Package: BNPmix (via r-universe)

October 17, 2026

Type Package

Title Bayesian Nonparametric Mixture Models

Version 1.2.3

Date 2026-08-04

Description Functions to perform Bayesian nonparametric univariate and multivariate density estimation and clustering, by means of Pitman-Yor mixtures, and dependent Dirichlet process mixtures for partially exchangeable data. See Corradin et al. (2021) <doi:10.18637/jss.v100.i15> for more details.

License LGPL-3 | file LICENSE

NeedsCompilation yes

Imports methods, stats, ggplot2, coda, ggpubr

Depends R (>= 3.5.0), Rcpp

LinkingTo RcppArmadillo, Rcpp(>= 0.12.13), RcppDist

Suggests R.rsp

VignetteBuilder R.rsp

RoxygenNote 7.3.3

Encoding UTF-8

Author Riccardo Corradin [aut, cre], Antonio Canale [ctb], Bernardo Nipoti [ctb]

Maintainer Riccardo Corradin <riccardo.corradin@gmail.com>

Repository https://cran.r-universe.dev

Date/Publication 2026-08-04 20:40:02 UTC

RemoteUrl https://github.com/cran/BNPmix

RemoteRef HEAD

RemoteSha e7011858e87c5d47596f7eb01b647d378ad53132

Contents

BNPdens 2
BNPdens2coda.BNPdens 3
BNPpart 4
dBNPdens.BNPdens 5
DDPdensity 5
partition.BNPdens 8
plot.BNPdens 9
print.BNPdens 11
PYcalibrate 12
PYdensity 12
PYregression 17
summary.BNPdens 20

Index **22**

`BNPdens` *BNPdens class constructor*

Description

A constructor for the `BNPdens` class. The class `BNPdens` is a named list containing the output generated by a specified Bayesian nonparametric mixture model implemented by means of a specified MCMC strategy, as in `PYdensity`, `DDPdensity`, and `PYregression`.

Usage

```
BNPdens(
  density = NULL,
  data = NULL,
  grideval = NULL,
  grid_x = NULL,
  grid_y = NULL,
  clust = NULL,
  mean = NULL,
  beta = NULL,
  sigma2 = NULL,
  probs = NULL,
  niter = NULL,
  nburn = NULL,
  tot_time = NULL,
  univariate = TRUE,
  regression = FALSE,
  dep = FALSE,
  group_log = NULL,
  group = NULL,
  wvals = NULL
)
```

Arguments

density	a matrix containing the values taken by the density at the grid points;
data	a dataset;
grideval	a set of values where to evaluate the density;
grid_x	regression grid, independent variable;
grid_y	regression grid, dependent variable;
clust	a (niter - nburn) $\times$ nrow(data)-dimensional matrix containing the cluster labels for each observation (cols) and MCMC iteration (rows);
mean	values for the location parameters;
beta	coefficients for regression model (only for PYregression);
sigma2	values of the scale parameters;
probs	values for the mixture weights;
niter	number of MCMC iterations;
nburn	number of MCMC iterations to discard as burn-in;
tot_time	total execution time;
univariate	logical, TRUE if the model is univariate;
regression	logical, TRUE for the output of PYregression;
dep	logical, TRUE for the output of DDPdensity;
group_log	group allocation for each iteration (only for DDPdensity);
group	vector, allocation of observations to strata (only for DDPdensity);
wvals	values of the processes weights (only for DDPdensity).

Examples

```
data_toy <- c(rnorm(100, -3, 1), rnorm(100, 3, 1))
grid <- seq(-7, 7, length.out = 50)
est_model <- PYdensity(y = data_toy, mcmc = list(niter = 100,
                       nburn = 10, nupd = 100), output = list(grid = grid))
str(est_model)
class(est_model)
```

BNPdens2coda.BNPdens *Export to coda interface*

Description

The method BNPdens2coda converts a BNPdens object into a coda mcmc object.

Usage

```
## S3 method for class 'BNPdens'
BNPdens2coda(object, dens = FALSE)
```

Arguments

object	a BNPdens object;
dens	logical, it can be TRUE only for models estimated with PYdensity. If TRUE, it converts to coda also the estimated density. Default FALSE.

Value

an mcmc object

Examples

```
data_toy <- cbind(c(rnorm(100, -3, 1), rnorm(100, 3, 1)),
                  c(rnorm(100, -3, 1), rnorm(100, 3, 1)))
grid <- expand.grid(seq(-7, 7, length.out = 50),
                    seq(-7, 7, length.out = 50))
est_model <- PYdensity(y = data_toy, mcmc = list(niter = 200, nburn = 100),
                       output = list(grid = grid))
coda_mcmc <- BNPdens2coda(est_model)
class(coda_mcmc)
```

BNPpart *BNPpart class constructor*

Description

A constructor for the BNPpart class. The class BNPpart is a named list containing the output of partition estimation methods.

