Package ‘MixSemiRob’

January 20, 2025

Title Mixture Models: Parametric, Semiparametric, and Robust

Version 1.1.0

Description Various functions are provided to estimate parametric mixture models (with Gaussian, t, Laplace, log-concave distributions, etc.) and non-parametric mixture models. The package performs hypothesis tests and addresses label switching issues in mixture models.
The package also allows for parameter estimation in mixture of regressions, proportion-varying mixture of regressions, and robust mixture of regressions.

License GPL (>= 2)

Encoding UTF-8

RoxygenNote 7.2.3

Imports GoFKernel, MASS, mixtools, mvtnorm, Rlab, robustbase, ucminf, pracma, quadprog, stats

Suggests knitr, rmarkdown

Depends R (>= 2.10)

LazyData true

NeedsCompilation no

Author Suyeon Kang [aut, cre] (<`https://orcid.org/0000-0001-6506-3035`>),
Xin Shen [aut] (<`https://orcid.org/0000-0002-7332-4669`>),
Weixin Yao [aut] (<`https://orcid.org/0000-0001-5925-5081`>),
Sijia Xiang [aut],
Yan Ge [aut, trl]

Maintainer Suyeon Kang <`suyeon.kang@ufl.edu`>

Repository CRAN

Date/Publication 2023-09-20 07:20:13 UTC

Contents

`AFDP` *AFDP data*

Description

The data contains the 11 sensor measures aggregated over one hour (by means of average or sum) from a gas turbine located in Turkey's north western region for the purpose of studying flue gas emissions.

Usage

```
AFDP
```

Format

A data frame containing 7411 observations.

`complh` *Complete Likelihood Frequency Method for Label Switching*

Description

'complh' is used to address the label switching problem by maximizing the complete likelihood (Yao, 2015). This method leverages information from the latent component label, which is the label the user used to generate the sample. The function supports both one-dimensional (with equal variances or unequal variances) and multi-dimensional data (with equal variances).

Usage

```
complh(est, lat)
```

Arguments

`est`	a list with four elements representing the estimated mixture model, which can be obtained using the `mixnorm` function. When specified, it has the form of `list(mu, sigma, pi, p)`, where `mu` is a C by p matrix of estimated component means where p is the dimension of data and C is the number of mixture components, `sigma` is a p by p matrix of estimated common standard deviation for all components (when the data is multi-dimensional) or a C-dimensional vector of estimated component standard deviations (when the data is one-dimensional), `pi` is a C-dimensional vector of mixing proportions, and `p` is a C by n matrix of the classification probabilities, where the (i, j)th element corresponds to the probability of the jth observation belonging to the ith component.
`lat`	a C by n zero-one matrix representing the latent component labels for all observations, where C is the number of components in the mixture model and n is the number of observations. If the (i, j)th cell is 1, it indicates that the jth observation belongs to the ith component.

Value

The estimation results adjusted to account for potential label switching problems are returned, as a list containing the following elements:

`mu`	C by p matrix of estimated component means.
`sigma`	C-dimensional vector of estimated component standard deviations (for univariate data) or p by p matrix of estimated component variance (for multivariate data).
`pi`	C-dimensional vector of estimated mixing proportions.

References

Yao, W. (2015). Label switching and its solutions for frequentist mixture models. Journal of Statistical Computation and Simulation, 85(5), 1000-1012.

See Also

`distlat`, `mixnorm`

Examples

```
#-------------------------------------------------------------------------------------------#
# Example 1: Two-component Univariate Normal Mixture
#-------------------------------------------------------------------------------------------#
# Simulate the data
set.seed(827)
n = 200
prop = 0.3
n1 = rbinom(1, n, prop)
mudif = 1.5
x1 = rnorm(n1, 0, 1)
x2 = rnorm(n - n1, mudif, 1)
x = c(x1, x2)
pm = c(2, 1, 3, 5, 4)

# Use the `mixnorm' function to get the MLE and the estimated classification probabilities
out = mixnorm(x, 2)

# Prepare latent component label
lat = rbind(rep(c(1, 0), times = c(n1, n - n1)),
            rep(c(0, 1), times = c(n1, n - n1)))

# Fit the complh/distlat function
clhest = complh(out, lat)
clhest
# Result:
# mean of the first component: -0.1037359,
# mean of the second component: 1.6622397,
# sigma is 0.8137515 for both components, and
# the proportions for the two components are
# 0.3945660 and 0.6054340, respectively.
ditlatest = distlat(out, lat)

#-------------------------------------------------------------------------------------------#
# Example 2: Two-component Multivariate Normal Mixture
#-------------------------------------------------------------------------------------------#
# Simulate the data
n = 400
prop = 0.3
n1 = rbinom(1, n, prop)
pi = c(prop, 1 - prop)
mu1 = 0.5
mu2 = 0.5
```

```
mu = matrix(c(0, mu1, 0, mu2), ncol = 2)
pm = c(2, 1, 4, 3, 6, 5)
sigma = diag(c(1, 1))
ini = list(sigma = sigma, mu = mu, pi = pi)
x1 = mvtnorm::rmvnorm(n1, c(0, 0), ini$sigma)
x2 = mvtnorm::rmvnorm(n - n1, c(mu1, mu2), ini$sigma)
x = rbind(x1, x2)

# Use the `mixnorm' function to get the MLE and the estimated classification probabilities
out = mixnorm(x, 2)

# Prepare latent component label
lat = rbind(rep(c(1, 0), times = c(n1, n - n1)),
            rep(c(0, 1), times = c(n1, n - n1)))

# Fit the complh/distlat function
clhest = complh(out, lat)
distlatest = distlat(out, lat)

#-------------------------------------------------------------------------------------------#
# Example 3: Three-component Multivariate Normal Mixture
#-------------------------------------------------------------------------------------------#
# Simulate the data
n = 100
pi = c(0.2, 0.3, 0.5)
ns = stats::rmultinom(1, n, pi)
n1 = ns[1]; n2 = ns[2]; n3 = ns[3]
mu1 = 1
mu2 = 1
mu = matrix(c(0, mu1, 2 * mu1, 0, mu2, 2 * mu2), ncol = 2)
sigma = diag(c(1, 1))
ini = list(sigma = sigma, mu = mu, pi = pi)
x1 = mvtnorm::rmvnorm(n1, c(0, 0), ini$sigma)
x2 = mvtnorm::rmvnorm(n2, c(mu1, mu2), ini$sigma)
x3 = mvtnorm::rmvnorm(n3, c(2 * mu1, 2 * mu2), ini$sigma)
x = rbind(x1, x2, x3)

# Use the `mixnorm' function to get the MLE and the estimated classification probabilities
out = mixnorm(x, 3)

# Prepare latent component label
lat = rbind(rep(c(1, 0), times = c(n1, n - n1)),
            rep(c(0, 1, 0), times = c(n1, n2, n3)),
            rep(c(0, 1), times = c(n - n3, n3)))

# Fit the complh/distlat function
clhest = complh(out, lat)
distlatest = distlat(out, lat)
```

`distlat` *Euclidean Distance Based Labeling Method for Label Switching*

Description

'distlat' is used to address the label switching problem by minimizing the distance between the classification probabilities and the latent component label, which is the label used by the user to generate the sample (Yao, 2015). The function supports both one-dimensional (with equal variances or unequal variances) and multi-dimensional data (with equal variances).

Usage

```
distlat(est, lat)
```

Arguments

`est`	a list with four elements representing the estimated mixture model, which can be obtained using the `mixnorm` function. When specified, it has the form of `list(mu, sigma, pi, p)`, where `mu` is a C by p matrix of estimated component means where p is the dimension of data and C is the number of mixture components, `sigma` is a p by p matrix of estimated common standard deviation for all components (when the data is multi-dimensional) or a C-dimensional vector of estimated component standard deviations (when the data is one-dimensional), `pi` is a C-dimensional vector of mixing proportions, and `p` is a C by n matrix of the classification probabilities, where the (i, j)th element corresponds to the probability of the jth observation belonging to the ith component.
`lat`	a C by n zero-one matrix representing the latent component labels for all observations, where C is the number of components in the mixture model and n is the number of observations. If the (i, j)th cell is 1, it indicates that the jth observation belongs to the ith component.

Value

The estimation results adjusted to account for potential label switching problems are returned, as a list containing the following elements:

`mu`	C by p matrix of estimated component means.
`sigma`	C-dimensional vector of estimated component standard deviations (for univariate data) or p by p matrix of estimated component variance (for multivariate data).
`pi`	C-dimensional vector of estimated mixing proportions.

References

Yao, W. (2015). Label switching and its solutions for frequentist mixture models. Journal of Statistical Computation and Simulation, 85(5), 1000-1012.

See Also

`complh`, `mixnorm`

Examples

```
# See examples for the `complh' function.
```

`elbow` *Elbow data*

Description

The data contains elbow dimension measurements on 507 individuals (247 men and 260 women), primarily in their twenties and thirties, all of whom exercise several hours a week.

Usage

```
elbow
```

Format

A data frame containing 507 observations (elbow diameter, sum of two elbows).

References

Heinz, G., Peterson, L. J., Johnson, R. W., and Kerk, C. J. (2003). Exploring relationships in body dimensions. Journal of Statistics Education, 11(2)

`EMnormal` *Parameter Estimation of Normal Mixture Using EM Algorithm*

Description

'EMnormal' is used to estimate the parameters of a univariate or multivariate normal mixture model using the expectation-maximization (EM) algorithm. The result can be used as the initial value for the `mixLogconc` and `mixLogconcHD` function.

