Package 'bbricks'

October 12, 2022

Type Package

Title Bayesian Methods and Graphical Model Structures for Statistical Modeling

Version 0.1.4

Description A set of frequently used Bayesian parametric and nonparametric model structures, as well as a set of tools for common analytical tasks. Structures include linear Gaussian systems, Gaussian and Normal-Inverse-Wishart conjugate structure, Gaussian and Normal-Inverse-Gamma conjugate structure, Categorical and Dirichlet conjugate structure, Dirichlet Process on positive integers, Dirichlet Process in general, Hierarchical Dirichlet Process ... Tasks include updating posteriors, sampling from posteriors, calculating marginal likelihood, calculating posterior predictive densities, sampling from posterior predictive distributions, calculating ``Maximum A Posteriori'' (MAP) estimates ... See <https://chenhaotian.github.io/Bayesian-Bricks/> to get started.

License MIT + file LICENSE

URL https://github.com/chenhaotian/Bayesian-Bricks

BugReports https://github.com/chenhaotian/Bayesian-Bricks/issues

Encoding UTF-8

LazyData true

RoxygenNote 7.0.2

Collate 'Bayesian_Bricks.r' 'Categorical_Inference.r' 'Gamma_Inference.r' 'Gaussian_Inference.r' 'Dirichlet_Process.r' 'MCMC.r' 'bbricks-package.R' 'testData.r'

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author Haotian Chen [aut, cre] (<https://orcid.org/0000-0001-9751-2093>)

Maintainer Haotian Chen <chenhaotian.jtt@gmail.com>

Depends R (>= 3.5.0)

Repository CRAN

Date/Publication 2020-05-07 19:10:02 UTC

R topics documented:

BasicBayesian *Create objects of type '"BasicBayesian"'.*

Description

A Basic Bayesian Object is with following conditional dependency structure:

$$theta|gamma \sim H(gamma)$$

$$X|theta \sim F(theta)$$

Where H(gamma) is usually called "the prior distribution", F(theta) is called "the observation distribution". Objects of type "LinearGaussianGaussian", "GaussianGaussian", "GaussianInvWishart", "GaussianNIW", "GaussianNIG", "CatDirichlet" and "CatDP" are all "BasicBayesian"s.

Usage

```
BasicBayesian(ENV = parent.frame())
```

Arguments

ENV	The environment where you want to create the BasicBayesian object

Value

An object of class "BasicBayesian".

See Also

LinearGaussianGaussian for Linear Gaussian and Gaussian conjugate structure, GaussianGaussian for Gaussian-Gaussian conjugate structure, GaussianInvWishart for Gaussian-Inverse-Wishart conjugate structure, GaussianNIW for Gaussian-NIW conjugate structure, GaussianNIG for Gaussian-NIG conjugate structure, CatDirichlet for Categorical-Dirichlet conjugate structure, CatDP for Categorical-DP conjugate structure ...

cancerData *Cancer mortality of 20 cities*

Description

Cancer mortality data from Ordinal Data Modeling(1999) p24. This dataset is a list of 20 character vectors, representing the individuals in 20 cities' cancer mortality information. There can be only two possible values "death" and "no death" in each character vector. For example "death" occurs 3 times the 10th character vector, while "no death" occurs 582 times, this mean there are 3 death among the 585 cancer patients in city 10.

Usage

```
data(cancerData)
```

Format

A list of 20 character vectors.

References

Johnson, Valen E., and James H. Albert. Ordinal data modeling. Springer Science & Business Media, 2006.

`CatDirichlet` *Create objects of type "CatDirichlet".*

Description

Create an object of type "CatDirichlet", which represents the Categorical (Multinomial) and Dirichlet conjugate structure:

$$pi|alpha \sim Dir(alpha)$$

$$x|pi \sim Categorical(pi)$$

Where Dir() is the Dirichlet distribution, Categorical() is the Categorical distribution. See `?dDir` and `dCategorical` for the definitions of these distribution.
The created object will be used as a place for recording and accumulating information in the related inference/sampling functions such as posterior(), posteriorDiscard(), MAP(), marginalLikelihood(), dPosteriorPredictive(), rPosteriorPredictive() and so on. A categorical distribution is defined on a set of unique labels, usually these labels are integers, they can also be characters and factors.

Usage

```
CatDirichlet(
  objCopy = NULL,
  ENV = parent.frame(),
  gamma = list(alpha = 1, uniqueLabels = 1L)
)
```

Arguments

`objCopy`	an object of type "CatDirichlet". If "objCopy" is not NULL, the function create a new "CatDirichlet" object by copying the content from objCopy, otherwise this new object will be created by using "ENV" and "gamma". Default NULL.
`ENV`	environment, specify where the object will be created.
`gamma`	list, a named list of parameters, gamma=list(alpha,uniqueLabels). Where gamma$alpha is a numeric vector specifying the parameters of the Dirichlet distribution, gamma$uniqueLabels is a integer/character vector specifying the unique category labels of the Categorical distribution.

Value

An object of class "CatDirichlet".

References

Murphy, Kevin P. Machine learning: a probabilistic perspective. MIT press, 2012.

See Also

posterior.CatDirichlet,posteriorDiscard.CatDirichlet,MAP.CatDirichlet,MPE.CatDirichlet,marginalLikeli
...

Examples

```
obj <- CatDirichlet(gamma=list(alpha=c(1,2,1),uniqueLabels = letters[1:3]))
obj #print the content
```

CatDP *Create objects of type "CatDP".*

Description

Create an object of type "CatDP", which represents the Categorical-Dirichlet-Process(Multinomial-Dirichlet-Process) conjugate structure on positive integers:

$$pi|alpha \sim DP(alpha, U)$$

$$x|pi \sim Categorical(pi)$$

where DP(alpha,U) is a Dirichlet Process on positive integers, alpha is the "concentration parameter" of the Dirichlet Process, U is the "base measure" of this Dirichlet process, it is an uniform distribution on all positive integers.Categorical() is the Categorical distribution. See dCategorical for the definition of the Categorical distribution.
In the case of CatDP, x can only be positive integers.
This object will be used as a place for recording and accumulating information in the related inference/sampling functions such as posterior(), posteriorDiscard(), MAP(), marginalLikelihood(), dPosteriorPredictive(), rPosteriorPredictive() and so on.

Usage

```
CatDP(objCopy = NULL, ENV = parent.frame(), gamma = list(alpha = 1))
```

Arguments

objCopy	an object of type "CatDP". If "objCopy" is not NULL, the function create a new "CatDP" object by copying the content from objCopy, otherwise this new object will be created by using "ENV" and "gamma". Default NULL.
ENV	environment, specify where the object will be created.
gamma	list, a named list of parameters, gamma=list(alpha). Where gamma$alpha is a numeric value specifying the concentration parameter of the Dirichlet Process.

Value

An object of class "CatDP".

References

Teh, Yee W., et al. "Sharing clusters among related groups: Hierarchical Dirichlet processes." Advances in neural information processing systems. 2005.

See Also

posterior.CatDP,posteriorDiscard.CatDP,MAP.CatDP,marginalLikelihood.CatDP, dPosteriorPredictive.CatDP, rPosteriorPredictive.CatDP

Examples

```
obj <- CatDP(gamma=list(alpha=2))
obj #print the content
```

CatHDP *Create objects of type "CatHDP".*

Description

Create an object of type "CatHDP", which represents the Categorical-Hierarchical-Dirichlet-Process(Multinomial-Hierarchical-Dirichlet-Process) conjugate structure on positive integers:

$$G|gamma \sim DP(gamma, U)$$

$$pi_j|G, alpha \sim DP(alpha, G), j = 1 : J$$

$$z|pi_j \sim Categorical(pi_j)$$

$$k|z, G \sim Categorical(G), \text{ if z is a sample from the base measure G}$$

where DP(gamma,U) is a Dirichlet Process on positive integers, gamma is the "concentration parameter", U is the "base measure" of this Dirichlet process, U is an uniform distribution on all positive integers. DP(alpha,G) is a Dirichlet Process on integers with concentration parameter alpha and base measure G. Categorical() is the Categorical distribution. See `dCategorical` for the definition of the Categorical distribution.
In the case of CatHDP, z and k can only be positive integers.
This object will be used as a place for recording and accumulating information in the related inference/sampling functions such as posterior(), posteriorDiscard(), rPosteriorPredictive() and so on.

Usage

```
CatHDP(
  objCopy = NULL,
  ENV = parent.frame(),
  gamma = list(gamma = 1, alpha = 1, j = 2)
)
```

Arguments

objCopy	an object of type "CatHDP". If "objCopy" is not NULL, the function create a new "CatHDP" object by copying the content from objCopy, otherwise this new object will be created by using "ENV" and "gamma". Default NULL.
ENV	environment, specify where the object will be created.
gamma	list, a named list of parameters, gamma=list(gamma,alpha,j). Where gamma$gamma is a numeric value specifying the concentration parameter of DP(gamma,U), gamma$alpha is a numeric value specifying the concentration parameter of DP(alpha,G), gamma$j is the number of groups J.

Value

An object of class "CatHDP".

References

Teh, Yee W., et al. "Sharing clusters among related groups: Hierarchical Dirichlet processes." Advances in neural information processing systems. 2005.

See Also

posterior.CatHDP,posteriorDiscard.CatHDP ...

Examples

```
obj <- CatHDP(gamma=list(gamma=1,alpha=1,j=2))
obj #print the content
```

CatHDP2 *Create objects of type "CatHDP2".*

Description

Create an object of type "CatHDP2" that represents the Categorical-Hierarchical-Dirichlet-Process of two Dirichlet Process hierarchies, which is basically CatHDP with an additional layer of Dirichlet Process:

$$G|eta \sim DP(eta, U)$$

$$G_m|gamma \sim DP(gamma, G), m = 1 : M$$

$$pi_{mj}|G_m, alpha \sim DP(alpha, G_m), j = 1 : J_m$$

$$z|pi_{mj} \sim Categorical(pi_{mj})$$

$$k|z, G_m \sim Categorical(G_m), \text{ if z is a sample from the base measure } G_m$$

$$u|k, G \sim Categorical(G), \text{ if k is a sample from the base measure G}$$

where DP(eta,U) is a Dirichlet Process on positive integers, eta is the "concentration parameter", U is the "base measure" of this Dirichlet process, U is an uniform distribution on all positive integers.

DP(gamma,G) is a Dirichlet Process on integers with concentration parameter gamma and base measure G. DP(alpha,G_m) is a Dirichlet Process on integers with concentration parameter alpha and base measure G_m. Categorical() is the Categorical distribution. See `dCategorical` for the definition of the Categorical distribution.
In the case of CatHDP2, u, z and k can only be positive integers.
This object will be used as a place for recording and accumulating information in the related inference/sampling functions such as posterior(), posteriorDiscard(), dPosteriorPredictive(), rPosteriorPredictive() and so on.

Usage

```
CatHDP2(
  objCopy = NULL,
  ENV = parent.frame(),
  gamma = list(eta = 1, gamma = 1, alpha = 1, m = 3, j = c(2, 3, 4))
)
```

Arguments

`objCopy`	an object of type "CatHDP2". If "objCopy" is not NULL, the function create a new "CatHDP2" object by copying the content from objCopy, otherwise this new object will be created by using "ENV" and "gamma". Default NULL.
`ENV`	environment, specify where the object will be created.
`gamma`	list, a named list of parameters, gamma=list(eta,gamma,alpha,m,j). Where gamma$eta is a numeric value specifying the concentration parameter of DP(eta,U), gamma$gamma is a numeric value specifying the concentration parameter of DP(gamma,G), gamma$alpha is a numeric value specifying the concentration parameter of DP(alpha,G_m), gamma$m is the number of groups M, gamma$j is the number of subgroups in each group, must satisfy length(gamma$j)=gamma$m.

Value

An object of class "CatHDP2".

References

Teh, Yee W., et al. "Sharing clusters among related groups: Hierarchical Dirichlet processes." Advances in neural information processing systems. 2005.

See Also

`posterior.CatHDP2`,`posteriorDiscard.CatHDP2` ...

Examples

```
obj <- CatHDP2(gamma=list(eta=1,gamma=1,alpha=1,m=2,j=c(2,3)))
obj #print the content
```

`dAllIndicators` *Get the probabilities of all possible values of the hidden indicator variables from the DP family objects.*

Description

This is a generic function that will get the probabilities of all possible values of the hidden indicator variables. i.e. for the model structure:

$$theta|gamma_1 \sim H1(gamma_1)$$
$$z|gamma_2 \sim H2(gamma_2)$$
$$x|theta, z \sim F(theta_z)$$

get p(z|x,gamma_2,gamma_1) for all possible values of z. Where H2 is either CatDP, CatHDP, CatHDP2, DP, HDP, or HDP2.

Usage

```
dAllIndicators(obj, ...)
```

Arguments

`obj`	A "CatDP", "CatHDP", "CatHDP2", "DP", "HDP", or a "HDP2" object used to select a method.
`...`	further arguments passed to or from other methods.

Value

a data.frame of indicator values and their corresponding probabilities.

`dAllIndicators.HDP` *Get the probabilities of all possible values of the hidden indicator variables of an "HDP" object.*

Description

Get p(z,k|gamma,alpha,psi,j,x), or p(z,k|gamma,alpha,psi,j) for the model structure:

$$G|gamma \sim DP(gamma, U)$$
$$pi_j|G, alpha \sim DP(alpha, G), j = 1 : J$$
$$z|pi_j \sim Categorical(pi_j)$$
$$k|z, G \sim Categorical(G), \text{ if z is a sample from the base measure G}$$
$$theta_k|psi \sim H0(psi)$$

$$x|theta_k, k \sim F(theta_k)$$

where DP(gamma,U) is a Dirichlet Process on positive integers, gamma is the "concentration parameter", U is the "base measure" of this Dirichlet process, U is an uniform distribution on all positive integers. DP(alpha,G) is a Dirichlet Process on integers with concentration parameter alpha and base measure G. The choice of F() and H0() can be described by an arbitrary "BasicBayesian" object such as "GaussianGaussian","GaussianInvWishart","GaussianNIW", "GaussianNIG", "CatDirichlet", and "CatDP". See `?BasicBayesian` for definition of "BasicBayesian" objects, and see for example `?GaussianGaussian` for specific "BasicBayesian" instances. As a summary, An "HDP" object is simply a combination of a "CatHDP" object (see `?CatHDP`) and an object of any "BasicBayesian" type.
In the case of HDP, z and k can only be positive integers.
This function will return all possible values of z, k and their corresponding probabilities

Usage

```
## S3 method for class 'HDP'
dAllIndicators(obj, j, x = NULL, ...)
```

Arguments

`obj`	A "HDP" object.
`j`	integer, the group ID.
`x`	the observation. The data type of x must fit the observation distribution specified by "H0aF" when initiating the "HDP" object.
`...`	further arguments passed to or from other methods.

Value

a data.frame of three columns, the first two columns are all possible values of z and k, the third column is the corresponding probabilities.

`dCategorical` *Probability mass function for Categorical distribution*

Description

Calculate probability masses for integer valued Categorical random samples. For a random variable x, the density function of categorical distribution is defined as

$$prod_{kin1:K} p_k^{I(x=k)}$$

Where K is the number of unique values.

Usage

```
dCategorical(x, p)
```

Arguments

x	integer, categorical samples.
p	numeric, probabilities.

Value

A numeric vector of the same length of 'x'.

See Also

rCategorical

Examples

```
dCategorical(x=c(1L,2L,1L),p=c(1,2))
```

---

dDir *Density function for Dirichelt distribution*

Description

Calculate the densities of a given set of Dirichlet samples. For a random vector x, the density function is defined as: 1/Beta(alpha) prod_i=1:p x_i^alpha_i -1 Where Beta() is the beta function. p is the dimension of x.

Usage

```
dDir(x, alpha, LOG = FALSE)
```

Arguments

x	matrix or numeric vector, if matrix every row of x is an observation, if numeric vector, it's the same as a matrix with only one row.
alpha	numeric, Dirichlet parameter.
LOG	logical, return the log density if set to "TRUE".

Value

A numeric vector of density values.

See Also

rDir

Examples

```
x <- rDir(5,c(1,2,3)) #generate 5 samples with parameters c(1,2,3)
dDir(x,c(1,2,3))
dDir(x,c(1,2,3),LOG=TRUE)
```

---

dGaussian *Density function of Gaussian distribution*

Description

Get the density of a set of samples from a (multivariate) Gaussian distribution. For a random vector x, the density function is defined as:

$$sqrt(2pi^p|Sigma|)^{-1}exp(-1/2(x-mu)^T Sigma^{-1}(x-mu))$$

where p is the dimension of x.

Usage

```
dGaussian(x, mu, Sigma = NULL, A = NULL, LOG = TRUE)
```

Arguments

x	matrix, when x is a numeric vector, it will be converted to a matrix with 1 column!
mu	numeric, mean vector.
Sigma	matrix, covariance matrix, one of Sigma and A should be non-NULL.
A	matrix, the Cholesky decomposition of Sigma, an upper triangular matrix, one of Sigma and A should be non-NULL.
LOG	logical, return log density of LOG=TRUE, default TRUE.

Value

A numeric vector.

See Also

rGaussian

Examples

```
plot(
   dGaussian(x=seq(-5,5,length.out = 1000),mu = 0,Sigma = 1,LOG = FALSE)
   ,type = "l"
)
```

dInvGamma *Density function of Inverse-Gamma distribution*

Description

For a random variable x, the density function of Inverse-Gamma distribution is defined as:

$$(rate^shape)/Gamma(shape)x^{-shape-1}exp(-rate/x)$$

Usage

```
dInvGamma(x, shape, scale, LOG = TRUE)
```

Arguments

x	numeric, positive numeric values.
shape	numeric, the shape parameter of gamma distribution.
scale	numeric, the scale, or inverse-scale parameter of gamma distribution. The 'rate' parameter in Gamma is the 'scale' parameter in InvGamma
LOG	logical, return log density of LOG=TRUE, default TRUE.

Value

A numeric vector, the density values.

dInvWishart *Density function of Inverse-Wishart distribution*

Description

For a random matrix x, The density function of Inverse-Wishart distribution is defined as:

$$(2^{(dfp)/2}Gamma_p(df/2)|scale|^{-df/2})^{-1}|x|^{(-df-p-1)/2}exp(-1/2tr(x^{-1}scale))$$

Where x is a pxp symmetric positive definite matrix, Gamma_p() is the multivariate Gamma function of dimension p.

Usage

```
dInvWishart(x, df, scale, LOG = TRUE)
```

Arguments

x	matrix, a symmetric positive-definite matrix.
df	numeric, the degree of freedom.
scale	matrix, a symmetric positive-definite matrix, the 'scale' parameter. The 'rate' parameter in Wishart is the 'scale' parameter in InvWishart.
LOG	logical, return log density of LOG=TRUE, default TRUE.

Value

A numeric vector, the density values.

References

Wishart, John. "The generalized product moment distribution in samples from a normal multivariate population." Biometrika (1928): 32-52.

MARolA, K. V., JT KBNT, and J. M. Bibly. Multivariate analysis. AcadeInic Press, Londres, 1979.

Examples

```
x <- crossprod(matrix(rnorm(15),5,3)) #generate a symmetric positive-definite matrix
scale <- crossprod(matrix(rnorm(15),5,3)) #the prior scale of x
dInvWishart(x,df = 5,scale = scale,LOG = TRUE)
dInvWishart(x,df = 5,scale = scale,LOG = FALSE)
```

---

dNIW *Density function for Normal-Inverse-Wishart (NIW) distribution.*

Description

Get the density of a NIW sample. For a random vector mu, and a random matrix Sigma, the density function is defined as:

$sqrt(2pi^p|Sigma/k|)^{-1}exp(-1/2(mu-m)^T(Sigma/k)^{-1}(mu-m))(2^{(vp)/2}Gamma_p(v/2)|S|^{-v/2})^{-1}|Sigma|^{(-v-p-$

Where p is the dimension of mu and Sigma.