Usage

```
BNPpart(partitions = NULL, scores = NULL, psm = NULL)
```

Arguments

partitions	a matrix, each row is a visited partition;
scores	a vector, each value is the score of a visited partition;
psm	a matrix, posterior similarity matrix.

Examples

```
data_toy <- c(rnorm(100, -3, 1), rnorm(100, 3, 1))
grid <- seq(-7, 7, length.out = 50)
est_model <- PYdensity(y = data_toy, mcmc = list(niter = 100,
                      nburn = 10, nupd = 100), output = list(grid = grid))
part <- partition(est_model)
class(part)
```

dBNPdens.BNPdens *Evaluate estimated univariate densities at a given point*

Description

The method dBNPdens provides an approximated evaluation of estimated univariate densities at a given point, for a BNPdens class object.

Usage

```
## S3 method for class 'BNPdens'
dBNPdens(object, x)
```

Arguments

object	a BNPdens object (only if univariate);
x	the point where to evaluate the density.

Value

a numeric value

Examples

```
data_toy <- c(rnorm(100, -3, 1), rnorm(100, 3, 1))
grid <- seq(-7, 7, length.out = 50)
est_model <- PYdensity(y = data_toy, mcmc = list(niter = 200, nburn = 100),
                       output = list(grid = grid))
x <- 1.4
dBNPdens(est_model, x)
```

DDPdensity *MCMC for GM-dependent Dirichlet process mixtures of Gaussians*

Description

The DDPdensity function generates posterior density samples for a univariate Griffiths-Milne dependent Dirichlet process mixture model with Gaussian kernel, for partially exchangeable data. The function implements the importance conditional sampler method.

Usage

```
DDPdensity(y, group, mcmc = list(), prior = list(), output = list())
```

Arguments

y	a vector or matrix giving the data based on which densities are to be estimated;
group	vector of length length(y) containing the group labels (integers) for the elements of y;
mcmc	list of MCMC arguments:

- niter (mandatory), number of iterations.
- nburn (mandatory), number of iterations to discard as burn-in.
- nupd, argument controlling the number of iterations to be displayed on screen: the function reports on standard output every time nupd new iterations have been carried out (default is niter/10).
- print_message, control option. If equal to TRUE, the status is printed to standard output every nupd iterations (default is TRUE).
- m_imp, number of generated values for the importance sampling step of the importance conditional sampler (default is 10). See details.
- var_MH_step, variance of the Gaussian proposal for the Metropolis-Hastings of the weights update (default is 0.25).

prior	a list giving the prior information, which contains:

- strength, the strength parameter, or total mass, of the marginal Dirichlet processes (default 1);
- m0, mean of the normal base measure on the location parameter (default is the sample mean of the data);
- k0, scale factor appearing in the normal base measure on the location parameter (default 1);
- a0, shape parameter of the inverse gamma base measure on the scale parameter (default 2);
- b0, scale parameter of the inverse gamma base measure on the scale parameter (default is the sample variance of the data);
- wei, parameter controlling the strength of dependence across Dirichlet processes (default 1/2).

output	a list of arguments for generating posterior output. It contains:

- grid, a grid of points at which to evaluate the estimated posterior mean densities (common for all the groups).
- out_type, if out_type = "FULL", return the estimated partitions and the realizations of the posterior density for each iterations. If out_type = "MEAN", return the estimated partitions and the mean of the densities sampled at each iterations. If out_type = "CLUST", return the estimated partitions. Default out_type = "FULL".

Details

This function fits a Griffiths-Milne dependent Dirichlet process (GM-DDP) mixture for density estimation for partially exchangeable data (Lijoi et al., 2014). For each observation the group variable allows the observations to be gathered into L=length(unique(group)) distinct groups. The

model assumes exchangeability within each group, with observations in the lth group marginally modelled by a location-scale Dirichlet process mixtures, i.e.

$$\tilde{f}_l(y) = \int \phi(y; \mu, \sigma^2)\tilde{p}_l(d\mu, d\sigma^2)$$

where each $\tilde{p}_l$ is a Dirichlet process with total mass `strength` and base measure P_0. The vector $\tilde{p} = (\tilde{p}_1, \ldots, \tilde{p}_L)$ is assumed to be jointly distributed as a vector of GM-DDP(`strength`, `wei`; P_0), where `strength` and P_0 are the total mass parameter and the base measure of each $\tilde{p}_l$, and `wei` controls the dependence across the components of $\tilde{p}$. Admissible values for `wei` are in $(0, 1)$, with the two extremes of the range corresponding to full exchangeability (`wei`$\to 0$) and independence across groups (`wei`$\to 1$).

P_0 is a normal-inverse gamma base measure, i.e.

$$P_0(d\mu, d\sigma^2) = N(d\mu; m_0, \sigma^2/k_0) \times IGa(d\sigma^2; a_0, b_0).$$

Posterior sampling is obtained by implementing the importance conditional sampler (Canale et al., 2019). See Corradin et al. (to appear) for more details.

