Package ‘BNPdensity’

January 20, 2025

Type Package

Title Ferguson-Klass Type Algorithm for Posterior Normalized Random Measures

Version 2023.3.8

Description Bayesian nonparametric density estimation modeling mixtures by a Ferguson-Klass type algorithm for posterior normalized random measures.

License GPL (>= 2)

Encoding UTF-8

LazyData yes

ByteCompile yes

RoxygenNote 7.2.3

Depends R(>= 3.5.0)

Maintainer Guillaume Kon Kam King <`guillaume.konkamking.work@gmail.com`>

Imports ggplot2, gridExtra, survival, coda, dplyr, tidyr, viridis

Suggests GreedyEPL, Rmpfr, gmp, knitr, rmarkdown, testthat, BNPmix

NeedsCompilation no

Author Julyan Arbel [ctb],
Ernesto Barrios [aut],
Guillaume Kon Kam King [aut, cre],
Antonio Lijoi [aut],
Luis E. Nieto-Barajas [aut],
Igor Prünster [aut]

Repository CRAN

Date/Publication 2023-03-24 15:10:02 UTC

Contents

`BNPdensity-package` *Bayesian nonparametric density estimation*

Description

This package performs Bayesian nonparametric density estimation for exact and censored data via a normalized random measure mixture model. The package allows the user to specify the mixture kernel, the mixing normalized measure and the choice of performing fully nonparametric mixtures on locations and scales, or semiparametric mixtures on locations only with common scale parameter. Options for the kernels are: two kernels with support in the real line (gaussian and double exponential), two more kernels in the positive line (gamma and lognormal) and one with bounded support (beta). The options for the normalized random measures are members of the class of normalized generalized gamma, which include the Dirichlet process, the normalized inverse gaussian process and the normalized stable process. The type of censored data handled by the package is right, left and interval.

Details

Package:	BNPdensity
Type:	Package
Version:	2016.10

Date:	2016-10-14
License:	GPL version 2 or later
LazyLoad:	yes

The package includes four main functions: MixNRMI1, MixNRMI2, MixNRMI1cens and MixN-RMI2cens which implement semiparametric and fully nonparametric mixtures for exact data, and semiparametric and fully nonparametric mixtures for censored data respectively. Additionally, the package includes several other functions required for sampling from conditional distributions in the MCMC implementation. These functions are intended for internal use only.

Author(s)

Barrios, E., Lijoi, A., Nieto-Barajas, L. E. and Prünster, I.; Contributor: Guillaume Kon Kam King.; Maintainer: Ernesto Barrios <ebarrios at itam.mx>

References

Barrios, E., Lijoi, A., Nieto-Barajas, L. E. and Prünster, I. (2013). Modeling with Normalized Random Measure Mixture Models. Statistical Science. Vol. 28, No. 3, 313-334.

Kon Kam King, G., Arbel, J. and Prünster, I. (2016). Species Sensitivity Distribution revisited: a Bayesian nonparametric approach. In preparation.

See Also

MixNRMI1, MixNRMI2, MixNRMI1cens, MixNRMI2cens

Examples

```
example(MixNRMI1)
example(MixNRMI2)
example(MixNRMI1cens)
example(MixNRMI2cens)
```

acidity *Acidity Index Dataset*

Description

Concerns an acidity index measured in a sample of 155 lakes in north-central Wisconsin.

Format

A real vector with 155 observations.

References

Crawford, S. L., DeGroot, M. H., Kadane, J. B. and Small, M. J. (1992). Modeling lake chemistry distributions: approximate Bayesian methods for estimating a finite mixture model. Technometrics, 34, 441-453.

Examples

```
data(acidity)
hist(acidity)
```

`add` *Add x and y*

Description

This is a helper function for use in Reduce() over a list of vectors

Usage

```
add(x, y)
```

Arguments

`x`	first argument of the sum
`y`	second argument of the sum

Value

x + y

`as.mcmc.multNRMI` *Convert the output of multMixNRMI into a coda mcmc object*

Description

Convert the output of multMixNRMI into a coda mcmc object

Usage

```
## S3 method for class 'multNRMI'
as.mcmc(x, ..., thinning_to = 1000, ncores = parallel::detectCores())
```

Arguments

`x`	Output of multMixNRMI.
`...`	Further arguments to be passed to specific methods
`thinning_to`	Final length of the chain after thinning.
`ncores`	Specify the number of cores to use in the conversion

Value

a coda::mcmc object

Examples

```
data(acidity)
out <- multMixNRMI1(acidity, parallel = TRUE, Nit = 10, ncores = 2)
coda::as.mcmc(out, ncores = 2)
```

`asNumeric_no_warning` *If the function Rmpfr::asNumeric returns a warning about inefficiency, silence it.*

Description

The function Rmpfr::asNumeric prints the following warning: In asMethod(object) : coercing "mpfr1" via "mpfr" (inefficient). It is not clear how to avoid it nor how to silence it, hence this function. A cleaner solution may be available at: https://stackoverflow.com/questions/4948361/how-do-i-save-warnings-and-errors-as-output-from-a-function/4952908#4952908

Usage

```
asNumeric_no_warning(x)
```

Arguments

`x`	An object of class Rmpfr::mpfr1

Value

a "numeric" number

comment_on_NRMI_type — *Comment on the NRMI process depending on the value of the parameters*

Description

Comment on the NRMI process depending on the value of the parameters

Usage

```
comment_on_NRMI_type(NRMI_param = list(Alpha = 1, Kappa = 0, Gamma = 0.4))
```

Arguments

`NRMI_param`	A named list of the form list("Alpha" = 1, "Kappa" = 0, "Gamma" = 0.4)

Value

A string containing a comment on the NRMI process

Examples

```
BNPdensity:::comment_on_NRMI_type(list("Alpha" = 1, "Kappa" = 0, "Gamma" = 0.4))
BNPdensity:::comment_on_NRMI_type(list("Alpha" = 1, "Kappa" = 0.1, "Gamma" = 0.4))
BNPdensity:::comment_on_NRMI_type(list("Alpha" = 1, "Kappa" = 0.1, "Gamma" = 0.5))
```

compute_optimal_clustering — *Compute the optimal clustering from an MCMC sample*

Description

Summarizes the posterior on all possible clusterings by an optimal clustering where optimality is defined as minimizing the posterior expectation of a specific loss function, the Variation of Information or Binder's loss function. Computation can be lengthy for large datasets, because of the large size of the space of all clusterings.

Usage

```
compute_optimal_clustering(fit, loss_type = "VI")
```

Arguments

`fit`	The fitted object, obtained from one of the MixNRMIx functions
`loss_type`	Defines the loss function to be used in the expected posterior loss minimization. Can be one of "VI" (Variation of Information), "B" (Binder's loss), "NVI" (Normalized Variation of Information) or "NID" (Normalized Information Distance). Defaults to "VI".

Value

A vector of integers with the same size as the data, indicating the allocation of each data point.

`compute_thinning_grid` *Compute the grid for thinning the MCMC chain*

Description

This function creates an real grid then rounds it. If the grid is fine enough, there is a risk that rounding ties, i.e. iteration which are kept twice. To avoid this, if the total number of iterations is smaller than twice the number of iterations desired after thinning, the chain is not thinned.

Usage

```
compute_thinning_grid(Nit, thinning_to = 10)
```

Arguments

`Nit`	Length of the MCMC chain
`thinning_to`	Desired number of iterations after thinning.

Value

an integer vector of the MCMC iterations retained.

`convert_to_mcmc` *Convert the output of multMixNRMI into a coda mcmc object*

Description

Convert the output of multMixNRMI into a coda mcmc object

Usage

```
convert_to_mcmc(fitlist, thinning_to = 1000, ncores = parallel::detectCores())
```

Arguments

`fitlist`	Output of multMixNRMI.
`thinning_to`	Final length of the chain after thinning.
`ncores`	Specify the number of cores to use in the conversion

Value

a coda::mcmc object

`cpo.multNRMI`	*Extract the Conditional Predictive Ordinates (CPOs) from a list of fitted objects*

Description

This function assumes that all chains have the same size. To allow for different chain sizes, care should be paid to proper weighting.

Usage

```
## S3 method for class 'multNRMI'
cpo(object, ...)
```

Arguments

`object`	A fit obtained through from the functions MixNRMI1/MixNRMI1cens
`...`	Further arguments to be passed to generic function, ignored at the moment

Value

A vector of Conditional Predictive Ordinates (CPOs)

Examples

```
data(acidity)
out <- multMixNRMI1(acidity, parallel = TRUE, Nit = 10, ncores = 2)
cpo(out)
```

`cpo.NRMI1`	*Extract the Conditional Predictive Ordinates (CPOs) from a fitted object*

Description

Extract the Conditional Predictive Ordinates (CPOs) from a fitted object

Usage

```
## S3 method for class 'NRMI1'
cpo(object, ...)
```

Arguments

`object`	A fit obtained through from the functions MixNRMI1/MixNRMI1cens
`...`	Further arguments to be passed to generic function, ignored at the moment

Value

A vector of Conditional Predictive Ordinates (CPOs)

Examples

```
data(acidity)
out <- MixNRMI1(acidity, Nit = 50)
cpo(out)
```

---

cpo.NRMI2 *Extract the Conditional Predictive Ordinates (CPOs) from a fitted object*

Description

Extract the Conditional Predictive Ordinates (CPOs) from a fitted object

Usage

```
## S3 method for class 'NRMI2'
cpo(object, ...)
```

Arguments

`object`	A fit obtained through from the function MixNRMI2/MixNRMI2cens
`...`	Further arguments to be passed to generic function, ignored at the moment

Value

A vector of Conditional Predictive Ordinates (CPOs)

Examples

```
data(acidity)
out <- MixNRMI2(acidity, Nit = 50)
cpo(out)
```

dist_name_k_index_converter

Convert distribution names to indices

Description

Convert distribution names to indices

Usage

```
dist_name_k_index_converter(distname)
```

Arguments

`distname`	a character representing the distribution name. Allowed names are "normal", "gamma", "beta", "exponential", "double exponential", "lognormal", "half-Cauchy", "half-normal", "half-student", "uniform" and "truncated normal", or their common abbreviations "norm", "exp", "lnorm", "halfcauchy", "halfnorm", "halft" and "unif".

Value

an index describing the distribution. 1 = Normal; 2 = Gamma; 3 = Beta; 4 = Double Exponential; 5 = Lognormal, 6 = Half-Cauchy, 7 = Half-normal, 8 = Half-Student, 9 = Uniform, 10 = Truncated normal

dt_

Non-standard student-t density

Description

Computes the density.

Usage

```
dt_(x, df, mean, sd)
```

Arguments

`x`	Numeric vector. Data set to which the density is evaluated.
`df`	Numeric constant. Degrees of freedom (> 0, maybe non-integer)
`mean`	Numeric constant. Location parameter.
`sd`	Positive numeric constant. Scale parameter.
	## The function is currently defined as function(x, df, mean, sd) dt((x - mean) / sd, df, ncp = 0) / sd

Details

For internal use

`enzyme` *Enzyme Dataset*

Description

Concerns the distribution of enzymatic activity in the blood, for an enzyme involved in the metabolism of carcinogenetic substances, among a group of 245 unrelated individuals.

Format

A data frame with 244 observations on the following variable:

list("enzyme") A numeric vector.

References

Bechtel, Y. C., Bonaiti-Pellie, C., Poisson, N., Magnette, J. and Bechtel, P.R. (1993). A population and family study of N-acetyltransferase using caffeine urinary metabolites. Clin. Pharm. Therp., 54, 134-141.

Examples

```
data(enzyme)
hist(enzyme)
```

`Enzyme1.out` *Fit of MixNRMI1 function to the enzyme dataset*

Description

This object contains the output when setting set.seed(150520) and running the function Enzyme1.out <- MixNRMI1(enzyme, Alpha = 1, Kappa = 0.007, Gama = 0.5, distr.k = "gamma", distr.p0 = "gamma", asigma = 1, bsigma = 1, Meps = 0.005, Nit = 5000, Pbi = 0.2)

Details

See function MixNRMI1

Examples

```
data(Enzyme1.out)
```

Enzyme2.out — *Fit of MixNRMI2 function to the enzyme dataset*

Description

This object contains the output when setting set.seed(150520) and running the function Enzyme2.out <- MixNRMI2(enzyme, Alpha = 1, Kappa = 0.007, Gama = 0.5, distr.k = "gamma", distr.py0 = "gamma", distr.pz0 = "gamma", mu.pz0 = 1, sigma.pz0 = 1, Meps = 0.005, Nit = 5000, Pbi = 0.2) See function MixNRMI2

Examples

```
data(Enzyme2.out)
```

expected_number_of_components_Dirichlet — *Computes the expected number of components for a Dirichlet process.*

Description

Computes the expected number of components for a Dirichlet process.