Usage

```
dNIW(mu, Sigma, m, k, v, S, LOG = TRUE)
```

Arguments

mu	numeric, the Gaussian sample.
Sigma	matrix, a symmetric positive definite matrix, the Inverse-Wishart sample.
m	numeric, mean of mu.
k	numeric, precision of mu.
v	numeric, degree of freedom of Sigma.
S	numeric, a symmetric positive definite scale matrix of Sigma, S is proportional to E(Sigma).
LOG	logical, return log density of LOG=TRUE, default TRUE.

Value

A numeric vector, the probability density of (mu,Sigma).

References

O'Hagan, Anthony, and Jonathan J. Forster. Kendall's advanced theory of statistics, volume 2B: Bayesian inference. Vol. 2. Arnold, 2004.

MARolA, K. V., JT KBNT, and J. M. Bibly. Multivariate analysis. AcadeInic Press, Londres, 1979.

See Also

rNIW

Examples

```
S <- crossprod(matrix(rnorm(15),5,3))
Sigma <- crossprod(matrix(rnorm(15),5,3))
mu <- runif(3)
m <- runif(3)
dNIW(mu=mu,Sigma=Sigma,m=m,k=2,v=4,S=S,LOG = TRUE)
```

DP *Create objects of type "DP".*

Description

Create an object of type "DP", which represents the Dirichlet-Process model structure:

$$pi|alpha \sim DP(alpha, U)$$

$$z|pi \sim Categorical(pi)$$

$$theta_z|psi \sim H0(psi)$$

$$x|theta_z, z \sim F(theta_z)$$

where DP(alpha,U) is a Dirichlet Process on positive integers, alpha is the "concentration parameter" of the Dirichlet Process, U is the "base measure" of this Dirichlet process. The choice of F() and H0() can be described by an arbitrary "BasicBayesian" object such as "GaussianGaussian","GaussianInvWishart","GaussianNIW", "GaussianNIG", "CatDirichlet", and "CatDP". See `?BasicBayesian` for definition of "BasicBayesian" objects, and see for example `?GaussianGaussian` for specific "BasicBayesian" instances. As a summary, An "DP" object is simply a combination of a "CatDP" object (see `?CatDP`) and an object of any "BasicBayesian" type.
The "DP" object will be used as a place for recording and accumulating information in the related inference/sampling functions such as posterior(), posteriorDiscard(), dPosteriorPredictive(), rPosteriorPredictive() and so on.

Usage

```
DP(
  objCopy = NULL,
  ENV = parent.frame(),
 gamma = list(alpha = 1, H0aF = "GaussianNIW", parH0 = list(m = 0, k = 1, v = 2, S =
    1))
)
```

Arguments

`objCopy`	an object of type "DP". If "objCopy" is not NULL, the function create a new "DP" object by copying the content from objCopy, otherwise this new object will be created by using "ENV" and "gamma". Default NULL.
`ENV`	environment, specify where the object will be created.
`gamma`	list, a named list of parameters, gamma=list(alpha,H0aF,parH0). Where gamma$alpha is a numeric value specifying the concentration parameter of the Dirichlet Process. gamma$H0aF is the name of the "BasicBayesian" object which specifies the structure of H0() and F(). gamma$parH0 is the parameters passed to the selected H0aF. For example, if gamma$H0aF="GaussianNIW", then gamma$parH0 should be a named list of NIW parameters: gamma$parH0=list(m,k,v,S), where gamma$parH0$m is a numeric "location" parameter; gamma$parH0$S is a symmetric positive definite matrix representing the "scale" parameters; gamma$parH0$k and gamma$parH0$v are numeric values.

Value

An object of class "DP".

References

Teh, Yee W., et al. "Sharing clusters among related groups: Hierarchical Dirichlet processes." Advances in neural information processing systems. 2005.

Li, Yuelin, Elizabeth Schofield, and Mithat Gönen. "A tutorial on Dirichlet process mixture modeling." Journal of Mathematical Psychology 91 (2019): 128-144.

See Also

BasicBayesian,GaussianNIW,GaussianNIG,CatDirichlet,CatDP,posterior.DP,posteriorDiscard.DP,marginalLikel
...

Examples

```
## This is an example of Gibbs sampling on Gaussian mixture models, using Dirichlet Process.

## generate some Gaussian mixture samples for demo purpose
x <- rbind(
    rGaussian(50,mu = c(-1.5,1.5),Sigma = matrix(c(0.1,0.03,0.03,0.1),2,2)),
    rGaussian(60,mu = c(-1.5,-1.5),Sigma = matrix(c(0.8,0.5,0.5,0.8),2,2)),
    rGaussian(70,mu = c(1.5,1.5),Sigma = matrix(c(0.3,0.05,0.05,0.3),2,2)),
    rGaussian(50,mu = c(1.5,-1.5),Sigma = matrix(c(0.5,-0.08,-0.08,0.5),2,2))
)

## Step1: Initialize---------------------------------------------
maxit <- 100                            #iterative for maxit times
burnin <- 50                            #number of burnin samples
z <- matrix(1L,nrow(x),maxit-burnin)    #labels
## create an "GaussianNIW" object to track all the changes.
obj <- DP(gamma = list(alpha=1,H0aF="GaussianNIW",parH0=list(m=c(0,0),k=0.001,v=2,S=diag(2))))
ss <- sufficientStatistics(obj,x=x,foreach = TRUE) #sufficient statistics of each x
N <- nrow(x)
for(i in 1L:N){ # initialize labels before Gibbs sampling
    z[i,1] <- rPosteriorPredictive(obj = obj,n=1,x=x[i,,drop=FALSE])
    posterior(obj = obj,ss = ss[[i]], z = z[i,1])
}

## Step2: Main Gibbs sampling loop-------------------------------
it <- 0                                 #iteration tracker
pb <- txtProgressBar(min = 0,max = maxit,style = 3)
while(TRUE){
    if(it>burnin) colIdx <- it-burnin
    else colIdx <- 1
    for(i in 1L:N){
        ## remove the sample information from the posterior
        posteriorDiscard(obj = obj,ss = ss[[i]],z=z[i,colIdx])
        ## get a new sample
        z[i,colIdx] <- rPosteriorPredictive(obj = obj,n=1,x=x[i,,drop=FALSE])
        ## add the new sample information to the posterior
        posterior(obj = obj,ss = ss[[i]],z=z[i,colIdx])
    }
   if(it>burnin & colIdx<ncol(z)) z[,colIdx+1] <- z[,colIdx] #copy result of previous iteration
    it <- it+1
    setTxtProgressBar(pb,it)
    if(it>=maxit){cat("\n");break}
    plot(x=x[,1],y=x[,2],col=z[,colIdx]) #to see how the labels change
}
```

```
## Step3: Estimate group labels of each observation---------------
col <- apply(z,1,function(l){
    tmp <- table(l)
    ## pick the most frequent group label in the samples to be the best estimate
    names(tmp)[which.max(tmp)]
})
plot(x=x[,1],y=x[,2],col=col)
```

---

dPosterior *Get the density from the posterior distribution.*

Description

This is a generic function that will generate the the density value of the posterior distribution. i.e. for the model structure:

$$theta|gamma \sim H(gamma)$$

$$x|theta \sim F(theta)$$

get the probability density/mass from the distribution $theta \sim H(gamma)$. For a given Bayesian bricks object obj and an observation of theta, `dPosterior()` will calculate the density value for different model structures:

class(obj)="LinearGaussianGaussian": Where

$$x \sim Gaussian(Az + b, Sigma)$$

$$z \sim Gaussian(m, S)$$

`dPosterior()` will return p(theta|m,S) See `?dPosterior.LinearGaussianGaussian` for details.

class(obj)="GaussianGaussian": Where

$$x \sim Gaussian(mu, Sigma)$$

$$mu \sim Gaussian(m, S)$$

Sigma is known. `dPosterior()` will return p(mu|m,S) See `?dPosterior.GaussianGaussian` for details.

class(obj)="GaussianInvWishart": Where

$$x \sim Gaussian(mu, Sigma)$$

$$Sigma \sim InvWishart(v, S)$$

mu is known. `dPosterior()` will return p(Sigma|v,S) See `?dPosterior.GaussianInvWishart` for details.

class(obj)="GaussianNIW": Where

$$x \sim Gaussian(mu, Sigma)$$

$$Sigma \sim InvWishart(v, S)$$

$$mu \sim Gaussian(m, Sigma/k)$$

`dPosterior()` will return p(mu,Sigma|m,k,v,S) See `?dPosterior.GaussianNIW` for details.

class(obj)="GaussianNIG": Where

$$x \sim Gaussian(Xbeta, sigma^2)$$

$$sigma^2 \sim InvGamma(a, b)$$

$$beta \sim Gaussian(m, sigma^2 V)$$

X is a row vector, or a design matrix where each row is an obervation. `dPosterior()` will return p(beta,sigma^2|m,V,a,b) See `?dPosterior.GaussianNIG` for details.

class(obj)="CatDirichlet": Where

$$x \sim Categorical(pi)$$

$$pi \sim Dirichlet(alpha)$$

`dPosterior()` will return p(pi|alpha) See `?dPosterior.CatDirichlet` for details.

Usage

```
dPosterior(obj, ...)
```

Arguments

`obj`	A "BayesianBrick" object used to select a method.
`...`	further arguments passed to or from other methods.

Value

numeric, the density value

See Also

`dPosterior.LinearGaussianGaussian` for Linear Gaussian and Gaussian conjugate structure, `dPosterior.GaussianGaussian` for Gaussian-Gaussian conjugate structure, `dPosterior.GaussianInvWishart` for Gaussian-Inverse-Wishart conjugate structure, `dPosterior.GaussianNIW` for Gaussian-NIW conjugate structure, `dPosterior.GaussianNIG` for Gaussian-NIG conjugate structure, `dPosterior.CatDirichlet` for Categorical-Dirichlet conjugate structure ...

dPosterior.CatDirichlet *Density function of the posterior distribution of a "CatDirichlet" object*

Description

Generate the the density value of the posterior distribution of the following structure:

$$pi|alpha \sim Dir(alpha)$$

$$x|pi \sim Categorical(pi)$$

Where Dir() is the Dirichlet distribution, Categorical() is the Categorical distribution. See `?dDir` and `dCategorical` for the definitions of these distribution.
The model structure and prior parameters are stored in a "CatDirichlet" object.
Posterior density is the density function of Dir(pi|alpha).

Usage

```
## S3 method for class 'CatDirichlet'
dPosterior(obj, pi, LOG = TRUE, ...)
```

Arguments

`obj`	A "CatDirichlet" object.
`pi`	matrix or a numeric vector. When pi is a matrix, each row is an observation. When pi is a vector, it will be treated as only one observation.
`LOG`	Return the log density if set to "TRUE".
`...`	Additional arguments to be passed to other inherited types.

Value

numeric vector, the posterior densities for each row of pi.

See Also

`CatDirichlet`, `rPosterior.CatDirichlet`

Examples

```
obj <- CatDirichlet(gamma=list(alpha=runif(26),uniqueLabels = letters))
dPosterior(obj = obj,pi = runif(26))
dPosterior(obj = obj,pi = matrix(runif(26*10),nrow = 10))
```

`dPosterior.GaussianGaussian` *Density function of the posterior distribution of a "GaussianGaussian" object*

Description

Generate the the density value of the posterior distribution of the following structure:

$$x \sim Gaussian(mu, Sigma)$$

$$mu \sim Gaussian(m, S)$$

Where Sigma is known. Gaussian() is the Gaussian distribution. See `?dGaussian` for the definition of Gaussian distribution.
The model structure and prior parameters are stored in a "GaussianGaussian" object.
Posterior density is the density function of Gaussian(mu|m,S).

Usage

```
## S3 method for class 'GaussianGaussian'
dPosterior(obj, mu, LOG = TRUE, ...)
```

Arguments

`obj`	A "GaussianGaussian" object.
`mu`	matrix, or the ones that can be converted to matrix. Each row of mu is an sample.
`LOG`	Return the log density if set to "TRUE".
`...`	Additional arguments to be passed to other inherited types.

Value

A numeric vector of the same length as nrow(mu), the posterior density.

See Also

`GaussianGaussian`, `rPosterior.GaussianGaussian`

Examples

```
obj <- GaussianGaussian(gamma=list(Sigma=matrix(c(2,1,1,2),2,2),m=c(0.2,0.5),S=diag(2)))
mu <- rGaussian(10,mu=c(0,0),Sigma = diag(2))
dPosterior(obj=obj,mu=mu)
```

dPosterior.GaussianInvWishart *Density function of the posterior distribution of a "GaussianInvWishart" object*

Description

Generate the the density value of the posterior distribution of the following structure:

$$x \sim Gaussian(mu, Sigma)$$

$$Sigma \sim InvWishart(v, S)$$

mu is known. Gaussian() is the Gaussian distribution. See `?dGaussian` and `?dInvWishart` for the definition of the distributions.
The model structure and prior parameters are stored in a "GaussianInvWishart" object.
Posterior density is the density function of InvWishart(Sigmalv,S).

Usage

```
## S3 method for class 'GaussianInvWishart'
dPosterior(obj, Sigma, LOG = TRUE, ...)
```

Arguments

`obj`	A "GaussianInvWishart" object.
`Sigma`	matrix.
`LOG`	Return the log density if set to "TRUE".
`...`	Additional arguments to be passed to other inherited types.

Value

numeric, the posterior density of Sigma.

See Also

`GaussianInvWishart`, `rPosterior.GaussianInvWishart`

Examples

```
obj <- GaussianInvWishart(gamma=list(mu=c(-1.5,1.5),v=3,S=diag(2)))
Sigma <- rInvWishart(df = 3,scale = diag(2))
dPosterior(obj = obj,Sigma=Sigma)
```

dPosterior.GaussianNIG *Density function of the posterior distribution of a "GaussianNIG" object*

Description

Generate the the density value of the posterior distribution of the following structure:

$$x \sim Gaussian(Xbeta, sigma^2)$$

$$sigma^2 \sim InvGamma(a, b)$$

$$beta \sim Gaussian(m, sigma^2 V)$$

Where X is a row vector, or a design matrix where each row is an obervation. InvGamma() is the Inverse-Gamma distribution, Gaussian() is the Gaussian distribution. See `?dInvGamma` and `dGaussian` for the definitions of these distribution.
The model structure and prior parameters are stored in a "GaussianNIG" object.
Posterior density is the density function of beta,sigma^2|a,b,m,V.

Usage

```
## S3 method for class 'GaussianNIG'
dPosterior(obj, beta, sigma2, LOG = TRUE, ...)
```

Arguments

`obj`	A "GaussianNIG" object.
`beta`	numeric vector.
`sigma2`	numeric.
`LOG`	Return the log density if set to "TRUE".
`...`	Additional arguments to be passed to other inherited types.

Value

numeric, the posterior density of (beta,sigma^2).

See Also

`GaussianNIG`, `rPosterior.GaussianNIG`

Examples

```
obj <- GaussianNIG(gamma=list(m=c(0,0),V=diag(2),a=1,b=1))
dPosterior(obj = obj,beta=runif(2),sigma2=3)
```

dPosterior.GaussianNIW *Density function of the posterior distribution of a "GaussianNIW" object*

Description

Generate the the density value of the posterior distribution of the following structure:

$$mu, Sigma|m, k, v, S \sim NIW(m, k, v, S)$$

$$x|mu, Sigma \sim Gaussian(mu, Sigma)$$

Where NIW() is the Normal-Inverse-Wishart distribution, Gaussian() is the Gaussian distribution. See ?dNIW and dGaussian for the definitions of these distribution.
The model structure and prior parameters are stored in a "GaussianNIW" object.
Posterior density is the density function of NIW(mu,Sigma|m,k,v,S).

Usage

```
## S3 method for class 'GaussianNIW'
dPosterior(obj, mu, Sigma, LOG = TRUE, ...)
```

Arguments

obj	A "GaussianNIW" object.
mu	vector.
Sigma	matrix, nrow(Sigma) = length(mu).
LOG	Return the log density if set to "TRUE".
...	Additional arguments to be passed to other inherited types.

Value

numeric, the posterior density of (mu,Sigma).

See Also

GaussianNIW, rPosterior.GaussianNIW

Examples

```
obj <- GaussianNIW(gamma=list(m=c(0,0),k=1,v=2,S=diag(2)))
mu <- rnorm(2)
Sigma <- rInvWishart(df = 3,scale = diag(2))
dPosterior(obj = obj,mu=mu,Sigma = Sigma)
```

`dPosterior.LinearGaussianGaussian` *Posterior density function of a "LinearGaussianGaussian" object*

Description

Generate the the density value of the posterior distribution of the following structure:

$$x \sim Gaussian(Az + b, Sigma)$$

$$z \sim Gaussian(m, S)$$

Where Sigma is known. A is a $dimxdimz$ matrix, x is a $dimx1$ random vector, z is a $dimz1$ random vector, b is a $dimmx1$ vector. Gaussian() is the Gaussian distribution. See `?dGaussian` for the definition of Gaussian distribution.
The model structure and prior parameters are stored in a "LinearGaussianGaussian" object.
Posterior density is the density function of Gaussian(z|m,S).

Usage

```
## S3 method for class 'LinearGaussianGaussian'
dPosterior(obj, z, LOG = TRUE, ...)
```

Arguments

`obj`	A "LinearGaussianGaussian" object.
`z`	matrix, or the ones that can be converted to matrix. Each row of z is an sample.
`LOG`	Return the log density if set to "TRUE".
`...`	Additional arguments to be passed to other inherited types.

Value

A numeric vector of the same length as nrow(z), the posterior density.

See Also

`LinearGaussianGaussian`, `rPosterior.LinearGaussianGaussian`

Examples

```
obj <- LinearGaussianGaussian(gamma=list(Sigma=matrix(c(2,1,1,2),2,2),
                                         m=c(0.2,0.5,0.6),S=diag(3)))
z <- rGaussian(10,mu = runif(3),Sigma = diag(3))
dPosterior(obj = obj,z=z)
dPosterior(obj = obj,z=z,LOG=FALSE)
```

dPosteriorPredictive *Get the density value of the posterior predictive distribution*

Description

This is a generic function that will generate the the density value of the posterior predictive distribution. i.e. for the model structure:

$$theta|gamma \sim H(gamma)$$

$$x|theta \sim F(theta)$$

get the probability density/mass of the posterior predictive distribution of a new sample x_new: p(x_new|gamma). For a given Bayesian bricks object obj and a new sample x, `dPosteriorPredictive()` will calculate the marginal likelihood for different model structures:

class(obj)="LinearGaussianGaussian": Where

$$x \sim Gaussian(Az + b, Sigma)$$

$$z \sim Gaussian(m, S)$$

`dPosteriorPredictive()` will return p(x|m,S,A,b,Sigma) See `?dPosteriorPredictive.LinearGaussianGaussian` for details.

class(obj)="GaussianGaussian": Where

$$x \sim Gaussian(mu, Sigma)$$

$$mu \sim Gaussian(m, S)$$

Sigma is known. `dPosteriorPredictive()` will return p(x|m,S,Sigma) See `?dPosteriorPredictive.GaussianGaussia` for details.

class(obj)="GaussianInvWishart": Where

$$x \sim Gaussian(mu, Sigma)$$

$$Sigma \sim InvWishart(v, S)$$

mu is known. `dPosteriorPredictive()` will return p(x|mu,v,S) See `?dPosteriorPredictive.GaussianInvWishart` for details.

class(obj)="GaussianNIW": Where

$$x \sim Gaussian(mu, Sigma)$$

$$Sigma \sim InvWishart(v, S)$$

$$mu \sim Gaussian(m, Sigma/k)$$

`dPosteriorPredictive()` will return p(x|m,k,v,S) See `?dPosteriorPredictive.GaussianNIW` for details.

class(obj)="GaussianNIG": Where

$$x \sim Gaussian(Xbeta, sigma^2)$$

$$sigma^2 \sim InvGamma(a, b)$$

$$beta \sim Gaussian(m, sigma^2 V)$$

X is a row vector, or a design matrix where each row is an obervation. `dPosteriorPredictive()` will return p(x,X|m,V,a,b) See `?dPosteriorPredictive.GaussianNIG` for details.

class(obj)="CatDirichlet": Where

$$x \sim Categorical(pi)$$

$$pi \sim Dirichlet(alpha)$$

`dPosteriorPredictive()` will return p(x|alpha) See `?dPosteriorPredictive.CatDirichlet` for details.

class(obj)="CatDP": Where

$$x \sim Categorical(pi)$$

$$pi \sim DirichletProcess(alpha)$$

`dPosteriorPredictive()` will return p(x|alpha) See `?dPosteriorPredictive.CatDP` for details.

