

Package ‘BNPdensity’

September 18, 2019

Type Package

Title Ferguson-Klass Type Algorithm for Posterior Normalized Random Measures

Version 2019.9.18

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 6.1.1

Depends R(>= 3.3.0)

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

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

Suggests GreedyEPL, Rmpfr, gmp

NeedsCompilation yes

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 2019-09-18 16:10:02 UTC

R topics documented:

BNPdensity-package	3
acidity	4
add	5

as.mcmc.multNRMI	5
asNumeric_no_warning	6
comment_on_NRMI_type	6
compute_optimal_clustering	7
compute_thinning_grid	7
convert_to_mcmc	8
enzyme	8
Enzyme1.out	9
Enzyme2.out	9
expected_number_of_components_Dirichlet	10
expected_number_of_components_stable	10
fill_sigmas	11
galaxy	12
Galaxy1.out	12
Galaxy2.out	13
give_kernel_name	13
GOFplots	14
GOFplots_censored	14
GOFplots_noncensored	15
grid_from_data	16
grid_from_data_censored	16
grid_from_data_noncensored	17
is_censored	17
is_semiparametric	18
MixNRMI1	18
MixNRMI1cens	23
MixNRMI2	27
MixNRMI2cens	31
multMixNRMI1	35
multMixNRMI1cens	37
multMixNRMI2	38
multMixNRMI2cens	40
plot.multNRMI	41
plot.NRMI1	42
plot.NRMI1cens	43
plot.NRMI2	43
plot.NRMI2cens	44
plotCDF_censored	45
plotCDF_noncensored	45
plotfit_censored	46
plotfit_noncensored	46
plotPDF_censored	47
plotPDF_noncensored	48
plot_clustering_and_CDF	48
plot_prior_number_of_components	49
pp_plot_censored	50
pp_plot_noncensored	50
print.multNRMI	51

print.NRMI1	52
print.NRMI1cens	52
print.NRMI2	53
print.NRMI2cens	54
qq_plot_censored	54
qq_plot_noncensored	55
salinity	55
summary.multNRMI	56
summary.NRMI1	57
summary.NRMI1cens	57
summary.NRMI2	58
summary.NRMI2cens	59
summarytext	59
traceplot	60

Index	61
--------------	-----------

BNPdensity-package	<i>Bayesian nonparametric density estimation</i>
--------------------	--

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 `MixNRMI2cens` 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 Prüenster, I.; Contributor: Guillaume Kon Kam King.; Maintainer: Ernesto Barrios <ebarrios at itam.mx>

References

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

Kon Kam King, G., Arbel, J. and Prüenster, 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 data frame with 155 observations on the following variable:

list("acidity") A numeric vector.

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	<i>Add x and y</i>
-----	--------------------

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

$x + y$

Examples

```
BNPdensity::add(1, 2)
```

as.mcmc.multNRMI	<i>Convert the output of multMixNRMI into a coda mcmc object</i>
------------------	--

Description

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

Usage

```
## S3 method for class 'multNRMI'  
as.mcmc(fitlist, thinning_to = 1000)
```

Arguments

fitlist	Output of <code>multMixNRMI</code> .
thinning_to	Final length of the chain after thinning.

Value

a coda::mcmc object

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

Summaizes 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

<code>fit</code>	The fitted object, obtained from one of the MixNRMix functions
<code>loss_type</code>	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" (Normalised Variation of Information) or "NID" (Normalised 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

<code>Nit</code>	Length of the MCMC chain
<code>thinning_to</code>	Desired number of iterations after thinning.

Value

an integer vector of the MCMC iterations retained.

convert_to_mcmc	<i>Convert the output of multMixNRMI into a coda mcmc object</i>
-----------------	--

Description

Convert the output of multMixNRMI into a coda mcmc object

Usage

```
convert_to_mcmc(fitlist, thinning_to = 1000)
```

Arguments

fitlist	Output of multMixNRMI.
thinning_to	Final length of the chain after thinning.

Value

a coda::mcmc object

enzyme	<i>Enzyme Dataset</i>
--------	-----------------------

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 `MixNRMI1(x, Alpha = 1, Kappa = 0.007, Gama = 0.5, distr.k = 2, distr.p0 = 2, asigma = 1, bsigma = 1, Meps=0.001, 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(x, Alpha = 1, Kappa = 0.007, Gama = 0.5, distr.k = 2, distr.py0 = 2, distr.pz0 = 2, mu.pz0 = 1, sigma.pz0 = 1, Meps=0.001, Nit = 5000, Pbi = 0.2)`

Details

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. $0 \leq \text{Gama} \leq 1$.
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	<i>Repeat the common scale parameter of a semiparametric model to match the dimension of the location parameters.</i>
-------------	---

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 MixNRMII.
--------------------	--

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(x, Alpha = 1, Kappa = 0.015, Gama = 0.5, distr.k = 1, distr.p0 = 2, asigma = 1, bsigma = 1, Meps=0.001, Nit = 5000, Pbi = 0.2)`

Details

See function `MixNRMI1`.