Usage

```
expected_number_of_components_Dirichlet(
  n,
  Alpha,
  ntrunc = NULL,
  silence = TRUE
)
```

Arguments

n	Number of data points
Alpha	Numeric constant. Total mass of the centering measure.
ntrunc	Level of truncation when computing the expectation. Defaults to n. If greater than n, it is fixed to n.
silence	Boolean. Whether to print the current calculation step for the Stable process, as the function can be long

Value

A real value which approximates the expected number of components

Reference: P. De Blasi, S. Favaro, A. Lijoi, R. H. Mena, I. Prünster, and M. Ruggiero, “Are Gibbs-type priors the most natural generalization of the Dirichlet process?,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 37, no. 2, pp. 212–229, 2015.

Examples

```
expected_number_of_components_Dirichlet(100, 1.2)
```

`expected_number_of_components_stable`
Computes the expected number of components for a stable process.

Description

Computes the expected number of components for a stable process.

Usage

```
expected_number_of_components_stable(n, Gama, ntrunc = NULL)
```

Arguments

`n`	Number of data points
`Gama`	Numeric constant. 0 <= Gama <=1.
`ntrunc`	Level of truncation when computing the expectation. Defaults to n. If greater than n, it is fixed to n.

Value

A real value of type mpfr1 which approximates the expected number of components

In spite of the high precision arithmetic packages used for in function, it can be numerically unstable for small values of Gama. This is because evaluating a sum with alternated signs, in the generalized factorial coefficients, is tricky. Reference: P. De Blasi, S. Favaro, A. Lijoi, R. H. Mena, I. Prünster, and M. Ruggiero, "Are gibbs-type priors the most natural generalization of the Dirichlet process?," IEEE Trans. Pattern Anal. Mach. Intell., vol. 37, no. 2, pp. 212–229, 2015.

Examples

```
expected_number_of_components_stable(100, 0.8)
```

`fill_sigmas` *Repeat the common scale parameter of a semiparametric model to match the dimension of the location parameters.*

Description

Repeat the common scale parameter of a semiparametric model to match the dimension of the location parameters.

Usage

```
fill_sigmas(semiparametric_fit)
```

Arguments

`semiparametric_fit`	The result of the fit, obtained through the function MixNRMI1.

Value

an adequate list of vectors of sigmas

`galaxy` *Galaxy Data Set*

Description

Velocities of 82 galaxies diverging from our own galaxy.

Format

A data frame with 82 observations on the following variable:

list("velocity") A numeric vector.

References

Roeder, K. (1990) "Density estimation with confidence sets exemplified by superclusters and voids in the galaxies". Journal of the American Statistical Association. 85, 617-624.

Examples

```
data(galaxy)
hist(galaxy)
```

`Galaxy1.out` *Fit of MixNRMI1 function to the galaxy dataset*

Description

This object contains the output when setting set.seed(150520) and running the function MixNRMI1(galaxy, Alpha = 1, Kappa = 0.015, Gama = 0.5, distr.k = "normal", distr.p0 = "gamma", asigma = 1, bsigma = 1, delta = 7, Meps = 0.005, Nit = 5000, Pbi = 0.2)

Details

See function MixNRMI1.

Examples

```
data(Galaxy1.out)
```

`Galaxy2.out` *Fit of MixNRMI2 function to the galaxy dataset*

Description

This object contains the output when setting set.seed(150520) and running the function Enzyme2.out <- MixNRMI2(x, Alpha = 1, Kappa = 0.007, Gama = 0.5, distr.k = "gamma", distr.py0 = "gamma", distr.pz0 = "gamma", mu.pz0 = 1, sigma.pz0 = 1, Meps = 0.005, Nit = 5000, Pbi = 0.2)

Details

See function MixNRMI2.

Examples

```
data(Galaxy2.out)
```

`give_kernel_name` *Gives the kernel name from the integer code*

Description

This function is used in the print methods for MixNRMI1, MixNRMI2, MixNRMI1cens, MixNRMI2cens, and all the multMixNRMIx versions

Usage

```
give_kernel_name(distr.k)
```

Arguments

`distr.k`	The distribution name for the kernel. Allowed names are "normal", "gamma", "beta", "double exponential", "lognormal" or their common abbreviations "norm", "exp", or an integer number identifying the mixture kernel: 1 = Normal; 2 = Gamma; 3 = Beta; 4 = Double Exponential; 5 = Lognormal.

Value

A character with the name of the distribution used as the kernel

Examples

```
BNPdensity:::give_kernel_name(4)
```

`GOFplots` *Plot Goodness of fits graphical checks for censored data*

Description

Plot Goodness of fits graphical checks for censored data

Usage

```
GOFplots(fit, qq_plot = FALSE, thinning_to = 500)
```

Arguments

`fit`	The result of the fit, obtained through the function MixNRMI1 or MixNRMI2, MixMRMI1cens or MixMRMI2cens
`qq_plot`	Whether to compute the QQ-plot
`thinning_to`	How many iterations to compute the mean posterior quantiles

Value

A density plot, a cumulative density plot with the Turnbull cumulative distribution, a percentile-percentile plot, and potentially a quantile-quantile plot.

Examples

```
set.seed(150520)
data(salinity)
out <- MixNRMI1cens(salinity$left, salinity$right, extras = TRUE, Nit = 100)
GOFplots(out)
```

`GOFplots_censored` *Plot Goodness of fits graphical checks for censored data*

Description

Plot Goodness of fits graphical checks for censored data

Usage

```
GOFplots_censored(fit, qq_plot = FALSE, thinning_to = 500)
```

Arguments

`fit`	The result of the fit, obtained through the function MixNRMI1 or MixNRMI2, MixMRMI1cens or MixMRMI2cens
`qq_plot`	Whether to compute the QQ-plot
`thinning_to`	How many iterations to compute the mean posterior quantiles

Value

A density plot, a cumulative density plot with the Turnbull cumulative distribution, and a percentile-percentile plot.

Examples

```
set.seed(150520)
data(salinty)
out <- MixNRMI1cens(salinity$left, salinity$right, extras = TRUE, Nit = 100)
BNPdensity:::GOFplots_censored(out)
```

GOFplots_noncensored *Plot Goodness of fits graphical checks for non censored data*

Description

Plot Goodness of fits graphical checks for non censored data

Usage

```
GOFplots_noncensored(fit, qq_plot = FALSE, thinning_to = 500)
```

Arguments

`fit`	The result of the fit, obtained through the function MixNRMI1 or MixNRMI2, MixMRMI1cens or MixMRMI2cens
`qq_plot`	Whether to compute the QQ-plot
`thinning_to`	How many iterations to compute the mean posterior quantiles

Value

A density plot with histogram, a cumulative density plot with the empirical cumulative distribution, and a percentile-percentile plot.

Examples

```
set.seed(150520)
data(acidity)
out <- MixNRMI1(acidity, extras = TRUE, Nit = 100)
BNPdensity:::GOFplots_noncensored(out)
```

grid_from_data *Create a plotting grid from censored or non-censored data.*

Description

Create a plotting grid from censored or non-censored data.

Usage

```
grid_from_data(data, npoints = 100)
```

Arguments

`data`	Input data from which to compute the grid.
`npoints`	Number of points on the grid.

Value

a vector containing the plotting grid

---

grid_from_data_censored

Create a plotting grid from censored data.

Description

Create a plotting grid from censored data.

Usage

```
grid_from_data_censored(data, npoints = 100)
```

Arguments

`data`	Censored input data from which to compute the grid.
`npoints`	Number of points on the grid.

Value

a vector containing the plotting grid

---

grid_from_data_noncensored

Create a plotting grid from non-censored data.

Description

Create a plotting grid from non-censored data.

Usage

```
grid_from_data_noncensored(data, npoints = 100)
```

Arguments

`data`	Non-censored input data from which to compute the grid.
`npoints`	Number of points on the grid.

Value

a vector containing the plotting grid

`is_censored` *Test if the data is censored*

Description

Test if the data is censored

Usage

```
is_censored(dat)
```

Arguments

`dat`	The dataset to be tested

Value

TRUE if the data is censored

Examples

```
data(salinity)
BNPdensity:::is_censored(salinity)
```

`is_semiparametric` *Tests if a fit is a semi parametric or nonparametric model.*

Description

Tests if a fit is a semi parametric or nonparametric model.

Usage

```
is_semiparametric(fit)
```

Arguments

`fit`	The result of the fit, obtained through the function MixNRMI1 or MixNRMI2.

Value

TRUE if the fit is a semiparametric model

Examples

```
set.seed(150520)
data(acidity)
x <- enzyme
out <- MixNRMI1(enzyme, extras = TRUE, Nit = 10)
BNPdensity:::is_semiparametric(out)
```

MixNRMI1 *Normalized Random Measures Mixture of Type I*

Description

Bayesian nonparametric estimation based on normalized measures driven mixtures for locations.

Usage

```
MixNRMI1(
  x,
  probs = c(0.025, 0.5, 0.975),
  Alpha = 1,
  Kappa = 0,
  Gama = 0.4,
  distr.k = "normal",
  distr.p0 = 1,
  asigma = 0.5,
  bsigma = 0.5,
  delta_S = 3,
  delta_U = 2,
  Meps = 0.01,
  Nx = 150,
  Nit = 1500,
  Pbi = 0.1,
  epsilon = NULL,
  printtime = TRUE,
  extras = TRUE,
  adaptive = FALSE
)
```

Arguments

x	Numeric vector. Data set to which the density is fitted.
probs	Numeric vector. Desired quantiles of the density estimates.
Alpha	Numeric constant. Total mass of the centering measure. See details.
Kappa	Numeric positive constant. See details.
Gama	Numeric constant. $0 \leq$ Gama ≤ 1. See details.

`distr.k`	The distribution name for the kernel. Allowed names are "normal", "gamma", "beta", "double exponential", "lognormal" or their common abbreviations "norm", "exp", or an integer number identifying the mixture kernel: 1 = Normal; 2 = Gamma; 3 = Beta; 4 = Double Exponential; 5 = Lognormal.
`distr.p0`	The distribution name for the centering measure. Allowed names are "normal", "gamma", "beta", or their common abbreviations "norm", "exp", or an integer number identifying the centering measure: 1 = Normal; 2 = Gamma; 3 = Beta.
`asigma`	Numeric positive constant. Shape parameter of the gamma prior on the standard deviation of the mixture kernel `distr.k`.
`bsigma`	Numeric positive constant. Rate parameter of the gamma prior on the standard deviation of the mixture kernel `distr.k`.
`delta_S`	Numeric positive constant. Metropolis-Hastings proposal variation coefficient for sampling sigma.
`delta_U`	Numeric positive constant. Metropolis-Hastings proposal variation coefficient for sampling the latent U.
`Meps`	Numeric constant. Relative error of the jump sizes in the continuous component of the process. Smaller values imply larger number of jumps.
`Nx`	Integer constant. Number of grid points for the evaluation of the density estimate.
`Nit`	Integer constant. Number of MCMC iterations.
`Pbi`	Numeric constant. Burn-in period proportion of Nit.
`epsilon`	Numeric constant. Extension to the evaluation grid range. See details.
`printtime`	Logical. If TRUE, prints out the execution time.
`extras`	Logical. If TRUE, gives additional objects: means, weights and Js.
`adaptive`	Logical. If TRUE, uses an adaptive MCMC strategy to sample the latent U (adaptive delta_U).

Details

This generic function fits a normalized random measure (NRMI) mixture model for density estimation (James et al. 2009). Specifically, the model assumes a normalized generalized gamma (NGG) prior for the locations (means) of the mixture kernel and a parametric prior for the common smoothing parameter sigma, leading to a semiparametric mixture model.