Usage

```
dPosteriorPredictive(obj, ...)
```

Arguments

`obj`	A "BayesianBrick" object used to select a method.
`...`	further arguments passed to or from other methods.

Value

numeric, the density value

See Also

`dPosteriorPredictive.LinearGaussianGaussian` for Linear Gaussian and Gaussian conjugate structure, `dPosteriorPredictive.GaussianGaussian` for Gaussian-Gaussian conjugate structure, `dPosteriorPredictive.GaussianInvWishart` for Gaussian-Inverse-Wishart conjugate structure, `dPosteriorPredictive.GaussianNIW` for Gaussian-NIW conjugate structure, `dPosteriorPredictive.GaussianNIG` for Gaussian-NIG conjugate structure, `dPosteriorPredictive.CatDirichlet` for Categorical-Dirichlet conjugate structure, `dPosteriorPredictive.CatDP` for Categorical-DP conjugate structure ...

dPosteriorPredictive.CatDirichlet
Posterior predictive density function of a "CatDirichlet" object

Description

Generate the the density value of the posterior predictive distribution of the following structure:

$$pi|alpha \sim Dir(alpha)$$

$$x|pi \sim Categorical(pi)$$

Where Dir() is the Dirichlet distribution, Categorical() is the Categorical distribution. See `?dDir` and `dCategorical` for the definitions of these distribution.
The model structure and prior parameters are stored in a "CatDirichlet" object.
Posterior predictive is a distribution of x|alpha.

Usage

```
## S3 method for class 'CatDirichlet'
dPosteriorPredictive(obj, x, LOG = TRUE, ...)
```

Arguments

`obj`	A "CatDirichlet" object.
`x`	numeric/integer/character vector, observed Categorical samples.
`LOG`	Return the log density if set to "TRUE".
`...`	Additional arguments to be passed to other inherited types.

Value

A numeric vector, the posterior predictive density.

References

Murphy, Kevin P. Machine learning: a probabilistic perspective. MIT press, 2012.

See Also

`CatDirichlet`, `dPosteriorPredictive.CatDirichlet`, `marginalLikelihood.CatDirichlet`

Examples

```
obj <- CatDirichlet(gamma=list(alpha=runif(26,1,2),uniqueLabels = letters))
x <- sample(letters,size = 20,replace = TRUE)
## res1 and res2 should provide the same result
res1 <- dPosteriorPredictive(obj = obj,x=x,LOG = TRUE)
res2 <- numeric(length(x))
for(i in seq_along(x)) res2[i] <- marginalLikelihood(obj=obj,x=x[i],LOG = TRUE)
```

dPosteriorPredictive.CatDP

Posterior predictive density function of a "CatDP" object

Description

Generate the the density value of the posterior predictive distribution of the following structure:

$$pi|alpha \sim DP(alpha, U)$$

$$x|pi \sim Categorical(pi)$$

where DP(alpha,U) is a Dirichlet Process on positive integers, alpha is the "concentration parameter" of the Dirichlet Process, U is the "base measure" of this Dirichlet process, it is an uniform distribution on all positive integers.Categorical() is the Categorical distribution. See `dCategorical` for the definition of the Categorical distribution.
In the case of CatDP, x can only be positive integers.
The model structure and prior parameters are stored in a "CatDP" object.
Posterior predictive density is p(x|alpha).

Usage

```
## S3 method for class 'CatDP'
dPosteriorPredictive(obj, x, LOG = TRUE, ...)
```

Arguments

`obj`	A "CatDP" object.
`x`	integer, the elements of the vector must all greater than 0, the samples of a Categorical distribution.
`LOG`	Return the log density if set to "TRUE".
`...`	Additional arguments to be passed to other inherited types.

Value

A numeric vector, the posterior predictive density.

References

Teh, Yee W., et al. "Sharing clusters among related groups: Hierarchical Dirichlet processes." Advances in neural information processing systems. 2005.

See Also

`CatDP`, `dPosteriorPredictive.CatDP`, `marginalLikelihood.CatDP`

Examples

```
x <- sample(1L:10L,size = 40,replace = TRUE)
obj <- CatDP()
ss <- sufficientStatistics(obj=obj,x=x)
posterior(obj = obj,ss = ss)
dPosteriorPredictive(obj = obj,x=1L:11L,LOG = FALSE)
```

`dPosteriorPredictive.CatHDP`
Posterior predictive density function of a "CatHDP" object

Description

Generate the the density value of the posterior predictive distribution of the following structure:

$$G|gamma \sim DP(gamma, U)$$

$$pi_j|G, alpha \sim DP(alpha, G), j = 1 : J$$

$$z|pi_j \sim Categorical(pi_j)$$

$$k|z, G \sim Categorical(G), \text{ if z is a sample from the base measure G}$$

where DP(gamma,U) is a Dirichlet Process on positive integers, gamma is the "concentration parameter", U is the "base measure" of this Dirichlet process, U is an uniform distribution on all positive integers. DP(alpha,G) is a Dirichlet Process on integers with concentration parameter alpha and base measure G. Categorical() is the Categorical distribution. See `dCategorical` for the definition of the Categorical distribution.
In the case of CatHDP, z and k can only be positive integers.
The model structure and prior parameters are stored in a "CatHDP" object.
Posterior predictive density = p(z,k|alpha,gamma,U,j)

Usage

```
## S3 method for class 'CatHDP'
dPosteriorPredictive(obj, z, k, j, LOG = TRUE, ...)
```

Arguments

`obj`	A "CatHDP" object.
`z`	integer, the elements of the vector must all greater than 0, the samples of a Categorical distribution.
`k`	integer, the elements of the vector must all greater than 0, the samples of a Categorical distribution.
`j`	integer, group label.
`LOG`	Return the log density if set to "TRUE".
`...`	Additional arguments to be passed to other inherited types.

Value

A numeric vector, the posterior predictive density.

References

Teh, Yee W., et al. "Sharing clusters among related groups: Hierarchical Dirichlet processes." Advances in neural information processing systems. 2005.

See Also

CatHDP, dPosteriorPredictive.CatHDP

---

dPosteriorPredictive.CatHDP2

Posterior predictive density function of a "CatHDP" object

Description

Generate the the density value of the posterior predictive distribution of the following structure:

$$G|eta \sim DP(eta, U)$$

$$G_m|gamma \sim DP(gamma, G), m = 1 : M$$

$$pi_{mj}|G_m, alpha \sim DP(alpha, G_m), j = 1 : J_m$$

$$z|pi_{mj} \sim Categorical(pi_{mj})$$

$$k|z, G_m \sim Categorical(G_m),$$ if z is a sample from the base measure G_m

$$u|k, G \sim Categorical(G),$$ if k is a sample from the base measure G

where DP(eta,U) is a Dirichlet Process on positive integers, eta is the "concentration parameter", U is the "base measure" of this Dirichlet process, U is an uniform distribution on all positive integers. DP(gamma,G) is a Dirichlet Process on integers with concentration parameter gamma and base measure G. DP(alpha,G_m) is a Dirichlet Process on integers with concentration parameter alpha and base measure G_m. Categorical() is the Categorical distribution. See `dCategorical` for the definition of the Categorical distribution.
In the case of CatHDP2, u, z and k can only be positive integers.
The model structure and prior parameters are stored in a "CatHDP" object.
Posterior predictive density = p(u,z,k|alpha,gamm,eta,U).

Usage

```
## S3 method for class 'CatHDP2'
dPosteriorPredictive(obj, u, k, z, m, j, LOG = TRUE, ...)
```

Arguments

obj	A "CatHDP" object.
u	integer, the elements of the vector must all greater than 0, the samples of a Categorical distribution.
k	integer, the elements of the vector must all greater than 0, the samples of a Categorical distribution.
z	integer, the elements of the vector must all greater than 0, the samples of a Categorical distribution.
m	integer, group label.
j	integer, subgroup label.
LOG	Return the log density if set to "TRUE".
...	Additional arguments to be passed to other inherited types.

Value

A numeric vector, the posterior predictive density.

References

Teh, Yee W., et al. "Sharing clusters among related groups: Hierarchical Dirichlet processes." Advances in neural information processing systems. 2005.

See Also

CatHDP, dPosteriorPredictive.CatHDP

---

dPosteriorPredictive.DP

Posterior predictive density function of a Dirichlet Process object

---

Description

Generate the the density value of the posterior predictive distribution of the following structure:

$$pi|alpha \sim DP(alpha, U)$$

$$z|pi \sim Categorical(pi)$$

$$theta_z|psi \sim H0(psi)$$

$$x|theta_z, z \sim F(theta_z)$$

where DP(alpha,U) is a Dirichlet Process on positive integers, alpha is the "concentration parameter" of the Dirichlet Process, U is the "base measure" of this Dirichlet process. The choice of F() and H0() can be described by an arbitrary "BasicBayesian" object such as "GaussianGaussian","GaussianInvWishart","GaussianNIW", "GaussianNIG", "CatDirichlet", and "CatDP". See ?BasicBayesian for definition of "BasicBayesian" objects, and see for example ?GaussianGaussian

for specific "BasicBayesian" instances. As a summary, An "DP" object is simply a combination of a "CatDP" object (see ?CatDP) and an object of any "BasicBayesian" type.
The model structure and prior parameters are stored in a "DP" object.
Posterior predictive density = p(x,z|alpha,psi).

Usage

```
## S3 method for class 'DP'
dPosteriorPredictive(obj, x, z, LOG = TRUE, ...)
```

Arguments

obj	A "DP" object.
x	Random samples of the "BasicBayesian" object.
z	integer, the partition label of the parameter space where the observation x is drawn from.
LOG	Return the log density if set to "TRUE".
...	Additional arguments to be passed to other inherited types.

Value

A numeric vector, the posterior predictive density.

References

Teh, Yee W., et al. "Sharing clusters among related groups: Hierarchical Dirichlet processes." Advances in neural information processing systems. 2005.

See Also

DP, dPosteriorPredictive.DP, marginalLikelihood.DP

Examples

```
x <- rnorm(4)
z <- sample(1L:10L,size = 4,replace = TRUE)
obj <- DP()
ss <- sufficientStatistics(obj = obj,x=x,foreach = TRUE)
for(i in 1L:length(x)) posterior(obj = obj,ss=ss[[i]],z=z[i])
xnew <- rnorm(10)
znew <- sample(1L:10L,size = 10,replace = TRUE)
dPosteriorPredictive(obj = obj,x=xnew,z=znew)
```

dPosteriorPredictive.GaussianGaussian *Posterior predictive density function of a "GaussianGaussian" object*

Description

Generate the the density value of the posterior predictive distribution of the following structure:

$$x \sim Gaussian(mu, Sigma)$$

$$mu \sim Gaussian(m, S)$$

Where Sigma is known. Gaussian() is the Gaussian distribution. See ?dGaussian for the definition of Gaussian distribution.
The model structure and prior parameters are stored in a "GaussianGaussian" object.
Posterior predictive density is p(x|m,S,Sigma).

Usage

```
## S3 method for class 'GaussianGaussian'
dPosteriorPredictive(obj, x, LOG = TRUE, ...)
```

Arguments

obj	A "GaussianGaussian" object.
x	matrix, or the ones that can be converted to matrix, each row of x is an observation.
LOG	Return the log density if set to "TRUE".
...	Additional arguments to be passed to other inherited types.

Value

A numeric vector of the same length as nrow(x), the posterior predictive density.

References

Gelman, Andrew, et al. Bayesian data analysis. CRC press, 2013.

See Also

GaussianGaussian, dPosteriorPredictive.GaussianGaussian, marginalLikelihood.GaussianGaussian

Examples

```
obj <- GaussianGaussian(gamma=list(Sigma=matrix(c(2,1,1,2),2,2),m=c(0.2,0.5),S=diag(2)))
x <- rGaussian(100,c(0,0),Sigma = matrix(c(2,1,1,2),2,2))
dPosteriorPredictive(obj = obj,x=x,LOG = TRUE)
dPosteriorPredictive(obj = obj,x=x,LOG = FALSE)
```

`dPosteriorPredictive.GaussianInvWishart` *Posterior predictive density function of a "GaussianInvWishart" object*

Description

Generate the the density value of the posterior predictive distribution of the following structure:

$$x \sim Gaussian(mu, Sigma)$$

$$Sigma \sim InvWishart(v, S)$$

mu is known. Gaussian() is the Gaussian distribution. See `?dGaussian` and `?dInvWishart` for the definition of the distributions.
The model structure and prior parameters are stored in a "GaussianInvWishart" object.
Posterior predictive density is p(x|v,S,mu).

Usage

```
## S3 method for class 'GaussianInvWishart'
dPosteriorPredictive(obj, x, LOG = TRUE, ...)
```

Arguments

`obj`	A "GaussianInvWishart" object.
`x`	matrix, or the ones that can be converted to matrix, each row of x is an observation.
`LOG`	Return the log density if set to "TRUE".
`...`	Additional arguments to be passed to other inherited types.

Value

A numeric vector of the same length as nrow(x), the posterior predictive density.

References

Gelman, Andrew, et al. Bayesian data analysis. CRC press, 2013.

MARolA, K. V., JT KBNT, and J. M. Bibly. Multivariate analysis. AcadeInic Press, Londres, 1979.

See Also

`GaussianInvWishart`, `dPosteriorPredictive.GaussianInvWishart`, `marginalLikelihood.GaussianInvWishart`

Examples

```
obj <- GaussianInvWishart(gamma=list(mu=c(-1.5,1.5),v=3,S=diag(2)))
x <- rGaussian(100,mu = c(-1.5,1.5),Sigma = matrix(c(0.1,0.03,0.03,0.1),2,2))
xNew <- rGaussian(100,mu = c(-1.5,1.5),Sigma = matrix(c(0.1,0.03,0.03,0.1),2,2))
ss <- sufficientStatistics(obj=obj,x=x,foreach = FALSE)
## update piror with x
posterior(obj=obj,ss = ss)
## use the posterior to calculate the probability of observing each xNew
dPosteriorPredictive(obj = obj,x = xNew,LOG = TRUE)
```

---

dPosteriorPredictive.GaussianNIG

Posterior predictive density function of a "GaussianNIG" object

---

Description

Generate the the density value of the posterior predictive distribution of the following structure:

$$x \sim Gaussian(Xbeta, sigma^2)$$

$$sigma^2 \sim InvGamma(a, b)$$

$$beta \sim Gaussian(m, sigma^2 V)$$

Where X is a row vector, or a design matrix where each row is an obervation. InvGamma() is the Inverse-Gamma distribution, Gaussian() is the Gaussian distribution. See `?dInvGamma` and `dGaussian` for the definitions of these distribution.
The model structure and prior parameters are stored in a "GaussianNIG" object.
Posterior predictive density is p(x|m,V,a,b,X).

Usage

```
## S3 method for class 'GaussianNIG'
dPosteriorPredictive(obj, x, X, LOG = TRUE, ...)
```

Arguments

`obj`	A "GaussianNIG" object.
`x`	numeric, must satisfy length(x) = nrow(X).
`X`	matrix, must satisfy length(x) = nrow(X).
`LOG`	Return the log density if set to "TRUE".
`...`	Additional arguments to be passed to other inherited types.

Value

A numeric vector, the posterior predictive density.

References

Banerjee, Sudipto. "Bayesian Linear Model: Gory Details." Downloaded from http://www. biostat. umn. edu/~ph7440 (2008).

See Also

GaussianNIG, dPosteriorPredictive.GaussianNIG, marginalLikelihood.GaussianNIG

Examples

```
obj <- GaussianNIG(gamma=list(m=0,V=1,a=1,b=1))
X <- 1:20
x <- rnorm(20)+ X*0.3
## out1 and out2 it should have the same values:
out1 <- dPosteriorPredictive(obj = obj, x = x,X=X,LOG = TRUE)
out2 <- numeric(length(x))
for(i in 1:length(x))
out2[i] <- marginalLikelihood(obj,x=x[i],X=X[i],LOG = TRUE)
max(abs(out1-out2))
```

---

dPosteriorPredictive.GaussianNIW

Posterior predictive density function of a "GaussianNIW" object

---

Description

Generate the the density value of the posterior predictive distribution of the following structure:

$$mu, Sigma|m, k, v, S \sim NIW(m, k, v, S)$$

$$x|mu, Sigma \sim Gaussian(mu, Sigma)$$

Where NIW() is the Normal-Inverse-Wishart distribution, Gaussian() is the Gaussian distribution. See ?dNIW and dGaussian for the definitions of these distribution.
The model structure and prior parameters are stored in a "GaussianNIW" object.
Posterior predictive density is p(x|m,k,v,S).

Usage

```
## S3 method for class 'GaussianNIW'
dPosteriorPredictive(obj, x, LOG = TRUE, ...)
```

Arguments

obj	A "GaussianNIW" object.
x	matrix, or the ones that can be converted to matrix, each row of x is an observation.
LOG	Return the log density if set to "TRUE".
...	Additional arguments to be passed to other inherited types.

Value

A numeric vector of the same length as nrow(x), the posterior predictive density.

References

Murphy, Kevin P. "Conjugate Bayesian analysis of the Gaussian distribution." def 1.22 (2007): 16.

Gelman, Andrew, et al. "Bayesian Data Analysis Chapman & Hall." CRC Texts in Statistical Science (2004).

See Also

GaussianNIW, dPosteriorPredictive.GaussianNIW, marginalLikelihood.GaussianNIW

Examples

```
x <- rGaussian(1000,mu = c(1,1),Sigma = matrix(c(1,0.5,0.5,3),2,2))
obj <- GaussianNIW(gamma=list(m=c(0,0),k=1,v=2,S=diag(2)))
## out1 and out2 it should have the same values:
out1 <- dPosteriorPredictive(obj = obj, x = x,LOG = TRUE)
out2 <- numeric(nrow(x))
for(i in 1:nrow(x))
out2[i] <- marginalLikelihood(obj,x=x[i,,drop=FALSE],LOG = TRUE)
max(abs(out1-out2))
```

---

dPosteriorPredictive.HDP

Posterior predictive density function of a "HDP" object

Description

Generate the the density value of the posterior predictive distribution of the following structure:

$$G|gamma \sim DP(gamma, U)$$

$$pi_j|G, alpha \sim DP(alpha, G), j = 1 : J$$

$$z|pi_j \sim Categorical(pi_j)$$

$$k|z, G \sim Categorical(G), \text{ if z is a sample from the base measure G}$$

$$theta_k|psi \sim H0(psi)$$

$$x|theta_k, k \sim F(theta_k)$$

where DP(gamma,U) is a Dirichlet Process on positive integers, gamma is the "concentration parameter", U is the "base measure" of this Dirichlet process, U is an uniform distribution on all positive integers. DP(alpha,G) is a Dirichlet Process on integers with concentration parameter alpha and base measure G. The choice of F() and H0() can be described by an arbitrary "BasicBayesian" object such as "GaussianGaussian","GaussianInvWishart","GaussianNIW", "GaussianNIG", "CatDirichlet", and "CatDP". See ?BasicBayesian for definition of "BasicBayesian" objects, and see for

example ?GaussianGaussian for specific "BasicBayesian" instances. As a summary, An "HDP" object is simply a combination of a "CatHDP" object (see ?CatHDP) and an object of any "BasicBayesian" type.
In the case of HDP, z and k can only be positive integers.
The model structure and prior parameters are stored in a "HDP" object.
Posterior predictive density = p(x,z,k|gamma,alpha,psi) when x is not NULL, or p(z,k|gamma,alpha,psi) when x is NULL.

