Package 'BNPmix'

May 18, 2019

Type Package

Title Algorithms for Pitman-Yor Process Mixtures

Version 0.1.2

Date 2019-05-06

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

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

Description Performs Bayesian nonparametric univariate and multivariate density estimation and clustering by means of Pitman-Yor mixtures and Dependent Dirichlet process mixtures for partially exchangeable data.

License LGPL-3 | file LICENSE

NeedsCompilation yes

Imports methods, ggplot2

Depends R (>= 3.3.0)

LinkingTo RcppArmadillo, Rcpp(>= 0.12.13)

RoxygenNote 6.1.1

Encoding UTF-8

Repository CRAN

Date/Publication 2019-05-18 17:20:08 UTC

R topics documented:

---

cDDP *C++ function to estimate DDP models with 1 grouping variables*

Description

C++ function to estimate DDP models with 1 grouping variables

Arguments

data	a vector of observations.
group	group allocation of the data.
ngr	number of groups.
grid	vector to evaluate the density.
niter	number of iterations.
nburn	number of burn-in iterations.
m0	expectation of location component.
k0	tuning parameter of variance of location component.
a0	parameter of scale component.
b0	parameter of scale component.
mass	mass of Dirichlet process.
wei	prior weight of the specific processes.
napprox	number of approximating values.
n_approx_unif	number of approximating values of the importance step for the weights updating.
nupd	number of iterations to show current updating.
out_dens	if TRUE, return also the estimated density (default TRUE).
print_message	print the status.
light_dens	if TRUE return only the posterior mean of the density

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 <- cDDP(data_toy, group_toy, 2, grid, 20, 10,
    0, 1, 2, 1, 1, 0.5, 10, 1000, 100, 1, 1, 1)
```

```
  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 <- cDDP(data_toy, group_toy, 2, grid, 1000, 100,
    0, 1, 2, 1, 1, 0.5, 10, 1000, 100, 1, 1, 1)

}
```

---

cICS *C++ function to estimate Pitman-Yor univariate mixtures via importance conditional sampler*

Description

C++ function to estimate Pitman-Yor univariate mixtures via importance conditional sampler

Arguments

data	a vector of observations
grid	vector to evaluate the density
niter	number of iterations
nburn	number of burn-in iterations
m0	expectation of location component
k0	tuning parameter of variance of location component
a0	parameter of scale component
b0	parameter of scale component
mass	mass of Dirichlet process
napprox	number of approximating values
nupd	number of iterations to show current updating
out_param	if TRUE, return also the location and scale paramteres lists
out_dens	if TRUE, return also the estimated density (default TRUE)
sigma_PY	second parameter of PY
print_message	print the status

Examples

```
{
  data_toy <- c(rnorm(100, -3, 1), rnorm(100, 3, 1))
  grid <- seq(-7, 7, length.out = 50)
  est_model <- cICS(data_toy, grid, 20, 10, 0, 1, 2, 1, 1, 10, 100, 0, 1, 0, 1)
```

```
    data_toy <- c(rnorm(100, -3, 1), rnorm(100, 3, 1))
    grid <- seq(-7, 7, length.out = 50)
    est_model <- cICS(data_toy, grid, 1000, 100, 0, 1, 2, 1, 1, 10, 100, 0, 1, 0, 1)

}
```

`cICS_mv` *C++ function to estimate Pitman-Yor multivariate mixtures via importance conditional sampler*

Description

C++ function to estimate Pitman-Yor multivariate mixtures via importance conditional sampler

Arguments

`data`	a matrix of observations
`grid`	matrix of points to evaluate the density
`niter`	number of iterations
`nburn`	number of burn-in iterations
`m0`	expectation of location component
`k0`	tuning parameter of variance of location component
`S0`	parameter of scale component
`n0`	parameter of scale component
`mass`	mass of Dirichlet process
`napprox`	number of approximating values
`nupd`	number of iterations to show current updating
`out_param`	if TRUE, return also the location and scale paramteres lists
`out_dens`	if TRUE, return also the estimated density (default TRUE)
`sigma_PY`	second parameter of PY
`print_message`	print the status
`light_dens`	if TRUE return only the posterior mean of the density

Examples