Usage

```
EMnormal(x, C = 2, nstart = 20, tol = 1e-05)
```

Arguments

`x`	an n by p data matrix where n is the number of observations and p is the dimension of the data.
`C`	number of mixture components. Default is 2.
`nstart`	number of initializations to try. Default is 20.
`tol`	stopping criteria (threshold value) for the EM algorithm. Default is 1e-05.

Value

A list containing the following elements:

`loglik`	final log-likelihood.
`pi`	estimated mixing proportions.
`mu`	estimated component means.
`sigma`	estimated component standard deviation or covariance matrix.

See Also

`mixLogconc`, `mixLogconcHD`

Examples

```
#-------------------------------------------------------------------------------------------#
# Univariate Case
#-------------------------------------------------------------------------------------------#
x = matrix(rnorm(100, 2, sqrt(2)), nrow = 100)
x[1:60] = x[1:60] + 5
ini = EMnormal(x)
```

`ethanol` *Ethanol data*

Description

The data contains 88 sets of measurements of peak nitrogen oxide emission levels from an experiment in which ethanol was burned in a single-cylinder engine. The emission levels of nitrogen oxides were recorded under different settings of the compression ratio and equivalence ratio of the engine.

Usage

```
ethanol
```

Format

A data frame containing 88 observations and the following 3 variables.

NO: concentration of nitric oxide (NO) and nitrogen dioxide (NO2) in engine exhaust.

Compression: compression ratio of engine.

Equivalence: equivalence ratio. This is a measure of the richness of the air and ethanol fuel mixture.

Source

Original source:

Brinkman, N. D. (1981). Ethanol fuel—single—cylinder engine study of efficiency and exhaust emissions. SAE transactions, 1410-1424.

R source:

Wand M (2018). SemiPar: Semiparametric Regression. R package version 1.0-4.2, `https://CRAN.R-project.org/package=SemiPar`.

Sarkar D (2008). Lattice: Multivariate Data Visualization with R. Springer, New York. ISBN 978-0-387-75968-5, `http://lmdvr.r-forge.r-project.org`.

References

Ruppert, D., Wand, M. P., and Carroll, R. J. (2003). Semiparametric regression (No. 12). Cambridge university press.

Hurn, M., Justel, A., and Robert, C. P. (2003). Estimating mixtures of regressions. Journal of computational and graphical statistics, 12(1), 55-79.

`kdeem` *Kernel Density-based EM-type algorithm for Semiparametric Mixture Regression with Unspecified Error Distributions*

Description

'kdeem' is used for semiparametric mixture regression using a kernel density-based expectation-maximization (EM)-type algorithm with unspecified homogeneous or heterogenous error distributions (Ma et al., 2012).

Usage

```
kdeem(x, y, C = 2, ini = NULL, maxiter = 200)
```

Arguments

`x`	an n by p data matrix where n is the number of observations and p is the number of explanatory variables (including the intercept).
`y`	an n-dimensional vector of response variable.
`C`	number of mixture components. Default is 2.
`ini`	initial values for the parameters. Default is NULL, which obtains the initial values using the `kdeem.lse` function. If specified, it can be a list with the form of `list(beta, prop, tau, pi, h)`, where `beta` is a p by C matrix for regression coefficients of C components, `prop` is an n by C matrix for probabilities of each observation belonging to each component, caculated based on the initial `beta` and `h`, `tau` is a vector of C precision parameters (inverse of standard deviation), `pi` is a vector of C mixing proportions, and `h` is the bandwidth for kernel estimation.
`maxiter`	maximum number of iterations for the algorithm. Default is 200.

Details

It can be used for a semiparametric mixture of linear regression models with unspecified component error distributions. The errors can be either homogeneous or heterogenous. The model is as follows:

$$f_{Y|\boldsymbol{X}}(y, \boldsymbol{x}, \boldsymbol{\theta}, g) = \sum_{j=1}^{C} \pi_j \tau_j g\{(y - \boldsymbol{x}^\top \boldsymbol{\beta}_j)\tau_j\}.$$

Here, $\boldsymbol{\theta} = (\pi_1, ..., \pi_{C-1}, \boldsymbol{\beta}_1^\top, .., \boldsymbol{\beta}_C^\top, \tau_1, ..., \tau_C)^\top$, $g(\cdot)$ is an unspecified density function with mean 0 and variance 1, and τ_j is a precision parameter. For the calculation of β in the M-step, this function employs the universal optimizer function `ucminf` from the 'ucminf' package.

Value

A list containing the following elements:

`posterior`	posterior probabilities of each observation belonging to each component.
`beta`	estimated regression coefficients.
`tau`	estimated precision parameters, the inverse of standard deviation.
`pi`	estimated mixing proportions.
`h`	bandwidth used for the kernel estimation.

References

Ma, Y., Wang, S., Xu, L., & Yao, W. (2021). Semiparametric mixture regression with unspecified error distributions. Test, 30, 429-444.

See Also

`kdeem.h`, `kdeem.lse`, and `ucminf` for beta calculation.

Examples

```
n = 300
C = 2
Dimen = 2
Beta.true.matrix = matrix(c(-3, 3, 3, -3), Dimen, C)
PI.true = c(0.5, 0.5)
x = runif(n)
X = cbind(1, x)
Group.ID = Rlab::rbern(n, prob = 0.5)
Error = rnorm(n, 0, 1)
n1 = sum(Group.ID)
n2 = n - n1
y = rep(0, n)
err = rep(0, n)

for(i in 1:n){
  if(Group.ID[i] == 1){
    err[i] = Error[i]
    y[i] = X[i, ] %*% Beta.true.matrix[, 1] + err[i]
  } else {
    err[i] = 0.5 * Error[i]
    y[i] = X[i, ] %*% Beta.true.matrix[, 2] + err[i]
  }
}
Result.kdeem.lse = kdeem.lse(x, y)
Result.kdeem.h = kdeem.h(x, y, 2, Result.kdeem.lse, maxiter = 200)
Result.kdeem = kdeem(x, y, 2, Result.kdeem.lse, maxiter = 200)
```

kdeem.h *Kernel Density-based EM-type algorithm for Semiparametric Mixture Regression with Unspecified Homogenous Error Distributions*

Description

'kdeem.h' is used for semiparametric mixture regression using a kernel density-based expectation-maximization (EM)-type algorithm with unspecified homogeneous error distributions (Hunter and Young, 2012).

Usage

```
kdeem.h(x, y, C = 2, ini = NULL, maxiter = 200)
```

Arguments

`x`	an n by p data matrix where n is the number of observations and p is the number of explanatory variables (including the intercept).
`y`	an n-dimensional vector of response variable.
`C`	number of mixture components. Default is 2.
`ini`	initial values for the parameters. Default is NULL, which obtains the initial values using the `kdeem.lse` function. If specified, it can be a list with the form of `list(beta, prop, tau, pi, h)`, where `beta` is a p by C matrix for regression coefficients of C components, `prop` is an n by C matrix for probabilities of each observation belonging to each component, calculated based on the initial `beta` and `h`, `tau` is a vector of C precision parameters (inverse of standard deviation), `pi` is a vector of C mixing proportions, and `h` is the bandwidth for kernel estimation.
`maxiter`	maximum number of iterations for the algorithm. Default is 200.

Details

'kdeem.h' can be used to estimate parameters in a mixture-of-regressions model with independent identically distributed errors. The model is defined as follows:

$$f_{Y|\boldsymbol{X}}(y, \boldsymbol{x}, \boldsymbol{\theta}, g) = \sum_{j=1}^{C} \pi_j g(y - \boldsymbol{x}^\top \boldsymbol{\beta}_j).$$

Here, $\boldsymbol{\theta} = (\pi_1, ..., \pi_{C-1}, \boldsymbol{\beta}_1^\top, \cdots, \boldsymbol{\beta}_C^\top)$, and $g(\cdot)$ represents identical unspecified density functions. The bandwidth of the kernel density estimation is calculated adaptively using the `bw.SJ` function from the 'stats' package, which implements the method of Sheather & Jones (1991) for bandwidth selection based on pilot estimation of derivatives.

For the calculation of β in the M-step, this function employs the universal optimizer `ucminf` from the 'ucminf' package.

Value

A list containing the following elements:

`posterior`	posterior probabilities of each observation belonging to each component.
`beta`	estimated regression coefficients.
`pi`	estimated mixing proportions.
`h`	bandwidth used for the kernel estimation.

References

Hunter, D. R., & Young, D. S. (2012). Semiparametric mixtures of regressions. Journal of Non-parametric Statistics, 24(1), 19-38.

Ma, Y., Wang, S., Xu, L., & Yao, W. (2021). Semiparametric mixture regression with unspecified error distributions. Test, 30, 429-444.

See Also

`kdeem`, `kdeem.lse`, `bw.SJ` for bandwidth calculation, and `ucminf` for beta calculation.

Examples

```
# See examples for the `kdeem' function.
```

`kdeem.lse` *Kernel Density-based EM-type algorithm with Least Square Estimation for Semiparametric Mixture Regression with Unspecified Homogenous Error Distributions*

Description

'kdeem.lse' is used for semiparametric mixture regression based on least squares estimation (Hunter and Young, 2012) using a kernel density-based expectation-maximization (EM)-type algorithm with unspecified homogeneous error distributions.

Usage

```
kdeem.lse(x, y, C = 2, ini = NULL)
```

Arguments

`x`	an n by p data matrix where n is the number of observations and p is the number of explanatory variables (including the intercept).
`y`	an n-dimensional vector of response variable.
`C`	number of mixture components. As of version 1.1.0, C must be set to 2.
`ini`	initial values for the parameters. Default is NULL, which obtains the initial values using the `regmixEM` function from the 'mixtools' package. If specified, it can be a list with the form of `list(beta, prop, tau, pi, h)`, where `beta` is a p by C matrix for regression coefficients of C components, `prop` is an n by C matrix for probabilities of each observation belonging to each component, caculated based on the initial `beta` and `h`, `tau` is a vector of C precision parameters (inverse of standard deviation), `pi` is a vector of C mixing proportions, and `h` is the bandwidth for kernel estimation.