Examples

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

Galaxy2.out

*Fit of MixNRM12 function to the galaxy dataset***Description**

This object contains the output when setting `set.seed(150520)` and running the function `MixNRM12(x, Alpha = 1, Kappa = 0.015, Gama = 0.5, distr.k = 1, distr.py0 = 2, distr.pz0 = 2, mu.pz0 = 1, sigma.pz0 = 1, Meps=0.001, Nit = 5000, Pbi = 0.2)`

Details

See function `MixNRM12`.

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 `MixNRM11`, `MixNRM12`, `MixNRM11cens`, `MixNRM12cens`, and all the `multMixNRM1x` versions

Usage

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

Arguments

`distr.k` 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

<code>fit</code>	The result of the fit, obtained through the function <code>MixNRM1</code> or <code>MixNRM2</code> , <code>MixMRM1cens</code> or <code>MixMRM2cens</code>
<code>qq_plot</code>	Whether to compute the QQ-plot
<code>thinning_to</code>	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 <- MixNRM1cens(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

<code>fit</code>	The result of the fit, obtained through the function <code>MixNRMI1</code> or <code>MixNRMI2</code> , <code>MixMRMI1cens</code> or <code>MixMRMI2cens</code>
<code>qq_plot</code>	Whether to compute the QQ-plot
<code>thinning_to</code>	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(salinty$left, salinty$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

<code>fit</code>	The result of the fit, obtained through the function <code>MixNRMI1</code> or <code>MixNRMI2</code> , <code>MixMRMI1cens</code> or <code>MixMRMI2cens</code>
<code>qq_plot</code>	Whether to compute the QQ-plot
<code>thinning_to</code>	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	<i>Create a plotting grid from censored or non-censored data.</i>
----------------	---

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	<i>Create a plotting grid from censored data.</i>
-------------------------	---

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

<code>data</code>	Non-censored input data from which to compute the grid.
<code>npoints</code>	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

<code>dat</code>	The dataset to be tested
------------------	--------------------------

Value

TRUE if the data is censored

Examples

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

<code>is_semiparametric</code>	<i>Tests if a fit is a semi parametric or nonparametric model.</i>
--------------------------------	--

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 = 1, distr.p0 = 1, asigma = 0.5,
  bsigma = 0.5, delta = 3, Delta = 2, Meps = 0.01, Nx = 150,
  Nit = 1500, Pbi = 0.1, 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 details.
Kappa	Numeric positive constant. See details.
Gama	Numeric constant. $0 \leq \text{Gama} \leq 1$. See details.
distr.k	Integer number identifying the mixture kernel: 1 = Normal; 2 = Gamma; 3 = Beta; 4 = Double Exponential; 5 = Lognormal.
distr.p0	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 <code>distr.k</code> .
bsigma	Numeric positive constant. Rate parameter of the gamma prior on the standard deviation of the mixture kernel <code>distr.k</code> .
delta	Numeric positive constant. Metropolis-Hastings proposal variation coefficient for sampling <code>sigma</code> .
Delta	Numeric positive constant. Metropolis-Hastings proposal variation coefficient for sampling the latent <code>U</code> .
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 <code>Nit</code> .
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 <code>Js</code> .

Details

This generic function fits a normalized random measure (NRM) 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:

$$\begin{aligned}
 X_i | Y_i, \sigma &\sim k(\cdot | Y_i, \sigma) \\
 Y_i | P &\sim P, \quad i = 1, \dots, n \\
 P &\sim \text{NGG}(\text{Alpha}, \text{Kappa}, \text{Gama}; P_\theta) \\
 \sigma &\sim \text{Gamma}(\text{asigma}, \text{bsigma})
 \end{aligned}$$

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, $(\text{Alpha}, \text{Kappa}, \text{Gama};$

P_{θ}) are the parameters of the NGG prior with P_{θ} being the centering measure whose parameters are assigned vague hyper prior distributions, and (a_{σ}, b_{σ}) are the hyper-parameters of the gamma prior on the smoothing parameter σ . In particular: $\text{NGG}(\text{Alpha}, 1, \theta; P_{\theta})$ defines a Dirichlet process; $\text{NGG}(1, \text{Kappa}, 1/2; P_{\theta})$ defines a Normalized inverse Gaussian process; and $\text{NGG}(1, \theta, \text{Gama}; P_{\theta})$ defines a normalized stable process.