Usage

```
## S3 method for class 'HDP'
dPosteriorPredictive(obj, x = NULL, z, k, j, LOG = TRUE, ...)
```

Arguments

obj	A "HDP" object.
x	Random samples of the "BasicBayesian" object.
z	integer.
k	integer, the partition label of the parameter space where the observation x is drawn from.
j	integer, group label.
LOG	Return the log density if set to "TRUE".
...	Additional arguments to be passed to other inherited types.

Value

A numeric vector, the posterior predictive density.

References

Teh, Yee W., et al. "Sharing clusters among related groups: Hierarchical Dirichlet processes." Advances in neural information processing systems. 2005.

See Also

HDP, dPosteriorPredictive.HDP, marginalLikelihood.HDP

---

dPosteriorPredictive.HDP2 *Posterior predictive density function of a "HDP2" object*

Description

Generate the the density value of the posterior predictive distribution of the following structure:

$$G|eta \sim DP(eta, U)$$

$$G_m|gamma, G \sim DP(gamma, G), m = 1 : M$$

$$pi_{mj}|G_m, alpha \sim DP(alpha, G_m), j = 1 : J_m$$

$$z|pi_{mj} \sim Categorical(pi_{mj})$$

$$k|z, G_m \sim Categorical(G_m), \text{ if z is a sample from the base measure } G_{mj}$$

$$u|k, G \sim Categorical(G), \text{ if k is a sample from the base measure G}$$

$$theta_u|psi \sim H0(psi)$$

$$x|theta_u, u \sim F(theta_u)$$

where DP(eta,U) is a Dirichlet Process on positive integers, eta is the "concentration parameter", U is the "base measure" of this Dirichlet process, U is an uniform distribution on all positive integers. DP(gamma,G) is a Dirichlet Process on integers with concentration parameter gamma and base measure G. DP(alpha,G_m) is a Dirichlet Process on integers with concentration parameter alpha and base measure G_m. The choice of F() and H0() can be described by an arbitrary "BasicBayesian" object such as "GaussianGaussian","GaussianInvWishart","GaussianNIW", "GaussianNIG", "CatDirichlet", and "CatDP". See `?BasicBayesian` for definition of "BasicBayesian" objects, and see for example `?GaussianGaussian` for specific "BasicBayesian" instances. As a summary, An "HDP2" object is simply a combination of a "CatHDP2" object (see `?CatHDP2`) and an object of any "BasicBayesian" type.
In the case of HDP2, u, z and k can only be positive integers.
The model structure and prior parameters are stored in a "HDP2" object.
Posterior predictive density = p(u,z,k,x|eta,gamma,alpha,psi) when x is not NULL, or p(u,z,k|eta,gamma,alpha,psi) when x is NULL.

Usage

```
## S3 method for class 'HDP2'
dPosteriorPredictive(obj, x = NULL, u, k, z, m, j, LOG = TRUE, ...)
```

Arguments

`obj`	A "HDP2" object.
`x`	Random samples of the "BasicBayesian" object.
`u`	integer, the partition label of the parameter space where the observation x is drawn from.
`k`	integer.
`z`	integer.
`m`	integer, group label.
`j`	integer, subgroup label.
`LOG`	Return the log density if set to "TRUE".
`...`	Additional arguments to be passed to other inherited types.

Value

A numeric vector, the posterior predictive density.

References

Teh, Yee W., et al. "Sharing clusters among related groups: Hierarchical Dirichlet processes." Advances in neural information processing systems. 2005.

See Also

HDP2, dPosteriorPredictive.HDP2, marginalLikelihood.HDP2

---

dPosteriorPredictive.LinearGaussianGaussian

Posterior predictive density function of a "LinearGaussianGaussian" object

---

Description

Generate the the density value of the posterior predictive distribution of the following structure:

$$x \sim Gaussian(Az + b, Sigma)$$

$$z \sim Gaussian(m, S)$$

Where Sigma is known. A is a $dimxxdimz$ matrix, x is a $dimxx1$ random vector, z is a $dimzx1$ random vector, b is a $dimmx1$ vector. Gaussian() is the Gaussian distribution. See ?dGaussian for the definition of Gaussian distribution.
The model structure and prior parameters are stored in a "LinearGaussianGaussian" object.
Posterior predictive density is p(x|m,S,A,b,Sigma).

Usage

```
## S3 method for class 'LinearGaussianGaussian'
dPosteriorPredictive(obj, x, A, b = NULL, LOG = TRUE, ...)
```

Arguments

obj	A "LinearGaussianGaussian" object.
x	matrix, or the ones that can be converted to matrix. Each row of x is an observation.
A	matrix or list. when x is a $Nx1$ matrix, A must be a matrix of $Nxdimz$, dimz is the dimension of z; When x is a $Nxdimx$ matrix, where $dimx > 1$, A can be either a list or a matrix. When A is a list, $A = A_1, A_2, ...A_N$ is a list of $dimxxdimz$ matrices. If A is a single $dimxxdimz$ matrix, it will be replicated N times into a length N list

b — matrix, when x is a $Nx1$ matrix, b must also be a $Nx1$ matrix or length N vector; When x is a $Nxdimx$ matrix, where $dimx > 1$, b can be either a matrix or a vector. When b is a matrix, $b = b_1^T, ..., b_N^T$ is a $Nxdimx$ matrix, each row is a transposed vector. When b is a length $dimx$ vector, it will be transposed into a row vector and replicated N times into a $Nxdimx$ matrix. When b = NULL, it will be treated as a vector of zeros. Default NULL.

LOG — Return the log density if set to "TRUE".

... — Additional arguments to be passed to other inherited types.

Value

A numeric vector of the same length as nrow(x), the posterior predictive density.

References

Murphy, Kevin P. Machine learning: a probabilistic perspective. MIT press, 2012.

See Also

LinearGaussianGaussian, rPosteriorPredictive.LinearGaussianGaussian, marginalLikelihood.LinearGaussian

Examples

```
obj <- LinearGaussianGaussian(gamma=list(Sigma=matrix(c(2,1,1,2),2,2),
                                         m=c(0.2,0.5,0.6),S=diag(3)))
x <- rGaussian(100,mu = runif(2),Sigma = diag(2))
A <- matrix(runif(6),2,3)
b <- runif(2)
dPosteriorPredictive(obj = obj,x=x,A=A,b=b)
```

---

dT *Density function for (multivariate) t distribution*

Description

Get the density of a set of samples from a t distribution. For a random vector x, the density function is defined as:

$$Gamma((df+p)/2)/(Gamma(df/2)df^{p/2}pi^{p/2}|Sigma|^{1/2})[1+1/df(x-df)^T Sigma^{-1}(x-df)]^{-(df+p)/2}$$

Where p is the dimension of x.

Usage

```
dT(x, mu, Sigma = NULL, A = NULL, df = 1, LOG = TRUE)
```

Arguments

x	matrix, when x is a numeric vector, it will be converted to a matrix with 1 column!
mu	numeric, mean vector.
Sigma	matrix, Sigma is proportional to the covariance matrix of x, one of Sigma and A should be non-NULL.
A	matrix, the Cholesky decomposition of Sigma, an upper triangular matrix, one of Sigma and A should be non-NULL.
df	numeric, degrees of freedom.
LOG	logical, return log density of LOG=TRUE, default TRUE.

Value

A numeric vector, the probability densities.

See Also

rT

Examples

```
plot(
   dT(x=seq(-5,5,length.out = 1000),mu = 0,Sigma = 1,LOG = FALSE)
   ,type = "l"
)
```

---

dWishart *Density function of Wishart distribution*

Description

For a random matrix x, the density function of Wishart distribution is defined as:

$$(2^{(dfp)/2}Gamma_p(df/2)|rate|^{-df/2})^{-1}|x|^{(df-p-1)/2}exp(-1/2tr(xrate))$$

Where x is a pxp symmetric positive definite matrix, Gamma_p() is the multivariate Gamma function of dimension p.

Usage

```
dWishart(x, df, rate, LOG = TRUE)
```

Arguments

x	matrix, a symmetric positive-definite matrix.
df	numeric, the degree of freedom.
rate	matrix, a symmetric positive-definite matrix, the 'rate', or 'inverse-scale' parameter. The 'rate' parameter in Wishart is the 'scale' parameter in InvWishart
LOG	logical, return log density of LOG=TRUE, default TRUE.

Value

A numeric vector, the density values.

References

Wishart, John. "The generalized product moment distribution in samples from a normal multivariate population." Biometrika (1928): 32-52.

MARolA, K. V., JT KBNT, and J. M. Bibly. Multivariate analysis. AcadeInic Press, Londres, 1979.

Examples

```
##generate a symmetric positive-definite matrix
x <- crossprod(matrix(rnorm(15),5,3))
rate <- crossprod(matrix(rnorm(15),5,3)) #the prior inverse-scale of x
dWishart(x,df = 5,rate = rate,LOG = TRUE)
dWishart(x,df = 5,rate = rate,LOG = FALSE)
```

---

farmadsData *farm ads data*

Description

A subset of farm ads data from https://archive.ics.uci.edu/ml/datasets/Farm+Ads

Usage

```
data(farmadsData)
```

Format

A list of two elements:

word : character, the words

document : integer, document id of each word

Source

Farm-Ads

GaussianGaussian *Create objects of type "GaussianGaussian".*

Description

Create an object of type "GaussianGaussian", which represents the Gaussian and Gaussian conjugate structure:

$$x \sim Gaussian(mu, Sigma)$$

$$mu \sim Gaussian(m, S)$$

Where Sigma is known. Gaussian() is the Gaussian distribution. See ?dGaussian for the definition of Gaussian distribution.
The created object will be used as a place for recording and accumulating information in the related inference/sampling functions such as posterior(), posteriorDiscard(), MAP(), marginalLikelihood(), dPosteriorPredictive(), rPosteriorPredictive() and so on.

Usage

```
GaussianGaussian(
  objCopy = NULL,
  ENV = parent.frame(),
  gamma = list(Sigma = 1, m = 0, S = 1)
)
```

Arguments

objCopy	an object of type "GaussianGaussian". If "objCopy" is not NULL, the function create a new "GaussianGaussian" object by copying the content from objCopy, otherwise this new object will be created by using "ENV" and "gamma". Default NULL.
ENV	environment, specify where the object will be created.
gamma	list, a named list of parameters, gamma=list(Sigma,m,S). Where gamma$Sigma is the known covariance matrix of x, gamma$m and gamma$S are the prior mean and covariance matrix of mu.

Value

An object of class "GaussianGaussian".

References

Gelman, Andrew, et al. Bayesian data analysis. CRC press, 2013.

See Also

posterior.GaussianGaussian,posteriorDiscard.GaussianGaussian,MAP.GaussianGaussian,MPE.GaussianGaussia

Examples

```
obj <- GaussianGaussian(gamma=list(Sigma=matrix(c(2,1,1,2),2,2),m=c(0.2,0.5),S=diag(2)))
obj #print the content
```

`GaussianInvWishart` *Create objects of type "GaussianInvWishart".*

Description

Create an object of type "GaussianInvWishart", which represents the Gaussian and Inverse-Wishart conjugate structure:

$$x \sim Gaussian(mu, Sigma)$$

$$Sigma \sim InvWishart(v, S)$$

mu is known. Gaussian() is the Gaussian distribution. See `?dGaussian` and `?dInvWishart` for the definition of the distributions.
The created object will be used as a place for recording and accumulating information in the related inference/sampling functions such as posterior(), posteriorDiscard(), MAP(), marginalLikelihood(), dPosteriorPredictive(), rPosteriorPredictive() and so on.

Usage

```
GaussianInvWishart(
  objCopy = NULL,
  ENV = parent.frame(),
  gamma = list(mu = 0, v = 3, S = 1)
)
```

Arguments

`objCopy`	an object of type "GaussianInvWishart". If "objCopy" is not NULL, the function create a new "GaussianInvWishart" object by copying the content from objCopy, otherwise this new object will be created by using "ENV" and "gamma". Default NULL.
`ENV`	environment, specify where the object will be created.
`gamma`	list, a named list of parameters, gamma=list(mu,v,S). Where gamma$mu is the known mean vector of x, gamma$v and gamma$S are the prior degree of freedom and scale of Sigma.

Value

An object of class "GaussianInvWishart".

References

Gelman, Andrew, et al. Bayesian data analysis. CRC press, 2013.

See Also

posterior.GaussianInvWishart,posteriorDiscard.GaussianInvWishart,MAP.GaussianInvWishart,MPE.GaussianI
...

Examples

```
obj <- GaussianInvWishart(gamma=list(mu=c(-1.5,1.5),v=3,S=diag(2)))
obj #print the content
```

GaussianNIG *Create objects of type "GaussianNIG".*

Description

Create an object of type "GaussianNIG", which represents the Gaussian and Normal-Inverse-Gamma (Gaussian-NIG) conjugate structure:

$$x \sim Gaussian(Xbeta, sigma^2)$$

$$sigma^2 \sim InvGamma(a, b)$$

$$beta \sim Gaussian(m, sigma^2 V)$$

Where X is a row vector, or a design matrix where each row is an obervation. InvGamma() is the Inverse-Gamma distribution, Gaussian() is the Gaussian distribution. See ?dInvGamma and dGaussian for the definitions of these distribution.
This object will be used as a place for recording and accumulating information in the related inference/sampling functions such as posterior(), posteriorDiscard(), MAP(), marginalLikelihood(), dPosteriorPredictive(), rPosteriorPredictive() and so on.

Usage

```
GaussianNIG(
  objCopy = NULL,
  ENV = parent.frame(),
  gamma = list(m = 0, V = 1, a = 1, b = 1)
)
```

Arguments

objCopy	An object of type "GaussianNIG". If "objCopy" is not NULL, the function create a new "GaussianNIG" object by copying the content from objCopy, otherwise this new object will be created by using "ENV" and "gamma". Default NULL.
ENV	environment, specify in which environment the object will be created
gamma	list, a named list of NIG parameters, gamma=list(m,V,a,b). Where gamma$m is a numeric "location" parameter; gamma$V is a symmetric positive definite matrix representing the "scale" parameters; gamma$a and gamma$b are the "shape" and "scale" parameter of the Inverse Gamma distribution.

Value

An object of class "GaussianNIG".

References

Banerjee, Sudipto. "Bayesian Linear Model: Gory Details." Downloaded from http://www. biostat. umn. edu/~ph7440 (2008).

See Also

posterior.GaussianNIG,posteriorDiscard.GaussianNIG,MAP.GaussianNIG,MPE.GaussianNIG,marginalLikelihood
rPosteriorPredictive.GaussianNIG ...

Examples

```
X <- 1:20                                #generate some linear data
x <- rnorm(20)+ X*0.3                    #generate some linear data
obj <- GaussianNIG(gamma=list(m=0,V=1,a=1,b=0)) #create a GaussianNIG object
ss <- sufficientStatistics(obj = obj,X=X,x=x)   #the sufficient statistics of X and x
posterior(obj = obj,ss = ss)                     #add the infomation to the posterior
MAP(obj)                                        #get the MAP estimate of beta and sigma^2
## print the whole content, "invV" and "mVm" in the output are temporary variables.
obj
```

GaussianNIW *Create objects of type "GaussianNIW".*

Description

Create an object of type "GaussianNIW", which represents the Gaussian-Normal-Inverse-Wishart (Gaussian-NIW) conjugate structure:

$$mu, Sigma|m, k, v, S \sim NIW(m, k, v, S)$$

$$x|mu, Sigma \sim Gaussian(mu, Sigma)$$

Where NIW() is the Normal-Inverse-Wishart distribution, Gaussian() is the Gaussian distribution. See ?dNIW and dGaussian for the definitions of these distribution.
This object will be used as a place for recording and accumulating information in the related inference/sampling functions such as posterior(), posteriorDiscard(), MAP(), marginalLikelihood(), dPosteriorPredictive(), rPosteriorPredictive() and so on.

Usage

```
GaussianNIW(
  objCopy = NULL,
  ENV = parent.frame(),
  gamma = list(m = 0, k = 1, v = 2, S = 1)
)
```

Arguments

objCopy	An object of type "GaussianNIW". If "objCopy" is not NULL, the function create a new "GaussianNIW" object by copying the content from objCopy, otherwise this new object will be created by using "ENV" and "gamma". Default NULL.
ENV	environment, specify in which environment the object will be created.
gamma	list, a named list of NIW parameters, gamma=list(m,k,v,S). Where gamma$m is a numeric "location" parameter; gamma$S is a symmetric positive definite matrix representing the "scale" parameters; gamma$k and gamma$v are numeric values.

Value

An object of class "GaussianNIW".

References

Murphy, Kevin P. "Conjugate Bayesian analysis of the Gaussian distribution." def 1.22 (2007): 16.

Gelman, Andrew, et al. "Bayesian Data Analysis Chapman & Hall." CRC Texts in Statistical Science (2004).

See Also

posterior.GaussianNIW,posteriorDiscard.GaussianNIW,MAP.GaussianNIW,MPE.GaussianNIW,marginalLikelihood
...

Examples

```
obj <- GaussianNIW(gamma=list(m=c(0,1),k=0.0001,v=2,S=diag(2)))
obj #print the content
```

HDP *Create objects of type "HDP".*

Description

Create an object of type "HDP", which represents the Hierarchical-Dirichlet-Process conjugate structure:

$$G|gamma \sim DP(gamma, U)$$

$$pi_j|G, alpha \sim DP(alpha, G), j = 1 : J$$

$$z|pi_j \sim Categorical(pi_j)$$

$k|z, G \sim Categorical(G)$, if z is a sample from the base measure G

$$theta_k|psi \sim H0(psi)$$

$$x|theta_k, k \sim F(theta_k)$$

where DP(gamma,U) is a Dirichlet Process on positive integers, gamma is the "concentration parameter", U is the "base measure" of this Dirichlet process, U is an uniform distribution on all positive integers. DP(alpha,G) is a Dirichlet Process on integers with concentration parameter alpha and base measure G. The choice of F() and H0() can be described by an arbitrary "BasicBayesian" object such as "GaussianGaussian","GaussianInvWishart","GaussianNIW", "GaussianNIG", "CatDirichlet", and "CatDP". See `?BasicBayesian` for definition of "BasicBayesian" objects, and see for example `?GaussianGaussian` for specific "BasicBayesian" instances. As a summary, An "HDP" object is simply a combination of a "CatHDP" object (see `?CatHDP`) and an object of any "BasicBayesian" type.
In the case of HDP, z and k can only be positive integers.
This object will be used as a place for recording and accumulating information in the related inference/sampling functions such as posterior(), posteriorDiscard(), dPosteriorPredictive(), rPosteriorPredictive() and so on.

Usage

```
HDP(
  objCopy = NULL,
  ENV = parent.frame(),
 gamma = list(gamma = 1, j = 2L, alpha = 1, H0aF = "GaussianNIW", parH0 = list(m = 0, k
    = 1, v = 2, S = 1))
)
```

Arguments

`objCopy`	an object of type "HDP". If "objCopy" is not NULL, the function create a new "HDP" object by copying the content from objCopy, otherwise this new object will be created by using "ENV" and "gamma". Default NULL.
`ENV`	environment, specify where the object will be created.
`gamma`	list, a named list of parameters, gamma=list(gamma,alpha,j,H0aF,parH0). Where gamma$gamma is a numeric value specifying the concentration parameter of DP(gamma,U), gamma$alpha is a numeric value specifying the concentration parameter of DP(alpha,G), gamma$j is the number of groups J. gamma$H0aF is the name of the "BasicBayesian" object which specifies the structure of H0() and F(). gamma$parH0 is the parameters passed to the selected H0aF. For example, if gamma$H0aF="GaussianNIW", then gamma$parH0 should be a named list of NIW parameters: gamma$parH0=list(m,k,v,S), where gamma$parH0$m is a numeric "location" parameter; gamma$parH0$S is a symmetric positive definite matrix representing the "scale" parameters; gamma$parH0$k and gamma$parH0$v are numeric values.

Value

An object of class "HDP".