Details

As of version 1.1.0, this function can only be used for a two-component mixture-of-regressions model with independent identically distributed errors. Assuming $C = 2$, the model is defined as follows:

$$f_{Y|\boldsymbol{X}}(y, \boldsymbol{x}, \boldsymbol{\theta}, g) = \sum_{j=1}^{C} \pi_j g(y - \boldsymbol{x}^\top \boldsymbol{\beta}_j).$$

Here, $\boldsymbol{\theta} = (\pi_1, ..., \pi_{C-1}, \boldsymbol{\beta}_1^\top, \cdots, \boldsymbol{\beta}_C^\top)$, and $g(\cdot)$ represents identical unspecified density functions. The bandwidth of the kernel density estimation is calculated adaptively using the `bw.SJ` function from the 'stats' package, which implements the method of Sheather & Jones (1991) for bandwidth selection based on pilot estimation of derivatives. This function employs weighted least square estimation for β in the M-step (Hunter and Young, 2012), where the weight is the posterior probability of an observation belonging to each component.

Value

A list containing the following elements:

`posterior`	posterior probabilities of each observation belonging to each component.
`beta`	estimated regression coefficients.
`tau`	estimated precision parameter, the inverse of standard deviation.
`pi`	estimated mixing proportions.
`h`	bandwidth used for the kernel estimation.

References

Hunter, D. R., and Young, D. S. (2012). Semiparametric mixtures of regressions. Journal of Nonparametric Statistics, 24(1), 19-38.

Ma, Y., Wang, S., Xu, L., and Yao, W. (2021). Semiparametric mixture regression with unspecified error distributions. Test, 30, 429-444.

See Also

kdeem, kdeem.h, bw.SJ for bandwidth calculation, and regmixEM for initial value calculation.

Examples

```
# See examples for the `kdeem' function.
```

---

mixLogconc — *Clustering with Mixtures of Log-concave Distributions using EM Algorithm (Univariate)*

Description

'mixLogconc' is used to estimate the parameters of a mixture of univariate log-concave distributions.

Usage

```
mixLogconc(x, C = 2, ini = NULL, nstart = 20, tol = 1e-05)
```

Arguments

x	an n by 1 data matrix where n is the number of observations.
C	number of mixture components. Default is 2.
ini	initial value for the EM algorithm. Default value is NULL, which obtains the initial value using the EMnormal function. It can be a list with the form of list(pi, mu, sigma), where pi is a 1 by C matrix of mixing proportions, mu is a C by 1 matrix of component means, and sigma is a p by p by 1 array of standard deviations or covariance matrices of C mixture components.
nstart	number of initializations to try. Default is 20.
tol	stopping criteria (threshold value) for the EM algorithm. Default is 1e-05.

Value

A list containing the following elements:

loglik	final log-likelihood.
pi	estimated mixing proportions.
f	component densities at x.

References

Chang, G. T., and Walther, G. (2007). Clustering with mixtures of log-concave distributions. Computational Statistics & Data Analysis, 51(12), 6242-6251.

Hu, H., Wu, Y., and Yao, W. (2016). Maximum likelihood estimation of the mixture of log-concave densities. Computational Statistics & Data Analysis, 101, 137-147.

See Also

EMnormal, mixLogconcHD

Examples

```
set.seed(4)
x = matrix(rnorm(100, 2, sqrt(2)), nrow = 100)
x[1:60] = x[1:60] + 5
EMlogc = mixLogconc(x, C = 2)
```

---

mixLogconcHD *Clustering with Mixtures of Log-concave Distributions using EM Algorithm (Multivariate)*

Description

'mixLogconcHD' is used to estimate the parameters of a mixture of multivariate log-concave distributions. The correlation structure among components is calculated by the normal copula.

Usage

```
mixLogconcHD(x, C, ini = NULL, nstart = 20, tol = 1e-05, maxiter = 100)
```

Arguments

`x`	an n by p data matrix where n is the number of observations and p is the dimension of the data.
`C`	number of mixture components.
`ini`	initial value for the EM algorithm. Default value is NULL, which obtains the initial value using the `EMnormal` function. It can be a list with the form of `list(pi, mu, sigma)`, where `pi` is a 1 by C matrix of mixing proportions, `mu` is a C by p matrix of component means, and `sigma` is a p by p by C array of standard deviations or covariance matrices of `C` mixture components.
`nstart`	number of initializations to try. Default is 20.
`tol`	stopping criteria (threshold value) for the EM algorithm. Default is 1e-05.
`maxiter`	maximum number of iterations for the EM algorithm. Default is 100.

Value

A list containing the following elements:

`loglik`	final log-likelihood.
`pi`	estimated mixing proportions.
`f`	component densities at x.
`sigma`	estimated standard deviation or covariance matrix.

References

Chang, G. T., and Walther, G. (2007). Clustering with mixtures of log-concave distributions. Computational Statistics & Data Analysis, 51(12), 6242-6251.

Hu, H., Wu, Y., and Yao, W. (2016). Maximum likelihood estimation of the mixture of log-concave densities. Computational Statistics & Data Analysis, 101, 137-147.

See Also

mixLogconc

Examples

```
x = mvtnorm::rmvnorm(100, c(0, 0), matrix(c(2, 1, 1, 2), nrow = 2))
x = matrix(x, nrow = 100)
x[1:60, ] = x[1:60, ] + 5
EMlogc = mixLogconcHD(x, C = 2)
```

---

mixMPHD *Semiparametric Mixture Model by Minimizing Profile Hellinger Distance*

Description

'mixMPHD' provides an efficient and robust estimation of a mixture of unknown location-shifted symmetric distributions using a semiparamatric method (Wu et al., 2017). As of version 1.1.0, 'mixMPHD' supports a two-component model, which is defined as

$$h(x; \boldsymbol{\theta}, f) = \pi f(x - \mu_1) + (1 - \pi) f(x - \mu_2),$$

where $\boldsymbol{\theta} = (\pi, \mu_1, \mu_2)^\top$ is the parameter to estimate, f is an unknown density function that is symmetric at zero. The parameters are estimated by minimizing the profile Hellinger distance (MPHD) between the parametric model and a non-parametric density estimate.

Usage

```
mixMPHD(x, sigma.known = NULL, ini = NULL)
```

Arguments

`x`	a vector of observations.
`sigma.known`	standard deviation of one component (if known). Default is NULL.
`ini`	initial values for the parameters. Default is NULL, which obtains the initial values using the `mixOnekn` function. It can be a list with the form of `list(mu, pi, sigma)`, where `mu` is a vector of component means, `pi` is a vector of component mixing proportions, `sigma` is a vector of component standard deviations.

Value

A list containing the following elements:

`lik`	final likelihood.
`pi`	estimated mixing proportion.
`sigma`	estimated component standard deviation. Only returned when `sigma.known` is not provided.
`mu`	estimated component mean.
`run`	total number of iterations after convergence.

References

Wu, J., Yao, W., and Xiang, S. (2017). Computation of an efficient and robust estimator in a semiparametric mixture model. Journal of Statistical Computation and Simulation, 87(11), 2128-2137.

See Also

`mixOnekn` for initial value calculation.

Examples

```
# Model: X ~ 0.3*N(0, 1) + 0.7*N(3, 1)
set.seed(4)
n = 100
p = 0.3
n1 = rbinom(1, n, p)
sigma1 = 1
sigma2 = 1
x1 = rnorm(n1, mean = 0, sd = sigma1)
x2 = rnorm(n - n1, mean = 3, sd = sigma2)
x = c(x1, x2)
ini = mixOnekn(x, sigma1)
mixMPHDest = mixMPHD(x, sigma1, ini = ini)
```

`mixnorm` *Parameter Estimation for Uni- or Multivariate Normal Mixture Models*

Description

‘mixnorm’ is used to estimate parameters of a normal mixture model with equal variance. The function supports both one-dimensional and multi-dimensional data.

Usage

```
mixnorm(x, C = 2, sigma.known = NULL, ini = NULL, tol = 1e-05)
```

Arguments

`x`	an n by p matrix of observations where n is the number of observations and s is the dimension of data.
`C`	number of mixture components. Default is 2.
`sigma.known`	a vector or matrix of component standard deviations. Default is NULL, which means the standard deviations are unknown.
`ini`	initial values for the parameters. Default is NULL, which randomly sets the initial values using the given observations. If specified, it can be a list with the form of `list(mu, pi, sigma)`, where `mu` is a vector of C component means, `pi` is a vector of C mixing proportions, and `sigma` is a vector of C component standard deviations (this element is only needed when `sigma.known` is not given).
`tol`	stopping criteria for the algorithm. Default is 1e-05.

Value

A list containing the following elements:

`mu`	estimated component means.
`sigma`	estimated component standard deviations. Only returned when `sigma.known` is not specified.
`pi`	estimated mixing proportions.
`p`	matrix containing estimated classification probabilities where the (i, j)th element is the probability of the jth observation belonging to the ith component.
`lik`	final likelihood.