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

Value

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

xx	Numeric vector. Evaluation grid.
qx	Numeric array. Matrix of dimension $N_x \times (\text{length}(\text{probs}) + 1)$ with the posterior mean and the desired quantiles input in <code>probs</code> .
cpo	Numeric vector of length(x) with conditional predictive ordinates.
R	Numeric vector of length($\text{Nit} * (1 - \text{Pbi})$) with the number of mixtures components (clusters).
S	Numeric vector of length($\text{Nit} * (1 - \text{Pbi})$) with the values of common standard deviation σ .
U	Numeric vector of length($\text{Nit} * (1 - \text{Pbi})$) with the values of the latent variable U.
Allocs	List of length($\text{Nit} * (1 - \text{Pbi})$) with the clustering allocations.
means	List of length($\text{Nit} * (1 - \text{Pbi})$) with the cluster means (locations). Only if <code>extras = TRUE</code> .
weights	List of length($\text{Nit} * (1 - \text{Pbi})$) with the mixture weights. Only if <code>extras = TRUE</code> .
Js	List of length($\text{Nit} * (1 - \text{Pbi})$) with the unnormalized weights (jump sizes). Only if <code>extras = TRUE</code> .
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 <code>proc.time</code> function.
distr.k	Integer corresponding to the kernel chosen for the mixture
data	Data used for the fit
NRMII_params	A named list with the parameters of the NRMII process

Warning

The function is computing intensive. Be patient.

Author(s)

Barrios, E., Lijoi, A., Nieto-Barajas, L.E. and Pruenster, I.

References

- 1.- Barrios, E., Lijoi, A., Nieto-Barajas, L. E. and Prüenster, I. (2013). Modeling with Normalized Random Measure Mixture Models. *Statistical Science*. Vol. 28, No. 3, 313-334.
- 2.- James, L.F., Lijoi, A. and Prüenster, 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
attach(out)
m <- ncol(qx)
ymax <- max(qx[, m])
par(mfrow = c(1, 1))
hist(x, probability = TRUE, breaks = 20, col = grey(.9), ylim = c(0, ymax))
lines(xx, qx[, 1], lwd = 2)
lines(xx, qx[, 2], lty = 3, col = 4)
lines(xx, qx[, m], lty = 3, col = 4)
detach()

## End(Not run)

### Example 2
## Do not run
# set.seed(150520)
# data(enzyme)
# x <- enzyme
# Enzyme1.out <- MixNRMI1(x, Alpha = 1, Kappa = 0.007, Gama = 0.5,
#                          distr.k = 2, distr.p0 = 2, asigma = 1, bsigma = 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(Enzyme1.out)
attach(Enzyme1.out)
# Plotting density estimate + 95% credible interval
m <- ncol(qx)
ymax <- max(qx[, m])
```

```

par(mfrow = c(1, 1))
hist(x, probability = TRUE, breaks = 20, col = grey(.9), ylim = c(0, ymax))
lines(xx, qx[, 1], lwd = 2)
lines(xx, qx[, 2], lty = 3, col = 4)
lines(xx, qx[, m], lty = 3, col = 4)
# 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()

### 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 = 1, distr.p0 = 2, asigma = 1, bsigma = 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(Galaxy1.out)
attach(Galaxy1.out)
# Plotting density estimate + 95% credible interval
m <- ncol(qx)
ymax <- max(qx[, m])
par(mfrow = c(1, 1))
hist(x, probability = TRUE, breaks = 20, col = grey(.9), ylim = c(0, ymax))
lines(xx, qx[, 1], lwd = 2)
lines(xx, qx[, 2], lty = 3, col = 4)
lines(xx, qx[, m], lty = 3, col = 4)
# 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 = 1, distr.p0 = 1, asigma = 0.5,
  bsigma = 0.5, delta = 3, Delta = 2, Meps = 0.01, Nx = 150,
  Nit = 1500, Pbi = 0.1, epsilon = NULL, printtime = TRUE,
  extras = TRUE)

```

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. $0 \leq \text{Gama} \leq 1$. See details.
distr.k	Integer number identifying the mixture kernel: 1 = Normal; 2 = Gamma; 3 = Beta; 4 = Double Exponential; 5 = Lognormal.
distr.p0	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 <code>distr.k</code> .
bsigma	Numeric positive constant. Rate parameter of the gamma prior on the standard deviation of the mixture kernel <code>distr.k</code> .
delta	Numeric positive constant. Metropolis-Hastings proposal variation coefficient for sampling sigma.
Delta	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.

Details

This generic function fits a normalized random measure (NRM) 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:

$$\begin{aligned}
 X_i | Y_i, \sigma &\sim k(\cdot | Y_i, \sigma) \\
 Y_i | P &\sim P, \quad i = 1, \dots, n \\
 P &\sim \text{NGG}(\text{Alpha}, \text{Kappa}, \text{Gama}; P_\theta) \\
 \sigma &\sim \text{Gamma}(\text{asigma}, \text{bsigma})
 \end{aligned}$$

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_θ`) are the parameters of the NGG prior with `P_θ` 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_θ)` defines a Dirichlet process; `NGG(1, Kappa, 1/2; P_θ)` defines a Normalized inverse Gaussian process; and `NGG(1, 0, Gama; P_θ)` defines a normalized stable process.

The evaluation grid ranges from $\min(x) - \text{epsilon}$ to $\max(x) + \text{epsilon}$. 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 $N_x \times (\text{length}(\text{probs}) + 1)$ 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($N_{it}*(1-P_{bi})$) with the number of mixtures components (clusters).
S	Numeric vector of length($N_{it}*(1-P_{bi})$) with the values of common standard deviation sigma.
U	Numeric vector of length($N_{it}*(1-P_{bi})$) with the values of the latent variable U.
Allocs	List of length($N_{it}*(1-P_{bi})$) with the clustering allocations.
means	List of length($N_{it}*(1-P_{bi})$) with the cluster means (locations). Only if extras = TRUE.
weights	List of length($N_{it}*(1-P_{bi})$) with the mixture weights. Only if extras = TRUE.
J _s	List of length($N_{it}*(1-P_{bi})$) with the unnormalized weights (jump sizes). Only if extras = TRUE.
N _m	Integer constant. Number of jumps of the continuous component of the unnormalized process.
N _x	Integer constant. Number of grid points for the evaluation of the density estimate.
N _{it}	Integer constant. Number of MCMC iterations.
P _{bi}	Numeric constant. Burn-in period proportion of N _{it} .
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 Pruenster, I. (2013). Modeling with Normalized Random Measure Mixture Models. *Statistical Science*. Vol. 28, No. 3, 313-334.
- 2.- James, L.F., Lijoi, A. and Pruenster, 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 Pruenster, 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
attach(out)
m <- ncol(qx)
ymax <- max(qx[, m])
par(mfrow = c(1, 1))
hist(x, probability = TRUE, breaks = 20, col = grey(.9), ylim = c(0, ymax))
lines(xx, qx[, 1], lwd = 2)
lines(xx, qx[, 2], lty = 3, col = 4)
lines(xx, qx[, m], lty = 3, col = 4)
detach()

## 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)
m <- ncol(qx)
ymax <- max(qx[, m])
par(mfrow = c(1, 1))
plot(xx, qx$"q0.5", lwd = 2, type = "l", ylab = "Density", xlab = "Data")
lines(xx, qx[, 2], lty = 3, col = 4)
lines(xx, qx[, m], lty = 3, col = 4)
# 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)