Value

A `BNPdensity` class object containing the estimated densities for each iteration, the allocations for each iteration; the grid used to evaluate the densities (for each group); the densities sampled from the posterior distribution (for each group); the groups; the weights of the processes. The function returns also informations regarding the estimation: the number of iterations, the number of burn-in iterations and the execution time.

References

Lijoi, A., Nipoti, B., and Pruenster, I. (2014). Bayesian inference with dependent normalized completely random measures. Bernoulli 20, 1260–1291, doi:10.3150/13-BEJ521

Canale, A., Corradin, R., & Nipoti, B. (2019). Importance conditional sampling for Bayesian nonparametric mixtures. arXiv preprint arXiv:1906.08147

Corradin, R., Canale, A., Nipoti, B. (2021), BNPmix: An R Package for Bayesian Nonparametric Modeling via Pitman-Yor Mixtures, Journal of Statistical Software, doi:10.18637/jss.v100.i15

Examples

```
data_toy <- c(rnorm(50, -4, 1), rnorm(100, 0, 1), rnorm(50, 4, 1))
group_toy <- c(rep(1,100), rep(2,100))
grid <- seq(-7, 7, length.out = 50)
est_model <- DDPdensity(y = data_toy, group = group_toy,
mcmc = list(niter = 200, nburn = 100, var_MH_step = 0.25),
output = list(grid = grid))
summary(est_model)
plot(est_model)
```

`partition.BNPdens` *Estimate the partition of the data*

Description

The `partition` method estimates the partition of the data based on the output generated by a Bayesian nonparametric mixture model, according to a specified criterion, for a `BNPdens` class object.

Usage

```
## S3 method for class 'BNPdens'
partition(object, dist = "VI", max_k = NULL, ...)
```

Arguments

`object`	an object of class `BNPdens`;
`dist`	a loss function defined on the space of partitions; it can be variation of information (`"VI"`) or `"Binder"`, default `"VI"`. See details;
`max_k`	maximum number of clusters passed to the `cutree` function. See value below;
`...`	additional arguments to be passed.

Details

This method returns point estimates for the clustering of the data induced by a nonparametric mixture model. This result is achieved exploiting two different loss fuctions on the space of partitions: variation of information (`dist = 'VI'`) and Binder's loss (`dist = 'Binder'`). The function is based on the `mcclust.ext` code by Sara Wade (Wade and Ghahramani, 2018).

Value

The method returns a list containing a matrix with `nrow(data)` columns and 3 rows. Each row reports the cluster labels for each observation according to three different approaches, one per row. The first and second rows are the output of an agglomerative clustering procedure obtained by applying the function `hclust` to the dissimilarity matrix, and by using the complete or average linkage, respectively. The number of clusters is between 1 and `max_k` and is choosen according to a lower bound on the expected loss, as described in Wade and Ghahramani (2018). The third row reports the partition visited by the MCMC with the minimum distance `dist` from the dissimilarity matrix.

In addition, the list reports a vector with three scores representing the lower bound on the expected loss for the three partitions.

References

Wade, S., Ghahramani, Z. (2018). Bayesian cluster analysis: Point estimation and credible balls. Bayesian Analysis, 13, 559-626.

Examples

```
data_toy <- c(rnorm(10, -3, 1), rnorm(10, 3, 1))
grid <- seq(-7, 7, length.out = 50)
fit <- PYdensity(y = data_toy, mcmc = list(niter = 100,
                      nburn = 10, nupd = 100), output = list(grid = grid))
class(fit)
partition(fit)
```

`plot.BNPdens` *Density plot for BNPdens class*

Description

Extension of the `plot` method to the `BNPdens` class. The method `plot.BNPdens` returns suitable plots for a `BNPdens` object. See details.

Usage

```
## S3 method for class 'BNPdens'
plot(
  x,
  dimension = c(1, 2),
  col = "#0037c4",
  show_points = F,
  show_hist = F,
  show_clust = F,
  bin_size = NULL,
  wrap_dim = NULL,
  xlab = "",
  ylab = "",
  band = T,
  conf_level = c(0.025, 0.975),
  ...
)
```

Arguments

`x`	an object of class `BNPdens`;
`dimension`	if `x` is the output of a model fitted to multivariate data, `dimensions` specifies the two dimensions for the bivariate contour plot (if they are equal, a marginal univarite plot is returned);
`col`	the color of the lines;
`show_points`	if `TRUE`, the function plots also the observations, default `FALSE`;
`show_hist`	if `TRUE`, and the model is univariate, the function plots also the histogram of the data, default `FALSE`;

show_clust	if TRUE the function plots also the points colored with respect to the estimated partition, default FALSE;
bin_size	if show_hist = TRUE, it correponds to the size of each bin, default range(data) / 30;
wrap_dim	bivariate vector, if x is the output of DDPdensity, it correponds to the number of rows and columns in the plot. Default c(ngroup, 1);
xlab	label of the horizontal axis;
ylab	label of the vertical axis;
band	if TRUE and x is the output of a univariate model or of DDPdensity, the plot method displays quantile-based posterior credible bands along with estimated densities;
conf_level	bivariate vector, order of the quantiles for the posterior credible bands. Default c(0.025, 0.975);
...	additional arguments to be passed.