The details of the model are:

$$X_i|Y_i,\sigma \sim k(\cdot|Y_i,\sigma)$$

$$Y_i|P \sim P, \quad i=1,\dots,n$$

$$P \sim \text{NGG}(\texttt{Alpha, Kappa, Gama; P\_0})$$

$$\sigma \sim \text{Gamma(asigma, bsigma)}$$

where X_i's are the observed data, Y_i's are latent (location) variables, `sigma` is the smoothing parameter, `k` is a parametric kernel parameterized in terms of mean and standard deviation, `(Alpha,Kappa, Gama; P_0)` are the parameters of the NGG prior with `P_0` being the centering measure whose parameters are assigned vague hyper prior distributions, and `(asigma,bsigma)` are the hyper-parameters of the gamma prior on the smoothing parameter `sigma`. In particular: `NGG(Alpha,` 1, 0; P_0) defines a Dirichlet process; NGG(1,Kappa, 1/2; P_0) defines a Normalized inverse Gaussian process; and NGG(1, 0, Gama; P_0) defines a normalized stable process.

The evaluation grid ranges from min(x) - epsilon to max(x) + epsilon. By default epsilon=sd(x)/4.

Value

The function returns a MixNRMI1 object. It is based on a list with the following components:

xx	Numeric vector. Evaluation grid.
qx	Numeric array. Matrix of dimension Nx $\times$ $($length(probs) $+ 1)$ with the posterior mean and the desired quantiles input in probs.
cpo	Numeric vector of length(x) with conditional predictive ordinates.
R	Numeric vector of length(Nit*(1-Pbi)) with the number of mixtures components (clusters).
S	Numeric vector of length(Nit*(1-Pbi)) with the values of common standard deviation sigma.
U	Numeric vector of length(Nit*(1-Pbi)) with the values of the latent variable U.
Allocs	List of length(Nit*(1-Pbi)) with the clustering allocations.
means	List of length(Nit*(1-Pbi)) with the cluster means (locations). Only if extras = TRUE.
weights	List of length(Nit*(1-Pbi)) with the mixture weights. Only if extras = TRUE.
Js	List of length(Nit*(1-Pbi)) with the unnormalized weights (jump sizes). Only if extras = TRUE.
Nm	Integer constant. Number of jumps of the continuous component of the unnormalized process.
Nx	Integer constant. Number of grid points for the evaluation of the density estimate.
Nit	Integer constant. Number of MCMC iterations.
Pbi	Numeric constant. Burn-in period proportion of Nit.
procTime	Numeric vector with execution time provided by proc.time function.
distr.k	Integer corresponding to the kernel chosen for the mixture
data	Data used for the fit
NRMI_params	A named list with the parameters of the NRMI process

Warning

The function is computing intensive. Be patient.

Author(s)

Barrios, E., Kon Kam King, G., Lijoi, A., Nieto-Barajas, L.E. and Prüenster, I.

References

1.- Barrios, E., Lijoi, A., Nieto-Barajas, L. E. and Prünster, I. (2013). Modeling with Normalized Random Measure Mixture Models. Statistical Science. Vol. 28, No. 3, 313-334.

2.- James, L.F., Lijoi, A. and Prünster, I. (2009). Posterior analysis for normalized random measure with independent increments. Scand. J. Statist 36, 76-97.

See Also

`MixNRMI2`, `MixNRMI1cens`, `MixNRMI2cens`, `multMixNRMI1`

Examples

```
### Example 1
## Not run:
# Data
data(acidity)
x <- acidity
# Fitting the model under default specifications
out <- MixNRMI1(x)
# Plotting density estimate + 95% credible interval
plot(out)
### Example 2
set.seed(150520)
data(enzyme)
x <- enzyme
Enzyme1.out <- MixNRMI1(x, Alpha = 1, Kappa = 0.007, Gama = 0.5,
                         distr.k = "gamma", distr.p0 = "gamma",
                         asigma = 1, bsigma = 1, Meps=0.005,
                         Nit = 5000, Pbi = 0.2)
attach(Enzyme1.out)
# Plotting density estimate + 95% credible interval
plot(Enzyme1.out)
# Plotting number of clusters
par(mfrow = c(2, 1))
plot(R, type = "l", main = "Trace of R")
hist(R, breaks = min(R - 0.5):max(R + 0.5), probability = TRUE)
# Plotting sigma
par(mfrow = c(2, 1))
plot(S, type = "l", main = "Trace of sigma")
hist(S, nclass = 20, probability = TRUE, main = "Histogram of sigma")
# Plotting u
par(mfrow = c(2, 1))
plot(U, type = "l", main = "Trace of U")
hist(U, nclass = 20, probability = TRUE, main = "Histogram of U")
# Plotting cpo
par(mfrow = c(2, 1))
plot(cpo, main = "Scatter plot of CPO's")
boxplot(cpo, horizontal = TRUE, main = "Boxplot of CPO's")
print(paste("Average log(CPO)=", round(mean(log(cpo)), 4)))
print(paste("Median log(CPO)=", round(median(log(cpo)), 4)))
detach()
```

```
## End(Not run)

### Example 3
## Do not run
# set.seed(150520)
# data(galaxy)
# x <- galaxy
#  Galaxy1.out <- MixNRMI1(x, Alpha = 1, Kappa = 0.015, Gama = 0.5,
#                          distr.k = "normal", distr.p0 = "gamma",
#                          asigma = 1, bsigma = 1, delta = 7, Meps=0.005,
#                          Nit = 5000, Pbi = 0.2)

# The output of this run is already loaded in the package
# To show results run the following
# Data
data(galaxy)
x <- galaxy
data(Galaxy1.out)
attach(Galaxy1.out)
# Plotting density estimate + 95% credible interval
plot(Galaxy1.out)
# Plotting number of clusters
par(mfrow = c(2, 1))
plot(R, type = "l", main = "Trace of R")
hist(R, breaks = min(R - 0.5):max(R + 0.5), probability = TRUE)
# Plotting sigma
par(mfrow = c(2, 1))
plot(S, type = "l", main = "Trace of sigma")
hist(S, nclass = 20, probability = TRUE, main = "Histogram of sigma")
# Plotting u
par(mfrow = c(2, 1))
plot(U, type = "l", main = "Trace of U")
hist(U, nclass = 20, probability = TRUE, main = "Histogram of U")
# Plotting cpo
par(mfrow = c(2, 1))
plot(cpo, main = "Scatter plot of CPO's")
boxplot(cpo, horizontal = TRUE, main = "Boxplot of CPO's")
print(paste("Average log(CPO)=", round(mean(log(cpo)), 4)))
print(paste("Median log(CPO)=", round(median(log(cpo)), 4)))
detach()
```

`MixNRMI1cens` *Normalized Random Measures Mixture of Type I for censored data*

Description

Bayesian nonparametric estimation based on normalized measures driven mixtures for locations.

Usage

```
MixNRMI1cens(
  xleft,
  xright,
  probs = c(0.025, 0.5, 0.975),
  Alpha = 1,
  Kappa = 0,
  Gama = 0.4,
  distr.k = "normal",
  distr.p0 = "normal",
  asigma = 0.5,
  bsigma = 0.5,
  delta_S = 3,
  delta_U = 2,
  Meps = 0.01,
  Nx = 150,
  Nit = 1500,
  Pbi = 0.1,
  epsilon = NULL,
  printtime = TRUE,
  extras = TRUE,
  adaptive = FALSE
)
```

Arguments

`xleft`	Numeric vector. Lower limit of interval censoring. For exact data the same as xright
`xright`	Numeric vector. Upper limit of interval censoring. For exact data the same as xleft.
`probs`	Numeric vector. Desired quantiles of the density estimates.
`Alpha`	Numeric constant. Total mass of the centering measure. See details.
`Kappa`	Numeric positive constant. See details.
`Gama`	Numeric constant. $0 \leq$ `Gama` ≤ 1. See details.
`distr.k`	The distribution name for the kernel. Allowed names are "normal", "gamma", "beta", "double exponential", "lognormal" or their common abbreviations "norm", "exp", or an integer number identifying the mixture kernel: 1 = Normal; 2 = Gamma; 3 = Beta; 4 = Double Exponential; 5 = Lognormal.
`distr.p0`	The distribution name for the centering measure. Allowed names are "normal", "gamma", "beta", or their common abbreviations "norm", "exp", or an integer number identifying the centering measure: 1 = Normal; 2 = Gamma; 3 = Beta.
`asigma`	Numeric positive constant. Shape parameter of the gamma prior on the standard deviation of the mixture kernel `distr.k`.
`bsigma`	Numeric positive constant. Rate parameter of the gamma prior on the standard deviation of the mixture kernel `distr.k`.

`delta_S`	Numeric positive constant. Metropolis-Hastings proposal variation coefficient for sampling sigma.
`delta_U`	Numeric positive constant. Metropolis-Hastings proposal variation coefficient for sampling the latent U.
`Meps`	Numeric constant. Relative error of the jump sizes in the continuous component of the process. Smaller values imply larger number of jumps.
`Nx`	Integer constant. Number of grid points for the evaluation of the density estimate.
`Nit`	Integer constant. Number of MCMC iterations.
`Pbi`	Numeric constant. Burn-in period proportion of Nit.
`epsilon`	Numeric constant. Extension to the evaluation grid range. See details.
`printtime`	Logical. If TRUE, prints out the execution time.
`extras`	Logical. If TRUE, gives additional objects: means, weights and Js.
`adaptive`	Logical. If TRUE, uses an adaptive MCMC strategy to sample the latent U (adaptive delta_U).

Details

This generic function fits a normalized random measure (NRMI) mixture model for density estimation (James et al. 2009) with censored data. Specifically, the model assumes a normalized generalized gamma (NGG) prior for the locations (means) of the mixture kernel and a parametric prior for the common smoothing parameter sigma, leading to a semiparametric mixture model.

This function coincides with `MixNRMI1` when the lower (xleft) and upper (xright) censoring limits correspond to the same exact value.

The details of the model are:

$$X_i|Y_i, \sigma \sim k(\cdot|Y_i, \sigma)$$

$$Y_i|P \sim P, \quad i = 1, \dots, n$$

$$P \sim \text{NGG}(\texttt{Alpha, Kappa, Gama; P\_0})$$

$$\sigma \sim \text{Gamma(asigma, bsigma)}$$

where X_i's are the observed data, Y_i's are latent (location) variables, `sigma` is the smoothing parameter, `k` is a parametric kernel parameterized in terms of mean and standard deviation, `(Alpha,Kappa, Gama; P_0)` are the parameters of the NGG prior with `P_0` being the centering measure whose parameters are assigned vague hyper prior distributions, and `(asigma,bsigma)` are the hyperparameters of the gamma prior on the smoothing parameter `sigma`. In particular: `NGG(Alpha, 1, 0; P_0)` defines a Dirichlet process; `NGG(1,Kappa, 1/2; P_0)` defines a Normalized inverse Gaussian process; and `NGG(1, 0, Gama; P_0)` defines a normalized stable process.

The evaluation grid ranges from `min(x) - epsilon` to `max(x) + epsilon`. By default `epsilon=sd(x)/4`.

Value

The function returns a list with the following components:

`xx`	Numeric vector. Evaluation grid.
`qx`	Numeric array. Matrix of dimension `Nx` $\times$ (`length(probs)` $+ 1$) with the posterior mean and the desired quantiles input in `probs`.
`cpo`	Numeric vector of `length(x)` with conditional predictive ordinates.
`R`	Numeric vector of `length(Nit*(1-Pbi))` with the number of mixtures components (clusters).
`S`	Numeric vector of `length(Nit*(1-Pbi))` with the values of common standard deviation sigma.
`U`	Numeric vector of `length(Nit*(1-Pbi))` with the values of the latent variable U.
`Allocs`	List of `length(Nit*(1-Pbi))` with the clustering allocations.
`means`	List of `length(Nit*(1-Pbi))` with the cluster means (locations). Only if extras = TRUE.
`weights`	List of `length(Nit*(1-Pbi))` with the mixture weights. Only if extras = TRUE.
`Js`	List of `length(Nit*(1-Pbi))` with the unnormalized weights (jump sizes). Only if extras = TRUE.
`Nm`	Integer constant. Number of jumps of the continuous component of the unnormalized process.
`Nx`	Integer constant. Number of grid points for the evaluation of the density estimate.
`Nit`	Integer constant. Number of MCMC iterations.
`Pbi`	Numeric constant. Burn-in period proportion of `Nit`.
`procTime`	Numeric vector with execution time provided by `proc.time` function.
`distr.k`	Integer corresponding to the kernel chosen for the mixture
`data`	Data used for the fit
`NRMI_params`	A named list with the parameters of the NRMI process

Warning

The function is computing intensive. Be patient.