```

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 = 1, distr.py0 = 1, distr.pz0 = 2,
  mu.pz0 = 3, sigma.pz0 = sqrt(10), delta = 4, kappa = 2,
  Delta = 2, Meps = 0.01, Nx = 150, Nit = 1500, Pbi = 0.1,
  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 details.
Kappa	Numeric positive constant. See details.
Gama	Numeric constant. $0 \leq Gama \leq 1$. See details.
distr.k	Integer number identifying the mixture kernel: 1 = Normal; 2 = Gamma; 3 = Beta; 4 = Double Exponential; 5 = Lognormal.
distr.py0	Integer number identifying the centering measure for locations: 1 = Normal; 2 = Gamma; 3 = Beta.
distr.pz0	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	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	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, sigmas, weights and Js.

Details

This generic function fits a normalized random measure (NRM) 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:

$$\begin{aligned}
 X_i | Y_i, Z_i &\sim k(\cdot | Y_i, Z_i) \\
 (Y_i, Z_i) | P &\sim P, i = 1, \dots, n \\
 P &\sim \text{NGG}(\text{Alpha}, \text{Kappa}, \text{Gama}; P_\theta)
 \end{aligned}$$

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, $(\text{Alpha}, \text{Kappa}, \text{Gama}; P_\theta)$ are the parameters of the NGG prior with a bivariate P_θ being the centering measure with independent components, that is, $P_\theta(Y, Z) = P_\theta(Y) * P_\theta(Z)$. The parameters of $P_\theta(Y)$ are assigned vague hyper prior distributions and $(\text{mu.pz}\theta, \text{sigma.pz}\theta)$ are the hyper-parameters of $P_\theta(Z)$. In particular, $\text{NGG}(\text{Alpha}, 1, \theta; P_\theta)$ defines a Dirichlet process; $\text{NGG}(1, \text{Kappa}, 1/2; P_\theta)$ defines a Normalized inverse Gaussian process; and $\text{NGG}(1, \theta, \text{Gama}; P_\theta)$ defines a normalized stable process. The evaluation grid ranges from $\min(x) - \text{epsilon}$ to $\max(x) + \text{epsilon}$. By default $\text{epsilon} = \text{sd}(x)/4$.

Value

The function returns a list with the following components:

xx	Numeric vector. Evaluation grid.
qx	Numeric array. Matrix of dimension $N_x \times (\text{length}(\text{probs}) + 1)$ with the posterior mean and the desired quantiles input in probs.
cpo	Numeric vector of $\text{length}(x)$ with conditional predictive ordinates.
R	Numeric vector of $\text{length}(\text{Nit} * (1 - \text{Pbi}))$ with the number of mixtures components (clusters).
U	Numeric vector of $\text{length}(\text{Nit} * (1 - \text{Pbi}))$ with the values of the latent variable U.
Allocs	List of $\text{length}(\text{Nit} * (1 - \text{Pbi}))$ with the clustering allocations.
means	List of $\text{length}(\text{Nit} * (1 - \text{Pbi}))$ with the cluster means (locations). Only if extras = TRUE.
sigmas	Numeric vector of $\text{length}(\text{Nit} * (1 - \text{Pbi}))$ with the cluster standard deviations. Only if extras = TRUE.
weights	List of $\text{length}(\text{Nit} * (1 - \text{Pbi}))$ 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., Lijoi, A., Nieto-Barajas, L.E. and Prüenster, I.