```
{
  data_toy <- cbind(c(rnorm(100, -3, 1), rnorm(100, 3, 1)),
                    c(rnorm(100, -3, 1), rnorm(100, 3, 1)))
  grid <- as.matrix(expand.grid(seq(-7, 7, length.out = 50),
                      seq(-7, 7, length.out = 50)))
  est_model <- cICS_mv(data_toy, grid, 20, 10,
    c(0,0), 1, diag(1,2), 4, 1, 100, 100, 0, 1, 0, 1, 1)
```

```
    data_toy <- cbind(c(rnorm(100, -3, 1), rnorm(100, 3, 1)),
                  c(rnorm(100, -3, 1), rnorm(100, 3, 1)))
    grid <- as.matrix(expand.grid(seq(-7, 7, length.out = 50),
                    seq(-7, 7, length.out = 50)))
    est_model <- cICS_mv(data_toy, grid, 1000, 100,
      c(0,0), 1, diag(1,2), 4, 1, 100, 100, 0, 1, 0, 1, 1)

}
```

---

condDDP *Conditional dependent Dirichlet process*

Description

Conditional dependent Dirichlet process

Estimate an univariate dependent Dirichlet process mixture model with Gaussian kernel using importance conditional sampler scheme.

Usage

```
condDDP(data, group, grid = NULL, niter, nburn, m0 = NULL, k0 = NULL,
  a0 = NULL, b0 = NULL, mass = 1, wei = 0.5, napprox = 10,
  n_approx_unif = 1000, nupd = 1000, out_dens = TRUE,
  print_message = TRUE, light_dens = FALSE)
```

Arguments

data	A dataset (vector).
group	group for the observed data, same length as data.
grid	A grid to evaluate the estimated density (vector).
niter	Number of iterations to estimate the model.
nburn	Number of burn-in iterations.
m0	Mean of the location component of the base measure.
k0	Tuning parameter of the variance for the location component of the base measure.
a0	Shape parameter of the scale component of the base measure.
b0	Rate parameter of the scale component of the base measure.
mass	Total mass.
wei	weigth of the idiosyncratic process.
napprox	Number of values to be sampled from the restricted process via "ICS" method, default 100.

n_approx_unif	number of values used in the importance sampling step for the multivariate beta distribution.
nupd	How frequently show the curren state of the estimation (number of iterations) - default 1000.
out_dens	If TRUE, save the parameters for each iteration, default TRUE.
print_message	Print the status of the estimation.
light_dens	Return only the posterior mean of the densities.

Value

A modCond class object contain the estimated density for each iterations, the allocations for each iterations. If out_param is TRUE, also the parameters.

Examples

```
set.seed(42)
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 <- condDDP(data = data_toy, group = group_toy, grid = grid, niter = 1000,
                     nburn = 100, napprox = 100, nupd = 100)
summary(est_model)
plot(est_model)
```

---

condMCMC *MCMC for univariate Pitman-Yor mixtures*

Description

The condMCMC function estimate a univariate Pitman-Yor process mixture model with Gaussian kernel. Three possible sampling strategies: importance conditional sampler, slice sampler and marginal sampler.

The models are of the form

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

where $k(x; \mu, \sigma^2)$ is an univariate gaussian kernel function, $\tilde{p}$ is distributed as a Pitman-Yor process with total mass ϑ, discount parameter σ and normal-inverse-gamma base measure P_0, i.e.

$$P_0 \sim N(\mu; m_0, k_0\sigma^2) \times IG(\sigma^2; a_0, b_0).$$

Usage

```
condMCMC(data, grid = NULL, niter, nburn, m0 = NULL, k0 = NULL,
  a0 = NULL, b0 = NULL, mass = 1, method = "ICS", napprox = 100,
  nupd = 1000, out_param = F, out_dens = TRUE, process = "DP",
  sigma_PY = 0, print_message = TRUE)
```

Arguments

data	A dataset (vector).
grid	A grid of points to evaluate the estimated density (vector).
niter	Number of iterations to estimate the model.
nburn	Number of burn-in iterations.
m0	Mean of the location component of the base measure.
k0	Tuning parameter of the variance for the location component of the base measure.
a0	Shape parameter of the scale component of the base measure.
b0	Rate parameter of the scale component of the base measure.
mass	Mass parameter.
method	Different methods to estimate the model. Possible vaule "ICS", "SLI" or "MAR", importance conditional sampler/slice sampler/marginal sampler, default "ICS".
napprox	Number of values to be sampled from the restricted process via "ICS" method, default 100.
nupd	How frequently show the curren state of the estimation (number of iterations) - default 1000.
out_param	If TRUE, save the parameters for each iteration, default FALSE.
out_dens	If TRUE, return also the estimated density, default TRUE.
process	Dirichlet process ("DP") or Pitman-Yor process ("PY"), default "DP".
sigma_PY	Discount parameter of the Pitman-Yor process, default 0.
print_message	If TRUE print the status of the estimation, default TRUE.

Value

A modCond class object contain the estimated density for each iterations, the allocations for each iterations. If out_param is TRUE, also the parameters for each iteration.

Examples

