

Package ‘BNPdensity’

October 7, 2011

Version 0.7.8

Date 2011/10/06

Title Ferguson-Klass type algorithm for posterior normalized random measures

Author I. Prunster and L. E. Nieto. R implementation by Ernesto Barrios

Maintainer Ernesto Barrios <ebarrios@itam.mx>

Description Ferguson-Klass type algorithm for porterior normalized random measures

Depends R(>= 2.11.0)

License GPL version 2 or later

Encoding latin1

R topics documented:

BNPdensity-package	2
acidity	3
enzyme	3
Enzyme1.out	4
Enzyme2.out	5
galaxy	5
Galaxy1.out	6
Galaxy2.out	6
MixNRMI1	7
MixNRMI2	11
Index	16

BNPdensity-package *Bayesian nonparametric density estimation*

Description

This package performs Bayesian nonparametric density estimation 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 inversed gaussian process and the normalized stable process.

Details

Package:	BNPdensity
Type:	Package
Version:	0.7.8
Date:	2011-10-06
License:	GPL version 2 or later
LazyLoad:	yes

The package includes two main functions: `MixNRMI1` and `MixNRMI2`, which implement semi-parametric mixtures and fully nonparametric mixtures, respectively. Additionally, the package includes several other functions required for sampling from conditional distributions in the MCMC implementation. These functions intended for internal use only.

Author(s)

Barrios, E., Nieto-Barajas, L.E. and Pruenster, I. Maintainer: <ebarrios@itam.mx> Ernesto Barrios

References

Barrios, E., Nieto-Barajas, L. and Pruenster, I. (2011). A study of normalized random measures mixture models. Preprint.

See Also

[MixNRMI1](#), [MixNRMI2](#)

Examples

```
example(MixNRMI1)
example(MixNRMI2)
```

`acidity`*Acidity Index Dataset*

Description

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

Usage

```
data(acidity)
```

Format

A data frame with 155 observations on the following variable:

```
acidity A numeric vector.
```

Source

<http://www.stats.bris.ac.uk/~peter/>

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)
```

`enzyme`*Enzyme Dataset*

Description

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

Usage

```
data(enzyme)
```

Format

A data frame with 244 observations on the following variable:

enzyme A numeric vector.

Source

<http://www.stats.bris.ac.uk/~peter/>

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(123456)` and running the function `MixNRMI1(x, Alpha = 1, Beta = 0.007, Gama = 0.5, distr.k = 2, distr.p0 = 2, mu.p0 = 10, sigma.p0 = 10, asigma = 1, bsigma = 1, Nit = 5000, Pbi = 0.2)`

Usage

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

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(123456)` and running the function `MixNRMI2(x, Alpha = 1, Beta = 0.007, Gama = 0.5, distr.k = 2, distr.py0 = 2, mu.py0 = 10, sigma.py0 = 10, distr.pz0 = 2, mu.pz0 = 1, sigma.pz0 = 1, Nit = 5000, Pbi = 0.2)`

Usage

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

Details

See function `MixNRMI2`

Examples

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

`galaxy`*Galaxy Data Set*

Description

Velocities of 82 galaxies diverging from our own galaxy.

Usage

```
data(galaxy)
```

Format

A data frame with 82 observations on the following variable:

`velocity` A numeric vector.

Source

<http://www.stats.bris.ac.uk/~peter/>

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(123456)` and running the function `MixNRMI1(x, Alpha = 1, Beta = 0.015, Gama = 0.5, distr.k = 1, distr.p0 = 2, mu.p0 = 20, sigma.p0 = 20, asigma = 1, bsigma = 1, Nit = 5000, Pbi = 0.2)`

Usage

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

Details

See function `MixNRMI1`.

Examples

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

Galaxy2.out

Fit of MixNRMI2 function to the galaxy dataset

Description

This object contains the output when setting `set.seed(123456)` and running the function `MixNRMI2(x, Alpha = 1, Beta = 0.015, Gama = 0.5, distr.k = 1, distr.py0 = 2, mu.py0 = 20, sigma.py0 = 20, distr.pz0 = 2, mu.pz0 = 1, sigma.pz0 = 1, Nit = 5000, Pbi = 0.2)`

Usage

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

Details

See function `MixNRMI2`.

Examples