References

- 1.- Barrios, E., Lijoi, A., Nieto-Barajas, L. E. and Prüenster, I. (2013). Modeling with Normalized Random Measure Mixture Models. Statistical Science. Vol. 28, No. 3, 313-334.
- 2.- James, L.F., Lijoi, A. and Prüenster, I. (2009). Posterior analysis for normalized random measure with independent increments. Scand. J. Statist 36, 76-97.

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
attach(out)
m <- ncol(qx)
ymax <- max(qx[, m])
par(mfrow = c(1, 1))
hist(x, probability = TRUE, breaks = 20, col = grey(.9), ylim = c(0, ymax))
```

```

lines(xx, qx[, 1], lwd = 2)
lines(xx, qx[, 2], lty = 3, col = 4)
lines(xx, qx[, m], lty = 3, col = 4)
detach()

## End(Not run)

### Example 2
## Do not run
# set.seed(150520)
# data(enzyme)
# x <- enzyme
# Enzyme2.out <- MixNRM12(x, Alpha = 1, Kappa = 0.007, Gama = 0.5,
#                         distr.k = 2, distr.py0 = 2,
#                         distr.pz0 = 2, 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
m <- ncol(qx)
ymax <- max(qx[, m])
par(mfrow = c(1, 1))
hist(x, probability = TRUE, breaks = 20, col = grey(.9), ylim = c(0, ymax))
lines(xx, qx[, 1], lwd = 2)
lines(xx, qx[, 2], lty = 3, col = 4)
lines(xx, qx[, m], lty = 3, col = 4)
# 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 <- MixNRM12(x, Alpha = 1, Kappa = 0.015, Gama = 0.5,

```

```

#           distr.k = 1, distr.py0 = 2,
#           distr.pz0 = 2, 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
m <- ncol(qx)
ymax <- max(qx[, m])
par(mfrow = c(1, 1))
hist(x, probability = TRUE, breaks = 20, col = grey(.9), ylim = c(0, ymax))
lines(xx, qx[, 1], lwd = 2)
lines(xx, qx[, 2], lty = 3, col = 4)
lines(xx, qx[, m], lty = 3, col = 4)
# 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 = 1, distr.py0 = 1,
  distr.pz0 = 2, mu.pz0 = 3, sigma.pz0 = sqrt(10), delta = 4,
  kappa = 2, Delta = 2, Meps = 0.01, Nx = 150, Nit = 1500,
  Pbi = 0.1, epsilon = NULL, printtime = TRUE, extras = TRUE)

```

Arguments

<code>xleft</code>	Numeric vector. Lower limit of interval censoring. For exact data the same as <code>xright</code>
<code>xright</code>	Numeric vector. Upper limit of interval censoring. For exact data the same as <code>xleft</code> .
<code>probs</code>	Numeric vector. Desired quantiles of the density estimates.
<code>Alpha</code>	Numeric constant. Total mass of the centering measure. See details.
<code>Kappa</code>	Numeric positive constant. See details.
<code>Gama</code>	Numeric constant. $0 \leq Gama \leq 1$. See details.
<code>distr.k</code>	Integer number identifying the mixture kernel: 1 = Normal; 2 = Gamma; 3 = Beta; 4 = Double Exponential; 5 = Lognormal.
<code>distr.py0</code>	Integer number identifying the centering measure for locations: 1 = Normal; 2 = Gamma; 3 = Beta.
<code>distr.pz0</code>	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.
<code>mu.pz0</code>	Numeric constant. Prior mean of the centering measure for scales.
<code>sigma.pz0</code>	Numeric constant. Prior standard deviation of the centering measure for scales.
<code>delta</code>	Numeric positive constant. Metropolis-Hastings proposal variation coefficient for sampling the scales.
<code>kappa</code>	Numeric positive constant. Metropolis-Hastings proposal variation coefficient for sampling the location parameters.
<code>Delta</code>	Numeric positive constant. Metropolis-Hastings proposal variation coefficient for sampling the latent U.
<code>Meps</code>	Numeric constant. Relative error of the jump sizes in the continuous component of the process. Smaller values imply larger number of jumps.
<code>Nx</code>	Integer constant. Number of grid points for the evaluation of the density estimate.
<code>Nit</code>	Integer constant. Number of MCMC iterations.
<code>Pbi</code>	Numeric constant. Burn-in period proportion of <code>Nit</code> .
<code>epsilon</code>	Numeric constant. Extension to the evaluation grid range. See details.
<code>printtime</code>	Logical. If TRUE, prints out the execution time.
<code>extras</code>	Logical. If TRUE, gives additional objects: means, sigmas, weights and Js.

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:

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

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

$$P \sim \text{NGG}(\text{Alpha}, \text{Kappa}, \text{Gama}; P_0)$$

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, $(\text{Alpha}, \text{Kappa}, \text{Gama}; P_0)$ 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 $(\text{mu.pz0}, \text{sigma.pz0})$ are the hyper-parameters of $P_0(Z)$. In particular, $\text{NGG}(\text{Alpha}, 1, \theta; P_0)$ defines a Dirichlet process; $\text{NGG}(1, \text{Kappa}, 1/2; P_0)$ defines a Normalized inverse Gaussian process; and $\text{NGG}(1, \theta, \text{Gama}; P_0)$ defines a normalized stable process. The evaluation grid ranges from $\min(x) - \text{epsilon}$ to $\max(x) + \text{epsilon}$. By default $\text{epsilon} = \text{sd}(x)/4$.

Value

The function returns a list with the following components:

xx	Numeric vector. Evaluation grid.
qx	Numeric array. Matrix of dimension $N_x \times (\text{length}(\text{probs}) + 1)$ 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.
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
NRM1_params	A named list with the parameters of the NRM1 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 Prüenster, I. (2013). Modeling with Normalized Random Measure Mixture Models. *Statistical Science*. Vol. 28, No. 3, 313-334.
- 2.- James, L.F., Lijoi, A. and Prüenster, 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 Prüenster, 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
attach(out)
m <- ncol(qx)
ymax <- max(qx[, m])
par(mfrow = c(1, 1))
hist(x, probability = TRUE, breaks = 20, col = grey(.9), ylim = c(0, ymax))
lines(xx, qx[, 1], lwd = 2)
lines(xx, qx[, 2], lty = 3, col = 4)
lines(xx, qx[, m], lty = 3, col = 4)
detach()

## 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)
m <- ncol(qx)
ymax <- max(qx[, m])
par(mfrow = c(1, 1))
plot(xx, qx$q0.5", lwd = 2, type = "l", ylab = "Density", xlab = "Data")
lines(xx, qx[, 2], lty = 3, col = 4)
lines(xx, qx[, m], lty = 3, col = 4)
# 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)