References

Teh, Yee W., et al. "Sharing clusters among related groups: Hierarchical Dirichlet processes." Advances in neural information processing systems. 2005.

See Also

BasicBayesian,GaussianNIW,GaussianNIG,CatDirichlet,CatHDP,posterior.HDP,posteriorDiscard.HDP,marginalLi
...

Examples

```
## This is an example of Gibbs sampling on an hierarchical mixture model,
## using Hierarchical Dirichlet Process.

## load some hierarchical mixture data, check ?mmhData for details.
data(mmhData)
x <- mmhData$x
js <- mmhData$groupLabel


## Step1: initialize---------------------------------------------------
z <- rep(1L,nrow(x))
k <- rep(1L,nrow(x))
## create a HDP object to track all the changes
obj <- HDP(gamma = list(gamma=1,j=max(js),alpha=1,
           H0aF="GaussianNIW",
           parH0=list(m=c(0,0),k=0.001,v=2,S=diag(2)*0.01)))
ss <- sufficientStatistics(obj$H,x=x,foreach = TRUE) #sufficient statistics
N <- length(ss)
for(i in 1L:N){# initialize k and z
    tmp <- rPosteriorPredictive(obj = obj,n=1,x=x[i,,drop=FALSE],j=js[i])
    z[i] <- tmp["z"]
    k[i] <- tmp["k"]
    posterior.HDP(obj = obj,ss = ss[[i]],ss1 = k[i],ss2 = z[i],j = js[i])
}

## Step2: main Gibbs loop----------------------------------------------
maxit <- 20                             #iterative for maxit times
it <- 0                                 #iteration tracker
pb <- txtProgressBar(min = 0,max = maxit,style = 3)
while(TRUE){
    for(i in 1L:N){
        ## remove the sample from the posterior info
        posteriorDiscard(obj = obj,ss = ss[[i]],ss1=k[i],ss2=z[i],j=js[i])
        ## resample a new partition
        tmp <- rPosteriorPredictive(obj = obj,n=1,x=x[i,,drop=FALSE],j=js[i])
        z[i] <- tmp["z"]
        k[i] <- tmp["k"]
        posterior(obj = obj,ss = ss[[i]], ss1=k[i],ss2 = z[i],j=js[i])
    }
    plot(x=x[,1],y=x[,2],col=k)         #to visualize the group label dynamics
    it <- it+1
    setTxtProgressBar(pb,it)
    if(it>=maxit){cat("\n");break}
}
```

HDP2 *Create objects of type "HDP2".*

Description

Create an object of type "HDP2", which represents the Hierarchical-Dirichlet-Process with two Dirichlet-Process hierarchies:

$$G|eta \sim DP(eta, U)$$

$$G_m|gamma, G \sim DP(gamma, G), m = 1 : M$$

$$pi_{mj}|G_m, alpha \sim DP(alpha, G_m), j = 1 : J_m$$

$$z|pi_{mj} \sim Categorical(pi_{mj})$$

$$k|z, G_m \sim Categorical(G_m),\ \text{if z is a sample from the base measure } G_m$$

$$u|k, G \sim Categorical(G),\ \text{if k is a sample from the base measure G}$$

$$theta_u|psi \sim H0(psi)$$

$$x|theta_u, u \sim F(theta_u)$$

where DP(eta,U) is a Dirichlet Process on positive integers, eta is the "concentration parameter", U is the "base measure" of this Dirichlet process, U is an uniform distribution on all positive integers. DP(gamma,G) is a Dirichlet Process on integers with concentration parameter gamma and base measure G. DP(alpha,G_m) is a Dirichlet Process on integers with concentration parameter alpha and base measure G_m. The choice of F() and H0() can be described by an arbitrary "BasicBayesian" object such as "GaussianGaussian","GaussianInvWishart","GaussianNIW", "GaussianNIG", "CatDirichlet", and "CatDP". See `?BasicBayesian` for definition of "BasicBayesian" objects, and see for example `?GaussianGaussian` for specific "BasicBayesian" instances. As a summary, An "HDP2" object is simply a combination of a "CatHDP2" object (see `?CatHDP2`) and an object of any "BasicBayesian" type.
In the case of HDP2, u, z and k can only be positive integers.
This object will be used as a place for recording and accumulating information in the related inference/sampling functions such as posterior(), posteriorDiscard(), MAP() and so on.

Usage

```
HDP2(
  objCopy = NULL,
  ENV = parent.frame(),
  gamma = list(eta = 1, gamma = 1, alpha = 1, m = 3L, j = c(2, 3, 4), H0aF =
    "GaussianNIW", parH0 = list(m = 0, k = 1, v = 2, S = 1))
)
```

Arguments

objCopy	an object of type "HDP2". If "objCopy" is not NULL, the function create a new "HDP2" object by copying the content from objCopy, otherwise this new object will be created by using "ENV" and "gamma". Default NULL.
ENV	environment, specify where the object will be created.
gamma	list, a named list of parameters, gamma=list(eta,gamma,alpha,m,j,H0aF,parH0), where gamma$eta is a numeric value specifying the concentration parameter of DP(eta,U), gamma$gamma is a numeric value specifying the concentration parameter of DP(gamma,G), gamma$alpha is a numeric value specifying the concentration parameter of DP(alpha,G_m), gamma$m is the number of groups M, gamma$j is the number of subgroups in each group, must satisfy length(gamma$j)=gamma$m. gamma$H0aF is the name of the "BasicBayesian" object which specifies the structure of H0() and F(). gamma$parH0 is the parameters passed to the selected H0aF. For example, if gamma$H0aF="GaussianNIW", then gamma$parH0 should be a named list of NIW parameters: gamma$parH0=list(m,k,v,S), where gamma$parH0$m is a numeric "location" parameter; gamma$parH0$S is a symmetric positive definite matrix representing the "scale" parameters; gamma$parH0$k and gamma$parH0$v are numeric values.

Value

An object of class "HDP2".

References

Teh, Yee W., et al. "Sharing clusters among related groups: Hierarchical Dirichlet processes." Advances in neural information processing systems. 2005.

See Also

BasicBayesian,GaussianNIW,GaussianNIG,CatDirichlet,CatHDP2,posterior.HDP2,posteriorDiscard.HDP2,margina
...

Examples

```
## This is an example of Gibbs sampling on a hierarchical mixture model, using HDP2.

## load some hierarchical mixture data, check ?mmhhData for details.
data(mmhhData)
x <- mmhhData$x
ms <- mmhhData$groupLabel
js <- mmhhData$subGroupLabel

## Step1: initialize-----------------------------------------------------
maxit <- 50                          #iterative for maxit times
z <- rep(1L,nrow(x))
k <- rep(1L,nrow(x))
```

```
u <- rep(1L,nrow(x))
obj <- HDP2(gamma = list(eta=1,gamma=1,alpha=1,m=2L,j=c(10L,20L),
            H0aF="GaussianNIW",
            parH0=list(m=c(0,0),k=0.001,v=2,S=diag(2)*0.001)))
ss <- sufficientStatistics(obj$H,x=x,foreach = TRUE) #sufficient statistics
N <- length(ss)
for(i in 1L:N){                          #initialize z k and u
   tmp <- rPosteriorPredictive(obj = obj,n=1,x=x[i,,drop=FALSE],m=ms[i],j=js[i])
   z[i] <- tmp["z"]
   k[i] <- tmp["k"]
   u[i] <- tmp["u"]
   posterior(obj = obj,ss = ss[[i]],ss1 = u[i],ss2 = k[i],ss3 = z[i],m=ms[i],j = js[i])
}

## Step2: main Gibbs loop-----------------------------------------------
it <- 0                                  #iteration tracker
pb <- txtProgressBar(min = 0,max = maxit,style = 3)
while(TRUE){
   for(i in 1L:N){
       ##remove the sample from the posterior info
     posteriorDiscard(obj = obj,ss = ss[[i]],ss1=u[i],ss2=k[i],ss3 = z[i],m=ms[i],j=js[i])
       ##resample a new partition
       tmp <- rPosteriorPredictive(obj = obj,n=1L,x=x[i,,drop=FALSE],m=ms[i],j=js[i])
       z[i] <- tmp["z"]
       k[i] <- tmp["k"]
       u[i] <- tmp["u"]
       posterior(obj = obj,ss = ss[[i]], ss1=u[i],ss2 = k[i],ss3 = z[i],m=ms[i],j=js[i])
   }
   plot(x=x[,1],y=x[,2],col=u)
   it <- it+1
   setTxtProgressBar(pb,it)
   if(it>=maxit){cat("\n");break}
}
```

---

hlrData *Samples from a hierarchical linear model*

Description

The data was part of the 2002 Educational Longitudinal Study (ELS), a survey of students from a large sample of schools across the United States. This dataset includes a population of schools as well as a population of students within each school.

Usage

```
data(lrData)
```

Format

A list of three elements:

mathScore : numeric, the mathScore of each student

socioeconomicStatus : numeric, the socioeconomic status score of each student

schoolID : integer, the school ID of each student

References

Hoff, Peter D. A first course in Bayesian statistical methods. Vol. 580. New York: Springer, 2009.

`hmmData` *Samples from a hidden Markov model*

Description

Random samples generated from a Hidden Markov Model (HMM) with 3 hidden states. The initial distribution is c(0.2,0.6,0.2), the transition matrix is matrix(c(0.9, 0.04, 0.06, 0.06, 0.9, 0.07, 0.04, 0.06, 0.87),3,3).

Usage

```
data(hmmData)
```

Format

A list of four elements:

x : matrix, two dimensional Gaussian observations. The observations are split into 'Nsegs' segment, see 'Nsegs' and 'breaks' below.

z : integer vector, the real hidden states.

Nsegs : integer, the number of segments.

breaks : integer vector, the starting and ending locations of the segments. The ith segment start at breaks[i]+1, ends at breaks[i+1]

inferenceJointGaussian

Inference in joint Gaussian distribution

Description

For the model structure

$$x1, x2|mu, Sigma \sim Gaussian(mu, Sigma)$$

$$x1|x2, mu, Sigma \sim Gaussian(mu12, Sigma12)$$

Usage

```
inferenceJointGaussian(x2, mu, Sigma = NULL, Precision = NULL)
```

Arguments

x2	numeric, an sample of X2, satisfying length(x2)<D, D is the dimension of the joint distribution.
mu	numeric, length D mean vector. mu=c(mu_X1,mu_X2)/.
Sigma	DxD covariance matrix. At least one of Sigma and Precision should be non-NULL.
Precision	DxD precision matrix, satisfying Precision = inverse(Sigma). At least one of Sigma and Precision should be non-NULL.

Value

A named list containing the conditional mean and covariance matrix.

Examples

```
tmp <- matrix(runif(100),20,5)
S <- crossprod(tmp)                #some synthetic covariance matrix
P <- solve(S)
m <- runif(5)
x2 <- runif(3)
inferenceJointGaussian(x2 = x2,mu = m,Precision = P)
```

LinearGaussianGaussian

Create objects of type "LinearGaussianGaussian".

Description

Create an object of type "LinearGaussianGaussian", which represents the Linear Gaussian and Gaussian conjugate structure:

$$x \sim Gaussian(Az + b, Sigma)$$

$$z \sim Gaussian(m, S)$$

Where Sigma is known. A is a mxn matrix, x is a $mx1$ random vector, z is a $nx1$ random vector, b is a $nx1$ vector. Gaussian() is the Gaussian distribution. See `?dGaussian` for the definition of Gaussian distribution.
The created object will be used as a place for recording and accumulating information in the related inference/sampling functions such as posterior(), posteriorDiscard(), MAP(), marginalLikelihood(), dPosteriorPredictive(), rPosteriorPredictive() and so on.

Usage

```
LinearGaussianGaussian(
  objCopy = NULL,
  ENV = parent.frame(),
  gamma = list(Sigma = 1, m = 0, S = 1)
)
```

Arguments

`objCopy`	an object of type "LinearGaussianGaussian". If "objCopy" is not NULL, the function create a new "LinearGaussianGaussian" object by copying the content from objCopy, otherwise this new object will be created by using "ENV" and "gamma". Default NULL.
`ENV`	environment, specify where the object will be created.
`gamma`	list, a named list of parameters, gamma=list(Sigma,m,S). Where gamma$Sigma is the known covariance matrix of x, gamma$m and gamma$S are the prior mean and covariance matrix of z.

Value

An object of class "LinearGaussianGaussian".

References

Murphy, Kevin P. Machine learning: a probabilistic perspective. MIT press, 2012.

See Also

posterior.LinearGaussianGaussian,posteriorDiscard.LinearGaussianGaussian,MAP.LinearGaussianGaussian,M

Examples

```
obj <- LinearGaussianGaussian(gamma=list(Sigma=matrix(c(2,1,1,2),2,2),
                              m=c(0.2,0.5,0.3),S=diag(3)))
obj #print the content
```

---

logsumexp *log sum exp*

Description

For each row l of a matrix x, calculate log(sum(exp(l))).

Usage

```
logsumexp(x)
```

Arguments

x	matrix, the values in x a all logged. If x is a numeric vector, it will be converted to a matrix with 1 row.

Value

numeric, the logsumexp of each row of x.

Examples

```
## Normalize the rows of x to make them sum up to 1
x <- matrix(runif(6,-1000,-20),nrow=2)
x <- x-logsumexp(x)
x <- exp(x)
rowSums(x)
```

lrData — *Samples from a simple linear model*

Description

A list of two elements, a matrix "X" and a numeric vector "x". They came from a linear model: x = X*0.3 + epsilon, where epsilon is Gaussian distributed with mean 0 and variance 25.

Usage

```
data(lrData)
```

Format

A list of two elements:

x : numeric, linear samples

X : matrix, the "locations" of the linear samples

MAP — *Get the Maximum A Posteriori(MAP) estimate of a "BayesianBrick" object*

Description

This is a generic function that will generate the MAP estimate of a given "BayesianBrick" object. For the model structure:

$$theta|gamma \sim H(gamma)$$
$$x|theta \sim F(theta)$$

MAP estimate of theta is theta_MAP = argmax_theta p(theta|gamma,x). For a given Bayesian bricks object obj, the MAP estimate will be:

class(obj)="LinearGaussianGaussian": Where

$$x \sim Gaussian(Az + b, Sigma)$$
$$z \sim Gaussian(m, S)$$

`MAP()` will return the MAP estimate of z. See `?MAP.LinearGaussianGaussian` for details.

class(obj)="GaussianGaussian": Where

$$x \sim Gaussian(mu, Sigma)$$
$$mu \sim Gaussian(m, S)$$

Sigma is known. `MAP()` will return the MAP estimate of mu. See `?MAP.GaussianGaussian` for details.

class(obj)="GaussianInvWishart": Where

$$x \sim Gaussian(mu, Sigma)$$

$$Sigma \sim InvWishart(v, S)$$

mu is known. `MAP()` will return the MAP estimate of Sigma. See `?MAP.GaussianInvWishart` for details.

class(obj)="GaussianNIW": Where

$$x \sim Gaussian(mu, Sigma)$$

$$Sigma \sim InvWishart(v, S)$$

$$mu \sim Gaussian(m, Sigma/k)$$

`MAP()` will return the MAP estimate of mu and Sigma. See `?MAP.GaussianNIW` for details.

class(obj)="GaussianNIG": Where

$$x \sim Gaussian(Xbeta, sigma^2)$$

$$sigma^2 \sim InvGamma(a, b)$$

$$beta \sim Gaussian(m, sigma^2 V)$$

X is a row vector, or a design matrix where each row is an obervation. `MAP()` will return the MAP estimate of beta and sigma^2. See `?MAP.GaussianNIG` for details.

class(obj)="CatDirichlet": Where

$$x \sim Categorical(pi)$$

$$pi \sim Dirichlet(alpha)$$

`MAP()` will return the MAP estimate of pi. See `?MAP.CatDirichlet` for details.

class(obj)="CatDP": Where

$$x \sim Categorical(pi)$$

$$pi \sim DirichletProcess(alpha)$$

`MAP()` will return the MAP estimate of pi. See `?MAP.CatDP` for details.

Usage

```
MAP(obj, ...)
```

Arguments

`obj`	A "BayesianBrick" object used to select a method.
`...`	further arguments passed to or from other methods.

Value

A list of the MAP estimates

See Also

MAP.LinearGaussianGaussian for Linear Gaussian and Gaussian conjugate structure, MAP.GaussianGaussian for Gaussian-Gaussian conjugate structure, MAP.GaussianInvWishart for Gaussian-Inverse-Wishart conjugate structure, MAP.GaussianNIW for Gaussian-NIW conjugate structure, MAP.GaussianNIG for Gaussian-NIG conjugate structure, MAP.CatDirichlet for Categorical-Dirichlet conjugate structure, MAP.CatDP for Categorical-DP conjugate structure ...

MAP.CatDirichlet *MAP estimate of a "CatDirichlet" object*

Description

Generate the MAP estimate of "pi" in following Categorical-Dirichlet structure:

$$pi|alpha \sim Dir(alpha)$$

$$x|pi \sim Categorical(pi)$$

Where Dir() is the Dirichlet distribution, Categorical() is the Categorical distribution. See ?dDir and dCategorical for the definitions of these distribution.
The model structure and prior parameters are stored in a "CatDirichlet" object.
MAP is pi_MAP = argmax p(pi|alpha,x).

Usage

```
## S3 method for class 'CatDirichlet'
MAP(obj, ...)
```

Arguments

obj	A "CatDirichlet" object.
...	Additional arguments to be passed to other inherited types.

Value

A numeric vector, the MAP estimate of "pi".

References

Murphy, Kevin P. Machine learning: a probabilistic perspective. MIT press, 2012.

See Also

CatDirichlet

Examples

```
obj <- CatDirichlet(gamma=list(alpha=rep(1,26),uniqueLabels = letters))
x <- sample(letters,size = 20,replace = TRUE)
w <- runif(20)
posterior(obj=obj,ss=x,w=w)
MAP(obj)
```

`MAP.CatDP` *Maximum A Posteriori(MAP) estimate of a "CatDP" object*

Description

Generate the MAP estimate of "pi" in following model structure:

$$pi|alpha \sim DP(alpha, U)$$

$$x|pi \sim Categorical(pi)$$

where DP(alpha,U) is a Dirichlet Process on positive integers, alpha is the "concentration parameter" of the Dirichlet Process, U is the "base measure" of this Dirichlet process, it is an uniform distribution on all positive integers.Categorical() is the Categorical distribution. See `dCategorical` for the definition of the Categorical distribution.
In the case of CatDP, x can only be positive integers.
The model structure and prior parameters are stored in a "CatDP" object.
The MAP estimate of pi is pi_MAP = argmax_pi p(pi|alpha,x).

Usage

```
## S3 method for class 'CatDP'
MAP(obj, ...)
```

Arguments

`obj`	A "CatDP" object.
`...`	Additional arguments to be passed to other inherited types.

Value

numeric.

References

Teh, Yee W., et al. "Sharing clusters among related groups: Hierarchical Dirichlet processes." Advances in neural information processing systems. 2005.

See Also

`CatDP`

Examples

```
x <- sample(1L:10L,size = 40,replace = TRUE)
obj <- CatDP()
posterior(obj = obj,ss = x)
MAP(obj)
```

`MAP.GaussianGaussian` *Maximum A Posteriori (MAP) estimate of a "GaussianGaussian" object*

Description

Generate the MAP estimate of mu in following model structure:

$$x \sim Gaussian(mu, Sigma)$$

$$mu \sim Gaussian(m, S)$$

Where Sigma is known. Gaussian() is the Gaussian distribution. See `?dGaussian` for the definition of Gaussian distribution.
The model structure and prior parameters are stored in a "GaussianGaussian" object.
The MAP estimates are:

- (mu_MAP) = argmax p(mu|m,S,x,Sigma)

Usage

```
## S3 method for class 'GaussianGaussian'
MAP(obj, ...)
```

Arguments

`obj`	A "GaussianGaussian" object.
`...`	Additional arguments to be passed to other inherited types.

Value

numeric vector, the MAP estimate of "mu".

