also

mixfit

Examples

```
x <- rmixnormal(200, c(0.5, 0.5), c(2, 5), c(1, 0.7))
fit <- mixfit(x, ncomp = 2)
print(x)
```

---

print.selectEM *Print Method for Class* selectEM

Description

The function prints the result of mixture model selection.

Usage

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

Arguments

x	an object of class selectEM
...	other arguments passed to print

Details

The function print.selectEM is the print method for the classselectEM, which is the output of the function select. It prints a data frame which contains the following information of each candidate mixture models: the number of components, whether the variance is the same for each component in a mixture model (only for normal), the value of BIC, and an indicator of the best model.

See Also

select

`reinstate` *Reinstate the Binned Data to the Raw Data*

Description

This function creates a numeric vector approximating the raw data from binned data

Usage

```
reinstate(data)
```

Arguments

`data`	a three-column matrix representing the raw data

Details

The function `reinstate` creates a numeric vector by generating n_i random data from the Uniform distribution $U(a_i, b_i)$ for $i = 1, \ldots, r$ and then combine all random data together. a_i, b_i, n_i are the first, second and the third column of the matrix `data` and r is the number of bins. It is used for enabling parameter initialization for EM algorithm when we fit mixture models for binned data.

Value

The function returns a numeric vector.

See Also

`bin`

Examples

```
x <- rnorm(100)
data <- bin(x, seq(-3, 3, 0.25))
y <- reinstate(data)
```

rmixgamma *Generating Random Data From A Gamma Mixture Model*

Description

The function rmixgamma generates random data from a Gamma mixture model.

Usage

```
rmixgamma(n, pi, mu, sd)
```

Arguments

n	a positive integer specifying the number of observations we want to generate from the mixture model
pi	a numeric vector for the proportion of each component
mu	a numeric vector for the mean of each component
sd	a numeric vector for the standard deviation of each component

Details

The number of random data from each component n_0 (a vector) is generated from a multinomial distribution Multinom(n, pi). Then the random data from each component is generated with the sample sized specified in n_0 and parameters of Gamma distributions specified in mu and sd.

Value

The function rmixgamma returns a numeric vector of random data from the specified Gamma mixture model.

See Also

rmixnormal, rmixweibull, rmixlnorm

Examples

```
x <- rmixgamma(1000, c(0.4, 0.6), c(2, 5), c(1, 0.5))
hist(x, breaks = 40)
```

`rmixlnorm` *Generating Random Data From A Lognormal Mixture Model*

Description

The function `rmixlnorm` generates random data from a lognormal mixture model.

Usage

```
rmixlnorm(n, pi, mu, sd)
```

Arguments

`n`	a positive integer specifying the number of observations we want to generate from the mixture model
`pi`	a numeric vector for the proportion of each component
`mu`	a numeric vector for the mean of each component
`sd`	a numeric vector for the standard deviation of each component

Details

The number of random data from each component n_0 (a vector) is generated from a multinomial distribution Multinom(n, pi). Then the random data from each component is generated with the sample sized specified in n_0 and parameters of lognormal distributions specified in `mu` and `sd`.

Value

The function `rmixlnorm` returns a numeric vector of random data from the specified lognormal mixture model.

See Also

`rmixnormal`, `rmixweibull`, `rmixgamma`

Examples

```
x <- rmixlnorm(1000, c(0.4, 0.6), c(2, 5), c(1, 0.5))
hist(x, breaks = 40)
```

rmixnormal *Generating Random Data From A Normal Mixture Model*

Description

The function rmixnormal generates random data from a normal mixture model.

Usage

```
rmixnormal(n, pi, mu, sd)
```

Arguments

n	a positive integer specifying the number of observations we want to generate from the mixture model
pi	a numeric vector for the proportion of each component
mu	a numeric vector for the mean of each component
sd	a numeric vector for the standard deviation of each component

Details

The number of random data from each component n_0 (a vector) is generated from a multinomial distribution Multinom(n, pi). Then the random data from each component is generated with the sample sized specified in n_0 and parameters of normal distributions specified in mu and sd.

Value

The function rmixnormal returns a numeric vector of random data from the specified normal mixture model.

See Also

rmixweibull, rmixgamma, rmixlnorm

Examples

```
x <- rmixnormal(1000, c(0.4, 0.6), c(2, 5), c(1, 0.5))
hist(x, breaks = 40)
```

`rmixweibull` *Generating Random Data From A Weibull Mixture Model*

Description

The function `rmixweibull` generates random data from a normal Weibull model.

Usage

```
rmixweibull(n, pi, mu, sd)
```

Arguments

`n`	a positive integer specifying the number of observations we want to generate from the mixture model
`pi`	a numeric vector for the proportion of each component
`mu`	a numeric vector for the mean of each component
`sd`	a numeric vector for the standard deviation of each component

Details

The number of random data from each component n_0 (a vector) is generated from a multinomial distribution Multinom(n, pi). Then the random data from each component is generated with the sample sized specified in n_0 and parameters of Weibull distributions specified in `mu` and `sd`.