See Also

`complh`, `distlat`

Examples

```
# See examples for the `complh' function.
```

`mixOnekn` *Two-component Normal Mixture Estimation with One Known Component*

Description

'mixOnekn' is used for the estimation of the following two-component mixture model:

$$h(x; \boldsymbol{\theta}, f) = \pi f(x - \mu_1) + (1 - \pi) f(x - \mu_2),$$

where $\boldsymbol{\theta} = (\pi, \mu_1, \mu_2)^\top$ is the parameter to estimate, f is an unknown density function that is symmetric at zero. The parameters are estimated by assuming f is the normal density and the first component has a mean of 0. This function can be used to obtain initial values for the `mixMPHD` function.

Usage

```
mixOnekn(x, sigma.known = NULL)
```

Arguments

`x`	a vector of observations.
`sigma.known`	standard deviation of the first component (if known). Default is NULL, which calculates the component standard deviations using the given observations `x`.

Value

A list containing the following elements:

`mu`	estimated 2 component means, where the first mean is 0.
`sigma`	estimated 2 component standard deviations.
`pi`	estimated 2 mixing proportions.
`lik`	final likelihood.

See Also

`mixMPHD`

Examples

```
# see examples for the `mixMPHD' function.
```

`mixpf` *Profile Likelihood Method for Normal Mixture with Unequal Variance*

Description

'mixpf' is used to estimate the following C-component univariate normal mixture model, using the profile likelihood method (Yao, 2010), with the assumption that the ratio of the smallest variance to the largest variance is k:

$$f(x;\boldsymbol{\theta}) = \sum_{j=1}^{C} \pi_j \phi(x;\mu_j,\sigma_j^2),$$

where $\boldsymbol{\theta} = (\pi_1,\mu_1,\sigma_1,..,\pi_C,\mu_C,\sigma_C)^\top$ is the parameter to estimate, $\phi(\cdot;\mu,\sigma^2)$ is the normal density with a mean of μ and a standard deviation of σ, and π's are mixing proportions that sum up to 1. Once the results are obtained, one can also find the maximum likelihood estimate (MLE) of k by plotting the likelihood vs. k for different k values and finding the maximum interior mode in the likelihood. See examples below.

Usage

```
mixpf(x, k = 0.5, C = 2, nstart = 20)
```

Arguments

`x`	a vector of observations.
`k`	ratio of the smallest variance to the largest variance. Default is 0.5.
`C`	number of mixture components. Default is 2.
`nstart`	number of initializations to try. Default is 20.

Value

A list containing the following elements:

`mu`	vector of estimated component means.
`sigma`	vector of estimated component standard deviations.
`pi`	vector of estimated mixing proportions.
`lik`	final likelihood.

References

Yao, W. (2010). A profile likelihood method for normal mixture with unequal variance. Journal of Statistical Planning and Inference, 140(7), 2089-2098.

Examples

```
set.seed(4)
n = 100
u = runif(n, 0, 1)
x2 = (u <= 0.3) * rnorm(n, 0, 0.5) + (u > 0.3) * rnorm(n, 1.5, 1)

# please set ngrid to 200 to get a smooth likelihood curve
ngrid = 5
grid = seq(from = 0.01, to = 1, length = ngrid)
likelihood = numeric()
for(i in 1:ngrid){
  k = grid[i]
  est = mixpf(x2, k)
  lh = est$lik
  likelihood[i] = lh
}

# visualize likelihood to find the best k
plot(grid, likelihood, type = "l", lty = 2, xlab = "k", ylab = "profile log-likelihood")
```

mixreg *MLE of Mixture Regression with Normal Errors*

Description

'mixreg' provides the MLE estimates of a mixture of regression models with normal errors. The result from this function can be used as initial values of the mixregRM2 function.

Usage

```
mixreg(x, y, C = 2, nstart = 20, tol = 1e-05)
```

Arguments

x	an n by p data matrix where n is the number of observations and p is the number of explanatory variables. The intercept term will automatically be added to the data.
y	an n-dimensional vector of response variable.
C	number of mixture components. Default is 2.
nstart	number of initializations to try. Default is 20.
tol	stopping criteria (threshold value) for the EM algorithm. Default is 1e-05.

Value

A list containing the following elements:

pi	C-dimensional vector of estimated mixing proportions.
beta	C by (p + 1) matrix of estimated regression coefficients.
sigma	C-dimensional vector of estimated standard deviations.
lik	final likelihood.
run	total number of iterations after convergence.

See Also

mixregRM2

Examples

```
data(tone)
y = tone$tuned
x = tone$stretchratio
k = 160
x[151:k] = 0
y[151:k] = 5
est = mixreg(x, y, 2, nstart = 1)
```

mixregBisq *Robust EM Algorithm For Mixture of Linear Regression Based on Bisquare Function*

Description

'mixregBisq' is used to robustly estimate the parameters of a mixture regression model using the bisquare function based on multiple initial values (Bai et al., 2012). The solution is the mode of the solutions obtained from all initial values.

Usage

```
mixregBisq(x, y, C = 2, nstart = 20)
```

Arguments

x	an n by p data matrix where n is the number of observations and p is the number of explanatory variables. The intercept term will automatically be added to the data.
y	an n-dimensional vector of response variable.
C	number of mixture components. Default is 2.
nstart	number of initializations to try. Default is 20.

Value

A list containing the following element:

pi	C-dimensional vector of estimated mixing proportions.
beta	C by (p + 1) matrix of estimated regression coefficients.
sigma	C-dimensional vector of estimated standard deviations.

References

Bai, X., Yao, W., and Boyer, J. E. (2012). Robust fitting of mixture regression models. Computational Statistics & Data Analysis, 56(7), 2347-2359.

Examples

```
data(tone)
y = tone$tuned
x = tone$stretchratio
k = 160
x[151:k] = 0
y[151:k] = 5
est_bi = mixregBisq(x, y, 2, nstart = 20)
```

`mixregLap` *Robust Mixture Regression with Laplace Distribution*

Description

'mixregLap' provides robust estimation for a mixture of linear regression models by assuming that the error terms follow the Laplace distribution (Song et al., 2014).

Usage

```
mixregLap(x, y, C = 2, nstart = 20, tol = 1e-05)
```

Arguments

`x`	an n by p matrix of observations (one observation per row). The intercept will be automatically added to `x`.
`y`	an n-dimensional vector of response variable.
`C`	number of mixture components. Default is 2.
`nstart`	number of initializations to try. Default is 20.
`tol`	stopping criteria (threshold value) for the EM algorithm. Default is 1e-05.

Value

A list containing the following elements:

`beta`	C by (p + 1) matrix of estimated regression coefficients.
`sigma`	C-dimensional vector of estimated component standard deviations.
`pi`	C-dimensional vector of estimated mixing proportions.
`lik`	final likelihood.
`run`	total number of iterations after convergence.

References

Song, W., Yao, W., and Xing, Y. (2014). Robust mixture regression model fitting by Laplace distribution. Computational Statistics & Data Analysis, 71, 128-137.

See Also

`mixregT` for robust estimation with t-distribution.

Examples

```
data(tone)
y = tone$tuned          # length(y) = 160
x = tone$stretchratio   # length(x) = 160
k = 160
x[151:k] = 0
y[151:k] = 5
est_lap = mixregLap(x, y, 2)
```

---

mixregPvary *Mixture of Regression Models with Varying Mixing Proportions*

Description

'mixregPvary' is used to estimate a mixture of regression models with varying proportions:

$$Y|_{\boldsymbol{x},Z=z} \sim \sum_{c=1}^{C} \pi_c(z) N(\boldsymbol{x}^\top \boldsymbol{\beta}_c, \sigma_c^2).$$

The varying proportions are estimated using a local constant regression method (kernel regression).

Usage

```
mixregPvary(x, y, C = 2, z = NULL, u = NULL, h = NULL,
            kernel = c("Gaussian", "Epanechnikov"), ini = NULL)
```

Arguments

`x`	an n by p matrix of explanatory variables. The intercept will be automatically added to x.
`y`	an n-dimensional vector of response variable.
`C`	number of mixture components. Default is 2.
`z`	a vector of a variable with varying-proportions. It can be any of the variables in `x`. Default is NULL, and the first variable in `x` will be used.
`u`	a vector of grid points for the local constant regression method to estimate the proportions. If NULL (default), 100 equally spaced grid points are automatically generated between the minimum and maximum values of z.
`h`	bandwidth for kernel density estimation. If NULL (default), the bandwidth is calculated based on the method of Botev et al. (2010).
`kernel`	character, determining the kernel function used in local constant method: `Gaussian` or `Epanechnikov`. Default is `Gaussian`.
`ini`	initial values for the parameters. Default is NULL, which obtains the initial values using the `regmixEM` function of the 'mixtools' package. If specified, it can be a list with the form of `list(pi, beta, var)`, where `pi` is a vector of length C of mixing proportions, `beta` is a (p + 1) by C matrix for component regression coefficients, and `var` is a vector of length C of component variances.

Value

A list containing the following elements:

`pi_u`	length(u) by C matrix of estimated mixing proportions at grid points.
`pi_z`	n by C matrix of estimated mixing proportions at z.
`beta`	(p + 1) by C matrix of estimated component regression coefficients.
`var`	C-dimensional vector of estimated component variances.
`loglik`	final log-likelihood.

References

Huang, M. and Yao, W. (2012). Mixture of regression models with varying mixing proportions: a semiparametric approach. Journal of the American Statistical Association, 107(498), 711-724.

Botev, Z. I., Grotowski, J. F., and Kroese, D. P. (2010). Kernel density estimation via diffusion. The Annals of Statistics, 38(5), 2916-2957.