References

Gelman, Andrew, et al. Bayesian data analysis. CRC press, 2013.

See Also

`GaussianGaussian`

Examples

```
obj <- GaussianGaussian(gamma=list(Sigma=matrix(c(2,1,1,2),2,2),m=c(0.2,0.5),S=diag(2)))
x <- rGaussian(100,c(0,0),Sigma = matrix(c(2,1,1,2),2,2))
ss <- sufficientStatistics(obj=obj,x=x,foreach = FALSE)
## update prior into posterior
posterior(obj = obj,ss = ss)
## get the MAP estimate of mu
MAP(obj)
```

`MAP.GaussianInvWishart` *Maximum A Posteriori (MAP) estimate of a "GaussianInvWishart" object*

Description

Generate the MAP estimate of Sigma in following model structure:

$$x \sim Gaussian(mu, Sigma)$$

$$Sigma \sim InvWishart(v, S)$$

mu is known. Gaussian() is the Gaussian distribution. See `?dGaussian` and `?dInvWishart` for the definition of the distributions.
The model structure and prior parameters are stored in a "GaussianInvWishart" object.
The MAP estimates are:

- (Sigma_MAP) = argmax p(Sigma|v,S,x,mu)

Usage

```
## S3 method for class 'GaussianInvWishart'
MAP(obj, ...)
```

Arguments

`obj`	A "GaussianInvWishart" object.
`...`	Additional arguments to be passed to other inherited types.

Value

matrix, the MAP estimate of "Sigma".

References

Gelman, Andrew, et al. Bayesian data analysis. CRC press, 2013.

MARolA, K. V., JT KBNT, and J. M. Bibly. Multivariate analysis. AcadeInic Press, Londres, 1979.

See Also

GaussianInvWishart

Examples

```
obj <- GaussianInvWishart(gamma=list(mu=c(-1.5,1.5),v=3,S=diag(2)))
x <- rGaussian(100,mu = c(-1.5,1.5),Sigma = matrix(c(0.1,0.03,0.03,0.1),2,2))
ss <- sufficientStatistics(obj=obj,x=x,foreach = FALSE)
posterior(obj=obj,ss = ss)
MAP(obj)
```

MAP.GaussianNIG *Maximum A Posteriori (MAP) estimate of a "GaussianNIG" object*

Description

Generate the MAP estimate of (beta,sigma^2) in following Gaussian-NIG structure:

$$x \sim Gaussian(Xbeta, sigma^2)$$

$$sigma^2 \sim InvGamma(a, b)$$

$$beta \sim Gaussian(m, sigma^2 V)$$

Where X is a row vector, or a design matrix where each row is an obervation. InvGamma() is the Inverse-Gamma distribution, Gaussian() is the Gaussian distribution. See ?dInvGamma and dGaussian for the definitions of these distribution.
The model structure and prior parameters are stored in a "GaussianNIG" object.
The MAP estimates are:

- (beta,sigma^2)_MAP = argmax p(beta,sigma^2|m,V,a,b,x,X)

Usage

```
## S3 method for class 'GaussianNIG'
MAP(obj, ...)
```

Arguments

obj	A "GaussianNIG" object.
...	Additional arguments to be passed to other inherited types.

Value

A named list, the MAP estimate of beta and sigma^2.

References

Banerjee, Sudipto. "Bayesian Linear Model: Gory Details." Downloaded from http://www. biostat. umn. edu/~ph7440 (2008).

See Also

GaussianNIG

Examples

```
obj <- GaussianNIG(gamma=list(m=0,V=1,a=1,b=1))
X <- 1:20
x <- rnorm(20)+ X*0.3
ss <- sufficientStatistics(obj = obj,X=X,x=x)
posterior(obj = obj,ss = ss)
MAP(obj)
```

MAP.GaussianNIW *Maximum A Posteriori (MAP) estimate of a "GaussianNIW" object*

Description

Generate the MAP estimate of (mu,Sigma) in following Gaussian-NIW structure:

$$mu, Sigma|m, k, v, S \sim NIW(m, k, v, S)$$

$$x|mu, Sigma \sim Gaussian(mu, Sigma)$$

Where NIW() is the Normal-Inverse-Wishart distribution, Gaussian() is the Gaussian distribution. See ?dNIW and dGaussian for the definitions of these distribution.
The model structure and prior parameters are stored in a "GaussianNIW" object.
The MAP estimates are:

- (mu_MAP,Sigma_MAP) = argmax p(mu,Sigma|m,k,v,S,x)

Usage

```
## S3 method for class 'GaussianNIW'
MAP(obj, ...)
```

Arguments

obj	A "GaussianNIW" object.
...	Additional arguments to be passed to other inherited types.

Value

A named list, the MAP estimate of mu and Sigma.

References

Murphy, Kevin P. "Conjugate Bayesian analysis of the Gaussian distribution." def 1.22 (2007): 16.

Gelman, Andrew, et al. "Bayesian Data Analysis Chapman & Hall." CRC Texts in Statistical Science (2004).

See Also

GaussianNIW

Examples

```
## update the piror with new observations then calculate the MAP estimate
x <- rGaussian(1000,mu = c(1,1),Sigma = matrix(c(1,0.5,0.5,3),2,2))
w <- runif(1000)
obj <- GaussianNIW(gamma=list(m=c(0,0),k=1,v=2,S=diag(2)))
ss <- sufficientStatistics_Weighted(obj = obj,x=x,w=w,foreach = TRUE)
for(i in 1L:length(ss)) posterior(obj = obj,ss=ss[[i]])
MAP(obj)
```

MAP.LinearGaussianGaussian

Maximum A Posteriori (MAP) estimate of a "LinearGaussianGaussian" object

---

Description

Generate the MAP estimate of mu in following model structure:

$$x \sim Gaussian(Az + b, Sigma)$$

$$z \sim Gaussian(m, S)$$

Where Sigma is known. A is a $dimxxdimz$ matrix, x is a $dimxx1$ random vector, z is a $dimzx1$ random vector, b is a $dimmx1$ vector. Gaussian() is the Gaussian distribution. See ?dGaussian for the definition of Gaussian distribution.
The model structure and prior parameters are stored in a "LinearGaussianGaussian" object.
The MAP estimates are:

- $z_MAP = argmaxp(z|m, S, A, b, x, Sigma)$

Usage

```
## S3 method for class 'LinearGaussianGaussian'
MAP(obj, ...)
```

Arguments

obj	A "LinearGaussianGaussian" object.
...	Additional arguments to be passed to other inherited types.

Value

numeric vector, the MAP estimate of "z".

References

Murphy, Kevin P. Machine learning: a probabilistic perspective. MIT press, 2012.

See Also

LinearGaussianGaussian

Examples

```
obj <- LinearGaussianGaussian(gamma=list(Sigma=matrix(c(2,1,1,2),2,2),
                                         m=c(0.2,0.5,0.6),S=diag(3)))
x <- rGaussian(100,mu = runif(2),Sigma = diag(2))
A <- matrix(runif(6),2,3)
b <- runif(2)
ss <- sufficientStatistics(obj,x=x,A=A,b=b)
## update prior into posterior
posterior(obj=obj,ss=ss)
## get the MAP estimate of z
MAP(obj)
```

---

marginalLikelihood *Get the marginal likelihood of a "BayesianBrick" object*

Description

This is a generic function that will generate the marginal likelihood of a set of observations conditioned on a given "BayesianBrick" object. i.e. for the model structure:

$$theta|gamma \sim H(gamma)$$
$$x|theta \sim F(theta)$$

Marginal likelihood is p(x|gamma), p() is the probability density/mass function for continuous/discrete x. For a given Bayesian bricks object obj and a sample set x, `marginalLikelihood()` will calculate the marginal likelihood for different model structures:

class(obj)="LinearGaussianGaussian": Where

$$x \sim Gaussian(Az + b, Sigma)$$
$$z \sim Gaussian(m, S)$$

`marginalLikelihood()` will return p(x|m,S,A,b,Sigma) See `?marginalLikelihood.LinearGaussianGaussian` for details.

class(obj)="GaussianGaussian": Where

$$x \sim Gaussian(mu, Sigma)$$
$$mu \sim Gaussian(m, S)$$

Sigma is known. `marginalLikelihood()` will return p(x|m,S,Sigma) See `?marginalLikelihood.GaussianGaussian` for details.

class(obj)="GaussianInvWishart": Where

$$x \sim Gaussian(mu, Sigma)$$

$$Sigma \sim InvWishart(v, S)$$

mu is known. `marginalLikelihood()` will return p(x|mu,v,S) See `?marginalLikelihood.GaussianInvWishart` for details.

class(obj)="GaussianNIW": Where

$$x \sim Gaussian(mu, Sigma)$$

$$Sigma \sim InvWishart(v, S)$$

$$mu \sim Gaussian(m, Sigma/k)$$

`marginalLikelihood()` will return p(x|m,k,v,S) See `?marginalLikelihood.GaussianNIW` for details.

class(obj)="GaussianNIG": Where

$$x \sim Gaussian(Xbeta, sigma^2)$$

$$sigma^2 \sim InvGamma(a, b)$$

$$beta \sim Gaussian(m, sigma^2 V)$$

X is a row vector, or a design matrix where each row is an obervation. `marginalLikelihood()` will return p(x,X|m,V,a,b) See `?marginalLikelihood.GaussianNIG` for details.

class(obj)="CatDirichlet": Where

$$x \sim Categorical(pi)$$

$$pi \sim Dirichlet(alpha)$$

`marginalLikelihood()` will return p(x|alpha) See `?marginalLikelihood.CatDirichlet` for details.

class(obj)="CatDP": Where

$$x \sim Categorical(pi)$$

$$pi \sim DirichletProcess(alpha)$$

`marginalLikelihood()` will return p(x|alpha) See `?marginalLikelihood.CatDP` for details.

Usage

```
marginalLikelihood(obj, ...)
```

Arguments

`obj`	A "BayesianBrick" object used to select a method.
`...`	further arguments passed to or from other methods.

Value

numeric, the marginal likelihood

See Also

marginalLikelihood.LinearGaussianGaussian for Linear Gaussian and Gaussian conjugate structure, marginalLikelihood.GaussianGaussian for Gaussian-Gaussian conjugate structure, marginalLikelihood.GaussianInvWishart for Gaussian-Inverse-Wishart conjugate structure, marginalLikelihood.Gau for Gaussian-NIW conjugate structure, marginalLikelihood.GaussianNIG for Gaussian-NIG conjugate structure, marginalLikelihood.CatDirichlet for Categorical-Dirichlet conjugate structure, marginalLikelihood.CatDP for Categorical-DP conjugate structure ...

---

marginalLikelihood.CatDirichlet

Marginal likelihood of a "CatDirichlet" object

Description

Generate the marginal likelihood of the following model structure:

$$pi|alpha \sim Dir(alpha)$$

$$x|pi \sim Categorical(pi)$$

Where Dir() is the Dirichlet distribution, Categorical() is the Categorical distribution. See ?dDir and dCategorical for the definitions of these distribution.
The model structure and prior parameters are stored in a "CatDirichlet" object.
Marginal likelihood is the likelihood of x|alpha.

Usage

```
## S3 method for class 'CatDirichlet'
marginalLikelihood(obj, x, LOG = TRUE, ...)
```

Arguments

obj	A "CatDirichlet" object.
x	numeric/integer/character vector, observed Categorical samples.
LOG	Return the log density if set to "TRUE".
...	Additional arguments to be passed to other inherited types.

Value

numeric, the marginal likelihood.

References

Murphy, Kevin P. Machine learning: a probabilistic perspective. MIT press, 2012.

See Also

CatDirichlet, marginalLikelihood_bySufficientStatistics.CatDirichlet

Examples

```
obj <- CatDirichlet(gamma=list(alpha=runif(26,1,2),uniqueLabels = letters))
x <- sample(letters,size = 20,replace = TRUE)
marginalLikelihood(obj=obj,x=x,LOG = TRUE) #marginal likelihood
ss <- sufficientStatistics(obj = obj,x=x)
marginalLikelihood_bySufficientStatistics(obj=obj,ss = ss,LOG = TRUE)
```

---

marginalLikelihood.CatDP

Marginal likelihood of a "CatDP" object

Description

Generate the marginal likelihood of the following model structure:

$$pi|alpha \sim DP(alpha, U)$$

$$x|pi \sim Categorical(pi)$$

where DP(alpha,U) is a Dirichlet Process on positive integers, alpha is the "concentration parameter" of the Dirichlet Process, U is the "base measure" of this Dirichlet process, it is an uniform distribution on all positive integers.Categorical() is the Categorical distribution. See `dCategorical` for the definition of the Categorical distribution.
In the case of CatDP, x can only be positive integers.
The model structure and prior parameters are stored in a "CatDP" object.
Marginal likelihood = p(x|alpha).

Usage

```
## S3 method for class 'CatDP'
marginalLikelihood(obj, x, LOG = TRUE, ...)
```

Arguments

`obj`	A "CatDP" object.
`x`	integer, the elements of the vector must all greater than 0, the samples of a Categorical distribution.
`LOG`	Return the log density if set to "TRUE".
`...`	Additional arguments to be passed to other inherited types.

Value

numeric, the marginal likelihood.

See Also

CatDP, marginalLikelihood_bySufficientStatistics.CatDP

---

marginalLikelihood.DP *Marginal likelihood for Dirichlet Process(DP)*

Description

Marginal likelihood for Dirichlet Process(DP)

Usage

```
## S3 method for class 'DP'
marginalLikelihood(obj, ...)
```

Arguments

obj	an object.
...	Additional arguments to be passed to other inherited types.

Value

numeric, the marginal likelihood.

---

marginalLikelihood.GaussianGaussian *Marginal likelihood of a "GaussianGaussian" object*

Description

Generate the marginal likelihood of the following model structure:

$$x \sim Gaussian(mu, Sigma)$$

$$mu \sim Gaussian(m, S)$$

Where Sigma is known. Gaussian() is the Gaussian distribution. See ?dGaussian for the definition of Gaussian distribution.
The model structure and prior parameters are stored in a "GaussianGaussian" object.
Marginal likelihood = p(x|m,S,Sigma)

Usage

```
## S3 method for class 'GaussianGaussian'
marginalLikelihood(obj, x, LOG = TRUE, ...)
```

Arguments

obj	A "GaussianGaussian" object.
x	matrix, or the ones that can be converted to matrix, each row of x is an observation.
LOG	Return the log density if set to "TRUE".
...	Additional arguments to be passed to other inherited types.

Value

numeric, the marginal likelihood.

References

Gelman, Andrew, et al. Bayesian data analysis. CRC press, 2013.

See Also

GaussianGaussian, marginalLikelihood_bySufficientStatistics.GaussianGaussian

Examples

```
obj <- GaussianGaussian(gamma=list(Sigma=matrix(c(2,1,1,2),2,2),m=c(0.2,0.5),S=diag(2)))
x <- rGaussian(100,c(0,0),Sigma = matrix(c(2,1,1,2),2,2))
marginalLikelihood(obj = obj,x=x,LOG = TRUE)
marginalLikelihood(obj = obj,x=x,LOG = FALSE)
```

---

marginalLikelihood.GaussianInvWishart

Marginal likelihood of a "GaussianInvWishart" object

Description

Generate the marginal likelihood of the following model structure:

$$x \sim Gaussian(mu, Sigma)$$

$$Sigma \sim InvWishart(v, S)$$

mu is known. Gaussian() is the Gaussian distribution. See ?dGaussian and ?dInvWishart for the definition of the distributions.
The model structure and prior parameters are stored in a "GaussianInvWishart" object.
Marginal likelihood = p(x|v,S,mu)

Usage

```
## S3 method for class 'GaussianInvWishart'
marginalLikelihood(obj, x, LOG = TRUE, ...)
```

Arguments

obj	A "GaussianInvWishart" object.
x	matrix, or the ones that can be converted to matrix, each row of x is an observation.
LOG	Return the log density if set to "TRUE".
...	Additional arguments to be passed to other inherited types.

Value

numeric, the marginal likelihood.

References

Gelman, Andrew, et al. Bayesian data analysis. CRC press, 2013.

MARolA, K. V., JT KBNT, and J. M. Bibly. Multivariate analysis. AcadeInic Press, Londres, 1979.

See Also

GaussianInvWishart, marginalLikelihood_bySufficientStatistics.GaussianInvWishart

Examples

```
obj <- GaussianInvWishart(gamma=list(mu=c(-1.5,1.5),v=3,S=diag(2)))
x <- rGaussian(100,mu = c(-1.5,1.5),Sigma = matrix(c(0.1,0.03,0.03,0.1),2,2))
xNew <- rGaussian(100,mu = c(-1.5,1.5),Sigma = matrix(c(0.1,0.03,0.03,0.1),2,2))
ss <- sufficientStatistics(obj=obj,x=x,foreach = FALSE)
## update piror with x
posterior(obj=obj,ss = ss)
## use the posterior to calculate the likelihood of xNew
marginalLikelihood(obj = obj,x = xNew,LOG = TRUE)
```

---

marginalLikelihood.GaussianNIG

Marginal likelihood of a "GaussianNIG" object

Description

Generate the marginal likelihood of the following model structure:

$$x \sim Gaussian(Xbeta, sigma^2)$$

$$sigma^2 \sim InvGamma(a, b)$$

$$beta \sim Gaussian(m, sigma^2 V)$$

Where X is a row vector, or a design matrix where each row is an obervation. InvGamma() is the Inverse-Gamma distribution, Gaussian() is the Gaussian distribution. See ?dInvGamma and dGaussian for the definitions of these distribution.
The model structure and prior parameters are stored in a "GaussianNIG" object.
Marginal likelihood = p(x|m,V,a,b,X).

Usage

```
## S3 method for class 'GaussianNIG'
marginalLikelihood(obj, x, X, LOG = TRUE, ...)
```

Arguments

obj	A "GaussianNIG" object.
x	numeric, must satisfy length(x) = nrow(X).
X	matrix, must satisfy length(x) = nrow(X).
LOG	Return the log density if set to "TRUE".
...	Additional arguments to be passed to other inherited types.

Value

numeric, the marginal likelihood.

References

Banerjee, Sudipto. "Bayesian Linear Model: Gory Details." Downloaded from http://www. biostat. umn. edu/~ph7440 (2008).

See Also

GaussianNIG, marginalLikelihood_bySufficientStatistics.GaussianNIG

Examples

```
obj <- GaussianNIG(gamma=list(m=0,V=1,a=1,b=1))
X <- 1:20
x <- rnorm(20)+ X*0.3
marginalLikelihood(obj = obj,x = x, X = X)
marginalLikelihood(obj = obj,x = x, X = X,LOG = FALSE)
```

---

marginalLikelihood.GaussianNIW

Marginal likelihood of a "GaussianNIW" object

Description

Generate the marginal likelihood of the following model structure:

$$mu, Sigma|m, k, v, S \sim NIW(m, k, v, S)$$

$$x|mu, Sigma \sim Gaussian(mu, Sigma)$$

Where NIW() is the Normal-Inverse-Wishart distribution, Gaussian() is the Gaussian distribution. See ?dNIW and dGaussian for the definitions of these distribution.
The model structure and prior parameters are stored in a "GaussianNIW" object.
Marginal likelihood = p(x|m,k,v,S)

Usage

```
## S3 method for class 'GaussianNIW'
marginalLikelihood(obj, x, LOG = TRUE, ...)
```

Arguments

obj	A "GaussianNIW" object.
x	matrix, or the ones that can be converted to matrix, each row of x is an observation.
LOG	Return the log density if set to "TRUE".
...	Additional arguments to be passed to other inherited types.

Value

numeric, the marginal likelihood.

References

Murphy, Kevin P. "Conjugate Bayesian analysis of the Gaussian distribution." def 1.22 (2007): 16.

Gelman, Andrew, et al. "Bayesian Data Analysis Chapman & Hall." CRC Texts in Statistical Science (2004).