Details

If the BNPdens object is generated by PYdensity, the function returns the univariate or bivariate estimated density plot. If the BNPdens object is generated by PYregression, the function returns the scatterplot of the response variable jointly with the covariates (up to four), coloured according to the estimated partition. up to four covariates. If x is a BNPdens object generated by DDPdensity, the function returns a wrapped plot with one density per group. The plots can be enhanced in several ways: for univariate densities, if show_hist = TRUE, the plot shows also the histogram of the data; if show_points = TRUE, the plot shows also the observed points along the x-axis; if show_points = TRUE and show_clust = TRUE, the points are colored according to the partition estimated with the partition function. For multivariate densities: if show_points = TRUE, the plot shows also the scatterplot of the data; if show_points = TRUE and show_clust = TRUE, the points are colored according to the estimated partition.

Value

A ggplot2 object.

Examples

```
# PYdensity example
data_toy <- c(rnorm(100, -3, 1), rnorm(100, 3, 1))
grid <- seq(-7, 7, length.out = 50)
est_model <- PYdensity(y = data_toy,
 mcmc = list(niter = 200, nburn = 100, nupd = 100),
 output = list(grid = grid))
class(est_model)
plot(est_model)

# PYregression example
x_toy <- c(rnorm(100, 3, 1), rnorm(100, 3, 1))
y_toy <- c(x_toy[1:100] * 2 + 1, x_toy[101:200] * 6 + 1) + rnorm(200, 0, 1)
grid_x <- c(0, 1, 2, 3, 4, 5)
```

```
grid_y <- seq(0, 35, length.out = 50)
est_model <- PYregression(y = y_toy, x = x_toy,
mcmc = list(niter = 200, nburn = 100),
output = list(grid_x = grid_x, grid_y = grid_y))
summary(est_model)
plot(est_model)

# DDPdensity example
data_toy <- c(rnorm(50, -4, 1), rnorm(100, 0, 1), rnorm(50, 4, 1))
group_toy <- c(rep(1,100), rep(2,100))
grid <- seq(-7, 7, length.out = 50)
est_model <- DDPdensity(y = data_toy, group = group_toy,
mcmc = list(niter = 200, nburn = 100, napprox_unif = 50),
output = list(grid = grid))
summary(est_model)
plot(est_model)
```

`print.BNPdens` *BNPdens print method*

Description

The `BNPdens` method prints the type of a `BNPdens` object.

Usage

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

Arguments

`x`	an object of class `BNPdens`;
`...`	additional arguments.

Examples

```
data_toy <- c(rnorm(100, -3, 1), rnorm(100, 3, 1))
grid <- seq(-7, 7, length.out = 50)
est_model <- PYdensity(y = data_toy, mcmc = list(niter = 100,
                     nburn = 10, napprox = 10), output = list(grid = grid))
class(est_model)
print(est_model)
```

PYcalibrate *Pitman-Yor prior elicitation*

Description

The function PYcalibrate elicits the strength parameter of the Pitman-Yor process, given the discount parameter and the prior expected number of clusters.

Usage

```
PYcalibrate(Ek, n, discount = 0)
```

Arguments

Ek	prior expected number of cluster;
n	sample size;
discount	discount parameter; default is set equal to 0, corresponding to a Dirichlet process prior.

Value

A named list containingtthe values of strength and discount parameters.

Examples

```
PYcalibrate(5, 100)

PYcalibrate(5, 100, 0.5)
```

PYdensity *MCMC for Pitman-Yor mixtures of Gaussians*

Description

The PYdensity function generates a posterior density sample for a selection of univariate and multivariate Pitman-Yor process mixture models with Gaussian kernels. See details below for the description of the different specifications of the implemented models.

Usage

```
PYdensity(y, mcmc = list(), prior = list(), output = list())
```

Arguments

y	a vector or matrix giving the data based on which the density is to be estimated;
mcmc	a list of MCMC arguments:

- niter (mandatory), number of iterations.
- nburn (mandatory), number of iterations to discard as burn-in.
- method, the MCMC sampling method to be used, options are 'ICS', 'MAR' and 'SLI' (default is 'ICS'). See details.
- model, the type of model to be fitted (default is 'LS'). See details.
- nupd, argument controlling the number of iterations to be displayed on screen: the function reports on standard output every time nupd new iterations have been carried out (default is niter/10).
- print_message, control option. If equal to TRUE, the status is printed to standard output every nupd iterations (default is TRUE).
- m_imp, number of generated values for the importance sampling step of method = 'ICS' (default is 10). See details.
- slice_type, when method = 'SLI' it specifies the type of slice sampler. Options are 'DEP' for dependent slice-efficient, and 'INDEP' for independent slice-efficient (default is 'DEP'). See details.
- hyper, if equal to TRUE, hyperprior distributions on the base measure's parameters are added, as specified in prior and explained in details (default is TRUE).

prior — a list giving the prior information. The list includes strength and discount, the strength and discount parameters of the Pitman-Yor process (default are strength = 1 and discount = 0, the latter leading to the Dirichlet process). The remaining parameters depend on the model choice.