See Also

`mixregPvaryGen`

Examples

```
n = 100
C = 2
u = seq(from = 0, to = 1, length = 100)
true_beta = cbind(c(4, - 2), c(0, 3))
true_var = c(0.09, 0.16)
data = mixregPvaryGen(n, C)
x = data$x
y = data$y
est = mixregPvary(x, y, C, z = x, u, h = 0.08)
```

`mixregPvaryGen` *Varying Proportion Mixture Data Generator*

Description

'mixregPvaryGen' is used to generate a mixture of normal distributions with varying proportions:

$$Y|_{\boldsymbol{x},Z=z} \sim \sum_{c=1}^{C} \pi_c(z) N(\boldsymbol{x}^\top \boldsymbol{\beta}_c, \sigma_c^2).$$

See `mixregPvary` for details.

Usage

```
mixregPvaryGen(n, C = 2)
```

Arguments

`n`	number of observations.
`C`	number of mixture components. Default is 2.

Value

A list containing the following elements:

`x`	vector of length n.
`y`	vector of length n.
`true_p`	n by C matrix of probabilities of an observations belonging to each component.

See Also

`mixregPvary`

Examples

```
mixregPvaryGen(n = 100, C = 2)
```

mixregRM2 — *Robust Mixture Regression with Thresholding-Embedded EM Algorithm for Penalized Estimation*

Description

A robust mixture regression model that simultaneously conducts outlier detection and robust parameter estimation. It uses a sparse, case-specific, and scale-dependent mean-shift mixture model parameterization (Yu et al., 2017):

$$f(y_i|\boldsymbol{x}_i, \boldsymbol{\theta}, \boldsymbol{\gamma}_i) = \sum_{j=1}^{C} \pi_j \phi(y_i; \boldsymbol{x}^\top \boldsymbol{\beta}_j + \gamma_{ij}\sigma_j, \sigma_j^2),$$

$i = 1, \cdots, n$, where C is the number of components in the model, $\boldsymbol{\theta} = (\pi_1, \boldsymbol{\beta}_1, \sigma_1, .., \pi_C, \boldsymbol{\beta}_C, \sigma_C)^\top$ is the parameter to estimate, and $\boldsymbol{\gamma}_i = (\gamma_{i1}, ..., \gamma_{iC})^\top$ is a vector of mean-shift parameter for the ith observation.

Usage

```
mixregRM2(x, y, C = 2, ini = NULL, nstart = 20, tol = 1e-02, maxiter = 50,
          method = c("HARD", "SOFT"), sigma.const = 0.001, lambda = 0.001)
```

Arguments

`x`	an n by p data matrix where n is the number of observations and p is the number of explanatory variables. The intercept term will automatically be added to the data.
`y`	an n-dimensional vector of response variable.
`C`	number of mixture components. Default is 2.
`ini`	initial values for the parameters. Default is NULL, which obtains the initial values using the `mixreg` function. It can be a list with the form of `list(pi, beta, sigma, gamma)`, where `pi` is a vector of C mixing proportions, `beta` is a C by (p + 1) matrix for regression coefficients of C components, `sigma` is a vector of C standard deviations, and `gamma` is a vector of C mean shift values.
`nstart`	number of initializations to try. Default is 20.
`tol`	stopping criteria (threshold value) for the EM algorithm. Default is 1e-02.
`maxiter`	maximum number of iterations for the EM algorithm. Default is 50.
`method`	character, determining which threshold method to use: `HARD` or `SOFT`. Default is `HARD`. See details.
`sigma.const`	constraint on the ratio of minimum and maximum values of sigma. Default is 0.001.
`lambda`	tuning parameter in the penalty term. It can be found based on BIC. See Yu et al. (2017) for more details.

Details

The parameters are estimated by maximizing the corresponding penalized log-likelihood function using an EM algorithm. The thresholding rule involes the estimation of γ_{ij} corresponding to different penalty:

- Soft threshold: $\hat{\gamma}_{ij} = sgn(\epsilon_{ij})(|\epsilon_{ij}| - \lambda^*_{ij})_+)$, corresponding to the l_1 penalty.
- Hard threshold: $\hat{\gamma}_{ij} = \epsilon_{ij} I(|\epsilon_{ij}| > \lambda^*_{ij}))$, corresponding to the l_0 penalty.

Here, $\epsilon_{ij} = (y_i - \boldsymbol{x}_i^\top \boldsymbol{\beta_j})/\sigma_j$ and $(\cdot)_+ = \max(\cdot, 0)$. Also, λ^*_{ij} is taken as $\lambda/p_{ij}^{(k+1)}$ for soft threshold and $\lambda/\sqrt{p_{ij}^{(k+1)}}$ for hard threshold.

Value

A list containing the following elements:

`pi`	C-dimensional vector of estimated mixing proportions.
`beta`	C by (p + 1) matrix of estimated regression coefficients.
`sigma`	C-dimensional vector of estimated standard deviations.
`gamma`	n-dimensional vector of estimated mean shift values.
`posterior`	n by C matrix of posterior probabilities of each observation belonging to each component.
`run`	total number of iterations after convergence.

References

Yu, C., Yao, W., and Chen, K. (2017). A new method for robust mixture regression. Canadian Journal of Statistics, 45(1), 77-94.

See Also

mixreg for initial value calculation.

Examples

```
data(tone)
y = tone$tuned
x = tone$stretchratio
k = 160
x[151:k] = 0
y[151:k] = 5
est_RM2 = mixregRM2(x, y, lambda = 1)
```

---

mixregT *Robust Mixture Regression with T-distribution*

Description

‘mixregT’ provides a robust estimation for a mixture of linear regression models by assuming that the error terms follow the t-distribution (Yao et al., 2014). The degrees of freedom is adaptively estimated.

Usage

```
mixregT(x, y, C = 2, maxdf = 30, nstart = 20, tol = 1e-05)
```

Arguments

x	an n by p data matrix where n is the number of observations and p is the number of explanatory variables. The intercept term will automatically be added to the data.
y	an n-dimensional vector of response variable.
C	number of mixture components. Default is 2.
maxdf	maximum degrees of freedom for the t-distribution. Default is 30.
nstart	number of initializations to try. Default is 20.
tol	threshold value (stopping criteria) for the EM algorithm. Default is 1e-05.

Value

A list containing the following elements:

`pi`	C-dimensional vector of estimated mixing proportions.
`beta`	C by (p + 1) matrix of estimated regression coefficients.
`sigma`	C-dimensional vector of estimated standard deviations.
`lik`	final likelihood.
`df`	estimated degrees of freedom of the t-distribution.
`run`	total number of iterations after convergence.

References

Yao, W., Wei, Y., and Yu, C. (2014). Robust mixture regression using the t-distribution. Computational Statistics & Data Analysis, 71, 116-127.

See Also

`mixregLap` for robust estimation with Laplace distribution.

Examples

```
data(tone)
y = tone$tuned
x = tone$stretchratio
k = 160
x[151:k] = 0
y[151:k] = 5
est_t = mixregT(x, y, 2, nstart = 20, tol = 0.1)
```

`mixregTrim` *Robust Regression Estimator Using Trimmed Likelihood*

Description

'mixregTrim' is used for robust regression estimation of a mixture model using the trimmed likelihood estimator (Neykov et al., 2007). It trims the data to reduce the impact of outliers on the model.

Usage

```
mixregTrim(x, y, C = 2, keep = 0.95, nstart = 20)
```

Arguments

`x`	an n by p data matrix where n is the number of observations and p is the number of explanatory variables. The intercept term will automatically be added to the data.
`y`	an n-dimensional vector of response variable.
`C`	number of mixture components. Default is 2.
`keep`	proportion of data to be kept after trimming, ranging from 0 to 1. Default is 0.95.
`nstart`	number of initializations to try. Default is 20.

Value

A list containing the following elements:

`pi`	C-dimensional vector of estimated mixing proportions.
`beta`	C by (p + 1) matrix of estimated regression coefficients.
`sigma`	C-dimensional vector of estimated standard deviations.
`lik`	final likelihood.

References

Neykov, N., Filzmoser, P., Dimova, R., and Neytchev, P. (2007). Robust fitting of mixtures using the trimmed likelihood estimator. Computational Statistics & Data Analysis, 52(1), 299-308.

Examples

```
data(tone)
y = tone$tuned
x = tone$stretchratio
k = 160
x[151:k] = 0
y[151:k] = 5
est_TLE = mixregTrim(x, y, 2, 0.95, nstart = 1)
```

`mixScale` *Continuous Scale Mixture Approach for Normal Scale Mixture Model*

Description

'mixScale' is used to estimate a two-component continuous normal scale mixture model, based on a backfitting method (Xiang et al., 2016):

$$p(x;\boldsymbol{\theta},f) = \pi f_1(x-\mu_1) + (1-\pi)f_2(x-\mu_2),$$

where $\boldsymbol{\theta} = (\pi,\mu_1,\mu_2)$. Here, f is assumed to be a member of $\mathcal{F} = \left\{f(x)\middle| \int \frac{1}{\sigma}\phi(x/\sigma)dQ(\sigma)\right\}$, where $\phi(x)$ is the standard normal density and Q is an unspecified probability measure on positive real numbers.

Usage

```
mixScale(x, ini = NULL, maxiter = 100)
```

Arguments

`x`	a vector of observations.
`ini`	initial values for the parameters. Default is NULL, which obtains the initial values using the `mixnorm` function. If specified, it can be a list with the form of `list(pi, mu, sigma)`, where `pi` is a vector of 2 mixing proportions, `mu` is a vector of 2 component means, and `sigma` is a vector of 2 component (common) standard deviations.
`maxiter`	maximum number of iterations for the EM algorithm. Default is 100.

Value

A list containing the following elements:

`mu`	estimated component means.
`pi`	estimated mixing proportions.
`suppQ`	support of Q.
`weightQ`	weight of Q corresponding to initial standard deviations.
`loglik`	final log-likelihood.
`run`	number of iterations after convergence.