Author(s)

Barrios, E., Kon Kam King, G. and Nieto-Barajas, L.E.

References

1.- Barrios, E., Lijoi, A., Nieto-Barajas, L. E. and Prünster, I. (2013). Modeling with Normalized Random Measure Mixture Models. Statistical Science. Vol. 28, No. 3, 313-334.

2.- James, L.F., Lijoi, A. and Prünster, I. (2009). Posterior analysis for normalized random measure with independent increments. Scand. J. Statist 36, 76-97.

3.- Kon Kam King, G., Arbel, J. and Prünster, I. (2016). Species Sensitivity Distribution revisited: a Bayesian nonparametric approach. In preparation.

See Also

MixNRMI2, MixNRMI1cens, MixNRMI2cens, multMixNRMI1

Examples

```
### Example 1
## Not run:
# Data
data(acidity)
x <- acidity
# Fitting the model under default specifications
out <- MixNRMI1cens(x, x)
# Plotting density estimate + 95% credible interval
plot(out)

## End(Not run)

## Not run:
### Example 2
# Data
data(salinity)
# Fitting the model under default specifications
out <- MixNRMI1cens(xleft = salinity$left, xright = salinity$right, Nit = 5000)
# Plotting density estimate + 95% credible interval
attach(out)
plot(out)
# Plotting number of clusters
par(mfrow = c(2, 1))
plot(R, type = "l", main = "Trace of R")
hist(R, breaks = min(R - 0.5):max(R + 0.5), probability = TRUE)
detach()

## End(Not run)
```

MixNRMI2 *Normalized Random Measures Mixture of Type II*

Description

Bayesian nonparametric estimation based on normalized measures driven mixtures for locations and scales.

Usage

```
MixNRMI2(
  x,
  probs = c(0.025, 0.5, 0.975),
  Alpha = 1,
  Kappa = 0,
  Gama = 0.4,
  distr.k = "normal",
  distr.py0 = "normal",
  distr.pz0 = "gamma",
  mu.pz0 = 3,
  sigma.pz0 = sqrt(10),
  delta_S = 4,
  kappa = 2,
  delta_U = 2,
  Meps = 0.01,
  Nx = 150,
  Nit = 1500,
  Pbi = 0.1,
  epsilon = NULL,
  printtime = TRUE,
  extras = TRUE,
  adaptive = FALSE
)
```

Arguments

`x`	Numeric vector. Data set to which the density is fitted.
`probs`	Numeric vector. Desired quantiles of the density estimates.
`Alpha`	Numeric constant. Total mass of the centering measure. See details.
`Kappa`	Numeric positive constant. See details.
`Gama`	Numeric constant. $0 \leq Gama \leq 1$. See details.
`distr.k`	The distribution name for the kernel. Allowed names are "normal", "gamma", "beta", "double exponential", "lognormal" or their common abbreviations "norm", "exp", or an integer number identifying the mixture kernel: 1 = Normal; 2 = Gamma; 3 = Beta; 4 = Double Exponential; 5 = Lognormal.
`distr.py0`	The distribution name for the centering measure for locations. Allowed names are "normal", "gamma", "beta", or their common abbreviations "norm", "exp", or an integer number identifying the centering measure for locations: 1 = Normal; 2 = Gamma; 3 = Beta.
`distr.pz0`	The distribution name for the centering measure for scales. Allowed names are "gamma", or an integer number identifying the centering measure for scales: 2 = Gamma. For more options use `MixNRMI2cens`.
`mu.pz0`	Numeric constant. Prior mean of the centering measure for scales.
`sigma.pz0`	Numeric constant. Prior standard deviation of the centering measure for scales.
`delta_S`	Numeric positive constant. Metropolis-Hastings proposal variation coefficient for sampling the scales.
`kappa`	Numeric positive constant. Metropolis-Hastings proposal variation coefficient for sampling the location parameters.

`delta_U`	Numeric positive constant. Metropolis-Hastings proposal variation coefficient for sampling the latent U. If ‘adaptive=TRUE‘, ‘delta_U‘is the starting value for the adaptation.
`Meps`	Numeric constant. Relative error of the jump sizes in the continuous component of the process. Smaller values imply larger number of jumps.
`Nx`	Integer constant. Number of grid points for the evaluation of the density estimate.
`Nit`	Integer constant. Number of MCMC iterations.
`Pbi`	Numeric constant. Burn-in period proportion of `Nit`.
`epsilon`	Numeric constant. Extension to the evaluation grid range. See details.
`printtime`	Logical. If TRUE, prints out the execution time.
`extras`	Logical. If TRUE, gives additional objects: means, sigmas, weights and Js.
`adaptive`	Logical. If TRUE, uses an adaptive MCMC strategy to sample the latent U (adaptive delta_U).

Details

This generic function fits a normalized random measure (NRMI) mixture model for density estimation (James et al. 2009). Specifically, the model assumes a normalized generalized gamma (NGG) prior for both, locations (means) and standard deviations, of the mixture kernel, leading to a fully nonparametric mixture model.

The details of the model are:

$$X_i|Y_i, Z_i \sim k(\cdot|Y_i, Z_i)$$

$$(Y_i, Z_i)|P \sim P, i = 1, \dots, n$$

$$P \sim \text{NGG}(\texttt{Alpha, Kappa, Gama; P\_0})$$

where, X_i's are the observed data, (Y_i, Z_i)'s are bivariate latent (location and scale) vectors, `k` is a parametric kernel parameterized in terms of mean and standard deviation, `(Alpha,Kappa, Gama; P_0)` are the parameters of the NGG prior with a bivariate `P_0` being the centering measure with independent components, that is, $P_0(Y, Z) = P_0(Y) * P_0(Z)$. The parameters of `P_0(Y)` are assigned vague hyper prior distributions and `(mu.pz0,sigma.pz0)` are the hyper-parameters of `P_0(Z)`. In particular, `NGG(Alpha, 1, 0; P_0)` defines a Dirichlet process; `NGG(1, Kappa, 1/2;P_0)` defines a Normalized inverse Gaussian process; and `NGG(1, 0, Gama; P_0)` defines a normalized stable process. The evaluation grid ranges from `min(x) -epsilon` to `max(x) + epsilon`. By default `epsilon=sd(x)/4`.

Value

The function returns a list with the following components:

`xx`	Numeric vector. Evaluation grid.
`qx`	Numeric array. Matrix of dimension `Nx` $\times$ (`length(probs)` $+ 1$) with the posterior mean and the desired quantiles input in `probs`.
`cpo`	Numeric vector of `length(x)` with conditional predictive ordinates.
`R`	Numeric vector of `length(Nit*(1-Pbi))` with the number of mixtures components (clusters).

`U`	Numeric vector of `length(Nit*(1-Pbi))` with the values of the latent variable U.
`Allocs`	List of `length(Nit*(1-Pbi))` with the clustering allocations.
`means`	List of `length(Nit*(1-Pbi))` with the cluster means (locations). Only if extras = TRUE.
`sigmas`	Numeric vector of `length(Nit*(1-Pbi))` with the cluster standard deviations. Only if extras = TRUE.
`weights`	List of `length(Nit*(1-Pbi))` with the mixture weights. Only if extras = TRUE.
`Js`	List of `length(Nit*(1-Pbi))` with the unnormalized weights (jump sizes). Only if extras = TRUE.
`Nm`	Integer constant. Number of jumps of the continuous component of the unnormalized process.
`delta_Us`	List of `length(Nit*(1-Pbi))` with the sequence of adapted delta_U used in the MH step for the latent variable U.
`Nx`	Integer constant. Number of grid points for the evaluation of the density estimate.
`Nit`	Integer constant. Number of MCMC iterations.
`Pbi`	Numeric constant. Burn-in period proportion of `Nit`.
`procTime`	Numeric vector with execution time provided by `proc.time` function.
`distr.k`	Integer corresponding to the kernel chosen for the mixture
`data`	Data used for the fit
`NRMI_params`	A named list with the parameters of the NRMI process

Warning

The function is computing intensive. Be patient.

Author(s)

Barrios, Kon Kam King, G., E., Lijoi, A., Nieto-Barajas, L.E. and Prüenster, I.

References

1.- Barrios, E., Lijoi, A., Nieto-Barajas, L. E. and Prünster, I. (2013). Modeling with Normalized Random Measure Mixture Models. Statistical Science. Vol. 28, No. 3, 313-334.

2.- James, L.F., Lijoi, A. and Prünster, I. (2009). Posterior analysis for normalized random measure with independent increments. Scand. J. Statist 36, 76-97.

3.- Arbel, J., Kon Kam King, G., Lijoi, A., Nieto-Barajas, L.E. and Prüenster, I. (2021). BNPdensity: a package for Bayesian Nonparametric density estimation using Normalised Random Measures with Independent Increments.. Australian and New Zealand Journal of Statistics, to appear

See Also

`MixNRMI2`, `MixNRMI1cens`, `MixNRMI2cens`, `multMixNRMI1`

Examples

```
## Not run:
### Example 1
# Data
data(acidity)
x <- acidity
# Fitting the model under default specifications
out <- MixNRMI2(x)
# Plotting density estimate + 95% credible interval
plot(out)

## End(Not run)

### Example 2
## Do not run
# set.seed(150520)
# data(enzyme)
# x <- enzyme
#  Enzyme2.out <- MixNRMI2(x, Alpha = 1, Kappa = 0.007, Gama = 0.5,
#                           distr.k = "gamma", distr.py0 = "gamma",
#                           distr.pz0 = "gamma", mu.pz0 = 1, sigma.pz0 = 1, Meps=0.005,
#                           Nit = 5000, Pbi = 0.2)
# The output of this run is already loaded in the package
# To show results run the following
# Data
data(enzyme)
x <- enzyme
data(Enzyme2.out)
attach(Enzyme2.out)
# Plotting density estimate + 95% credible interval
plot(Enzyme2.out)
# Plotting number of clusters
par(mfrow = c(2, 1))
plot(R, type = "l", main = "Trace of R")
hist(R, breaks = min(R - 0.5):max(R + 0.5), probability = TRUE)
# Plotting u
par(mfrow = c(2, 1))
plot(U, type = "l", main = "Trace of U")
hist(U, nclass = 20, probability = TRUE, main = "Histogram of U")
# Plotting cpo
par(mfrow = c(2, 1))
plot(cpo, main = "Scatter plot of CPO's")
boxplot(cpo, horizontal = TRUE, main = "Boxplot of CPO's")
print(paste("Average log(CPO)=", round(mean(log(cpo)), 4)))
print(paste("Median log(CPO)=", round(median(log(cpo)), 4)))
detach()

### Example 3
## Do not run
# set.seed(150520)
# data(galaxy)
# x <- galaxy
#  Galaxy2.out <- MixNRMI2(x, Alpha = 1, Kappa = 0.015, Gama = 0.5,
#                         distr.k = "normal", distr.py0 = "gamma",
#                         distr.pz0 = "gamma", mu.pz0 = 1, sigma.pz0 = 1,  Meps=0.005,
#                         Nit = 5000, Pbi = 0.2)
# The output of this run is already loaded in the package
# To show results run the following
# Data
data(galaxy)
x <- galaxy
data(Galaxy2.out)
attach(Galaxy2.out)
# Plotting density estimate + 95% credible interval
plot(Galaxy2.out)
# Plotting number of clusters
par(mfrow = c(2, 1))
plot(R, type = "l", main = "Trace of R")
hist(R, breaks = min(R - 0.5):max(R + 0.5), probability = TRUE)
# Plotting u
par(mfrow = c(2, 1))
plot(U, type = "l", main = "Trace of U")
hist(U, nclass = 20, probability = TRUE, main = "Histogram of U")
# Plotting cpo
par(mfrow = c(2, 1))
plot(cpo, main = "Scatter plot of CPO's")
boxplot(cpo, horizontal = TRUE, main = "Boxplot of CPO's")
print(paste("Average log(CPO)=", round(mean(log(cpo)), 4)))
print(paste("Median log(CPO)=", round(median(log(cpo)), 4)))
detach()
```

MixNRMI2cens *Normalized Random Measures Mixture of Type II for censored data*

Description

Bayesian nonparametric estimation based on normalized measures driven mixtures for locations and scales.