- If model = 'L' (location mixture) and y is univariate:
  m0 and s20 are mean and variance of the base measure on the location parameter (default are sample mean and sample variance of the data); a0 and b0 are shape and scale parameters of the inverse gamma prior on the common scale parameter (default are 2 and the sample variance of the data). If hyper = TRUE, optional hyperpriors on the base measure's parameters are added: specifically, m1 and k1 are the mean parameter and the scale factor defining the normal hyperprior on m0 (default are the sample mean of the data and 1), and a1 and b1 are shape and rate parameters of the inverse gamma hyperprior on b0 (default are 2 and the sample variance of the data). See details.
- If model = 'LS' (location-scale mixture) and y is univariate:
  m0 and k0 are the mean parameter and the scale factor defining the normal base measure on the location parameter (default are the sample mean of the data and 1), and a0 and b0 are shape and scale parameters of the inverse gamma base measure on the scale parameter (default are 2 and the sample variance of the data). If hyper = TRUE, optional hyperpriors on the base measure's parameters are added: specifically, m1 and s21 are mean and variance parameters of the normal hyperprior on m0 (default are the sample mean and the sample variance of the data); tau1 and zeta1 are shape

and rate parameters of the gamma hyperprior on k0 (default is 1 for both); a1 and b1 are shape and rate parameters of the gamma hyperprior on b0 (default are the sample variance of the data and 1). See details.

- If model = 'L' (location mixture) and y is multivariate (p-variate):
m0 and S20 are mean and covariance of the base measure on the location parameter (default are the sample mean and the sample covariance of the data); Sigma0 and n0 are the parameters of the inverse Whishart prior on the common scale matrix (default are the sample covariance of the data and p+2). If hyper = TRUE, optional hyperpriors on the base measure's parameters are added: specifically, m1 and k1 are the mean parameter and the scale factor defining the normal hyperprior on m0 (default are the sample mean of the data and 1), and lambda and Lambda1 are the parameters (degrees of freedom and scale) of the inverse Wishart prior on S20 (default are p+2 and the sample covariance of the data). See details.
- If model = 'LS' (location-scale mixture) and y is multivariate (p-variate):
m0 and k0 are the mean parameter and the scale factor defining the normal base measure on the location parameter (default are the sample mean of the data and 1), and n0 and Sigma0 are the parameters (degrees of freedom and scale) of the inverse Wishart base measure on the location parameter (default are p+2 and the sample covariance of the data). If hyper = TRUE, optional hyperpriors on the base measure's parameters are added: specifically, m1 and S1 are mean and covariance matrix parameters of the normal hyperprior on m0 (default are the sample mean and the sample covariance of the data); tau1 and zeta1 are shape and rate parameters of the gamma hyperprior on k0 (default is 1 for both); n1 and Sigma1 are the parameters (degrees of freedom and scale) of the Wishart prior for Sigma0 (default are p+2 and the sample covariance of the data divided p+2). See details.
- If model = 'DLS' (diagonal location-scale mixture):
m0 and k0 are the mean vector parameter and the vector of scale factors defining the normal base measure on the location parameter (default are the sample mean and a vector of ones), and a0 and b0 are vectors of shape and scale parameters defining the base measure on the scale parameters (default are a vector of twos and the diagonal of the sample covariance of the data). If hyper = TRUE, optional hyperpriors on the base measure's parameters are added: specifically, m1 and s21 are vectors of mean and variance parameters for the normal hyperpriors on the components of m0 (default are the sample mean and the diagonal of the sample covariance of the data); tau1 and zeta1 are vectors of shape and rate parameters of the gamma hyperpriors on the components of k0 (default is a vector of ones for both); a1 and b1 are vectors of shape and rate parameters of the gamma hyperpriors on the components of b0 (default is the diagonal of the sample covariance of the data and a vector of ones). See details.

output — a list of arguments for generating posterior output. It contains:

- grid, a grid of points at which to evaluate the estimated posterior mean density; a data frame obtained with the expand.grid function.
- out_param, if equal to TRUE, the function returns the draws of the kernel's parameters for each MCMC iteration, default is FALSE. See value for details.

- `out_type`, if `out_type = "FULL"`, the function returns the visited partitions and the realizations of the posterior density for each iterations. If `out_type = "MEAN"`, the function returns the estimated partitions and the mean of the densities sampled at each iterations. If `out_type = "CLUST"`, the function returns the estimated partition. Default `"FULL"`.

Details

This generic function fits a Pitman-Yor process mixture model for density estimation and clustering. The general model is

$$\tilde{f}(y) = \int K(y;\theta)\tilde{p}(d\theta),$$

where $K(y;\theta)$ is a kernel density with parameter $\theta \in \Theta$. Univariate and multivariate Gaussian kernels are implemented with different specifications for the parametric space Θ, as described below. The mixing measure $\tilde{p}$ has a Pitman-Yor process prior with strength parameter ϑ, discount parameter α, and base measure P_0 admitting the specifications presented below. For posterior sampling, three MCMC approaches are implemented. See details below.