```
data(Galaxy2.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, Beta = 0, Gama = 0.4,
          distr.k = 1, distr.p0 = 1, mu.p0 = mean(x), sigma.p0 = 1.5 * sd(x),
          asigma = 0.1, bsigma = 0.1, delta = 3, Delta = 2,
          Nm = 50, Nx = 100, Nit = 1000, Pbi = 0.1,
          epsilon = NULL, printtime = 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.
Beta	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.
mu.p0	Numeric constant. Prior mean of the centering measure.
sigma.p0	Numeric constant. Prior standard deviation of the centering measure.
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.
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.
epsilon	Numeric constant. Extension to the evaluation grid range. See details.
printtime	Logical. If TRUE, prints out the execution time.

Details

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

The details of the model are:

$$\begin{aligned} X_i | Y_i, \sigma &\sim k(\cdot | Y_i, \sigma) \\ Y_i | P &\sim P, \quad i = 1, \dots, n \\ P &\sim \text{NGG}(\text{Alpha}, \text{Beta}, \text{Gama}; \text{P}_0) \\ \sigma &\sim \text{Gamma}(\text{asigma}, \text{bsigma}) \end{aligned}$$

where X_i 's are the observed data, Y_i 's are latent (location) variables, σ is the smoothing parameter, k is a parametric kernel parameterized in terms of mean and standard deviation, $(\text{Alpha}, \text{Beta}, \text{Gama}; \text{P}_0)$ are the parameters of the NGG prior with P_0 being the centering measure, and $(\text{mu}_0, \text{sigma}_0)$ are the hyper-parameters of the gamma prior on the smoothing parameter σ . In particular: $\text{NGG}(\text{Alpha}, 1, 0; \text{P}_0)$ defines a Dirichlet process; $\text{NGG}(1, \text{Beta}, 1/2; \text{P}_0)$ defines a Normalized inverse Gaussian process; and $\text{NGG}(1, 0, \text{Gama}; \text{P}_0)$ 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:

<code>xx</code>	Numeric vector. Evaluation grid.
<code>qx</code>	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> .
<code>cpo</code>	Numeric vector of $\text{length}(x)$ with conditional predictive ordinates.
<code>R</code>	Numeric vector of $\text{length}(\text{Nit} * (1 - \text{Pbi}))$ with the number of mixtures components (clusters).
<code>S</code>	Numeric vector of $\text{length}(\text{Nit} * (1 - \text{Pbi}))$ with the values of common standard deviation σ .
<code>U</code>	Numeric vector of $\text{length}(\text{Nit} * (1 - \text{Pbi}))$ with the values of the latent variable U .
<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>procTime</code>	Numeric vector with execution time provided by <code>proc.time</code> function.

Warning

The function is computing intensive. Be patient.

Author(s)

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

References

- 1.- Barrios, E., Nieto-Barajas, L.E. and Pruenster, I. (2011). A study of normalized random measures mixture models. Preprint.
- 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.

See Also

[MixNRMI2](#)

Examples

```
### Example 1
## Not run:
# Data
data(acidity)
x <- acidity
# Fitting the model under default specifications
out <- MixNRMI1(x)
# Plotting density estimate + 95
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(123456)
# data(enzyme)
# x <- enzyme
# Enzyme1.out <- MixNRMI1(x, Alpha = 1, Beta = 0.007, Gama = 0.5, distr.k = 2,
#                       distr.p0 = 2, mu.p0 = 10, sigma.p0 = 10, asigma = 1, bsigma = 1,
#                       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-1):max(R),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(123456)
# data(galaxy)
# x <- galaxy
# Galaxy1.out <- MixNRMII(x, Alpha = 1, Beta = 0.015, Gama = 0.5,
#   distr.k = 1, distr.p0 = 2, mu.p0 = 20, sigma.p0 = 20,
#   asigma = 1, bsigma = 1, 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-1):max(R),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()