Usage

```
MixNRMI2cens(
  xleft,
  xright,
  probs = c(0.025, 0.5, 0.975),
  Alpha = 1,
  Kappa = 0,
  Gama = 0.4,
  distr.k = "normal",
  distr.py0 = "normal",
  distr.pz0 = "gamma",
  mu.pz0 = 3,
  sigma.pz0 = sqrt(10),
  delta_S = 4,
  kappa = 2,
  delta_U = 2,
  Meps = 0.01,
  Nx = 150,
  Nit = 1500,
  Pbi = 0.1,
  epsilon = NULL,
  printtime = TRUE,
  extras = TRUE,
  adaptive = FALSE
)
```

Arguments

xleft	Numeric vector. Lower limit of interval censoring. For exact data the same as xright
xright	Numeric vector. Upper limit of interval censoring. For exact data the same as xleft.
probs	Numeric vector. Desired quantiles of the density estimates.
Alpha	Numeric constant. Total mass of the centering measure. See details.
Kappa	Numeric positive constant. See details.
Gama	Numeric constant. $0 \leq Gama \leq 1$. See details.
distr.k	The distribution name for the kernel. Allowed names are "normal", "gamma", "beta", "double exponential", "lognormal" or their common abbreviations "norm", "exp", or an integer number identifying the mixture kernel: 1 = Normal; 2 = Gamma; 3 = Beta; 4 = Double Exponential; 5 = Lognormal.
distr.py0	The distribution name for the centering measure for locations. Allowed names are "normal", "gamma", "beta", or their common abbreviations "norm", "exp", or an integer number identifying the centering measure for locations: 1 = Normal; 2 = Gamma; 3 = Beta.
distr.pz0	The distribution name for the centering measure for scales. Allowed names are "gamma", "lognormal", "half-Cauchy", "half-normal", "half-student", "uniform" and "truncated normal", or their common abbreviations "norm", "exp", "lnorm", "halfcauchy", "halfnorm", "halft" and "unif", or an integer number identifying the centering measure for scales: 2 = Gamma, 5 = Lognormal, 6 = Half Cauchy, 7 = Half Normal, 8 = Half Student-t, 9 = Uniform, 10 = Truncated Normal.
mu.pz0	Numeric constant. Prior mean of the centering measure for scales.
sigma.pz0	Numeric constant. Prior standard deviation of the centering measure for scales.
delta_S	Numeric positive constant. Metropolis-Hastings proposal variation coefficient for sampling the scales.
kappa	Numeric positive constant. Metropolis-Hastings proposal variation coefficient for sampling the location parameters.

`delta_U`	Numeric positive constant. Metropolis-Hastings proposal variation coefficient for sampling the latent U. If 'adaptive=TRUE', 'delta_U'is the starting value for the adaptation.
`Meps`	Numeric constant. Relative error of the jump sizes in the continuous component of the process. Smaller values imply larger number of jumps.
`Nx`	Integer constant. Number of grid points for the evaluation of the density estimate.
`Nit`	Integer constant. Number of MCMC iterations.
`Pbi`	Numeric constant. Burn-in period proportion of `Nit`.
`epsilon`	Numeric constant. Extension to the evaluation grid range. See details.
`printtime`	Logical. If TRUE, prints out the execution time.
`extras`	Logical. If TRUE, gives additional objects: means, sigmas, weights and Js.
`adaptive`	Logical. If TRUE, uses an adaptive MCMC strategy to sample the latent U (adaptive delta_U).

Details

This generic function fits a normalized random measure (NRMI) mixture model for density estimation (James et al. 2009). Specifically, the model assumes a normalized generalized gamma (NGG) prior for both, locations (means) and standard deviations, of the mixture kernel, leading to a fully nonparametric mixture model.

The details of the model are:

$$X_i|Y_i, Z_i \sim k(\cdot|Y_i, Z_i)$$

$$(Y_i, Z_i)|P \sim P, i = 1, \ldots, n$$

$$P \sim \text{NGG}(\texttt{Alpha, Kappa, Gama; P\_0})$$

where, X_i's are the observed data, (Y_i, Z_i)'s are bivariate latent (location and scale) vectors, `k` is a parametric kernel parameterized in terms of mean and standard deviation, `(Alpha,Kappa, Gama; P_0)` are the parameters of the NGG prior with a bivariate `P_0` being the centering measure with independent components, that is, $P_0(Y, Z) = P_0(Y) * P_0(Z)$. The parameters of `P_0(Y)` are assigned vague hyper prior distributions and `(mu.pz0,sigma.pz0)` are the hyper-parameters of `P_0(Z)`. In particular, `NGG(Alpha, 1, 0; P_0)` defines a Dirichlet process; `NGG(1, Kappa, 1/2;P_0)` defines a Normalized inverse Gaussian process; and `NGG(1, 0, Gama; P_0)` defines a normalized stable process. The evaluation grid ranges from `min(x) -epsilon` to `max(x) + epsilon`. By default `epsilon=sd(x)/4`.

Value

The function returns a list with the following components:

`xx`	Numeric vector. Evaluation grid.
`qx`	Numeric array. Matrix of dimension `Nx` $\times$ (`length(probs)` $+ 1$) with the posterior mean and the desired quantiles input in `probs`.
`cpo`	Numeric vector of `length(x)` with conditional predictive ordinates.
`R`	Numeric vector of `length(Nit*(1-Pbi))` with the number of mixtures components (clusters).

`U`	Numeric vector of `length(Nit*(1-Pbi))` with the values of the latent variable U.
`Allocs`	List of `length(Nit*(1-Pbi))` with the clustering allocations.
`means`	List of `length(Nit*(1-Pbi))` with the cluster means (locations). Only if extras = TRUE.
`sigmas`	Numeric vector of `length(Nit*(1-Pbi))` with the cluster standard deviations. Only if extras = TRUE.
`weights`	List of `length(Nit*(1-Pbi))` with the mixture weights. Only if extras = TRUE.
`Js`	List of `length(Nit*(1-Pbi))` with the unnormalized weights (jump sizes). Only if extras = TRUE.
`Nm`	Integer constant. Number of jumps of the continuous component of the unnormalized process.
`delta_Us`	List of `length(Nit*(1-Pbi))` with the sequence of adapted delta_U used in the MH step for the latent variable U.
`Nx`	Integer constant. Number of grid points for the evaluation of the density estimate.
`Nit`	Integer constant. Number of MCMC iterations.
`Pbi`	Numeric constant. Burn-in period proportion of `Nit`.
`procTime`	Numeric vector with execution time provided by `proc.time` function.
`distr.k`	Integer corresponding to the kernel chosen for the mixture
`data`	Data used for the fit
`NRMI_params`	A named list with the parameters of the NRMI process

Warning

The function is computing intensive. Be patient.

Author(s)

Barrios, E., Kon Kam King, G. and Nieto-Barajas, L.E.

References

1.- Barrios, E., Lijoi, A., Nieto-Barajas, L. E. and Prünster, I. (2013). Modeling with Normalized Random Measure Mixture Models. Statistical Science. Vol. 28, No. 3, 313-334.

2.- James, L.F., Lijoi, A. and Prünster, I. (2009). Posterior analysis for normalized random measure with independent increments. Scand. J. Statist 36, 76-97.

3.- Kon Kam King, G., Arbel, J. and Prünster, I. (2016). Species Sensitivity Distribution revisited: a Bayesian nonparametric approach. In preparation.

See Also

`MixNRMI2`, `MixNRMI1cens`, `MixNRMI2cens`, `multMixNRMI1`

Examples

```
## Not run:
### Example 1
# Data
data(acidity)
x <- acidity
# Fitting the model under default specifications
out <- MixNRMI2cens(x, x)
# Plotting density estimate + 95% credible interval
plot(out)

## End(Not run)

## Not run:
### Example 2
# Data
data(salinity)
# Fitting the model under special specifications
out <- MixNRMI2cens(
  xleft = salinity$left, xright = salinity$right, Nit = 5000, distr.pz0 = 10,
  mu.pz0 = 1, sigma.pz0 = 2
)
# Plotting density estimate + 95% credible interval
attach(out)
plot(out)
# Plotting number of clusters
par(mfrow = c(2, 1))
plot(R, type = "l", main = "Trace of R")
hist(R, breaks = min(R - 0.5):max(R + 0.5), probability = TRUE)
detach()

## End(Not run)
```

MixPY1 *Pitman-Yor process mixture of Type I*

Description

This function calls the PYdensity function from package BNPmix, to allow fitting a Pitman-Yor process mixture to the data.

Usage

```
MixPY1(
  x,
  probs = c(0.025, 0.5, 0.975),
  Alpha = 1,
  Gama = 0.4,
  asigma = 2,
  bsigma = 1/var(x),
  Nx = 100,
  Nit = 1500,
  Pbi = 0.5,
  epsilon = NULL,
  printtime = TRUE,
  extras = TRUE
)
```

Arguments

`x`	Numeric vector. Data set to which the density is fitted.
`probs`	Numeric vector. Desired quantiles of the density estimates.
`Alpha`	Numeric constant. Total mass of the centering measure. See
`Gama`	Numeric constant. $0 \leq$ `Gama` ≤ 1. See details.
`asigma`	Numeric positive constant. Shape parameter of the gamma prior on the standard deviation of the mixture kernel. Default value suggested by package BNPmix.
`bsigma`	Numeric positive constant. Rate parameter of the gamma prior on the standard deviation of the mixture kernel. Default value suggested by package BNPmix.
`Nx`	Integer constant. Number of grid points for the evaluation of the density estimate.
`Nit`	Integer constant. Number of MCMC iterations.
`Pbi`	Numeric constant. Burn-in period proportion of Nit.
`epsilon`	Numeric constant. Extension to the evaluation grid range. See details.
`printtime`	Logical. If TRUE, prints out the execution time.
`extras`	Logical. If TRUE, gives additional objects: means and weights

Value

The function returns a MixPY1 object. It is based on a list with the following components:

`xx`	Numeric vector. Evaluation grid.
`qx`	Numeric array. Matrix of dimension `Nx` $\times$ (`length(probs)` $+ 1$) with the posterior mean and the desired quantiles input in `probs`.
`R`	Numeric vector of `length(Nit*(1-Pbi))` with the number of mixtures components (clusters).
`S`	Numeric vector of `length(Nit*(1-Pbi))` with the values of common standard deviation sigma.
`Allocs`	List of `length(Nit*(1-Pbi))` with the clustering allocations.
`means`	List of `length(Nit*(1-Pbi))` with the cluster means (locations). Only if extras = TRUE.
`weights`	List of `length(Nit*(1-Pbi))` with the mixture weights. Only if extras = TRUE.
`Nit`	Integer constant. Number of MCMC iterations.

Pbi	Numeric constant. Burn-in period proportion of Nit.
distr.k	Integer corresponding to the kernel chosen for the mixture. Always 1, since the Pitman-Yor process is only written to work with Gaussian kernels.
data	Data used for the fit
PY_params	A named list with the parameters of the Pitman-Yor process

Examples

```
# Data
data(acidity)
x <- acidity
# Fitting the model under default specifications
out <- MixPY1(x)
# Plotting density estimate + 95% credible interval
plot(out)
```

MixPY2 *Pitman-Yor process mixture of Type II*

Description

This function calls the PYdensity function from package BNPmix, to allow fitting a Pitman-Yor process mixture to the data.