```
data_toy <- c(rnorm(100, -3, 1), rnorm(100, 3, 1))
grid <- seq(-7, 7, length.out = 50)
est_model <- condMCMC(data = data_toy, grid = grid, niter = 1000,
                      nburn = 100, napprox = 100, nupd = 100)
summary(est_model)
plot(est_model)
```

condMCMCmv *MCMC for multivariate Pitman-Yor mixtures*

Description

The condMCMCmv function estimate a multivariate Pitman-Yor process mixture model with Gaussian kernel. Three possible sampling strategies: importance conditional sampler, slice sampler and marginal sampler.

The models are of the form

$$\tilde{f}(\mathbf{x}) = \int k(\mathbf{x}; \mu, \Sigma)\tilde{p}(d\mu, d\Sigma)$$

where $k(\mathbf{x}; \mu, \Sigma)$ is a multivariate gaussian kernel function, $\tilde{p}$ is distributed as a Pitman-Yor process with total mass ϑ, discount parameter σ and normal-inverse-wishart base measure P_0, i.e.

$$P_0 \sim N(\mu; \mathbf{m}_0, k_0\Sigma) \times IW(\Sigma; n_0, S_0).$$

Usage

```
condMCMCmv(data, grid = NULL, niter, nburn, m0 = NULL, k0 = NULL,
  S0 = NULL, n0 = NULL, mass = 1, method = "ICS", napprox = 100,
  nupd = 1000, out_param = F, out_dens = TRUE, process = "DP",
  sigma_PY = 0, print_message = TRUE, light_dens = TRUE)
```

Arguments

data	A dataset (matrix).
grid	A grid of points to evaluate the estimated density (matrix).
niter	Number of iterations to estimate the model.
nburn	Number of burn-in iterations.
m0	Mean of location component of the base measure.
k0	Tuning parameter of the variance for the location component of the base measure.
S0	Matrix of the Inverse Wishart distribution of the scale component.
n0	Degree of freedom of the Inverse Wishart distribution of the scale component.
mass	Mass parameter.
method	Different methods to estimate the model. Possible vaule "ICS", "SLI" or "MAR", importance conditional sampler/slice sampler/marginal sampler, default "ICS".
napprox	Number of values to be sampled from the restricted process via "ICS" method, default 100.
nupd	How frequently show the curren state of the estimation (number of iterations) - default 1000.
out_param	If TRUE, save the parameters for each iteration, default FALSE.
out_dens	If TRUE, return also the estimated density, default TRUE.

process	Dirichlet process ("DP") or Pitman-Yor process ("PY"), default "DP".
sigma_PY	Discount parameter of the Pitman-Yor process, default 0.
print_message	If TRUE print the status of the estimation, default TRUE.
light_dens	If TRUE return only the mean of the estimated densities, default TRUE.

Value

A modCondMv class object contain the estimated density for each iterations, the allocations for each iterations. If out_param is TRUE, also the parameters for each iteration.

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 <- condMCMCmv(data = data_toy, grid = grid, niter = 1000,
                        nburn = 100, napprox = 100, nupd = 100)
summary(est_model)
plot(est_model)
```

---

cSLI *C++ function to estimate Pitman-Yor univariate mixtures via slice sampler*

Description

C++ function to estimate Pitman-Yor univariate mixtures via slice sampler

Arguments

data	a vector of observations
grid	vector to evaluate the density
niter	number of iterations
nburn	number of burn-in iterations
m0	expectation of location component
k0	tuning parameter of variance of location component
a0	parameter of scale component
b0	parameter of scale component
mass	mass of Dirichlet process
nupd	number of iterations to show current updating
out_param	if TRUE, return also the location and scale paramteres lists
out_dens	if TRUE, return also the estimated density (default TRUE)
sigma_PY	second parameter of PY
print_message	print the status