```

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 = 1, distr.p0 = 1, asigma = 0.5,
  bsigma = 0.5, delta = 3, Delta = 2, Meps = 0.01, Nx = 150,
  Nit = 1500, Pbi = 0.1, epsilon = NULL, printtime = TRUE,
  extras = TRUE, 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. $0 \leq \text{Gama} \leq 1$. See details.
distr.k	Integer number identifying the mixture kernel: 1 = Normal; 2 = Gamma; 3 = Beta; 4 = Double Exponential; 5 = Lognormal.
distr.p0	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.

<code>bsigma</code>	Numeric positive constant. Rate parameter of the gamma prior on the standard deviation of the mixture kernel distr.k.
<code>delta</code>	Numeric positive constant. Metropolis-Hastings proposal variation coefficient for sampling sigma.
<code>Delta</code>	Numeric positive constant. Metropolis-Hastings proposal variation coefficient for sampling the latent U.
<code>Meps</code>	Numeric constant. Relative error of the jump sizes in the continuous component of the process. Smaller values imply larger number of jumps.
<code>Nx</code>	Integer constant. Number of grid points for the evaluation of the density estimate.
<code>Nit</code>	Integer constant. Number of MCMC iterations.
<code>Pbi</code>	Numeric constant. Burn-in period proportion of Nit.
<code>epsilon</code>	Numeric constant. Extension to the evaluation grid range. See details.
<code>printtime</code>	Logical. If TRUE, prints out the execution time.
<code>extras</code>	Logical. If TRUE, gives additional objects: means, weights and Js (the jump sizes).
<code>nchains</code>	The number of chains to run.
<code>parallel</code>	Whether to run the chains in parallel. Only works on UNIX-like systems as it rests on Fork parallelism
<code>ncores</code>	Number of cores for the parallel run. Defaults to <code>parallel::detectCores()</code> , 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 = 1, distr.p0 = 1,
  asigma = 0.5, bsigma = 0.5, delta = 3, Delta = 2, Meps = 0.01,
  Nx = 150, Nit = 1500, Pbi = 0.1, epsilon = NULL,
  printtime = TRUE, extras = TRUE, 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. $0 \leq \text{Gama} \leq 1$. See details.
distr.k	Integer number identifying the mixture kernel: 1 = Normal; 2 = Gamma; 3 = Beta; 4 = Double Exponential; 5 = Lognormal.
distr.p0	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	Numeric positive constant. Metropolis-Hastings proposal variation coefficient for sampling sigma.
Delta	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 (the jump sizes).
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 = 1, distr.py0 = 1, distr.pz0 = 2,
  mu.pz0 = 3, sigma.pz0 = sqrt(10), delta = 4, kappa = 2,
  Delta = 2, Meps = 0.01, Nx = 150, Nit = 1500, Pbi = 0.1,
  epsilon = NULL, printtime = TRUE, extras = TRUE, nchains = 4,
  parallel = FALSE, ncores = parallel::detectCores())
```

Arguments

<code>x</code>	Numeric vector. Data set to which the density is fitted.
<code>probs</code>	Numeric vector. Desired quantiles of the density estimates.
<code>Alpha</code>	Numeric constant. Total mass of the centering measure. See details.
<code>Kappa</code>	Numeric positive constant. See details.
<code>Gama</code>	Numeric constant. $0 \leq \text{Gama} \leq 1$. See details.
<code>distr.k</code>	Integer number identifying the mixture kernel: 1 = Normal; 2 = Gamma; 3 = Beta; 4 = Double Exponential; 5 = Lognormal.
<code>distr.py0</code>	Integer number identifying the centering measure for locations: 1 = Normal; 2 = Gamma; 3 = Beta.
<code>distr.pz0</code>	Integer number identifying the centering measure for scales: 2 = Gamma. For more options use <code>MixNRMI2cens</code> .
<code>mu.pz0</code>	Numeric constant. Prior mean of the centering measure for scales.
<code>sigma.pz0</code>	Numeric constant. Prior standard deviation of the centering measure for scales.
<code>delta</code>	Numeric positive constant. Metropolis-Hastings proposal variation coefficient for sampling the scales.
<code>kappa</code>	Numeric positive constant. Metropolis-Hastings proposal variation coefficient for sampling the location parameters.
<code>Delta</code>	Numeric positive constant. Metropolis-Hastings proposal variation coefficient for sampling the latent U.
<code>Meps</code>	Numeric constant. Relative error of the jump sizes in the continuous component of the process. Smaller values imply larger number of jumps.