See Also

GaussianNIW, marginalLikelihood_bySufficientStatistics.GaussianNIW

Examples

```
x <- rGaussian(1000,mu = c(1,1),Sigma = matrix(c(1,0.5,0.5,3),2,2))
obj <- GaussianNIW(gamma=list(m=c(0,0),k=1,v=2,S=diag(2)))
marginalLikelihood(obj = obj,x=x)
## or...
ss <- sufficientStatistics(obj=obj,x=x,foreach = FALSE)
marginalLikelihood_bySufficientStatistics(obj = obj,ss=ss)
```

---

marginalLikelihood.HDP

Marginal likelihood for HDP

Description

Marginal likelihood for HDP

Usage

```
## S3 method for class 'HDP'
marginalLikelihood(obj, ...)
```

Arguments

obj	an object.
...	Additional arguments to be passed to other inherited types.

Value

numeric, the marginal likelihood.

---

marginalLikelihood.HDP2

Marginal likelihood for HDP2

Description

Marginal likelihood for HDP2

Usage

```
## S3 method for class 'HDP2'
marginalLikelihood(obj, ...)
```

Arguments

obj	an object.
...	Additional arguments to be passed to other inherited types.

Value

numeric, the marginal likelihood.

---

marginalLikelihood.LinearGaussianGaussian

Marginal likelihood of a "LinearGaussianGaussian" object

Description

Generate the marginal likelihood of the following model structure:

$$x \sim Gaussian(Az + b, Sigma)$$

$$z \sim Gaussian(m, S)$$

Where Sigma is known. A is a $dimxxdimz$ matrix, x is a $dimxx1$ random vector, z is a $dimzx1$ random vector, b is a $dimmx1$ vector. Gaussian() is the Gaussian distribution. See ?dGaussian for the definition of Gaussian distribution.
The model structure and prior parameters are stored in a "LinearGaussianGaussian" object.
Marginal likelihood = p(x|m,S,A,b,Sigma)

Usage

```
## S3 method for class 'LinearGaussianGaussian'
marginalLikelihood(obj, x, A, b = NULL, LOG = TRUE, ...)
```

Arguments

obj	A "LinearGaussianGaussian" object.
x	matrix, or the ones that can be converted to matrix. Each row of x is an observation.
A	matrix or list. when x is a $Nx1$ matrix, A must be a matrix of $Nxdimz$, dimz is the dimension of z; When x is a $Nxdimx$ matrix, where $dimx > 1$, A can be either a list or a matrix. When A is a list, $A = A_1, A_2, ...A_N$ is a list of $dimxxdimz$ matrices. If A is a single $dimxxdimz$ matrix, it will be replicated N times into a length N list
b	matrix, when x is a $Nx1$ matrix, b must also be a $Nx1$ matrix or length N vector; When x is a $Nxdimx$ matrix, where $dimx > 1$, b can be either a matrix or a vector. When b is a matrix, $b = b_1^T, ..., b_N^T$ is a $Nxdimx$ matrix, each row is a transposed vector. When b is a length $dimx$ vector, it will be transposed into a row vector and replicated N times into a $Nxdimx$ matrix. When b = NULL, it will be treated as a vector of zeros. Default NULL.
LOG	Return the log density if set to "TRUE".
...	Additional arguments to be passed to other inherited types.

Value

numeric, the marginal likelihood.

References

Gelman, Andrew, et al. Bayesian data analysis. CRC press, 2013.

See Also

LinearGaussianGaussian, marginalLikelihood_bySufficientStatistics.LinearGaussianGaussian

Examples

```
obj <- LinearGaussianGaussian(gamma=list(Sigma=matrix(c(2,1,1,2),2,2),
                                         m=c(0.2,0.5,0.6),S=diag(3)))
x <- rGaussian(100,mu = runif(2),Sigma = diag(2))
A <- matrix(runif(6),2,3)
b <- runif(2)
marginalLikelihood(obj = obj,x=x,A=A,b=b)
marginalLikelihood(obj = obj,x=x,A=A,b=b,LOG=FALSE)
```

`marginalLikelihood_bySufficientStatistics`
Get the marginal likelihood of a "BayesianBrick" object

Description

This is a generic function that will generate the marginal likelihood of a set of observations conditioned on a given "BayesianBrick" object. i.e. for the model structure:

$$theta|gamma \sim H(gamma)$$
$$x|theta \sim F(theta)$$

Marginal likelihood is p(x|gamma), p() is the probability density/mass function for continuous/discrete x. For a given Bayesian bricks object obj and a sample set x, `marginalLikelihood_bySufficientStatistics()` will calculate the marginal likelihood for different model structures:

class(obj)="LinearGaussianGaussian": Where

$$x \sim Gaussian(Az + b, Sigma)$$
$$z \sim Gaussian(m, S)$$

`marginalLikelihood_bySufficientStatistics()` will return p(x|m,S,A,b,Sigma) See `?marginalLikelihood_bySuff` for details.

class(obj)="GaussianGaussian": Where

$$x \sim Gaussian(mu, Sigma)$$
$$mu \sim Gaussian(m, S)$$

Sigma is known. `marginalLikelihood_bySufficientStatistics()` will return p(x|m,S,Sigma) See `?marginalLikelihood_bySufficientStatistics.GaussianGaussian` for details.

class(obj)="GaussianInvWishart": Where

$$x \sim Gaussian(mu, Sigma)$$
$$Sigma \sim InvWishart(v, S)$$

mu is known. `marginalLikelihood_bySufficientStatistics()` will return p(x|mu,v,S) See `?marginalLikelihood_bySufficientStatistics.GaussianInvWishart` for details.

class(obj)="GaussianNIW": Where

$$x \sim Gaussian(mu, Sigma)$$
$$Sigma \sim InvWishart(v, S)$$
$$mu \sim Gaussian(m, Sigma/k)$$

`marginalLikelihood_bySufficientStatistics()` will return p(x|m,k,v,S) See `?marginalLikelihood_bySufficient` for details.

class(obj)="GaussianNIG": Where

$$x \sim Gaussian(Xbeta, sigma^2)$$

$$sigma^2 \sim InvGamma(a, b)$$

$$beta \sim Gaussian(m, sigma^2 V)$$

X is a row vector, or a design matrix where each row is an obervation. `marginalLikelihood_bySufficientStatistics()` will return p(x,X|m,V,a,b) See `?marginalLikelihood_bySufficientStatistics.GaussianNIG` for details.

class(obj)="CatDirichlet": Where

$$x \sim Categorical(pi)$$

$$pi \sim Dirichlet(alpha)$$

`marginalLikelihood_bySufficientStatistics()` will return p(x|alpha) See `?marginalLikelihood_bySufficientSt` for details.

class(obj)="CatDP": Where

$$x \sim Categorical(pi)$$

$$pi \sim DirichletProcess(alpha)$$

`marginalLikelihood_bySufficientStatistics()` will return p(x|alpha) See `?marginalLikelihood_bySufficientSt` for details.

Usage

```
marginalLikelihood_bySufficientStatistics(obj, ...)
```

Arguments

`obj`	A "BayesianBrick" object used to select a method.
`...`	further arguments passed to or from other methods.

Value

numeric, the marginal likelihood

See Also

`marginalLikelihood_bySufficientStatistics.LinearGaussianGaussian` for Linear Gaussian and Gaussian conjugate structure, `marginalLikelihood_bySufficientStatistics.GaussianGaussian` for Gaussian-Gaussian conjugate structure, `marginalLikelihood_bySufficientStatistics.GaussianInvWishart` for Gaussian-Inverse-Wishart conjugate structure, `marginalLikelihood_bySufficientStatistics.GaussianNIW` for Gaussian-NIW conjugate structure, `marginalLikelihood_bySufficientStatistics.GaussianNIG` for Gaussian-NIG conjugate structure, `marginalLikelihood_bySufficientStatistics.CatDirichlet` for Categorical-Dirichlet conjugate structure, `marginalLikelihood_bySufficientStatistics.CatDP` for Categorical-DP conjugate structure ...

`marginalLikelihood_bySufficientStatistics.CatDirichlet` *Marginal likelihood of a "CatDirichlet" object, using sufficient statistics*

Description

Generate the marginal likelihood of a set of observations of the following model structure:

$$pi|alpha \sim Dir(alpha)$$

$$x|pi \sim Categorical(pi)$$

Where Dir() is the Dirichlet distribution, Categorical() is the Categorical distribution. See `?dDir` and `dCategorical` for the definitions of these distribution.
The model structure and prior parameters are stored in a "CatDirichlet" object.
Marginal likelihood is the likelihood of x|alpha

Usage

```
## S3 method for class 'CatDirichlet'
marginalLikelihood_bySufficientStatistics(obj, ss, LOG = TRUE, ...)
```

Arguments

`obj`	A "CatDirichlet" object.
`ss`	Sufficient statistics of x. In Categorical-Dirichlet case the sufficient statistic of sample x can be either x itself, of an "ssCat" object generated by the function sufficientStatistics.CatDirichlet().
`LOG`	Return the log density if set to "TRUE".
`...`	Additional arguments to be passed to other inherited types.

Value

numeric, the marginal likelihood.

References

Murphy, Kevin P. Machine learning: a probabilistic perspective. MIT press, 2012.

See Also

`CatDirichlet`, `marginalLikelihood.CatDirichlet`

Examples

```
obj <- CatDirichlet(gamma=list(alpha=runif(26,1,2),uniqueLabels = letters))
x <- sample(letters,size = 20,replace = TRUE)
marginalLikelihood(obj=obj,x=x,LOG = TRUE) #marginal likelihood
ss <- sufficientStatistics(obj = obj,x=x)
marginalLikelihood_bySufficientStatistics(obj=obj,ss = ss,LOG = TRUE)
```

---

marginalLikelihood_bySufficientStatistics.CatDP

Marginal likelihood of a "CatDP" object, using sufficient statistics

---

Description

Generate the marginal likelihood of the following model structure:

$$pi|alpha \sim DP(alpha, U)$$

$$x|pi \sim Categorical(pi)$$

where DP(alpha,U) is a Dirichlet Process on positive integers, alpha is the "concentration parameter" of the Dirichlet Process, U is the "base measure" of this Dirichlet process, it is an uniform distribution on all positive integers.Categorical() is the Categorical distribution. See `dCategorical` for the definition of the Categorical distribution.
In the case of CatDP, x can only be positive integers.
The model structure and prior parameters are stored in a "CatDP" object.
Marginal likelihood = p(x|alpha).

Usage

```
## S3 method for class 'CatDP'
marginalLikelihood_bySufficientStatistics(obj, ss, LOG = TRUE, ...)
```

Arguments

`obj`	A "CatDP" object.
`ss`	Sufficient statistics of x. In Categorical-DP case the sufficient statistic of sample x can either be an object of type "ssCatDP" generated by sufficientStatistics(), or x itself(if x is a integer vector with all positive values).
`LOG`	Return the log density if set to "TRUE".
`...`	Additional arguments to be passed to other inherited types.

Value

numeric, the marginal likelihood.

See Also

`CatDP`, `marginalLikelihood_bySufficientStatistics.CatDP`

`marginalLikelihood_bySufficientStatistics.GaussianGaussian` *Marginal likelihood of a "GaussianGaussian" object, using sufficient statistics*

Description

Generate the marginal likelihood of the following model structure:

$$x \sim Gaussian(mu, Sigma)$$

$$mu \sim Gaussian(m, S)$$

Where Sigma is known. Gaussian() is the Gaussian distribution. See `?dGaussian` for the definition of Gaussian distribution.
The model structure and prior parameters are stored in a "GaussianGaussian" object.
Marginal likelihood = p(x|m,S,Sigma)

Usage

```
## S3 method for class 'GaussianGaussian'
marginalLikelihood_bySufficientStatistics(obj, ss, LOG = TRUE, ...)
```

Arguments

`obj`	A "GaussianGaussian" object.
`ss`	Sufficient statistics of x. In Gaussian-Gaussian case the sufficient statistic of sample x is a object of type "ssGaussianMean", it can be generated by the function sufficientStatistics().
`LOG`	Return the log density if set to "TRUE".
`...`	Additional arguments to be passed to other inherited types.

Value

numeric, the marginal likelihood.

See Also

`GaussianGaussian`, `marginalLikelihood.GaussianGaussian`

`marginalLikelihood_bySufficientStatistics.GaussianInvWishart` *Marginal likelihood of a "GaussianInvWishart" object, using sufficient statistics*

Description

Generate the marginal likelihood of the following model structure:

$$x \sim Gaussian(mu, Sigma)$$

$$Sigma \sim InvWishart(v, S)$$

mu is known. Gaussian() is the Gaussian distribution. See `?dGaussian` and `?dInvWishart` for the definition of the distributions.
The model structure and prior parameters are stored in a "GaussianInvWishart" object.
Marginal likelihood = p(x|v,S,mu)

Usage

```
## S3 method for class 'GaussianInvWishart'
marginalLikelihood_bySufficientStatistics(obj, ss, LOG = TRUE, ...)
```

Arguments

`obj`	A "GaussianInvWishart" object.
`ss`	Sufficient statistics of x. In Gaussian and Inverse-Wishart case the sufficient statistic of sample x is a object of type "ssGaussianVar", it can be generated by the function sufficientStatistics().
`LOG`	Return the log density if set to "TRUE".
`...`	Additional arguments to be passed to other inherited types.

Value

numeric, the marginal likelihood.

See Also

`GaussianInvWishart`, `marginalLikelihood.GaussianInvWishart`

`marginalLikelihood_bySufficientStatistics.GaussianNIG` *Marginal likelihood of a "GaussianNIG" object, using sufficient statistics*

Description

Generate the marginal likelihood of a set of observations of the following model structure:

$$x \sim Gaussian(Xbeta, sigma^2)$$

$$sigma^2 \sim InvGamma(a, b)$$

$$beta \sim Gaussian(m, sigma^2 V)$$

Where X is a row vector, or a design matrix where each row is an obervation. InvGamma() is the Inverse-Gamma distribution, Gaussian() is the Gaussian distribution. See `?dInvGamma` and `dGaussian` for the definitions of these distribution.
The model structure and prior parameters are stored in a "GaussianNIG" object.
Marginal likelihood = p(x|m,V,a,b,X)

Usage

```
## S3 method for class 'GaussianNIG'
marginalLikelihood_bySufficientStatistics(obj, ss, LOG = TRUE, ...)
```

Arguments

`obj`	A "GaussianNIG" object.
`ss`	Sufficient statistics of (x,X). In Gaussian-NIG case the sufficient statistic of sample (x,X) is a object of type "ssGaussianLinear", it can be generated by the function sufficientStatistics().
`LOG`	Return the log density if set to "TRUE".
`...`	Additional arguments to be passed to other inherited types.

Value

numeric, the marginal likelihood.

References

Banerjee, Sudipto. "Bayesian Linear Model: Gory Details." Downloaded from http://www. biostat. umn. edu/~ph7440 (2008).

See Also

`GaussianNIG`, `marginalLikelihood.GaussianNIG`

Examples

```
obj <- GaussianNIG(gamma=list(m=0,V=1,a=1,b=1))
X <- 1:20
x <- rnorm(20)+ X*0.3
ss <- sufficientStatistics(obj=obj,x=x,X=X,foreach=FALSE)
marginalLikelihood_bySufficientStatistics(obj = obj,ss = ss)
marginalLikelihood_bySufficientStatistics(obj = obj,ss = ss,LOG = FALSE)
```

---

marginalLikelihood_bySufficientStatistics.GaussianNIW

Marginal likelihood of a "GaussianNIW" object, using sufficient statistics

---

Description

Generate the marginal likelihood of a set of observations of the following model structure:

$$mu, Sigma|m, k, v, S \sim NIW(m, k, v, S)$$

$$x|mu, Sigma \sim Gaussian(mu, Sigma)$$

Where NIW() is the Normal-Inverse-Wishart distribution, Gaussian() is the Gaussian distribution. See `?dNIW` and `dGaussian` for the definitions of these distribution.
The model structure and prior parameters are stored in a "GaussianNIW" object.
Marginal likelihood = p(x|m,k,v,S)

Usage

```
## S3 method for class 'GaussianNIW'
marginalLikelihood_bySufficientStatistics(obj, ss, LOG = TRUE, ...)
```

Arguments

`obj`	A "GaussianNIW" object.
`ss`	Sufficient statistics of x. In Gaussian-NIW case the sufficient statistic of sample x is a object of type "ssGaussian", it can be generated by the function sufficientStatistics().
`LOG`	Return the log density if set to "TRUE".
`...`	Additional arguments to be passed to other inherited types.

Value

numeric, the marginal likelihood.

References

Murphy, Kevin P. "Conjugate Bayesian analysis of the Gaussian distribution." def 1.22 (2007): 16.

Gelman, Andrew, et al. "Bayesian Data Analysis Chapman & Hall." CRC Texts in Statistical Science (2004).

See Also

GaussianNIW, marginalLikelihood.GaussianNIW

Examples

```
x <- rGaussian(1000,mu = c(1,1),Sigma = matrix(c(1,0.5,0.5,3),2,2))
obj <- GaussianNIW(gamma=list(m=c(0,0),k=1,v=2,S=diag(2)))
marginalLikelihood(obj = obj,x=x)
## or...
ss <- sufficientStatistics(obj=obj,x=x,foreach = FALSE)
marginalLikelihood_bySufficientStatistics(obj = obj,ss=ss)
```

---

marginalLikelihood_bySufficientStatistics.LinearGaussianGaussian

Marginal likelihood of a "LinearGaussianGaussian" object, using sufficient statistics

---

Description

Generate the marginal likelihood of the following model structure:

$$x \sim Gaussian(Az + b, Sigma)$$

$$z \sim Gaussian(m, S)$$

Where Sigma is known. A is a $dimxxdimz$ matrix, x is a $dimxx1$ random vector, z is a $dimzx1$ random vector, b is a $dimmx1$ vector. Gaussian() is the Gaussian distribution. See ?dGaussian for the definition of Gaussian distribution.
The model structure and prior parameters are stored in a "LinearGaussianGaussian" object.
Marginal likelihood = p(x|m,S,Sigma)

Usage

```
## S3 method for class 'LinearGaussianGaussian'
marginalLikelihood_bySufficientStatistics(obj, ss, LOG = TRUE, ...)
```

Arguments

obj	A "LinearGaussianGaussian" object.
ss	Sufficient statistics of x. In Gaussian-Gaussian case the sufficient statistic of sample x is a object of type "ssGaussianMean", it can be generated by the function sufficientStatistics().
LOG	Return the log density if set to "TRUE".
...	Additional arguments to be passed to other inherited types.

Value

numeric, the marginal likelihood.

See Also

LinearGaussianGaussian, marginalLikelihood.LinearGaussianGaussian

---

MetropolisHastings *Metropolis-Hastings sampler*

Description

sample xhat from the target distribution p(xhat), with given proposal distribution q(xhat|x), the acceptance rate is:
min(1 , dp(xhat)/dp(x) * dq(x|xhat)dq(xhat|x))
where dp() is the density function of the target distribution, dq() the the density function of the proposal distribution. A new sample xhat is drawn from the sampler of the proposal distribution rq(). See examples.

Usage

```
MetropolisHastings(nsamples, xini, dp, dq, rq)
```

Arguments

nsamples	integer, number of samples to draw
xini	initial sample, the chain of samples starts from here. xini must be a matrix of one row, or a numeric vector that will be converted to a matrix of one row.
dp	function(x), the LOG density function of the target distribution log dp(x), DON'T FORGET THE LOG.
dq	function(xhat,x), the LOG density function of the proposal distribution log dq(xhat | x), DON'T FORGET THE LOG.
rq	function(x), the generator of the proposal distribution rq(xhat | x).

Value

a matrix of nsamples rows.