Univariate data

For univariate y the function implements both a location and location-scale mixture model. The former assumes

$$\tilde{f}(y) = \int \phi(y;\mu,\sigma^2)\tilde{p}(d\mu)\pi(\sigma^2),$$

where $\phi(y;\mu,\sigma^2)$ is a univariate Gaussian kernel function with mean μ and variance σ^2, and $\pi(\sigma^2)$ is an inverse gamma prior. The base measure is specified as

$$P_0(d\mu) = N(d\mu; m_0, \sigma_0^2),$$

and $\sigma^2 \sim IGa(a_0, b_0)$. Optional hyperpriors for the base measure's parameters are

$$(m_0, \sigma_0^2) \sim N(m_1, \sigma_0^2/k_1) \times IGa(a_1, b_1).$$

The location-scale mixture model, instead, assumes

$$\tilde{f}(y) = \int \phi(y;\mu,\sigma^2)\tilde{p}(d\mu, d\sigma^2)$$

with normal-inverse gamma base measure

$$P_0(d\mu, d\sigma^2) = N(d\mu; m_0, \sigma^2/k_0) \times IGa(d\sigma^2; a_0, b_0).$$

and (optional) hyperpriors

$$m_0 \sim N(m_1, \sigma_1^2), \quad k_0 \sim Ga(\tau_1, \zeta_1), \quad b_0 \sim Ga(a_1, b_1).$$

Multivariate data

For multivariate y (p-variate) the function implements a location mixture model (with full covariance matrix) and two different location-scale mixture models, with either full or diagonal covariance matrix. The location mixture model assumes

$$\tilde{f}(y) = \int \phi_p(y;\mu,\Sigma)\tilde{p}(d\mu)\pi(\Sigma)$$

where $\phi_p(y; \mu, \Sigma)$ is a p-dimensional Gaussian kernel function with mean vector μ and covariance matrix Σ. The prior on Σ is inverse Whishart with parameters Σ_0 and ν_0, while the base measure is

$$P_0(d\mu) = N(d\mu; m_0, S_0),$$

with optional hyperpriors

$$m_0 \sim N(m_1, S_0/k_1), \quad S_0 \sim IW(\lambda_1, \Lambda_1).$$

The location-scale mixture model assumes

$$\tilde{f}(x) = \int \phi_p(y; \mu, \Sigma)\tilde{p}(d\mu, d\Sigma).$$

Two possible structures for Σ are implemented, namely full and diagonal covariance. For the full covariance mixture model, the base measure is the normal-inverse Wishart

$$P_0(d\mu, d\Sigma) = N(d\mu; m_0, \Sigma/k_0) \times IW(d\Sigma; \nu_0, \Sigma_0),$$

with optional hyperpriors

$$m_0 \sim N(m_1, S_1), \quad k_0 \sim Ga(\tau_1, \zeta_1), \quad b_0 \sim W(\nu_1, \Sigma_1).$$

The second location-scale mixture model assumes a diagonal covariance structure. This is equivalent to write the mixture model as a mixture of products of univariate normal kernels, i.e.

$$\tilde{f}(y) = \int \prod_{r=1}^{p} \phi(y_r; \mu_r, \sigma_r^2)\tilde{p}(d\mu_1, \ldots, d\mu_p, d\sigma_1^2, \ldots, d\sigma_p^2).$$

For this specification, the base measure is assumed defined as the product of p independent normal-inverse gamma distributions, that is

$$P_0 = \prod_{r=1}^{p} P_{0r}$$

where

$$P_{0r}(d\mu_r, d\sigma_r^2) = N(d\mu_r; m_{0r}, \sigma_r^2/k_{0r}) \times Ga(d\sigma_r^2; a_{0r}, b_{0r}).$$

Optional hyperpriors can be added, and, for each component, correspond to the set of hyperpriors considered for the univariate location-scale mixture model.

Posterior simulation methods

This generic function implements three types of MCMC algorithms for posterior simulation. The default method is the importance conditional sampler `'ICS'` (Canale et al. 2019). Other options are the marginal sampler `'MAR'` (Neal, 2000) and the slice sampler `'SLI'` (Kalli et al. 2011). The importance conditional sampler performs an importance sampling step when updating the values of individual parameters θ, which requires to sample `m_imp` values from a suitable proposal. Large values of `m_imp` are known to improve the mixing of the chain at the cost of increased running time (Canale et al. 2019). Two options are available for the slice sampler, namely the dependent slice-efficient sampler (`slice_type = 'DEP'`), which is set as default, and the independent slice-efficient sampler (`slice_type = 'INDEP'`) (Kalli et al. 2011). See Corradin et al. (to appear) for more details.