<code>Nx</code>	Integer constant. Number of grid points for the evaluation of the density estimate.
<code>Nit</code>	Integer constant. Number of MCMC iterations.
<code>Pbi</code>	Numeric constant. Burn-in period proportion of Nit.
<code>epsilon</code>	Numeric constant. Extension to the evaluation grid range. See details.
<code>printtime</code>	Logical. If TRUE, prints out the execution time.
<code>extras</code>	Logical. If TRUE, gives additional objects: means, sigmas, weights and Js.
<code>nchains</code>	The number of chains to run.
<code>parallel</code>	Whether to run the chains in parallel. Only works on UNIX-like systems as it rests on Fork parallelism
<code>ncores</code>	Number of cores for the parallel run. Defaults to <code>parallel::detectCores()</code> , 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	<i>Multiple chains of MixNRMI2cens</i>
------------------	--

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 = 1, distr.py0 = 1,
  distr.pz0 = 2, mu.pz0 = 3, sigma.pz0 = sqrt(10), delta = 4,
  kappa = 2, Delta = 2, Meps = 0.01, Nx = 150, Nit = 1500,
  Pbi = 0.1, epsilon = NULL, printtime = TRUE, extras = TRUE,
  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. $0 \leq \text{Gama} \leq 1$. See details.
distr.k	Integer number identifying the mixture kernel: 1 = Normal; 2 = Gamma; 3 = Beta; 4 = Double Exponential; 5 = Lognormal.
distr.py0	Integer number identifying the centering measure for locations: 1 = Normal; 2 = Gamma; 3 = Beta.
distr.pz0	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	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	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, sigmas, weights and Js.
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)
```

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	<i>Plot the density estimate and the 95% credible interval</i>
------------	--

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

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

plot.NRMI1cens	<i>Plot the density estimate and the 95% credible interval</i>
----------------	--

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

```
## S3 method for class 'NRMI1cens'
plot(x, ...)
```

Arguments

x	A fitted object of class MixNRMI1cens
...	Further arguments to be passed to generic functions, ignored at the moment

Value

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

Examples

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

plot.NRMI2	<i>Plot the density estimate and the 95% credible interval</i>
------------	--

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

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

plot.NRMI2cens	<i>Plot the density estimate and the 95% credible interval</i>
----------------	--

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

```
## S3 method for class 'NRMI2cens'
plot(x, ...)
```

Arguments

x	A fitted object of class MixNRMI2cens
...	Further arguments to be passed to generic functions, ignored at the moment

Value

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

Examples

```
data(salinity)
out <- MixNRMI2cens(salinity$left, salinity$right, 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	<i>Plot the density estimate and the 95% credible interval for censored data</i>
------------------	--

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, the 95% credible interval

Examples

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

plotfit_noncensored	<i>Plot the density estimate and the 95% credible interval for noncensored data</i>
---------------------	---

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 visualise 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. $0 \leq \text{Gama} \leq 1$.
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 `MixNRMI2cens`.

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

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

```
print.NRMI1cens      S3 method for class 'MixNRMI1cens'
```

Description

S3 method for class 'MixNRMI1cens'

Usage

```
## S3 method for class 'NRMI1cens'
print(x, ...)
```

Arguments

```
x          A fitted object of class MixNRMI1cens
...        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 <- 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