Value

The function `rmixweibull` returns a numeric vector of random data from the specified Weibull mixture model.

See Also

`rmixnormal`, `rmixgamma`, `rmixlnorm`

Examples

```
x <- rmixweibull(1000, c(0.4, 0.6), c(2, 5), c(1, 0.5))
hist(x, breaks = 40)
```

`select` *Finite Mixture Model Selection by Information Criterion*

Description

This function selects the best model from a candidate of mixture models based on the information criterion BIC.

Usage

```
select(x, ncomp, family = c("normal", "weibull", "gamma", "lnorm"),
  mstep.method = c("bisection", "newton"), init.method = c("kmeans",
  "hclust"), tol = 1e-06, max_iter = 500)
```

Arguments

`x`	a numeric vector for raw data or a three-column matrix for the binned data
`ncomp`	a vector of positive integers specifying the number of components of the candidate mixture models
`family`	a character string specifying the family of the mixture model. It can only be one element from `normal`, `weibull`, `gamma` or `lnorm`.
`mstep.method`	a character string specifying the method used in M-step of the EM algorithm when fitting weibull or gamma mixture models. It can be either `bisection` or `newton`. The default is `bisection`.
`init.method`	a character string specifying the method used for providing initial values for the parameters for EM algorithm. It can be one of `kmeans` or `hclust`. The default is `kmeans`
`tol`	the tolerance for the stopping rule of EM algorithm. It is the value to stop EM algorithm when the two consecutive iterations produces loglikelihood with difference less than `tol`. The default value is 1e-6.
`max_iter`	the maximum number of iterations for the EM algorithm (default 500).

Details

By specifying different number of components, the function `select` fits a series of mixture models for a given family, and a mixture model with minimum value of BIC is regarded as the best.

Value

The function returns an object of class `selectEM` which contains the following items.

`ncomp`	the specified number of components of the candidate mixture models
`equal.var`	a logical vector indicating whether the variances of each component in each mixture model are constrained to be the same (only for `normal` family)
`bic`	the value of BIC for each mixture model
`best`	an indicator of the best model
`family`	the family of the mixture model

See Also

plot.selectEM, bs.test, mixfit

Examples

```
## selecting the optimal normal mixture model by BIC
set.seed(105)
x <- rmixnormal(1000, c(0.3, 0.4, 0.3), c(-4, 0, 4), c(1, 1, 1))
hist(x, breaks = 40)
ret <- select(x, ncomp = 2:5)
## [1] "The final model: normal mixture (equal variance) with 3 components"

## (not run) selecting the optimal Weibull mixture model by BIC
## set.seed(106)
## x <- rmixweibull(1000, c(0.3, 0.4, 0.3), c(2, 5, 8), c(0.7, 0.6, 1))
## ret <- select(x, ncomp = 2:5, family = "weibull")
## [1] "The final model: weibull mixture with 3 components"

## (not run) selecting the optimal Gamma mixture model by BIC
## set.seed(107)
## x <- rmixgamma(1000, c(0.3, 0.7), c(2, 5), c(0.7, 1))
## ret <- select(x, ncomp = 2:5, family = "gamma")
## [1] "The final model: gamma mixture with 2 components"


## (not run) selecting the optimal lognormal mixture model by BIC
## set.seed(108)
## x <- rmixlnorm(1000, c(0.2, 0.3, 0.2, 0.3), c(4, 7, 9, 12), c(1, 0.5, 0.7, 1))
## ret <- select(x, ncomp = 2:6, family = "lnorm")
## [1] "The final model: lnorm mixture with 4 components"
```

Stamp *1872 Hidalgo Stamp Data*

Description

A vector containing the 1872 Hidalgo stamp data

Usage

```
Stamp
```

Format

A vector with 485 measurements of the thickness (nm) of the stamps

References

Izenman, A. J. and Sommer, C. J. Philatelic mixtures and multimodal densities. *Journal of the American Statistical association*, 83(404):941-953, 1988.

`Stamp2` *1872 Hidalgo Stamp Data (Binned)*

Description

A dataset containing the 1872 Hidalgo stamp data in the form of binned data

Usage

```
Stamp2
```

Format

A matrix with 62 rows and 3 columns:

lower the lower bin values

upper the upper bin values

freq the number of observations in each bin

`to_k_lambda_weibull` *Parameter Conversion for Weibull Distribution*

Description

The function `to_k_lambda_weibull` converts the mean and standard deviation to the shape and scale for the Weibull distributions.

Usage

```
to_k_lambda_weibull(mu, sd)
```

Arguments

`mu`	a numeric vector representing the means of Weibull distributions
`sd`	a numeric vector representing the standard deviations of Weibull distributions. `mu` and `sd` should have the same length.

Details

The purpose of this function is to convert the parameterization of Weibull distribution in the form of mean and standard deviation to the form of shape and scale. It can be used for specifying the initial values for the EM algorithm when the first-hand intial values are in the form of mean and standard deviation from K-means clustering algorithm.