Examples

```
{
  data_toy <- c(rnorm(100, -3, 1), rnorm(100, 3, 1))
  grid <- seq(-7, 7, length.out = 50)
  est_model <- cSLI(data_toy, grid, 20, 10, 0, 1, 2, 1, 1, 100, 0, 1, 0, 1)


    data_toy <- c(rnorm(100, -3, 1), rnorm(100, 3, 1))
    grid <- seq(-7, 7, length.out = 50)
    est_model <- cSLI(data_toy, grid, 1000, 100, 0, 1, 2, 1, 1, 100, 0, 1, 0, 1)

}
```

`cSLI_mv` *C++ function to estimate Pitman-Yor multivariate mixtures via slice sampler*

Description

C++ function to estimate Pitman-Yor multivariate mixtures via slice sampler

Arguments

`data`	a matrix of observations
`grid`	matrix of points to evaluate the density
`niter`	number of iterations
`nburn`	number of burn-in iterations
`m0`	expectation of location component
`k0`	tuning parameter of variance of location component
`S0`	parameter of scale component
`n0`	parameter of scale component
`mass`	mass of Dirichlet process
`nupd`	number of iterations to show current updating
`out_param`	if TRUE, return also the location and scale paramteres lists
`out_dens`	if TRUE, return also the estimated density (default TRUE)
`sigma_PY`	second parameter of PY
`print_message`	print the status
`light_dens`	if TRUE return only the posterior mean of the density

Examples

```
{
  data_toy <- cbind(c(rnorm(100, -3, 1), rnorm(100, 3, 1)),
                    c(rnorm(100, -3, 1), rnorm(100, 3, 1)))
  grid <- as.matrix(expand.grid(seq(-7, 7, length.out = 50),
                      seq(-7, 7, length.out = 50)))
 est_model <- cSLI_mv(data_toy, grid, 20, 10, c(0,0), 1, diag(1,2), 4, 1, 100, 0, 1, 0, 1, 1)


   data_toy <- cbind(c(rnorm(100, -3, 1), rnorm(100, 3, 1)),
                    c(rnorm(100, -3, 1), rnorm(100, 3, 1)))
   grid <- as.matrix(expand.grid(seq(-7, 7, length.out = 50),
                      seq(-7, 7, length.out = 50)))
  est_model <- cSLI_mv(data_toy, grid, 1000, 100, c(0,0), 1, diag(1,2), 4, 1, 100, 0, 1, 0, 1, 1)

}
```

MAR *C++ function to estimate Pitman-Yor univariate mixtures via marginal sampler*

Description

C++ function to estimate Pitman-Yor univariate mixtures via marginal sampler

Arguments

data	a vector of observations
grid	vector to evaluate the density
niter	number of iterations
nburn	number of burn-in iterations
m0	expectation of location component
k0	tuning parameter of variance of location component
a0	parameter of scale component
b0	parameter of scale component
mass	mass of Dirichlet process
nupd	number of iterations to show current updating
out_param	if TRUE, return also the location and scale paramteres lists
out_dens	if TRUE, return also the estimated density (default TRUE)
process	if 0 DP, if 1 PY
sigma_PY	second parameter of PY
print_message	print the status

Examples

```
{
  data_toy <- c(rnorm(100, -3, 1), rnorm(100, 3, 1))
  grid <- seq(-7, 7, length.out = 50)
  est_model <- MAR(data_toy, grid, 50, 10, 0, 1, 2, 1, 1, 100, 0, 1, 0, 1)


    data_toy <- c(rnorm(100, -3, 1), rnorm(100, 3, 1))
    grid <- seq(-7, 7, length.out = 50)
    est_model <- MAR(data_toy, grid, 1000, 100, 0, 1, 2, 1, 1, 100, 0, 1, 0, 1)

}
```

MAR_mv *C++ function to estimate Pitman-Yor multivariate mixtures via marginal sampler*