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

```
print.NRMI2cens      S3 method for class 'MixNRMI2cens'
```

Description

S3 method for class 'MixNRMI2cens'

Usage

```
## S3 method for class 'NRMI2cens'
print(x, ...)
```

Arguments

```
x                A fitted object of class MixNRMI2cens
...              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 <- MixNRMI2cens(salinity$left, salinity$right, Nit = 50)
print(out)
```

```
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., Nuggeoda, 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	<i>S3 method for class 'multNRMI'</i>
------------------	---------------------------------------

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

summary.NRMI1cens *S3 method for class 'MixNRMI1cens'*

Description

S3 method for class 'MixNRMI1cens'

Usage

```
## S3 method for class 'NRMI1cens'
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

Summary of some relevant information on the fit

Examples

```
data(salinity)
out <- MixNRMI1cens(salinity$left, salinity$right, 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 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(acidity)
out <- MixNRMI2(acidity, Nit = 50)
summary(out)
```

```
summary.NRMI2cens      S3 method for class 'MixNRMI2cens'
```

Description

S3 method for class 'MixNRMI2cens'

Usage

```
## S3 method for class 'NRMI2cens'
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 <- MixNRMI2cens(salinity$left, salinity$right, 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, number_of_clusters = FALSE)
```

Arguments

```
fit              NRMIx object
kernel_comment   Text specific to the parametric and nonparametric nature of the model
number_of_clusters
                 Flag to decide whether to compute the optimal clustering
```

Value

Prints out the text for the summary S3 methods

traceplot	<i>Draw a traceplot for multiple chains</i>
-----------	---

Description

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

Usage

```
traceplot(fitlist)
```

Arguments

fitlist Output of multMixNRMI.

Value

A traceplot for multiple chains.

Index

- *Topic **datasets**
 - acidity, 4
 - enzyme, 8
 - Enzyme1.out, 9
 - Enzyme2.out, 9
 - galaxy, 12
 - Galaxy1.out, 12
 - Galaxy2.out, 13
 - salinity, 55
- *Topic **distribution**
 - MixNRMI1, 18
 - MixNRMI1cens, 23
 - MixNRMI2, 27
 - MixNRMI2cens, 31
- *Topic **models**
 - MixNRMI1, 18
 - MixNRMI1cens, 23
 - MixNRMI2, 27
 - MixNRMI2cens, 31
- *Topic **nonparametrics**
 - MixNRMI1, 18
 - MixNRMI1cens, 23
 - MixNRMI2, 27
 - MixNRMI2cens, 31
- *Topic **package**
 - BNPdensity-package, 3
- acidity, 4
- add, 5
- as.mcmc.multNRMI, 5
- asNumeric_no_warning, 6
- BNPdensity (BNPdensity-package), 3
- BNPdensity-package, 3
- comment_on_NRMI_type, 6
- compute_optimal_clustering, 7, 56–59
- compute_thinning_grid, 7
- convert_to_mcmc, 8
- enzyme, 8
- Enzyme1.out, 9
- Enzyme2.out, 9
- expected_number_of_components_Dirichlet, 10
- expected_number_of_components_stable, 10
- fill_sigmas, 11
- galaxy, 12
- Galaxy1.out, 12
- Galaxy2.out, 13
- give_kernel_name, 13
- GOFplots, 14
- GOFplots_censored, 14
- GOFplots_noncensored, 15
- grid_from_data, 16
- grid_from_data_censored, 16
- grid_from_data_noncensored, 17
- is_censored, 17
- is_semiparametric, 18
- MixNRMI1, 4, 18, 24
- MixNRMI1cens, 4, 21, 23, 26, 29, 34, 36, 38, 39, 41
- MixNRMI2, 4, 21, 26, 27, 29, 34, 36, 38, 39, 41
- MixNRMI2cens, 4, 21, 26, 27, 29, 31, 34, 36, 38, 39, 41
- multMixNRMI1, 21, 26, 29, 34, 35, 38, 39, 41
- multMixNRMI1cens, 37
- multMixNRMI2, 38
- multMixNRMI2cens, 40
- plot.multNRMI, 41
- plot.NRMI1, 42
- plot.NRMI1cens, 43
- plot.NRMI2, 43
- plot.NRMI2cens, 44
- plot_clustering_and_CDF, 48
- plot_prior_number_of_components, 49

plotCDF_censored, 45
plotCDF_noncensored, 45
plotfit_censored, 46
plotfit_noncensored, 46
plotPDF_censored, 47
plotPDF_noncensored, 48
pp_plot_censored, 50
pp_plot_noncensored, 50
print.multNRMI, 51
print.NRMI1, 52
print.NRMI1cens, 52
print.NRMI2, 53
print.NRMI2cens, 54

qq_plot_censored, 54
qq_plot_noncensored, 55

salinity, 55
summary.multNRMI, 56
summary.NRMI1, 57
summary.NRMI1cens, 57
summary.NRMI2, 58
summary.NRMI2cens, 59
summarytext, 59

traceplot, 60