Value

A BNPdens class object containing the estimated density and the cluster allocations for each iterations. If out_param = TRUE the output contains also the kernel specific parameters for each iteration. If mcmc_dens = TRUE the output contains also a realization from the posterior density for each iteration. IF mean_dens = TRUE the output contains just the mean of the realizations from the posterior density. The output contains also informations as the number of iterations, the number of burn-in iterations, the used computational time and the type of estimated model (univariate = TRUE or FALSE).

References

Canale, A., Corradin, R., Nipoti, B. (2019), Importance conditional sampling for Bayesian nonparametric mixtures, arXiv preprint, arXiv:1906.08147

Corradin, R., Canale, A., Nipoti, B. (2021), BNPmix: An R Package for Bayesian Nonparametric Modeling via Pitman-Yor Mixtures, Journal of Statistical Software, 100, doi:10.18637/jss.v100.i15

Kalli, M., Griffin, J. E., and Walker, S. G. (2011), Slice sampling mixture models. Statistics and Computing 21, 93-105, doi:10.1007/s11222-009-9150-y

Neal, R. M. (2000), Markov Chain Sampling Methods for Dirichlet Process Mixture Models, Journal of Computational and Graphical Statistics 9, 249-265, doi:10.2307/1390653

Examples

```
data_toy <- cbind(c(rnorm(100, -3, 1), rnorm(100, 3, 1)),
                  c(rnorm(100, -3, 1), rnorm(100, 3, 1)))
grid <- expand.grid(seq(-7, 7, length.out = 50),
                    seq(-7, 7, length.out = 50))
est_model <- PYdensity(y = data_toy, mcmc = list(niter = 200, nburn = 100),
output = list(grid = grid))
summary(est_model)
plot(est_model)
```

PYregression *MCMC for Pitman-Yor mixture of Gaussian regressions*

Description

The PYregression function generates a posterior sample for mixtures of linear regression models inspired by the ANOVA-DDP model introduced in De Iorio et al. (2004). See details below for model specification.

Usage

```
PYregression(y, x, mcmc = list(), prior = list(), output = list())
```

Arguments

y	a vector of observations, univariate dependent variable;
x	a matrix of observations, multivariate independent variable;
mcmc	a list of MCMC arguments: • niter (mandatory), number of iterations. • nburn (mandatory), number of iterations to discard as burn-in. • method, the MCMC sampling method to be used. Options are 'ICS', 'MAR' and 'SLI' (default is 'ICS'). See details. • model the type of model to be fitted (default is 'LS'). See details. • nupd, argument controlling the number of iterations to be displayed on screen: the function reports on standard output every time nupd new iterations have been carried out (default is niter/10). • print_message, control option. If equal to TRUE, the status is printed to standard output every nupd iterations (default is TRUE). • m_imp, number of generated values for the importance sampling step of method = 'ICS' (default is 10). See details. • slice_type, when method = 'SLI' it specifies the type of slice sampler. Options are 'DEP' for dependent slice-efficient, and 'INDEP' for independent slice-efficient (default is 'DEP'). See details. • m_marginal, number of generated values for the augmentation step needed, if method = 'MAR', to implement Algorithm 8 of Neal, 2000. (Default is 100). See details. • hyper, if equal to TRUE, hyperprior distributions on the base measure's parameters are added, as specified in prior and explained in details (default is TRUE).
prior	a list giving the prior information. The list includes strength and discount, the strength and discount parameters of the Pitman-Yor process (default are strength = 1 and discount = 0, the latter leading to the Dirichlet process). The remaining parameters specify the base measure: m0 and S0 are the mean and covariance of normal base measure on the regression coefficients (default are a vector of zeroes, except for the first element equal to mean(y), and a diagonal matrix with each element equal to 100); a0 and b0 are the shape and scale parameters of the inverse gamma base measure on the scale component (default are 2 and var(y)). If hyper = TRUE, optional hyperpriors on the base measure's parameters are added: specifically, m1 and k1 are the mean parameter and scale factor defining the normal hyperprior on m0 (default are a vector of zeroes, except for the first element equal to the sample mean of the dependent observed variable, and 1); tau1 and zeta1 are the shape and rate parameters of the gamma hyperprior on b0 (default is 1 for both); n1 and S1 are the parameters (degrees of freedom and scale) of the Wishart prior for S0 (default 4 and a diagonal matrix with each element equal to 100); See details.
output	list of posterior summaries: • grid_y, a vector of points where to evaluate the estimated posterior mean density of y, conditionally on each value of x in grid_x;

- grid_x, a matrix of points where to evaluate the realization of the posterior conditional densities of y given x;
- out_type, if out_type = "FULL", the function returns the estimated partitions and the realizations of the posterior density for each iteration; If out_type = "MEAN", return the estimated partitions and the mean of the densities sampled at each iteration; If out_type = "CLUST", return the estimated partitions. Default out_type = "FULL";
- out_param, if equal to TRUE, the function returns the draws of the kernel's parameters for each MCMC iteration, default is FALSE. See value for details.