Description

C++ function to estimate Pitman-Yor multivariate mixtures via marginal sampler

Arguments

data	a matrix of observations
grid	matrix of points to evaluate the density
niter	number of iterations
nburn	number of burn-in iterations
m0	expectation of location component
k0	tuning parameter of variance of location component
S0	parameter of scale component
n0	parameter of scale component
mass	mass of Dirichlet process
nupd	number of iterations to show current updating
out_param	if TRUE, return also the location and scale paramteres lists
out_dens	if TRUE, return also the estimated density (default TRUE)
sigma_PY	second parameter of PY
print_message	print the status
light_dens	if TRUE return only the posterior mean of the density

Examples

```
{
  data_toy <- cbind(c(rnorm(100, -3, 1), rnorm(100, 3, 1)),
                    c(rnorm(100, -3, 1), rnorm(100, 3, 1)))
  grid <- as.matrix(expand.grid(seq(-7, 7, length.out = 50),
                      seq(-7, 7, length.out = 50)))
 est_model <- MAR_mv(data_toy, grid, 20, 10, c(0,0), 1, diag(1,2), 4, 1, 100, 0, 1, 0, 1, 1)


   data_toy <- cbind(c(rnorm(100, -3, 1), rnorm(100, 3, 1)),
                  c(rnorm(100, -3, 1), rnorm(100, 3, 1)))
   grid <- as.matrix(expand.grid(seq(-7, 7, length.out = 50),
                    seq(-7, 7, length.out = 50)))
  est_model <- MAR_mv(data_toy, grid, 1000, 100, c(0,0), 1, diag(1,2), 4, 1, 100, 0, 1, 0, 1, 1)

}
```

---

modCond *modCond class constructor*

Description

define the modCond class, could be for univariate or multivariate estimated models

Usage

```
modCond(density = NULL, grideval = NULL, clust = NULL, mean = NULL,
  sigma2 = NULL, probs = NULL, niter = NULL, nburn = NULL,
  nnew = NULL, tot_time = NULL, univariate = TRUE, dep = FALSE,
  group_log = NULL, nclust = NULL, group = NULL, wvals = NULL)
```

Arguments

density	a matrix containing the density value corresponding to the grid points
grideval	a set of values to evaluate the density
clust	a matrix containing the corresponding clusters for each observation, for each iteration
mean	values for the location parameters
sigma2	values for the scale parameters
probs	values for the mixing measure
niter	number of iterations
nburn	number of burn-in iterations
nnew	number of new clusters sampled at each iteration
tot_time	execution time

univariate	if the model is univariate (TRUE/FALSE)
dep	if the model is dependent Dirichlet process (TRUE/FALSE)
group_log	group allocation for dependent Dirichlet process
nclust	number of cluster for dependent Dirichlet process
group	true group allocation for dependent Dirichlet process
wvals	values of the processes weights

Examples

```
data_toy <- c(rnorm(100, -3, 1), rnorm(100, 3, 1))
grid <- seq(-7, 7, length.out = 50)
est_model <- condMCMC(data = data_toy, grid = grid, niter = 1000,
                      nburn = 100, napprox = 100, nupd = 100)
str(est_model)
class(est_model)
```

---

plot.modCond *modCond summary method*

Description

modCond summary method

Usage

```
## S3 method for class 'modCond'
plot(x, dimension = c(1, 2), col = "#0037c4", ...)
```

Arguments

x	an object of class modCond
dimension	if multivariate, the two dimensions for the plot (if they are equal, a marginal plot is performed)
col	the color of the lines
...	additional arguments to be passed

Examples

```
data_toy <- c(rnorm(100, -3, 1), rnorm(100, 3, 1))
grid <- seq(-7, 7, length.out = 50)
est_model <- condMCMC(data = data_toy, grid = grid, niter = 1000,
                      nburn = 100, napprox = 100, nupd = 100)
class(est_model)
plot(est_model)
```

summary.modCond *modCond summary method*

Description

modCond summary method

Usage

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

Arguments

object	an object of class modCond
...	additional arguments to be passed

Examples

```
data_toy <- c(rnorm(100, -3, 1), rnorm(100, 3, 1))
grid <- seq(-7, 7, length.out = 50)
est_model <- condMCMC(data = data_toy, grid = grid, niter = 1000,
                      nburn = 100, napprox = 100, nupd = 100)
class(est_model)
summary(est_model)
```

Index