```

MixNRMI2

Normalized Random Measures Mixture of Type II

Description

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

Usage

```

MixNRMI2(x, probs = c(0.025, 0.5, 0.975), Alpha = 1, Beta = 0, Gama = 0.4,
distr.k = 1, distr.py0 = 1, mu.py0 = mean(x), sigma.py0 = 1.5 * sd(x),
distr.pz0 = 2, mu.pz0 = 1, sigma.pz0 = 10,
delta = 3, Delta = 2, Nm = 50, Nx = 100, Nit = 1000, Pbi = 0.1,
epsilon = NULL, printtime = 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.
Beta	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.
mu.py0	Numeric constant. Prior mean of the centering measure for locations.

<code>sigma.py0</code>	Numeric constant. Prior standard deviation of the centering measure for locations.
<code>distr.pz0</code>	Integer number identifying the centering measure for scales: 2 = Gamma, currently the only option.
<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>Delta</code>	Numeric positive constant. Metropolis-Hastings proposal variation coefficient for sampling the latent U.
<code>Nm</code>	Integer constant. Number of jumps of the continuous component of the unnormalized process.
<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.

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:

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

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

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

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, $(\text{Alpha}, \text{Beta}, \text{Gama}; P_0)$ are the parameters of the NGG prior with a bivariate P_0 being the centering measure. In particular, $\text{NGG}(\text{Alpha}, 1, 0; P_0)$ defines a Dirichlet process; $\text{NGG}(1, \text{Beta}, 1/2; P_0)$ defines a Normalized inverse Gaussian process; and $\text{NGG}(1, 0, \text{Gama}; P_0)$ defines a normalized stable process. The bivariate measure P_0 is assumed to have independent components, that is, $P_0(Y, Z) = P_0(Y) * P_0(Z)$.

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:

<code>xx</code>	Numeric vector. Evaluation grid.
<code>qx</code>	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> .
<code>cpo</code>	Numeric vector of <code>length(x)</code> with conditional predictive ordinates.
<code>R</code>	Numeric vector of <code>length(Nit*(1-Pbi))</code> with the number of mixtures components (clusters).
<code>U</code>	Numeric vector of <code>length(Nit*(1-Pbi))</code> with the values of the latent variable <code>U</code> .
<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>procTime</code>	Numeric vector with execution time provided by <code>proc.time</code> function.

Warning

The function is computing intensive. Be patient.

Author(s)

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

References

- 1.- Barrios, E., Nieto-Barajas, L.E. and Pruenster, I. (2011). A study of normalized random measures mixture models. Preprint.
- 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.

See Also

[MixNRMI1](#)

Examples

```
## Not run:
### Example 1
# Data
data(acidity)
x <- acidity
# Fitting the model under default specifications
out <- MixNRMI2(x)
# Plotting density estimate + 95
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(123456)
# data(enzyme)
# x <- enzyme
# Enzyme2.out <- MixNRM12(x, Alpha = 1, Beta = 0.007, Gama = 0.5, distr.k = 2,
#                         distr.py0 = 2, mu.py0 = 10, sigma.py0 = 10,
#                         distr.pz0 = 2, mu.pz0 = 1, sigma.pz0 = 1,
#                         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-1):max(R),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(123456)
# data(galaxy)
# x <- galaxy
# Galaxy2.out <- MixNRM12(x, Alpha = 1, Beta = 0.015, Gama = 0.5, distr.k = 1,
#                         distr.py0 = 2, mu.py0 = 20, sigma.py0 = 20,
#                         distr.pz0 = 2, mu.pz0 = 1, sigma.pz0 = 1,
#                         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-1):max(R),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()
```

Index

*Topic **datasets**

- acidity, [3](#)
- enzyme, [3](#)
- Enzyme1.out, [4](#)
- Enzyme2.out, [5](#)
- galaxy, [5](#)
- Galaxy1.out, [6](#)
- Galaxy2.out, [6](#)

*Topic **distribution**

- MixNRMI1, [7](#)
- MixNRMI2, [11](#)

*Topic **models**

- MixNRMI1, [7](#)
- MixNRMI2, [11](#)

*Topic **nonparametrics**

- MixNRMI1, [7](#)
- MixNRMI2, [11](#)

*Topic **package**

- BNPdensity-package, [2](#)

acidity, [3](#)

BNPdensity (*BNPdensity-package*), [2](#)

BNPdensity-package, [2](#)

enzyme, [3](#)

Enzyme1.out, [4](#)

Enzyme2.out, [5](#)

galaxy, [5](#)

Galaxy1.out, [6](#)

Galaxy2.out, [6](#)

MixNRMI1, [2](#), [7](#), [13](#)

MixNRMI2, [2](#), [9](#), [11](#)