Usage

```
MixPY2(
  x,
  probs = c(0.025, 0.5, 0.975),
  Alpha = 1,
  Gama = 0.4,
  asigma = 2,
  bsigma = 1/var(x),
  Nx = 100,
  Nit = 1500,
  Pbi = 0.5,
  epsilon = NULL,
  printtime = TRUE,
  extras = TRUE
)
```

Arguments

x	Numeric vector. Data set to which the density is fitted.
probs	Numeric vector. Desired quantiles of the density estimates.
Alpha	Numeric constant. Total mass of the centering measure. See

`Gama`	Numeric constant. $0 \leq$ `Gama` ≤ 1. See details.
`asigma`	Numeric positive constant. Shape parameter of the gamma prior on the standard deviation of the mixture kernel. Default value suggested by package BNPmix.
`bsigma`	Numeric positive constant. Rate parameter of the gamma prior on the standard deviation of the mixture kernel. Default value suggested by package BNPmix.
`Nx`	Integer constant. Number of grid points for the evaluation of the density estimate.
`Nit`	Integer constant. Number of MCMC iterations.
`Pbi`	Numeric constant. Burn-in period proportion of Nit.
`epsilon`	Numeric constant. Extension to the evaluation grid range. See details.
`printtime`	Logical. If TRUE, prints out the execution time.
`extras`	Logical. If TRUE, gives additional objects: means and weights

Value

The function returns a MixPY2 object. It is based on a list with the following components:

`xx`	Numeric vector. Evaluation grid.
`qx`	Numeric array. Matrix of dimension `Nx` $\times$ (`length(probs)` $+ 1$) with the posterior mean and the desired quantiles input in `probs`.
`R`	Numeric vector of `length(Nit*(1-Pbi))` with the number of mixtures components (clusters).
`Allocs`	List of `length(Nit*(1-Pbi))` with the clustering allocations.
`means`	List of `length(Nit*(1-Pbi))` with the cluster means (locations). Only if extras = TRUE.
`sigmas`	List of `length(Nit*(1-Pbi))` with the cluster standard deviations (scales). Only if extras = TRUE.
`weights`	List of `length(Nit*(1-Pbi))` with the mixture weights. Only if extras = TRUE.
`Nit`	Integer constant. Number of MCMC iterations.
`Pbi`	Numeric constant. Burn-in period proportion of `Nit`.
`distr.k`	Integer corresponding to the kernel chosen for the mixture. Always 1, since the Pitman-Yor process is only written to work with Gaussian kernels.
`data`	Data used for the fit
`PY_params`	A named list with the parameters of the Pitman-Yor process

Examples

```
# Data
data(acidity)
x <- acidity
# Fitting the model under default specifications
out <- MixPY2(x)
# Plotting density estimate + 95% credible interval
plot(out)
```

multMixNRMI1 *Multiple chains of MixNRMI1*

Description

Multiple chains of MixNRMI1

Usage

```
multMixNRMI1(
  x,
  probs = c(0.025, 0.5, 0.975),
  Alpha = 1,
  Kappa = 0,
  Gama = 0.4,
  distr.k = "normal",
  distr.p0 = "normal",
  asigma = 0.5,
  bsigma = 0.5,
  delta_S = 3,
  delta_U = 2,
  Meps = 0.01,
  Nx = 150,
  Nit = 1500,
  Pbi = 0.1,
  epsilon = NULL,
  printtime = TRUE,
  extras = TRUE,
  adaptive = FALSE,
  nchains = 4,
  parallel = TRUE,
  ncores = parallel::detectCores()
)
```

Arguments

`x`	Numeric vector. Data set to which the density is fitted.
`probs`	Numeric vector. Desired quantiles of the density estimates.
`Alpha`	Numeric constant. Total mass of the centering measure. See details.
`Kappa`	Numeric positive constant. See details.
`Gama`	Numeric constant. $0 \leq \texttt{Gama} \leq 1$. See details.
`distr.k`	The distribution name for the kernel. Allowed names are "normal", "gamma", "beta", "double exponential", "lognormal" or their common abbreviations "norm", "exp", or an integer number identifying the mixture kernel: 1 = Normal; 2 = Gamma; 3 = Beta; 4 = Double Exponential; 5 = Lognormal.

`distr.p0`	The distribution name for the centering measure. Allowed names are "normal", "gamma", "beta", or their common abbreviations "norm", "exp", or an integer number identifying the centering measure: 1 = Normal; 2 = Gamma; 3 = Beta.
`asigma`	Numeric positive constant. Shape parameter of the gamma prior on the standard deviation of the mixture kernel `distr.k`.
`bsigma`	Numeric positive constant. Rate parameter of the gamma prior on the standard deviation of the mixture kernel `distr.k`.
`delta_S`	Numeric positive constant. Metropolis-Hastings proposal variation coefficient for sampling sigma.
`delta_U`	Numeric positive constant. Metropolis-Hastings proposal variation coefficient for sampling the latent U.
`Meps`	Numeric constant. Relative error of the jump sizes in the continuous component of the process. Smaller values imply larger number of jumps.
`Nx`	Integer constant. Number of grid points for the evaluation of the density estimate.
`Nit`	Integer constant. Number of MCMC iterations.
`Pbi`	Numeric constant. Burn-in period proportion of Nit.
`epsilon`	Numeric constant. Extension to the evaluation grid range. See details.
`printtime`	Logical. If TRUE, prints out the execution time.
`extras`	Logical. If TRUE, gives additional objects: means, weights and Js.
`adaptive`	Logical. If TRUE, uses an adaptive MCMC strategy to sample the latent U (adaptive delta_U).
`nchains`	The number of chains to run.
`parallel`	Whether to run the chains in parallel. Only works on UNIX-like systems as it rests on Fork parallelism
`ncores`	Number of cores for the parallel run. Defaults to parallel::detectCores(), i.e. the maximum number of cores detected by R on your system.

Value

a list containing the multiple fits.

See Also

`MixNRMI2`, `MixNRMI1cens`, `MixNRMI2cens`

Examples

```
data(acidity)
multMixNRMI1(acidity, parallel = TRUE, Nit = 10, ncores = 2)
```

`multMixNRMI1cens` *Multiple chains of MixNRMI1cens*

Description

Multiple chains of MixNRMI1cens

Usage

```
multMixNRMI1cens(
  xleft,
  xright,
  probs = c(0.025, 0.5, 0.975),
  Alpha = 1,
  Kappa = 0,
  Gama = 0.4,
  distr.k = "normal",
  distr.p0 = "normal",
  asigma = 0.5,
  bsigma = 0.5,
  delta_S = 3,
  delta_U = 2,
  Meps = 0.01,
  Nx = 150,
  Nit = 1500,
  Pbi = 0.1,
  epsilon = NULL,
  printtime = TRUE,
  extras = TRUE,
  adaptive = FALSE,
  nchains = 4,
  parallel = TRUE,
  ncores = parallel::detectCores()
)
```

Arguments

`xleft`	Numeric vector. Lower limit of interval censoring. For exact data the same as xright
`xright`	Numeric vector. Upper limit of interval censoring. For exact data the same as xleft.
`probs`	Numeric vector. Desired quantiles of the density estimates.
`Alpha`	Numeric constant. Total mass of the centering measure. See details.
`Kappa`	Numeric positive constant. See details.
`Gama`	Numeric constant. $0 \leq$ `Gama` ≤ 1. See details.

`distr.k`	The distribution name for the kernel. Allowed names are "normal", "gamma", "beta", "double exponential", "lognormal" or their common abbreviations "norm", "exp", or an integer number identifying the mixture kernel: 1 = Normal; 2 = Gamma; 3 = Beta; 4 = Double Exponential; 5 = Lognormal.
`distr.p0`	The distribution name for the centering measure. Allowed names are "normal", "gamma", "beta", or their common abbreviations "norm", "exp", or an integer number identifying the centering measure: 1 = Normal; 2 = Gamma; 3 = Beta.
`asigma`	Numeric positive constant. Shape parameter of the gamma prior on the standard deviation of the mixture kernel `distr.k`.
`bsigma`	Numeric positive constant. Rate parameter of the gamma prior on the standard deviation of the mixture kernel `distr.k`.
`delta_S`	Numeric positive constant. Metropolis-Hastings proposal variation coefficient for sampling sigma.
`delta_U`	Numeric positive constant. Metropolis-Hastings proposal variation coefficient for sampling the latent U.
`Meps`	Numeric constant. Relative error of the jump sizes in the continuous component of the process. Smaller values imply larger number of jumps.
`Nx`	Integer constant. Number of grid points for the evaluation of the density estimate.
`Nit`	Integer constant. Number of MCMC iterations.
`Pbi`	Numeric constant. Burn-in period proportion of Nit.
`epsilon`	Numeric constant. Extension to the evaluation grid range. See details.
`printtime`	Logical. If TRUE, prints out the execution time.
`extras`	Logical. If TRUE, gives additional objects: means, weights and Js.
`adaptive`	Logical. If TRUE, uses an adaptive MCMC strategy to sample the latent U (adaptive delta_U).
`nchains`	The number of chains to run.
`parallel`	Whether to run the chains in parallel. Only works on UNIX-like systems as it rests on Fork parallelism
`ncores`	Number of cores for the parallel run. Defaults to parallel::detectCores(), i.e. the maximum number of cores detected by R on your system.

Value

a list containing the multiple fits.

See Also

`MixNRMI2`, `MixNRMI1cens`, `MixNRMI2cens`, `multMixNRMI1`

Examples

```
data(salinity)
multMixNRMI1cens(salinity$left, salinity$right, parallel = TRUE, Nit = 10, ncores = 2)
```

multMixNRMI2 *Multiple chains of MixNRMI2*

Description

Multiple chains of MixNRMI2

Usage

```
multMixNRMI2(
  x,
  probs = c(0.025, 0.5, 0.975),
  Alpha = 1,
  Kappa = 0,
  Gama = 0.4,
  distr.k = "normal",
  distr.py0 = "normal",
  distr.pz0 = "gamma",
  mu.pz0 = 3,
  sigma.pz0 = sqrt(10),
  delta_S = 4,
  kappa = 2,
  delta_U = 2,
  Meps = 0.01,
  Nx = 150,
  Nit = 1500,
  Pbi = 0.1,
  epsilon = NULL,
  printtime = TRUE,
  extras = TRUE,
  adaptive = FALSE,
  nchains = 4,
  parallel = FALSE,
  ncores = parallel::detectCores()
)
```

Arguments

`x`	Numeric vector. Data set to which the density is fitted.
`probs`	Numeric vector. Desired quantiles of the density estimates.
`Alpha`	Numeric constant. Total mass of the centering measure. See details.
`Kappa`	Numeric positive constant. See details.
`Gama`	Numeric constant. $0 \leq Gama \leq 1$. See details.
`distr.k`	The distribution name for the kernel. Allowed names are "normal", "gamma", "beta", "double exponential", "lognormal" or their common abbreviations "norm",

	"exp", or an integer number identifying the mixture kernel: 1 = Normal; 2 = Gamma; 3 = Beta; 4 = Double Exponential; 5 = Lognormal.
`distr.py0`	The distribution name for the centering measure for locations. Allowed names are "normal", "gamma", "beta", or their common abbreviations "norm", "exp", or an integer number identifying the centering measure for locations: 1 = Normal; 2 = Gamma; 3 = Beta.
`distr.pz0`	The distribution name for the centering measure for scales. Allowed names are "gamma", or an integer number identifying the centering measure for scales: 2 = Gamma. For more options use `MixNRMI2cens`.
`mu.pz0`	Numeric constant. Prior mean of the centering measure for scales.
`sigma.pz0`	Numeric constant. Prior standard deviation of the centering measure for scales.
`delta_S`	Numeric positive constant. Metropolis-Hastings proposal variation coefficient for sampling the scales.
`kappa`	Numeric positive constant. Metropolis-Hastings proposal variation coefficient for sampling the location parameters.
`delta_U`	Numeric positive constant. Metropolis-Hastings proposal variation coefficient for sampling the latent U. If ‘adaptive=TRUE‘, ‘delta_U‘is the starting value for the adaptation.
`Meps`	Numeric constant. Relative error of the jump sizes in the continuous component of the process. Smaller values imply larger number of jumps.
`Nx`	Integer constant. Number of grid points for the evaluation of the density estimate.
`Nit`	Integer constant. Number of MCMC iterations.
`Pbi`	Numeric constant. Burn-in period proportion of `Nit`.
`epsilon`	Numeric constant. Extension to the evaluation grid range. See details.
`printtime`	Logical. If TRUE, prints out the execution time.
`extras`	Logical. If TRUE, gives additional objects: means, sigmas, weights and Js.
`adaptive`	Logical. If TRUE, uses an adaptive MCMC strategy to sample the latent U (adaptive delta_U).
`nchains`	The number of chains to run.
`parallel`	Whether to run the chains in parallel. Only works on UNIX-like systems as it rests on Fork parallelism
`ncores`	Number of cores for the parallel run. Defaults to parallel::detectCores(), i.e. the maximum number of cores detected by R on your system.