References

Xiang, S., Yao, W., and Seo, B. (2016). Semiparametric mixture: Continuous scale mixture approach. Computational Statistics & Data Analysis, 103, 413-425.

See Also

`mixnorm` for initial value calculation.

Examples

```
require(quadprog)

#-------------------------------------------------------------------------------------------#
# Example 1: simulation
#-------------------------------------------------------------------------------------------#
n = 10
mu = c(-2.5, 0)
sd = c(0.8, 0.6)
pi = c(0.3, 0.7)
set.seed(2023)
n1 = rbinom(n, 1, pi[1])
x = c(rnorm(sum(n1), mu[1], sd[1]), rnorm(n - sum(n1), mu[2], sd[2]))
ini = list(pi = pi, mu = mu, sigma = sd)
```

```
out = mixScale(x, ini)

#--------------------------------------------------------------------------------------#
# Example 2: elbow data
#--------------------------------------------------------------------------------------#
ini = mixnorm(elbow)
res = mixScale(elbow, ini)
```

---

mixTest *Goodness of Fit Test for Finite Mixture Models*

Description

‘mixTest’ is used to perform a goodness-of-fit test for finite mixture models (Wichitchan et al., 2019). It returns five types of goodness-of-fit statistics and determines the number of components in the mixture model based on the Kolmogorov-Smirnov (KS) statistic. The test is performed using bootstrapping.

Usage

```
mixTest(x, alpha = 0.10, C.max = 10, nboot = 500, nstart = 5)
```

Arguments

`x`	a vector of observations.
`alpha`	significance level of the test.
`C.max`	maximum number of mixture components considered in the test. The test is performed for 2 to `C.max` components.
`nboot`	number of bootstrap resampling. Default is 500.
`nstart`	number of initializations to try. Default if 5.

Value

A list containing the following elements:

`GOFstats`	vector of test statistics calculated from data, in the order of `c(ks, cvm, kui, wat, ad)`.
`ks`	vector of the Kolmogorov-Smirnov (KS) statistic for each bootstrap sample.
`cvm`	vector of the Cramer-Von Mises statistic for each bootstrap sample.
`kui`	vector of the Kuiper’s statistic for each bootstrap sample.
`wat`	vector of the Watson statistic for each bootstrap sample.
`ad`	vector of the Anderson Darling statistic for each bootstrap sample.
`result`	vector of test results based on the KS statistic. If the kth element in the vector is 1, k-component mixture model is significant based on the KS statistic; If 0, otherwise. See examples for details.

References

Wichitchan, S., Yao, W., and Yang, G. (2019). Hypothesis testing for finite mixture models. Computational Statistics & Data Analysis, 132, 180-189.

Examples

```
n = 100
mu = c(-2.5, 0)
sd = c(0.8, 0.6)
n1 = rbinom(n, 1, 0.3)
x = c(rnorm(sum(n1), mu[1], sd[1]), rnorm(n - sum(n1), mu[2], sd[2]))

# The result shows that two-component mixture model is statistically significant based on the KS.
out = mixTest(x, alpha = 0.10, C.max = 10, nboot = 500, nstart = 5)
```

`NBA` *NBA data*

Description

The data contains four descriptive statistics for all 95 guards for the 1992-1993 season. There are many ways to measure the (statistical) performance of guards in the NBA. Of interest is how the the player's height (Height), minutes per game (MPG), and free throw percentage (FTP) affect points per game (PPM).

Usage

```
NBA
```

Format

A data frame containing 95 observations and the following 4 variables.

Height: height of the player.

MPG: minutes per game.

FTP: free throw percentage.

PPM: points per game.

References

Chatterjee, S., Handcock, M. S., and Simonoff, J. S. (1995). A casebook for a first course in statistics and data analysis (No. 04; QA276. 12, C4.). New York: Wiley.

Xiang, S. and Yao, W. (2020). Semiparametric mixtures of regressions with single-index for model based clustering. Advances in Data Analysis and Classification, 14(2), 261-292.

ROE *ROE data*

Description

The data contains a total of 2110 Chinese listed companies on their Return on Equity (ROE), which represents the amount of net income returned as a percentage of shareholders' equity. ROE is an important index used to measure a corporation's profitability and is also a useful indicator for fundamental analysts to assess the value of stocks.

Usage

```
ROE
```

Format

A data frame containing 2110 observations.

References

Huang, M., Wang, S., Wang, H., and Jin, T. (2018). Maximum smoothed likelihood estimation for a class of semiparametric Pareto mixture densities. Statistics and Its Interface, 11(1), 31-40.

semimrBin *Semiparametric Mixture of Binomial Regression with a Degenerate Component with Time-Varying Proportion and Time-Varying Success Probability*

Description

'semimrBin' is used for semiparametric estimation of a mixture of binomial distributions with one degenerate component, with time-varying proportions and time-varying success probability (Cao and Yao, 2012).

Usage

```
semimrBin(t, x, N, tg = NULL, tune = 1, tol = 1e-02)
```

Arguments

t	a vector of time variable along which $w(t)$ and $p(t)$ vary. See details for the explanations of those notations.
x	a vector of observed number of successes. The length of t and x must be the same.
N	a scalar, specifying the number of trials for the Binomial distribution.

`tg`	grid points of time used in the kernel regression for the estimation of $w(t)$ and $p(t)$. Default is NULL, and 100 equally spaced grid points will automatically generated using the minimum and maximum values of `t`.
`tune`	a scalar related to the bandwidth selection and local estimation. Default is 1. If greater than 0.2, the bandwidth is found based on the method in Cao and Yao (2012). If smaller than or equal to 0.2, this value is used as the percentage of data included in local estimation.
`tol`	stopping criteria for the algorithm.

Details

The semiparametric mixture of binomial regression model is as follows:

$$w(t) \times B(N, p(t)) + (1 - w(t)) \times B(N, 0),$$

where $B(N, p)$ is the probability mass function of a binomial distribution with the number of trials N and the success probability p. Here, the second component is a degenerate distribution with mass 1 on 0. The time-varying proportion $w(t)$ and success probability $p(t)$ for the binomial components are estimated by the kernel regression with some bandwidth.

Value

A list containing the following elements:

`pt`	estimated time-varying success probabilities for the first component.
`wt`	estimated time-varying proportions for the first component.
`h`	bandwidth for the kernel regression. The bandwidth calculation can be found in Section 4 of Cao and Yao (2012).

References

Cao, J. and Yao, W. (2012). Semiparametric mixture of binomial regression with a degenerate component. Statistica Sinica, 27-46.

See Also

`semimrBinOne`, `semimrBinFull`

Examples

```
n = 100
tg = seq(from = 0, to = 1, length.out = 50)
t = seq(from = 0, to = 1, length.out = n)
pt = 0.5 * (1 - cos(2 * pi * t))
b = rbinom(n, 1, 0.2)
y = apply(X = matrix(pt), 1, rbinom, n = 1, size = 7)
y = ifelse(b == 1, 0, y)
ft = semimrBin(t = t, x = y, N = 7, tg = tg)
```

`semimrBinFull` *Semiparametric Mixture of Binomial Regression with a Degenerate Component with Constant Proportion and Time-Varying Success Probability (Backfitting)*

Description

'semimrBinFull' implements the backfitting method (Cao and Yao, 2012) for semiparametric estimation of a mixture of binomial distributions with one degenerate component, with constant proportion and time-varying success probability p.

Usage

```
semimrBinFull(t, x, N, tg = NULL, tune = 1, tol = 1e-02)
```

Arguments

`t`	a vector of time variable along which $p(t)$ varies.
`x`	a vector of observed number of successes. The length of `t` and `x` must be the same.
`N`	a scalar, specifying the number of trials for the Binomial distribution.
`tg`	grid points of time used in the kernel regression for the estimation of $p(t)$. Default is NULL, and 100 equally spaced grid points will automatically generated using the minimum and maximum values of `t`.
`tune`	a scalar, specifying the percentage of data included in local estimation. related to the bandwidth selection and local estimation. Default is 1.
`tol`	stopping criteria for the algorithm.

Details

The semiparametric mixture of binomial regression model is as follows:

$$w \times (N, p(t)) + (1 - w) \times B(N, 0),$$

where $B(N, p)$ is the probability mass function of a binomial distribution with the number of trials N and the success probability p. Here, the second component is a degenerate distribution with mass 1 on 0. The time-varying success probability $p(t)$ for the binomial components are estimated by the kernel regression using a full iterative backfitting procedure with some bandwidth.

Value

A list containing the following elements:

`pt`	estimated time-varying success probabilities for the first component.
`w`	estimated constant proportion for the first component.
`h`	bandwidth for the kernel regression. The bandwidth calculation can be found in Section 4 of Cao and Yao (2012).

References

Cao, J. and Yao, W. (2012). Semiparametric mixture of binomial regression with a degenerate component. Statistica Sinica, 27-46.

See Also

semimrBin, semimrBinOne

Examples

```
nobs = 50
tobs = seq(from = 0, to = 1, length.out = nobs)
pi1Tru = 0.4
ptTru = 0.3 * (1.5 + cos(2 * pi * tobs))
nfine = nobs
tfine = seq(from = 0, to = 1, length.out = nfine)
b = rbinom(nobs, size = 1, pi1Tru)
yobs = apply(X = matrix(ptTru), 1, rbinom, n = 1, size = 7)
yobs = ifelse(b == 1, 0, yobs)
ftfull = semimrBinFull(t = tobs, x = yobs, N = 7, tg = tfine)
```

---

semimrBinOne — *Semiparametric Mixture of Binomial Regression with a Degenerate Component with Constant Proportion and Time-Varying Success Probability (One-step Backfitting)*

Description

'semimrBinOne' implements the one-step backfitting method (Cao and Yao, 2012) for semiparametric estimation of a mixture of binomial distributions with one degenerate component, with constant proportion and time-varying success probability.