Value

a list of two items

k	a vector of the shapes of Weibull distributions
lambda	a vector of the scales of Weibull distributions

See Also

to_mu_sd_weibull

Examples

```
to_k_lambda_weibull(2, 1)
to_k_lambda_weibull(c(2, 5), c(1, 0.7))
```

---

to_mulog_sdlog_lnorm *Parameter Conversion for Lognormal Distribution*

Description

The function to_mulog_sdlog_lnorm converts the mean and standard deviation to the logarithm mean and logarithm standard deviation

Usage

```
to_mulog_sdlog_lnorm(mu, sd)
```

Arguments

mu	a vector of means of lognormal distributions
sd	a vector of standard deviations of lognormal distributions

Details

The purpose of this function is to convert the parameterization of lognormal distribution in the form of mean and standard deviation to the form of logarithm mean and logarithm standard deviation. It can be used for specifying the initial values for the EM algorithm when the first-hand intial values are in the form of mean and standard deviation from K-means clustering algorithm.

Value

a list of two items

mulog	a vector of lognormal means of lognormal distributions
sdlog	a vector of lognormal standard deviations of lognormal distributions

See Also

to_mu_sd_lnorm

Examples

```
to_mulog_sdlog_lnorm(2, 1)
to_mulog_sdlog_lnorm(c(2, 4), c(1, 1))
```

---

to_mu_sd_gamma *Parameter Conversion for Gamma Distribution*

Description

The function to_mu_sd_gamma converts the shape and rate to the mean and standard deviation

Usage

```
to_mu_sd_gamma(alpha, lambda)
```

Arguments

alpha	a numeric vector representing the shape of one or more than one gamma distributions
lambda	a numeric vector representing the rate of one or more than one gamma distributions. alpha and lambda should have the same length.

Details

The purpose of this function is to convert the parameterization of gamma distribution in the form of shape and rate to the form of mean and standard deviation.

Value

a list of two items

mu	a vector of the means of gamma distributions
sd	a vector of the standard deviations of gamma distributions

See Also

to_shape_rate_gamma

Examples

```
to_mu_sd_gamma(2, 1)
to_mu_sd_gamma(c(2, 4), c(1, 1))
```

---

to_mu_sd_lnorm *Parameter Conversion for Lognormal Distribution*

Description

The function to_mu_sd_lnorm converts the logarithm mean and logarithm standard deviation to the mean and standard deviation

Usage

```
to_mu_sd_lnorm(mulog, sdlog)
```

Arguments

mulog	a vector of logarithm means of lognormal distributions
sdlog	a vector of logarithm standard deviations of lognormal distributions

Details

The purpose of this function is to convert the parameterization of lognormal distribution in the form of logarithm mean and logarithm standard deviation to the form of mean and standard deviation.

Value

a list of two items

mu	a vector of the means of lognormal distributions
sd	a vector of the standard deviations of lognormal distributions

See Also

to_mulog_sdlog_lnorm

Examples

```
to_mu_sd_lnorm(2, 1)
to_mu_sd_lnorm(c(2, 4), c(1, 1))
```

`to_mu_sd_weibull` *Parameter Conversion for Weibull Distribution*

Description

The function `to_mu_sd_weibull` converts the parameters of shape and scale of weibull distributions to the parameters of the mean and standard deviation.

Usage

```
to_mu_sd_weibull(k, lambda)
```

Arguments

`k`	a numeric vector representing the shape of a series of Weibull distributions
`lambda`	a numeric vector representing the scale of a series of Weibull distributions. `k` and `lambda` should have the same length.

Details

The purpose of this function is to convert the parameterization of Weibull distribution in the form of shape and scale to the form of mean and standard deviation.

Value

a list of two items

`mu`	a vector of the means of Weibull distributions
`sd`	a vector of the standard deviations of Weibull distributions

See Also

`to_k_lambda_weibull`

Examples

```
to_mu_sd_weibull(2, 1)
to_mu_sd_weibull(c(2, 4), c(1, 1))
```

`to_shape_rate_gamma` *Parameter Conversion for Gamma Distribution*

Description

The function `to_shape_rate_gamma` converts the mean and standard deviation to the shape and rate

Usage

```
to_shape_rate_gamma(mu, sd)
```

Arguments

`mu`	a numeric vector representing the means of gamma distributions
`sd`	a numeric vector representing the standard deviations of gamma distributions. `mu` and `sd` should have the same length.

Details

The purpose of this function is to convert the parameterization of gamma distribution in the form of mean and standard deviation to the form of shape and rate. It can be used for specifying the initial values for the EM algorithm when the first-hand intial values are in the form of mean and standard deviation from K-means clustering algorithm.

Value

a list of two items

`alpha`	a vector of the shapes of gamma distributions
`lambda`	a vector of the rates of gamma distributions

See Also

`to_mu_sd_gamma`

Examples

```
to_shape_rate_gamma(2, 1)
to_shape_rate_gamma(c(2, 4), c(1, 1))
```

Index