Value

a list containing the multiple fits.

See Also

`MixNRMI2`, `MixNRMI1cens`, `MixNRMI2cens`, `multMixNRMI1`

Examples

```
data(acidity)
multMixNRMI2(acidity, parallel = TRUE, Nit = 10, ncores = 2)
```

`multMixNRMI2cens` *Multiple chains of MixNRMI2cens*

Description

Multiple chains of MixNRMI2cens

Usage

```
multMixNRMI2cens(
  xleft,
  xright,
  probs = c(0.025, 0.5, 0.975),
  Alpha = 1,
  Kappa = 0,
  Gama = 0.4,
  distr.k = "normal",
  distr.py0 = "normal",
  distr.pz0 = "gamma",
  mu.pz0 = 3,
  sigma.pz0 = sqrt(10),
  delta_S = 4,
  kappa = 2,
  delta_U = 2,
  Meps = 0.01,
  Nx = 150,
  Nit = 1500,
  Pbi = 0.1,
  epsilon = NULL,
  printtime = TRUE,
  extras = TRUE,
  adaptive = FALSE,
  nchains = 4,
  parallel = TRUE,
  ncores = parallel::detectCores()
)
```

Arguments

`xleft`	Numeric vector. Lower limit of interval censoring. For exact data the same as xright
`xright`	Numeric vector. Upper limit of interval censoring. For exact data the same as xleft.

`probs`	Numeric vector. Desired quantiles of the density estimates.
`Alpha`	Numeric constant. Total mass of the centering measure. See details.
`Kappa`	Numeric positive constant. See details.
`Gama`	Numeric constant. $0 \leq Gama \leq 1$. See details.
`distr.k`	The distribution name for the kernel. Allowed names are "normal", "gamma", "beta", "double exponential", "lognormal" or their common abbreviations "norm", "exp", or an integer number identifying the mixture kernel: 1 = Normal; 2 = Gamma; 3 = Beta; 4 = Double Exponential; 5 = Lognormal.
`distr.py0`	The distribution name for the centering measure for locations. Allowed names are "normal", "gamma", "beta", or their common abbreviations "norm", "exp", or an integer number identifying the centering measure for locations: 1 = Normal; 2 = Gamma; 3 = Beta.
`distr.pz0`	The distribution name for the centering measure for scales. Allowed names are "gamma", "lognormal", "half-Cauchy", "half-normal", "half-student", "uniform" and "truncated normal", or their common abbreviations "norm", "exp", "lnorm", "halfcauchy", "halfnorm", "halft" and "unif", or an integer number identifying the centering measure for scales: 2 = Gamma, 5 = Lognormal, 6 = Half Cauchy, 7 = Half Normal, 8 = Half Student-t, 9 = Uniform, 10 = Truncated Normal.
`mu.pz0`	Numeric constant. Prior mean of the centering measure for scales.
`sigma.pz0`	Numeric constant. Prior standard deviation of the centering measure for scales.
`delta_S`	Numeric positive constant. Metropolis-Hastings proposal variation coefficient for sampling the scales.
`kappa`	Numeric positive constant. Metropolis-Hastings proposal variation coefficient for sampling the location parameters.
`delta_U`	Numeric positive constant. Metropolis-Hastings proposal variation coefficient for sampling the latent U. If ‘adaptive=TRUE‘, ‘delta_U‘is the starting value for the adaptation.
`Meps`	Numeric constant. Relative error of the jump sizes in the continuous component of the process. Smaller values imply larger number of jumps.
`Nx`	Integer constant. Number of grid points for the evaluation of the density estimate.
`Nit`	Integer constant. Number of MCMC iterations.
`Pbi`	Numeric constant. Burn-in period proportion of `Nit`.
`epsilon`	Numeric constant. Extension to the evaluation grid range. See details.
`printtime`	Logical. If TRUE, prints out the execution time.
`extras`	Logical. If TRUE, gives additional objects: means, sigmas, weights and Js.
`adaptive`	Logical. If TRUE, uses an adaptive MCMC strategy to sample the latent U (adaptive delta_U).
`nchains`	The number of chains to run.
`parallel`	Whether to run the chains in parallel. Only works on UNIX-like systems as it rests on Fork parallelism
`ncores`	Number of cores for the parallel run. Defaults to parallel::detectCores(), i.e. the maximum number of cores detected by R on your system.

Value

a list containing the multiple fits.

See Also

MixNRMI2, MixNRMI1cens, MixNRMI2cens, multMixNRMI1

Examples

```
data(salinity)
## Not run:
multMixNRMI2cens(salinity$left, salinity$right, parallel = TRUE, Nit = 20, ncores = 2)

## End(Not run)
```

MvInv *Invert jump heights function*

Description

Determines the jump heights of an increasing additive process by inverting the M(v) function. Use a truncation level based on expected moments of the NGG process (thresholdGG). For internal use.

Usage

```
MvInv(eps, u = 0.5, alpha = 1, kappa = 1, gama = 1/2, N = 3001)
```

Arguments

eps	Dummy argument kept for consistency with past versions of the functions
u	Real number. The value of the latent variable at the current step.
alpha	Numeric constant. Total mass of the centering measure.
kappa	Numeric positive constant.
gama	Numeric constant. Discount parameter of the NRMI process.
N	Number of steps in the discretization scheme for the grid inversion. ## The function has been optimised but it is morally defined as: function(eps, u = 0.5, alpha = 1, kappa = 1, gama = 1 / 2, N = 3001) n <- length(w) v <- rep(NA, n) x <- -log(seq(from = exp(-1e-05), to = exp(-10), length = N)) f <- alpha / gamma(1 - gama) * x^(-(1 + gama)) * exp(-(u + kappa) * x) dx <- diff(x) h <- (f[-1] + f[-N]) / 2 Mv <- rep(0, N) for (i in seq(N - 1, 1)) Mv[i] <- Mv[i + 1] + dx[i] * h[i] for (j in seq(n)) v[j] <- x[which.min(Mv > w[j])] return(v)

plot.multNRMI — *Plot the density estimate and the 95% credible interval*

Description

The density estimate is the mean posterior density computed on the data points.

Usage

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

Arguments

`x`	An object of class multNRMI
`...`	Further arguments to be passed to generic functions, ignored at the moment

Value

A graph with the density estimate, the 95% credible interval. Includes a histogram if the data is non censored.

Examples

```
data(salinity)
fit <- multMixNRMI2cens(salinity$left, salinity$right, parallel = TRUE, Nit = 10, ncores = 2)
plot(fit)
```

plot.NRMI1 — *Plot the density estimate and the 95% credible interval*

Description

The density estimate is the mean posterior density computed on the data points.

Usage

```
## S3 method for class 'NRMI1'
plot(x, ...)
```

Arguments

`x`	A fitted object of class NRMI1
`...`	Further arguments to be passed to generic function, ignored at the moment

Value

A graph with the density estimate, the 95% credible interval and a histogram of the data

Examples

```
## Example for non censored data

data(acidity)
out <- MixNRMI1(acidity, Nit = 50)
plot(out)

## Example for censored data

data(salinity)
out <- MixNRMI1cens(salinity$left, salinity$right, Nit = 50)
plot(out)
```

`plot.NRMI2` *Plot the density estimate and the 95% credible interval*

Description

The density estimate is the mean posterior density computed on the data points.

Usage

```
## S3 method for class 'NRMI2'
plot(x, ...)
```

Arguments

`x`	A fitted object of class NRMI2
`...`	Further arguments to be passed to generic function, ignored at the moment

Value

A graph with the density estimate, the 95% credible interval and a histogram of the data

Examples

```
## Example for non censored data

data(acidity)
out <- MixNRMI2(acidity, Nit = 20)
plot(out)

## Example for censored data

data(salinity)
```

```
out <- MixNRMI2cens(salinity$left, salinity$right, Nit = 20)
plot(out)
```

plot.PY1 *Plot the density estimate and the 95% credible interval*

Description

Plot the density estimate and the 95% credible interval

Usage

```
## S3 method for class 'PY1'
plot(x, ...)
```

Arguments

x	A fitted object of class PY1
...	Further arguments to be passed to generic function, ignored at the moment

Value

A graph with the density estimate, the 95% credible interval and a histogram of the data

Examples

```
data(acidity)
out <- MixPY1(acidity, Nit = 50)
plot(out)
```

plot.PY2 *Plot the density estimate and the 95% credible interval*

Description

Plot the density estimate and the 95% credible interval

Usage

```
## S3 method for class 'PY2'
plot(x, ...)
```

Arguments

x	A fitted object of class PY2
...	Further arguments to be passed to generic function, ignored at the moment

Value

A graph with the density estimate, the 95% credible interval and a histogram of the data

Examples

```
data(acidity)
out <- MixPY2(acidity, Nit = 50)
plot(out)
```

---

plotCDF_censored *Plot the Turnbull CDF and fitted CDF for censored data.*

Description

Plot the Turnbull CDF and fitted CDF for censored data.

Usage

```
plotCDF_censored(fit)
```

Arguments

`fit`	The result of the fit, obtained through the function MixNRMI1cens or MixNRMI2cens.

Value

Plot of the empirical and fitted CDF for non censored data.

Examples

```
set.seed(150520)
data(salinity)
out <- MixNRMI1cens(salinity$left, salinity$right, extras = TRUE, Nit = 100)
BNPdensity:::plotCDF_censored(out)
```

plotCDF_noncensored *Plot the empirical and fitted CDF for non censored data.*

Description

Plot the empirical and fitted CDF for non censored data.

Usage

```
plotCDF_noncensored(fit)
```

Arguments

`fit`	The result of the fit, obtained through the function MixNRMI1 or MixNRMI2.

Value

Plot of the empirical and fitted CDF for non censored data.

Examples

```
set.seed(150520)
data(acidity)
out <- MixNRMI1(acidity, extras = TRUE, Nit = 10)
BNPdensity:::plotCDF_noncensored(out)
```

plotfit_censored *Plot the density estimate and the 95% credible interval for censored data*

Description

The density estimate is the mean posterior density computed on the data points. It is not possible to display a histogram for censored data.

Usage

```
plotfit_censored(fit)
```

Arguments

`fit`	A fitted object of class NRMI1cens or NRMI2cens

Value

A graph with the density estimate and the 95% credible interval

Examples

```
data(acidity)
out <- MixNRMI1(acidity, Nit = 50)
plot(out)
```

`plotfit_noncensored` *Plot the density estimate and the 95% credible interval for noncensored data*

Description

The density estimate is the mean posterior density computed on the data points.

Usage

```
plotfit_noncensored(fit)
```

Arguments

`fit`	A fitted object of class NRMI1 or NRMI2

Value

A graph with the density estimate, the 95% credible interval and a histogram of the data

Examples

```
data(acidity)
out <- MixNRMI1(acidity, Nit = 50)
plot(out)
```

`plotPDF_censored` *Plot the density for censored data.*

Description

Plot the density for censored data.

Usage

```
plotPDF_censored(fit)
```

Arguments

`fit`	The result of the fit, obtained through the function MixNRMI1cens or MixNRMI2cens.

Value

Plot of the density and a histogram for non censored data.

Examples

```
set.seed(150520)
data(salinity)
out <- MixNRMI1cens(xleft = salinity$left, xright = salinity$right, extras = TRUE, Nit = 100)
BNPdensity:::plotPDF_censored(out)
```

`plotPDF_noncensored` *Plot the density and a histogram for non censored data.*

Description

Plot the density and a histogram for non censored data.

Usage

```
plotPDF_noncensored(fit)
```

Arguments

`fit`	The result of the fit, obtained through the function MixNRMI1 or MixNRMI2.

Value

Plot of the density and a histogram for non censored data.

Examples

```
set.seed(150520)
data(acidity)
out <- MixNRMI1(acidity, extras = TRUE, Nit = 100)
BNPdensity:::plotPDF_noncensored(out)
```

`plot_clustering_and_CDF` *Plot the clustering and the Cumulative Distribution Function*

Description

This is a function to visualize the clustering induced by the BNP model. The data points are plotted with a color reflecting their cluster.