Usage

```
semimrBinOne(t, x, N, tg = NULL, tune = 1, tol = 1e-02)
```

Arguments

t	a vector of time variable along which $p(t)$ varies.
x	a vector of observed number of successes. The length of t and x must be the same.
N	a scalar, specifying the number of trials for the Binomial distribution.
tg	grid points of time used in the kernel regression for the estimation of $p(t)$. Default is NULL, and 100 equally spaced grid points will automatically generated using the minimum and maximum values of t.
tune	a scalar, specifying the percentage of data included in local estimation. related to the bandwidth selection and local estimation. Default is 1.
tol	stopping criteria for the algorithm.

Details

The semiparametric mixture of binomial regression model is as follows:

$$w \times B(N, p(t)) + (1 - w) \times B(N, 0),$$

where $B(N, p)$ is the probability mass function of a binomial distribution with the number of trials N and the success probability p. Here, the second component is a degenerate distribution with mass 1 on 0. The time-varying success probability $p(t)$ for the binomial components are estimated by the kernel regression using one-step estimation for faster computation with some bandwidth.

Value

A list containing the following elements:

`pt`	estimated time-varying success probabilities for the first component.
`w`	estimated constant proportion for the first component.
`h`	bandwidth for the kernel regression. The bandwidth calculation can be found in Section 4 of Cao and Yao (2012).

References

Cao, J. and Yao, W. (2012). Semiparametric mixture of binomial regression with a degenerate component. Statistica Sinica, 27-46.

See Also

`semimrBin`, `semimrBinFull`

Examples

```
nobs = 50
tobs = seq(from = 0, to = 1, length.out = nobs)
pi1Tru = 0.4
ptTru = 0.3 * (1.5 + cos(2 * pi * tobs))
nfine = nobs
tfine = seq(from = 0, to = 1, length.out = nfine)
b = rbinom(nobs, size = 1, pi1Tru)
yobs = apply(X = matrix(ptTru), 1, rbinom, n = 1, size = 7)
yobs = ifelse(b == 1, 0, yobs)
ftonestep = semimrBinOne(t = tobs, x = yobs, N = 7, tg = tfine)
```

`semimrFull` *Semiparametric Mixture Regression Models with Single-index Proportion and Fully Iterative Backfitting*

Description

Assume that $\boldsymbol{x} = (\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n)$ is an n by p matrix and $Y = (Y_1, \cdots, Y_n)$ is an n-dimensional vector of response variable. The conditional distribution of Y given $\boldsymbol{x}$ can be written as:

$$f(y|\boldsymbol{x}, \boldsymbol{\alpha}, \pi, m, \sigma^2) = \sum_{j=1}^{C} \pi_j(\boldsymbol{\alpha}^\top \boldsymbol{x})\phi(y|m_j(\boldsymbol{\alpha}^\top \boldsymbol{x}), \sigma_j^2(\boldsymbol{\alpha}^\top \boldsymbol{x})).$$

'semimrFull' is used to estimate the mixture of single-index models described above, where $\phi(y|m_j(\boldsymbol{\alpha}^\top \boldsymbol{x}), \sigma_j^2(\boldsymbol{\alpha}^\top \boldsymbol{x}))$ represents the normal density with a mean of $m_j(\boldsymbol{\alpha}^\top \boldsymbol{x})$ and a variance of $\sigma_j^2(\boldsymbol{\alpha}^\top \boldsymbol{x})$, and $\pi_j(\cdot), \mu_j(\cdot), \sigma_j^2(\cdot)$ are unknown smoothing single-index functions capable of handling high-dimensional non-parametric problem. This function employs kernel regression and a fully iterative backfitting (FIB) estimation procedure (Xiang and Yao, 2020).

Usage

```
semimrFull(x, y, h = NULL, coef = NULL, ini = NULL, grid = NULL, maxiter = 100)
```

Arguments

`x`	an n by p matrix of observations where n is the number of observations and p is the number of explanatory variables.
`y`	an n-dimensional vector of response values.
`h`	bandwidth for the kernel regression. Default is NULL, and the bandwidth is computed in the function by cross-validation.
`coef`	initial value of $\boldsymbol{\alpha}^\top$ in the model, which plays a role of regression coefficient in a regression model. Default is NULL, and the value is computed in the function by sliced inverse regression (Li, 1991).
`ini`	initial values for the parameters. Default is NULL, which obtains the initial values, assuming a linear mixture model. If specified, it can be a list with the form of `list(pi, mu, var)`, where `pi` is a vector of mixing proportions, `mu` is a vector of component means, and `var` is a vector of component variances.
`grid`	grid points at which nonparametric functions are estimated. Default is NULL, which uses the estimated mixing proportions, component means, and component variances as the grid points after the algorithm converges.
`maxiter`	maximum number of iterations. Default is 100.

Value

A list containing the following elements:

`pi`	matrix of estimated mixing proportions.
`mu`	estimated component means.
`var`	estimated component variances.
`coef`	estimated regression coefficients.
`run`	total number of iterations after convergence.

References

Xiang, S. and Yao, W. (2020). Semiparametric mixtures of regressions with single-index for model based clustering. Advances in Data Analysis and Classification, 14(2), 261-292.

Li, K. C. (1991). Sliced inverse regression for dimension reduction. Journal of the American Statistical Association, 86(414), 316-327.

See Also

`semimrOne`, `sinvreg` for initial value calculation of $\boldsymbol{\alpha}^\top$.

Examples

```
xx = NBA[, c(1, 2, 4)]
yy = NBA[, 3]
x = xx/t(matrix(rep(sqrt(diag(var(xx))), length(yy)), nrow = 3))
y = yy/sd(yy)
ini_bs = sinvreg(x, y)
ini_b = ini_bs$direction[, 1]
est = semimrFull(x[1:50, ], y[1:50], h = 0.3442, coef = ini_b)
```

semimrGen *Semiparametric Mixture Data Generator*

Description

'semimrGen' is used to generate data for a two-component semiparametric mixture of regression models:

$$pm_1(x) + (1-p)m_2(x),$$

where $m_1(x) = 4 - \sin(2\pi x)$ and $m_2(x) = 1.5 + \cos(3\pi x)$. This function is used in the examples for the `semimrLocal` and `semimrGlobal` functions. See the examples for details.

Usage

```
semimrGen(n, p = 0.5, var = c(.1, .1), u)
```

Arguments

`n`	a scalar, specifying the number of observations in x.
`p`	a scalar, specifying the probability of an observation belonging to the first component, i.e., p in the model.
`var`	a vector of variances of observations for the two components.
`u`	a vector of grid points for x. If some specific explanatory variable are needed, create a vector and assign to `u`.

Value

A list containing the following elements:

`x`	vector of length n, which represents the explanatory variable that is randomly generated from Uniform(0,1).
`y`	vector of length n, which represent the response variable that is generated based on the mean functions $m_1(x)$ and $m_2(x)$, with the addition of normal errors having a mean of 0 and a standard deviation specified by the user.
`true_mu`	n by 2 matrix containing the values of $m_1(x)$ and $m_2(x)$ at x.
`true_mu_u`	length(u) by 2 matrix containing the values of $m_1(x)$ and $m_2(x)$ at u.

See Also

`semimrLocal`, `semimrGlobal`, `semimrBinFull`

Examples

```
n = 100
u = seq(from = 0, to = 1, length = 100)
true_p = c(0.3, 0.7)
true_var = c(0.09, 0.16)
out = semimrGen(n = n, p = true_p[1], var = true_var, u = u)
```

`semimrGlobal` *Semiparametric Mixtures of Nonparametric Regressions with Global EM-type Algorithm*

Description

'semimrGlobal' is used to estimate a mixture of regression models, where the mixing proportions and variances remain constant, but the component regression functions are smooth functions $(m(\cdot))$ of a covariate. The model is expressed as follows:

$$\sum_{j=1}^{C} \pi_j \phi(y|m(x_j), \sigma_j^2).$$

This function provides the one-step backfitting estimate using the global EM-type algorithm (GEM) (Xiang and Yao, 2018). As of version 1.1.0, this function supports a two-component model.

Usage

```
semimrGlobal(x, y, u = NULL, h = NULL, ini = NULL)
```

Arguments

x	a vector of covariate values.
y	a vector of response values.
u	a vector of grid points for spline method to estimate the proportions. If NULL (default), 100 equally spaced grid points are automatically generated between the minimum and maximum values of x.
h	bandwidth for the nonparametric regression. If NULL (default), the bandwidth is calculated based on the method of Botev et al. (2010).
ini	initial values for the parameters. Default is NULL, which obtains the initial values using regression spline approximation. If specified, it can be a list with the form of `list(pi, mu, var)`, where `pi` is a vector of length 2 of mixing proportions, `mu` is a length(x) by 2 matrix of component means, and `var` is a vector of length 2 of component variances.

Value

A list containing the following elements:

pi	vector of length 2 of estimated mixing proportions.
mu	length(x) by 2 matrix of estimated mean functions at x, $m(x)$.
mu_u	length(u) by 2 matrix of estimated mean functions at grid point u, $m(u)$.
var	vector of length 2 estimated component variances.
lik	final likelihood.

References

Xiang, S. and Yao, W. (2018). Semiparametric mixtures of nonparametric regressions. Annals of the Institute of Statistical Mathematics, 70, 131-154.