Details

This function fits a Pitman-Yor process mixture of Gaussian linear regression models, i.e

$$\tilde{f}(y) = \int \phi(y; x^T\beta, \sigma^2)\tilde{p}(d\beta, d\sigma^2)$$

where x is a bivariate vector containing the dependent variable in x and a value of 1 for the intercept term. The mixing measure $\tilde{p}$ has a Pitman-Yor process prior with strength ϑ, discount parameter α. The location model assume a base measures P_0 specified as

$$P_0(d\beta) = N(d\beta; m_0, S_0).$$

while the location-scale model assume a base measures P_0 specified as

$$P_0(d\beta, d\sigma^2) = N(d\beta; m_0, S_0) \times IGa(d\sigma^2; a_0, b_0).$$

Optional hyperpriors complete the model specification:

$$m_0 \sim N(m_1, S_0/k_1), \quad S_0 \sim IW(\nu_1, S_1), \quad b_0 \sim G(\tau_1, \zeta_1).$$

Posterior simulation methods

This generic function implements three types of MCMC algorithms for posterior simulation. The default method is the importance conditional sampler 'ICS' (Canale et al. 2019). Other options are the marginal sampler 'MAR' (algorithm 8 of Neal, 2000) and the slice sampler 'SLI' (Kalli et al. 2011). The importance conditional sampler performs an importance sampling step when updating the values of individual parameters θ, which requires to sample m_imp values from a suitable proposal. Large values of m_imp are known to improve the mixing of the posterior distribution at the cost of increased running time (Canale et al. 2019). When updateing the individual parameter θ, Algorithm 8 of Neal, 2000, requires to sample m_marginal values from the base measure. m_marginal can be chosen arbitrarily. Two options are available for the slice sampler, namely the dependent slice-efficient sampler (slice_type = 'DEP'), which is set as default, and the independent slice-efficient sampler (slice_type = 'INDEP') (Kalli et al. 2011). See Corradin et al. (to appear) for more details.

Value

A BNPdens class object containing the estimated density and the cluster allocations for each iterations. The output contains also the data and the grids. If out_param = TRUE the output contains

also the kernel specific parameters for each iteration. If `mcmc_dens = TRUE`, the function returns also a realization from the posterior density for each iteration. If `mean_dens = TRUE`, the output contains just the mean of the densities sampled at each iteration. The output retuns also the number of iterations, the number of burn-in iterations, the computational time and the type of model.

References

Canale, A., Corradin, R., Nipoti, B. (2019), Importance conditional sampling for Bayesian nonparametric mixtures, arXiv preprint, arXiv:1906.08147

Corradin, R., Canale, A., Nipoti, B. (2021), BNPmix: An R Package for Bayesian Nonparametric Modeling via Pitman-Yor Mixtures, Journal of Statistical Software, doi:10.18637/jss.v100.i15

De Iorio, M., Mueller, P., Rosner, G.L., and MacEachern, S. (2004), An ANOVA Model for Dependent Random Measures, Journal of the American Statistical Association 99, 205-215, doi:10.1198/016214504000000205

Kalli, M., Griffin, J. E., and Walker, S. G. (2011), Slice sampling mixture models. Statistics and Computing 21, 93-105, doi:10.1007/s11222-009-9150-y

Neal, R. M. (2000), Markov Chain Sampling Methods for Dirichlet Process Mixture Models, Journal of Computational and Graphical Statistics 9, 249-265, doi:10.2307/1390653

Examples

```
x_toy <- c(rnorm(100, 3, 1), rnorm(100, 3, 1))
y_toy <- c(x_toy[1:100] * 2 + 1, x_toy[101:200] * 6 + 1) + rnorm(200, 0, 1)
grid_x <- c(0, 1, 2, 3, 4, 5)
grid_y <- seq(0, 35, length.out = 50)
est_model <- PYregression(y = y_toy, x = x_toy,
mcmc = list(niter = 200, nburn = 100),
output = list(grid_x = grid_x, grid_y = grid_y))
summary(est_model)
plot(est_model)
```

---

summary.BNPdens *BNPdens summary method*

Description

The `summary.BNPdens` method provides summary information on `BNPdens` objects.

Usage

```
## S3 method for class 'BNPdens'
summary(object, ...)
```

Arguments

`object`	an object of class `BNPdens`;
`...`	additional arguments

Examples

```
data_toy <- c(rnorm(100, -3, 1), rnorm(100, 3, 1))
grid <- seq(-7, 7, length.out = 50)
est_model <- PYdensity(y = data_toy, mcmc = list(niter = 100,
                     nburn = 10, napprox = 10), output = list(grid = grid))
class(est_model)
summary(est_model)
```

Index

BNPdens, 2
BNPdens2coda.BNPdens, 3
BNPpart, 4

dBNPdens.BNPdens, 5
DDPdensity, 5

partition.BNPdens, 8
plot.BNPdens, 9
print.BNPdens, 11
PYcalibrate, 12
PYdensity, 12
PYregression, 17

summary.BNPdens, 20