Usage

```
plot_clustering_and_CDF(fit, clustering, label_vector = NULL)
```

Arguments

`fit`	The fitted object, obtained from one of the MixNRMIx functions
`clustering`	A vector of integers with the same length as the data, representing the allocation variable for data each point.
`label_vector`	A vector of data labels to be plotted, to provide some identification to each point.

Value

A plot of the Cumulative Distribution Function (or Turnbull estimate for censored data) with data points whose color denotes the cluster allocation. For censored data, right or left censored data points are not represented, while interval censored data points are represented at the middle of the censoring interval.

`plot_prior_number_of_components` *This plots the prior distribution on the number of components for the stable process. The Dirichlet process is provided for comparison.*

Description

This plots the prior distribution on the number of components for the stable process. The Dirichlet process is provided for comparison.

Usage

```
plot_prior_number_of_components(
  n,
  Gama,
  Alpha = 1,
  grid = NULL,
  silence = TRUE
)
```

Arguments

`n`	Number of data points
`Gama`	Numeric constant. 0 <= Gama <=1.
`Alpha`	Numeric constant. Total mass of the centering measure for the Dirichlet process.
`grid`	Integer vector. Level of truncation when computing the expectation. Defaults to n. If greater than n, it is fixed to n.
`silence`	Boolean. Whether to print the current calculation step for the Stable process, as the function can be long

Value

A plot with the prior distribution on the number of components.

Examples

```
plot_prior_number_of_components(50, 0.4)
```

`pp_plot_censored` *Plot the percentile-percentile graph for non censored data, using the Turnbull estimator the position of the percentiles.*

Description

Plot the percentile-percentile graph for non censored data, using the Turnbull estimator the position of the percentiles.

Usage

```
pp_plot_censored(fit)
```

Arguments

`fit`	The result of the fit, obtained through the function MixNRMI1cens or MixN-RMI2cens.

Value

Percentile-percentile graph using the Turnbull estimator

Examples

```
set.seed(150520)
data(salinity)
out <- MixNRMI1cens(xleft = salinity$left, xright = salinity$right, extras = TRUE, Nit = 100)
BNPdensity:::pp_plot_censored(out)
```

`pp_plot_noncensored` *Plot the percentile-percentile graph for non censored data.*

Description

Plot the percentile-percentile graph for non censored data.

Usage

```
pp_plot_noncensored(fit)
```

Arguments

`fit`	The result of the fit, obtained through the function MixNRMI1 or MixNRMI2.

Value

Percentile-percentile plot for non censored data.

Examples

```
set.seed(150520)
data(acidity)
out <- MixNRMI1(acidity, extras = TRUE, Nit = 100)
BNPdensity:::pp_plot_noncensored(out)
```

`print.multNRMI` *S3 method for class 'multNRMI'*

Description

S3 method for class 'multNRMI'

Usage

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

Arguments

`x`	An object of class multNRMI
`...`	Further arguments to be passed to generic functions, ignored at the moment

Value

A visualization of the important information about the object

Examples

```
data(salinity)
out <- multMixNRMI2cens(salinity$left, salinity$right, parallel = TRUE, Nit = 10, ncores = 2)
print(out)
```

`print.NRMI1` *S3 method for class 'MixNRMI1'*

Description

S3 method for class 'MixNRMI1'

Usage

```
## S3 method for class 'NRMI1'
print(x, ...)
```

Arguments

`x`	A fitted object of class NRMI1
`...`	Further arguments to be passed to generic function, ignored at the moment

Value

A visualization of the important information about the object

Examples

```
## Example for non censored data

data(acidity)
out <- MixNRMI1(acidity, Nit = 50)
print(out)

## Example for censored data

data(salinity)
out <- MixNRMI1cens(salinity$left, salinity$right, Nit = 50)
print(out)
```

`print.NRMI2` *S3 method for class 'MixNRMI2'*

Description

S3 method for class 'MixNRMI2'

Usage

```
## S3 method for class 'NRMI2'
print(x, ...)
```

Arguments

`x`	A fitted object of class NRMI2
`...`	Further arguments to be passed to generic function, ignored at the moment

Value

A visualization of the important information about the object

Examples

```
#' ## Example for censored data
data(acidity)
out <- MixNRMI2(acidity, Nit = 20)
print(out)

data(salinity)
out <- MixNRMI2cens(salinity$left, salinity$right, Nit = 20)
print(out)
```

`print.PY1` *S3 method for class 'PY1'*

Description

S3 method for class 'PY1'

Usage

```
## S3 method for class 'PY1'
print(x, ...)
```

Arguments

`x`	A fitted object of class PY1
`...`	Further arguments to be passed to generic function, ignored at the moment

Value

A visualization of the important information about the object

Examples

```
## Example for non censored data

data(acidity)
out <- MixPY1(acidity, Nit = 50)
print(out)
```

`print.PY2` *S3 method for class 'PY2'*

Description

S3 method for class 'PY2'

Usage

```
## S3 method for class 'PY2'
print(x, ...)
```

Arguments

`x`	A fitted object of class PY2
`...`	Further arguments to be passed to generic function, ignored at the moment

Value

A visualization of the important information about the object

Examples

```
## Example for non censored data

data(acidity)
out <- MixPY2(acidity, Nit = 50)
print(out)
```

`process_dist_name` *Process the distribution name argument into a distribution index*

Description

This function is intended to help with compatibility with the previous versions of the package.

Usage

```
process_dist_name(distname)
```

Arguments

`distname`	Can be an integer or a distribution name. Allowed names are "normal", "gamma", "beta", "exponential", "lognormal", "half-Cauchy", "half-normal", "half-student", "uniform" and "truncated normal", or their common abbreviations "norm", "exp", "halfcauchy", "halfnorm", "halft" and "unif".

Value

an integer both if distname is an integer or a character

`qq_plot_censored` *Plot the quantile-quantile graph for censored data.*

Description

This function may be rather slow for many iterations/many data because it relies on numerical inversion of the mixture Cumulative Distribution Function. set.seed(150520) data(salinity) out <- MixNRMI1cens(xleft = salinity$left, xright = salinity$right, extras = TRUE, Nit = 100) BNPdensity:::qq_plot_censored(out)

Usage

```
qq_plot_censored(fit, thinning_to = 500)
```

Arguments

`fit`	The result of the fit, obtained through the function MixNRMI1 or MixNRMI2, MixMRMI1cens or MixMRMI2cens
`thinning_to`	How many iterations to compute the mean posterior quantiles

Value

quantile-quantile plot for non censored data.

qq_plot_noncensored *Plot the quantile-quantile graph for non censored data.*

Description

This function may be rather slow for many iterations/many data because it relies on numerical inversion of the mixture Cumulative Distribution Function.

Usage

```
qq_plot_noncensored(fit, thinning_to = 500)
```

Arguments

`fit`	The result of the fit, obtained through the function MixNRMI1 or MixNRMI2, MixMRMI1cens or MixMRMI2cens
`thinning_to`	How many iterations to compute the mean posterior quantiles

Value

quantile-quantile plot for non censored data.

Examples

```
### Not run
# set.seed(150520)
# data(acidity)
# out <- MixNRMI1(acidity, extras = TRUE, Nit = 100)
# BNPdensity:::qq_plot_noncensored(out)
```

salinity *Salinity tolerance*

Description

72-hour acute salinity tolerance (LC50 values) of riverine macro-invertebrates.

Format

A data frame with 108 observations on the following two variables:

left A numeric vector.

right A numeric vector.

Source

`fitdistrplus` R-package

References

Kefford, B.J., Nugegoda, D., Metzeling, L., Fields, E. 2006. Validating species sensitivity distributions using salinity tolerance of riverine macroinvertebrates in the southern Murray-darling Basin (Victoria, Australia). Canadian Journal of Fisheries and Aquatic Science, 63, 1865-1877.

Examples

```
data(salinity)
hist(salinity$left)
```

`summary.multNRMI` *S3 method for class 'multNRMI'*

Description

S3 method for class 'multNRMI'

Usage

```
## S3 method for class 'multNRMI'
summary(object, number_of_clusters = FALSE, ...)
```

Arguments

`object`	A fitted object of class NRMI1cens
`number_of_clusters`	Whether to compute the optimal number of clusters, which can be a time-consuming operation (see `compute_optimal_clustering`)
`...`	Further arguments to be passed to generic function, ignored at the moment

Value

Prints out the text for the summary S3 methods

Examples

```
data(salinity)
out <- multMixNRMI2cens(salinity$left, salinity$right, parallel = TRUE, Nit = 10, ncores = 2)
summary(out)
```

summary.NRMI1 *S3 method for class 'MixNRMI1'*

Description

S3 method for class 'MixNRMI1'

Usage

```
## S3 method for class 'NRMI1'
summary(object, number_of_clusters = FALSE, ...)
```

Arguments

`object`	A fitted object of class NRMI1
`number_of_clusters`	Whether to compute the optimal number of clusters, which can be a time-consuming operation (see `compute_optimal_clustering`)
`...`	Further arguments to be passed to generic function, ignored at the moment

Value

Prints out the text for the summary S3 methods

Examples

```
## Example for non censored data

data(acidity)
out <- MixNRMI1(acidity, Nit = 50)
summary(out)
```

summary.NRMI2 *S3 method for class 'MixNRMI2'*

Description

S3 method for class 'MixNRMI2'

Usage

```
## S3 method for class 'NRMI2'
summary(object, number_of_clusters = FALSE, ...)
```

Arguments

`object`	A fitted object of class NRMI2
`number_of_clusters`	Whether to compute the optimal number of clusters, which can be a time-consuming operation (see `compute_optimal_clustering`)
`...`	Further arguments to be passed to generic function, ignored at the moment

Value

Prints out the text for the summary S3 methods

Examples

```
data(acidity)
out <- MixNRMI2(acidity, Nit = 20)
summary(out)

data(salinity)
out <- MixNRMI2cens(salinity$left, salinity$right, Nit = 20)
summary(out)
```

`summary.PY1` *S3 method for class 'PY1'*

Description

S3 method for class 'PY1'

Usage

```
## S3 method for class 'PY1'
summary(object, number_of_clusters = FALSE, ...)
```

Arguments

`object`	A fitted object of class PY1
`number_of_clusters`	Whether to compute the optimal number of clusters, which can be a time-consuming operation (see `compute_optimal_clustering`)
`...`	Further arguments to be passed to generic function, ignored at the moment

Value

Prints out the text for the summary S3 methods

Examples

```
## Example for non censored data

data(acidity)
out <- MixPY1(acidity, Nit = 50)
summary(out)
```

`summary.PY2` *S3 method for class 'PY2'*

Description

S3 method for class 'PY2'

Usage

```
## S3 method for class 'PY2'
summary(object, number_of_clusters = FALSE, ...)
```

Arguments

`object`	A fitted object of class PY2
`number_of_clusters`	Whether to compute the optimal number of clusters, which can be a time-consuming operation (see `compute_optimal_clustering`)
`...`	Further arguments to be passed to generic function, ignored at the moment

Value

Prints out the text for the summary S3 methods

Examples

```
## Example for non censored data

data(acidity)
out <- MixPY2(acidity, Nit = 50)
summary(out)
```

`summarytext` *Common text for the summary S3 methods*

Description

Common text for the summary S3 methods

Usage

```
summarytext(
  fit,
  kernel_comment,
  BNP_process_comment,
  number_of_clusters = FALSE
)
```

Arguments

`fit`	NRMIx or PYx object
`kernel_comment`	Text specific to the parametric and nonparametric nature of the model
`BNP_process_comment`	Text specific to the nonparametric process, NRMI or Pitman-Yor
`number_of_clusters`	Flag to decide whether to compute the optimal clustering

Value

Prints out the text for the summary S3 methods

`traceplot` *Draw a traceplot for multiple chains*

Description

This is a convenience function which works when coda is not yet loaded by the user. If coda is loaded, it gets masked. See also file multMixNRMI.R

Usage

```
traceplot(fitlist)
```

Arguments

`fitlist`	Output of multMixNRMI.

Value

A traceplot for multiple chains.

Index