Botev, Z. I., Grotowski, J. F., and Kroese, D. P. (2010). Kernel density estimation via diffusion. The Annals of Statistics, 38(5), 2916-2957.

See Also

semimrLocal, semimrGen

Examples

```
# produce data that matches the description using semimrGen function
# true_mu = (4 - sin(2 * pi * x), 1.5 + cos(3 * pi * x))
n = 100
u = seq(from = 0, to = 1, length = 100)
true_p = c(0.3, 0.7)
true_var = c(0.09, 0.16)
```

```
out = semimrGen(n, true_p[1], true_var, u)

x = out$x
y = out$y
true_mu = out$true_mu
true = list(true_p = true_p, true_mu = true_mu, true_var = true_var)

# estimate parameters using semimrGlobal function.
est = semimrGlobal(x, y)
```

---

semimrLocal — *Semiparametric Mixtures of Nonparametric Regressions with Local EM-type Algorithm*

Description

'semimrLocal' is used to estimate a mixture of regression models, where the mixing proportions and variances remain constant, but the component regression functions are smooth functions ($m(\cdot)$) of a covariate. The model is expressed as follows:

$$\sum_{j=1}^{C} \pi_j \phi(y|m(x_j), \sigma_j^2).$$

This function provides the one-step backfitting estimate using the local EM-type algorithm (LEM) (Xiang and Yao, 2018). As of version 1.1.0, this function supports a two-component model.

Usage

```
semimrLocal(x, y, u = NULL, h = NULL, ini = NULL)
```

Arguments

`x`	a vector of covariate values.
`y`	a vector of response values.
`u`	a vector of grid points for spline method to estimate the proportions. If NULL (default), 100 equally spaced grid points are automatically generated between the minimum and maximum values of x.
`h`	bandwidth for the nonparametric regression. If NULL (default), the bandwidth is calculated based on the method of Botev et al. (2010).
`ini`	initial values for the parameters. Default is NULL, which obtains the initial values using regression spline approximation. If specified, it can be a list with the form of `list(pi, mu, var)`, where `pi` is a vector of length 2 of mixing proportions, `mu` is a length(x) by 2 matrix of component means with length(x) rows, and `var` is a vector of length 2 of variances.

Value

A list containing the following elements:

pi	vector of length 2 of estimated mixing proportions.
mu	length(x) by 2 matrix of estimated mean functions at x, $m(x)$.
mu_u	length(u) by 2 matrix of estimated mean functions at grid point u, $m(u)$.
var	vector of length 2 estimated component variances.
lik	final likelihood.

References

Xiang, S. and Yao, W. (2018). Semiparametric mixtures of nonparametric regressions. Annals of the Institute of Statistical Mathematics, 70, 131-154.

Botev, Z. I., Grotowski, J. F., and Kroese, D. P. (2010). Kernel density estimation via diffusion. The Annals of Statistics, 38(5), 2916-2957.

See Also

semimrGlobal, semimrGen

Examples

```
# produce data that matches the description using semimrGen function
# true_mu = (4 - sin(2 * pi * x), 1.5 + cos(3 * pi * x))
n = 100
u = seq(from = 0, to = 1, length = 100)
true_p = c(0.3, 0.7)
true_var = c(0.09, 0.16)
out = semimrGen(n, true_p[1], true_var, u)

x = out$x
y = out$y
true_mu = out$true_mu
true = list(true_p = true_p, true_mu = true_mu, true_var = true_var)

# estimate parameters using semimrLocal function.
est = semimrLocal(x, y)
```

semimrOne *Semiparametric Mixture Regression Models with Single-index and One-step Backfitting*

Description

Assume that $\boldsymbol{x} = (\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n)$ is an n by p matrix and $Y = (Y_1, \cdots, Y_n)$ is an n-dimensional vector of response variable. The conditional distribution of Y given $\boldsymbol{x}$ can be written as:

$$f(y|\boldsymbol{x}, \boldsymbol{\alpha}, \pi, m, \sigma^2) = \sum_{j=1}^{C} \pi_j(\boldsymbol{\alpha}^\top \boldsymbol{x})\phi(y|m_j(\boldsymbol{\alpha}^\top \boldsymbol{x}), \sigma_j^2(\boldsymbol{\alpha}^\top \boldsymbol{x})).$$

'semimrFull' is used to estimate the mixture of single-index models described above, where $\phi(y|m_j(\boldsymbol{\alpha}^\top \boldsymbol{x}), \sigma_j^2(\boldsymbol{\alpha}^\top \boldsymbol{x}))$ represents the normal density with a mean of $m_j(\boldsymbol{\alpha}^\top \boldsymbol{x})$ and a variance of $\sigma_j^2(\boldsymbol{\alpha}^\top \boldsymbol{x})$, and $\pi_j(\cdot), \mu_j(\cdot), \sigma_j^2(\cdot)$ are unknown smoothing single-index functions capable of handling high-dimensional non-parametric problem. This function employs kernel regression and a one-step estimation procedure (Xiang and Yao, 2020).

Usage

```
semimrOne(x, y, h, coef = NULL, ini = NULL, grid = NULL)
```

Arguments

`x`	an n by p matrix of observations where n is the number of observations and p is the number of explanatory variables.
`y`	a vector of response values.
`h`	bandwidth for the kernel regression. Default is NULL, and the bandwidth is computed in the function by cross-validation.
`coef`	initial value of $\boldsymbol{\alpha}^\top$ in the model, which plays a role of regression coefficient in a regression model. Default is NULL, and the value is computed in the function by sliced inverse regression (Li, 1991).
`ini`	initial values for the parameters. Default is NULL, which obtains the initial values, assuming a linear mixture model. If specified, it can be a list with the form of `list(pi, mu, var)`, where `pi` is a vector of mixing proportions, `mu` is a vector of component means, and `var` is a vector of component variances.
`grid`	grid points at which nonparametric functions are estimated. Default is NULL, which uses the estimated mixing proportions, component means, and component variances as the grid points after the algorithm converges.

Value

A list containing the following elements:

`pi`	estimated mixing proportions.
`mu`	estimated component means.
`var`	estimated component variances.
`coef`	estimated regression coefficients.

References

Xiang, S. and Yao, W. (2020). Semiparametric mixtures of regressions with single-index for model based clustering. Advances in Data Analysis and Classification, 14(2), 261-292.

Li, K. C. (1991). Sliced inverse regression for dimension reduction. Journal of the American Statistical Association, 86(414), 316-327.

See Also

`semimrFull`, `sinvreg` for initial value calculation of $\boldsymbol{\alpha}^\top$.

Examples

```
xx = NBA[, c(1, 2, 4)]
yy = NBA[, 3]
x = xx/t(matrix(rep(sqrt(diag(var(xx))), length(yy)), nrow = 3))
y = yy/sd(yy)
ini_bs = sinvreg(x, y)
ini_b = ini_bs$direction[, 1]

# used a smaller sample for a quicker demonstration of the function
set.seed(123)
est_onestep = semimrOne(x[1:50, ], y[1:50], h = 0.3442, coef = ini_b)
```

`sinvreg` *Dimension Reduction Based on Sliced Inverse Regression*

Description

'sinvreg' is used in the examples for the `semimrFull` and `semimrOne` functions to obtain initial values based on sliced inverse regression (Li, 1991).

Usage

```
sinvreg(x, y, nslice = NULL)
```

Arguments

`x`	an n by p matrix of observations where n is the number of observations and p is the number of explanatory variables.
`y`	an n-dimentionsl vector of response values.
`nslice`	number of slices. Default is 10.

Value

A list containing the following elements:

`direction`	direction vector.
`reducedx`	reduced x.
`eigenvalue`	eigenvalues for reduced x.
`nobs`	number of observations within each slice.

References

Li, K. C. (1991). Sliced inverse regression for dimension reduction. Journal of the American Statistical Association, 86(414), 316-327.

See Also

`semimrFull`, `semimrOne`

Examples

```
# See examples for the 'semimrFull' function.
```

`tone` *Tone perception data*

Description

The data originates from an experiment of Cohen (1980) and has been analyzed in de Veaux (1989) and Viele and Tong (2002). In this experiment, a pure fundamental tone was played to a trained musician. To create different auditory conditions, electronically generated overtones were introduced, which were determined by a stretching ratio. When `stretchratio = 2.0`, it aligns with 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. The variable `tuned` gives the ratio of the adjusted tone to the fundamental tone. For example, `tuned = 2.0`, would be the correct tuning for all `stretchratio` values. This dataset comprises data collected from 150 trials conducted with the same musician. In the original study, data were gathered from an additional four musicians as well. The dataset and the description have been sourced from the `tonedata` of the ‘fpc’ package.

Usage

```
tone
```

Format

A data frame containing 150 observations and the following 2 variables.

stretchratio: electronically generated overtones added to a pure fundamental tone.

tuned: ratio of the adjusted tone to the fundamental tone.

Source

Original source:

Cohen, E. (1980). Inharmonic tone perception. Unpublished Ph. D. Dissertation, Stanford University.

R source:

Hennig C (2023). fpc: Flexible Procedures for Clustering. R package version 2.2-10, https://CRAN.R-project.org/package=fpc

Benaglia, T., Chauveau, D., Hunter, D. R., and Young, D. S. (2010). mixtools: an R package for analyzing mixture models. Journal of statistical software, 32, 1-29.

References

De Veaux, R. D. (1989). Mixtures of linear regressions. Computational Statistics & Data Analysis, 8(3), 227-245.

Viele, K. and Tong, B. (2002). Modeling with mixtures of linear regressions. Statistics and Computing, 12, 315-330.

Index