Examples

```
## example1: independent Metropolis-Hastings algorithm, get 5000 samples from Beta(2.7,6.3)
## with independent uniform proposal U(0,1), and independent normal proposal N(0.5,1).

## step1: define p() and q()
dp <- function(x) if(x>0&x<1) dbeta(x,2.7,6.3,log = TRUE) else -Inf
dq1 <- function(xnew,x) 0               #uniform proposal log density
dq2 <- function(xnew,x) dnorm(xnew,0.5,1) #normal proposal log density
rq1 <- function(x) runif(1,0,1)           #uniform proposal sampler
rq2 <- function(x) rnorm(1,0.5,1)         #normal proposal sampler
```

```
## step2: get 5000 samples, with two different proposals
X1 <- MetropolisHastings(nsamples = 5000,xini = runif(1,0,1),dp=dp,dq=dq1,rq=rq1)
X2 <- MetropolisHastings(nsamples = 5000,xini = runif(1,0,1),dp=dp,dq=dq2,rq=rq2)
## step3: plot the result, calculate acceptance rate
sum(diff(X1)!=0)/nrow(X1)                #the acceptance rate of uniform proposal
sum(diff(X2)!=0)/nrow(X2)                #the acceptance rate of normal proposal
## Clearly Uniform, compare to Normal, can better resemble Beta, so the acceptance rate is higher
## plot the results
hist(X1)
hist(X2)
hist(rbeta(5000,2.7,6.3))

## example2: independent Metropolis-Hastings algorithm, sample from an improper distribution
## p(x) = -|x|+1, where -1<x<1, with independent uniform proposal U(-1,1)

## step1: define p() and q()
dp <- function(x) log(-abs(x)+1)        #log dp
dq <- function(xnew,x) 1
rq <- function(x) runif(1,-1,1)         #make sure -1<x<1
## step2: get 5000 samples
X <- MetropolisHastings(nsamples = 5000,xini = runif(1,-1,1),dp=dp,dq=dq,rq=rq)
## step3: plot the result, calculate acceptance rate
hist(X)
sum(diff(X)!=0)/nrow(X)                 #the acceptance rate

## example3: random walk Metropolis-Hastings algorithm, sample from a
## normal mixture 0.2*N(1,1)+0.8*N(-5,1), with symmetric proposal xhat ~ U(x-l,x+l),
## compare different values of l.

## step1: define p() and q()
dp <- function(x) log(dnorm(x,1,1)*0.2+dnorm(x,-5,1)*0.8)
## a symmetric proposal has no influence to the acceptance rate, so a constant function
## would suffice.
dq <- function(xnew,x) 1
rq1 <- function(x) runif(1,x-0.01,x+0.01)
rq2 <- function(x) runif(1,x-2,x+2)
## step2: get 5000 samples
X1 <- MetropolisHastings(nsamples = 5000,xini = rnorm(1),dp=dp,dq=dq,rq=rq1)
X2 <- MetropolisHastings(nsamples = 50000,xini = rnorm(1),dp=dp,dq=dq,rq=rq2)
## step3: plot the result, calculate acceptance rate
sum(diff(X1)!=0)/nrow(X1)
sum(diff(X2)!=0)/nrow(X2)
## plot the results
hist(X1,xlim = c(-10,5))
hist(X2,xlim = c(-10,5))
hist(c(rnorm(1000,1,1),rnorm(4000,-5,1)),xlim = c(-10,5))

## note that X1 has a higher acceptance rate comparing to X2, though it performs poorer.
## So we use Kolmogorov-Smirnov to test the real performance of X1 and X2:
ks.test(jitter(X1),c(rnorm(1000,1,1),rnorm(4000,-5,1))) #ks.test() assumes continuous
## samples doesn't contain equal values, otherwise there will be a warning.so use jitter() to
## remove the equals
ks.test(jitter(X2),c(rnorm(1000,1,1),rnorm(4000,-5,1)))
```

```
## it turns out that even though X2 looks better from the histogram, it still doesn't
## pass the KS test.

## example4: hybrid Metropolis-Hastings algorithm, questions same as previous example,
## but use a mixutre proposal instead.

## we use mixture proposal to capture both the local and global areas of the target distribution
## step1: define p() and q()
dp <- function(x) log(dnorm(x,1,1)*0.2+dnorm(x,-5,1)*0.8)
## a symmetric proposal has no influence to the acceptance rate,
## so a constant function would suffice.
dq <- function(xnew,x) 1
##70% local, 30% global
rq <- function(x) if(runif(1)<0.7) runif(1,x-0.1,x+0.1) else runif(1,x-3,x+3)
## step2: get 5000 samples
X <- MetropolisHastings(nsamples = 5000,xini = rnorm(1),dp=dp,dq=dq,rq=rq)
## step3: plot the result, calculate acceptance rate
sum(diff(X)!=0)/nrow(X)
## plot the results
hist(X,xlim = c(-10,5))
hist(c(rnorm(1000,1,1),rnorm(4000,-5,1)),xlim = c(-10,5))

## perform the KS test again, this time it says there's no significance difference between the MH
## and the real samples. ks.test() assumes continuous samples doesn't contain equal values,
## otherwise there will be a warning.so use jitter() to remove the equals
ks.test(jitter(X),c(rnorm(1000,1,1),rnorm(4000,-5,1)))
```

`mmData` *Samples from a mixture model*

Description

A matrix of 2-dimensional Gaussian mixture samples, the samples came from 4 different Gaussian distributions

Usage

```
data(mmData)
```

Format

A matrix of Gaussian samples, each row is one sample.

mmhData — *Samples from a hierarchical mixture model*

Description

A list of 2-dimensional Gaussian mixture samples and the corresponding group labels. The samples are generated by 4 different Gaussian distributions. The samples are separated into 30 groups.

Usage

```
data(mmhData)
```

Format

A list of two elements:

x : matrix, 2-dimensional Gaussian samples, each row is a sample

groupLabel : integer, the group label of each Gaussian sample

mmhhData — *Samples from a hierarchical mixture model with two layers of hierarchies*

Description

A list of 2-dimensional Gaussian mixture samples and the corresponding group labels. The samples are generated by 4 different Gaussian distributions. Each sample is assigned to a group and a subgroup. There are 2 groups, group 1 has 10 subgroups, group 2 has 20 subgroups.

Usage

```
data(mmhhData)
```

Format

A list of three elements:

x : matrix, 2-dimensional Gaussian samples, each row is a sample

groupLabel : integer, the group label of each Gaussian sample

subGroupLabel : integer, the subgroup label of each Gaussian sample

MPE *Get the Mean Posterior Estimate(MPE) of a "BayesianBrick" object*

Description

This is a generic function that will generate the MPE estimate of a given "BayesianBrick" object. i.e. for the model structure:

$$theta|gamma \sim H(gamma)$$

$$x|theta \sim F(theta)$$

MPE estimate of theta is theta_MPE = E(theta|gamma,x), E() is the expectation function. For a given Bayesian bricks object obj, the MPE estimate will be:

class(obj)="LinearGaussianGaussian": Where

$$x \sim Gaussian(Az + b, Sigma)$$

$$z \sim Gaussian(m, S)$$

`MPE()` will return the MPE estimate of z. See `?MPE.LinearGaussianGaussian` for details.

class(obj)="GaussianGaussian": Where

$$x \sim Gaussian(mu, Sigma)$$

$$mu \sim Gaussian(m, S)$$

Sigma is known. `MPE()` will return the MPE estimate of mu. See `?MPE.GaussianGaussian` for details.

class(obj)="GaussianInvWishart": Where

$$x \sim Gaussian(mu, Sigma)$$

$$Sigma \sim InvWishart(v, S)$$

mu is known. `MPE()` will return the MPE estimate of Sigma. See `?MPE.GaussianInvWishart` for details.

class(obj)="GaussianNIW": Where

$$x \sim Gaussian(mu, Sigma)$$

$$Sigma \sim InvWishart(v, S)$$

$$mu \sim Gaussian(m, Sigma/k)$$

`MPE()` will return the MPE estimate of mu and Sigma. See `?MPE.GaussianNIW` for details.

class(obj)="GaussianNIG": Where

$$x \sim Gaussian(Xbeta, sigma^2)$$

$$sigma^2 \sim InvGamma(a, b)$$

$$beta \sim Gaussian(m, sigma^2 V)$$

X is a row vector, or a design matrix where each row is an obervation. `MPE()` will return the MPE estimate of beta and sigma^2. See `?MPE.GaussianNIG` for details.

class(obj)="CatDirichlet": Where

$$x \sim Categorical(pi)$$

$$pi \sim Dirichlet(alpha)$$

`MPE()` will return the MPE estimate of pi. See `?MPE.CatDirichlet` for details.

class(obj)="CatDP": Where

$$x \sim Categorical(pi)$$

$$pi \sim DirichletProcess(alpha)$$

`MPE()` will return the MPE estimate of pi. See `?MPE.CatDP` for details.

Usage

```
MPE(obj, ...)
```

Arguments

`obj`	A "BayesianBrick" object used to select a method.
`...`	further arguments passed to or from other methods.

Value

A list of MPE estimates

See Also

`MPE.LinearGaussianGaussian` for Linear Gaussian and Gaussian conjugate structure, `MPE.GaussianGaussian` for Gaussian-Gaussian conjugate structure, `MPE.GaussianInvWishart` for Gaussian-Inverse-Wishart conjugate structure, `MPE.GaussianNIW` for Gaussian-NIW conjugate structure, `MPE.GaussianNIG` for Gaussian-NIG conjugate structure, `MPE.CatDirichlet` for Categorical-Dirichlet conjugate structure, `MPE.CatDP` for Categorical-DP conjugate structure ...

`MPE.CatDirichlet` *MPE of a "CatDirichlet" object*

Description

Generate the MPE of "pi" in following Categorical-Dirichlet structure:

$$pi|alpha \sim Dir(alpha)$$

$$x|pi \sim Categorical(pi)$$

Where Dir() is the Dirichlet distribution, Categorical() is the Categorical distribution. See `?dDir` and `dCategorical` for the definitions of these distribution.
The model structure and prior parameters are stored in a "CatDirichlet" object.
MPE is pi_MPE = E(pi|alpha,x), E() is the expectation function.

Usage

```
## S3 method for class 'CatDirichlet'
MPE(obj, ...)
```

Arguments

`obj`	A "CatDirichlet" object.
`...`	Additional arguments to be passed to other inherited types.

Value

A numeric vector, the MPE of "pi".

References

Murphy, Kevin P. Machine learning: a probabilistic perspective. MIT press, 2012.

See Also

`CatDirichlet`

Examples

```
obj <- CatDirichlet(gamma=list(alpha=rep(1,26),uniqueLabels = letters))
x <- sample(letters,size = 20,replace = TRUE)
w <- runif(20)
posterior(obj=obj,ss=x,w=w)
MPE(obj)
```

MPE.CatDP *Mean Posterior Estimate(MPE) of a "CatDP" object*

Description

Generate the MPE estimate of "pi" in following model structure:

$$pi|alpha \sim DP(alpha, U)$$

$$x|pi \sim Categorical(pi)$$

where DP(alpha,U) is a Dirichlet Process on positive integers, alpha is the "concentration parameter" of the Dirichlet Process, U is the "base measure" of this Dirichlet process, it is an uniform distribution on all positive integers.Categorical() is the Categorical distribution. See `dCategorical` for the definition of the Categorical distribution.
In the case of CatDP, x can only be positive integers.
The model structure and prior parameters are stored in a "CatDP" object.
The MPE of pi is pi_MPE = E(pi|alpha,x), E() is the expectation function.

Usage

```
## S3 method for class 'CatDP'
MPE(obj, ...)
```

Arguments

`obj`	A "CatDP" object.
`...`	Additional arguments to be passed to other inherited types.

Value

numeric.

References

Teh, Yee W., et al. "Sharing clusters among related groups: Hierarchical Dirichlet processes." Advances in neural information processing systems. 2005.

See Also

`CatDP`

Examples

```
x <- sample(1L:10L,size = 40,replace = TRUE)
obj <- CatDP()
posterior(obj = obj,ss = x)
MPE(obj)
```

`MPE.GaussianGaussian` *Mean Posterior Estimate (MPE) of a "GaussianGaussian" object*

Description

Generate the MPE estimate of mu in following model structure:

$$x \sim Gaussian(mu, Sigma)$$

$$mu \sim Gaussian(m, S)$$

Where Sigma is known. Gaussian() is the Gaussian distribution. See `?dGaussian` for the definition of Gaussian distribution.
The model structure and prior parameters are stored in a "GaussianGaussian" object.
The MPE estimates is:

- mu_MPE = E(mu|m,S,x,Sigma)

Usage

```
## S3 method for class 'GaussianGaussian'
MPE(obj, ...)
```

Arguments

`obj`	A "GaussianGaussian" object.
`...`	Additional arguments to be passed to other inherited types.

Value

numeric vector, the MPE estimate of "mu".

References

Gelman, Andrew, et al. Bayesian data analysis. CRC press, 2013.

See Also

`GaussianGaussian`

Examples

```
obj <- GaussianGaussian(gamma=list(Sigma=matrix(c(2,1,1,2),2,2),m=c(0.2,0.5),S=diag(2)))
x <- rGaussian(100,c(0,0),Sigma = matrix(c(2,1,1,2),2,2))
ss <- sufficientStatistics(obj=obj,x=x,foreach = FALSE)
## update prior into posterior
posterior(obj = obj,ss = ss)
## get the MPE estimate of mu
MPE(obj)
```

`MPE.GaussianInvWishart` *Mean Posterior Estimate (MPE) of a "GaussianInvWishart" object*

Description

Generate the MPE estimate of Sigma in following model structure:

$$x \sim Gaussian(mu, Sigma)$$

$$Sigma \sim InvWishart(v, S)$$

mu is known. Gaussian() is the Gaussian distribution. See `?dGaussian` and `?dInvWishart` for the definition of the distributions.
The model structure and prior parameters are stored in a "GaussianInvWishart" object.
The MPE estimates are:

- (Sigma_MPE) = E(Sigma|v,S,x,mu)

Usage

```
## S3 method for class 'GaussianInvWishart'
MPE(obj, ...)
```

Arguments

`obj`	A "GaussianInvWishart" object.
`...`	Additional arguments to be passed to other inherited types.

Value

matrix, the MPE estimate of "Sigma".

References

Gelman, Andrew, et al. Bayesian data analysis. CRC press, 2013.

MARolA, K. V., JT KBNT, and J. M. Bibly. Multivariate analysis. AcadeInic Press, Londres, 1979.

See Also

`GaussianInvWishart`

Examples

```
obj <- GaussianInvWishart(gamma=list(mu=c(-1.5,1.5),v=3,S=diag(2)))
x <- rGaussian(100,mu = c(-1.5,1.5),Sigma = matrix(c(0.1,0.03,0.03,0.1),2,2))
ss <- sufficientStatistics(obj=obj,x=x,foreach = FALSE)
posterior(obj=obj,ss = ss)
MPE(obj)
```

MPE.GaussianNIG *Mean Posterior Estimate (MPE) of a "GaussianNIG" object*

Description

Generate the MPE estimate of (beta,sigma^2) in following Gaussian-NIG structure:

$$x \sim Gaussian(Xbeta, sigma^2)$$

$$sigma^2 \sim InvGamma(a, b)$$

$$beta \sim Gaussian(m, sigma^2 V)$$

Where X is a row vector, or a design matrix where each row is an obervation. InvGamma() is the Inverse-Gamma distribution, Gaussian() is the Gaussian distribution. See `?dInvGamma` and `dGaussian` for the definitions of these distribution.
The model structure and prior parameters are stored in a "GaussianNIG" object.
The MPEs are E(beta,sigma^2|m,V,a,b,X,x)

Usage

```
## S3 method for class 'GaussianNIG'
MPE(obj, ...)
```

Arguments

`obj`	A "GaussianNIG" object.
`...`	Additional arguments to be passed to other inherited types.

Value

A named list, the MPE estimate of beta and sigma^2.

References

Banerjee, Sudipto. "Bayesian Linear Model: Gory Details." Downloaded from http://www. biostat. umn. edu/~ph7440 (2008).

See Also

GaussianNIG

`MPE.GaussianNIW` *Mean Posterior Estimate (MPE) of a "GaussianNIW" object*

Description

Generate the MPE of (mu,Sigma) in following GaussianNIW structure:

$$mu, Sigma|m, k, v, S \sim NIW(m, k, v, S)$$

$$x|mu, Sigma \sim Gaussian(mu, Sigma)$$

Where NIW() is the Normal-Inverse-Wishart distribution, Gaussian() is the Gaussian distribution. See `?dNIW` and `dGaussian` for the definitions of these distribution.
The model structure and prior parameters are stored in a "GaussianNIW" object.
The MPE estimates are:

- (mu_MPE,Sigma_MPE) = E(mu,Sigma|m,k,v,S,x)

Usage

```
## S3 method for class 'GaussianNIW'
MPE(obj, ...)
```

Arguments

`obj`	A "GaussianNIW" object.
`...`	Additional arguments to be passed to other inherited types.

Value

A named list, the MPE estimate of mu and Sigma.

References

Murphy, Kevin P. "Conjugate Bayesian analysis of the Gaussian distribution." def 1.22 (2007): 16.

Gelman, Andrew, et al. "Bayesian Data Analysis Chapman & Hall." CRC Texts in Statistical Science (2004).

See Also

`GaussianNIW`

`MPE.LinearGaussianGaussian` *Mean Posterior Estimate (MPE) of a "LinearGaussianGaussian" object*

Description

Generate the MPE estimate of mu in following model structure:

$$x \sim Gaussian(Az + b, Sigma)$$

$$z \sim Gaussian(m, S)$$

Where Sigma is known. A is a $dimxxdimz$ matrix, x is a $dimxx1$ random vector, z is a $dimzx1$ random vector, b is a $dimmx1$ vector. Gaussian() is the Gaussian distribution. See `?dGaussian` for the definition of Gaussian distribution.
The model structure and prior parameters are stored in a "LinearGaussianGaussian" object.
The MPE estimates is:

- z_MPE = E(z|m,S,A,b,x,Sigma)

Usage

```
## S3 method for class 'LinearGaussianGaussian'
MPE(obj, ...)
```

Arguments

`obj`	A "LinearGaussianGaussian" object.
`...`	Additional arguments to be passed to other inherited types.

Value

numeric vector, the MPE estimate of "z".

References

Murphy, Kevin P. Machine learning: a probabilistic perspective. MIT press, 2012.

See Also

`LinearGaussianGaussian`

Examples

```
obj <- LinearGaussianGaussian(gamma=list(Sigma=matrix(c(2,1,1,2),2,2),
                                         m=c(0.2,0.5,0.6),S=diag(3)))
x <- rGaussian(100,mu = runif(2),Sigma = diag(2))
A <- matrix(runif(6),2,3)
b <- runif(2)
ss <- sufficientStatistics(obj,x=x,A=A,b=b)
## update prior into posterior
posterior(obj=obj,ss=ss)
## get the MAP estimate of z
MPE(obj)
```

---

pdsDeterminant *determinant of a positive definite symmetric matrix*

Description

Use Cholesky decomposition to calculate the determinant of S, where $S = A'A$, A is a upper diagonal matrix. det(S) = det(A)*det(A).

Usage

```
pdsDeterminant(S, LOG = FALSE)
```

Arguments

S	a symmetric positive definitive matrix.
LOG	logical, return log(det(S)) if TRUE, return det(S) if FALSE, default FALSE.

Value

A matrix, the determinant of "S".

Examples

```
Sigma = matrix(c(1.3,1,1,2),2,2)      # some positive definite symmetric matrix
pdsDeterminant(Sigma)                 # get inv(Sigma)
pdsDeterminant(Sigma,LOG=TRUE)        # get inv(Sigma)
pdsInverse(Sigma,returnUpper=TRUE)  # get inv(A), where Sigma=A'A, A is upper triangle
```

pdsInverse *Inverse of a positive definite symmetric matrix*

Description

Use Cholesky decomposition to calculate the inverse of S, where $S = A'A$, A is a upper diagonal matrix. inv(S) = inv(A)inv(A)'.

Usage

```
pdsInverse(S, returnUpper = FALSE)
```

Arguments

S	a symmetric positive definitive matrix.
returnUpper	logical, return inv(A) if returnUpper=TRUE,return inv(S) if returnUpper=FALSE, default FALSE.

Value

A matrix, the inverse of "S".

Examples

```
Sigma = matrix(c(2,1,1,2),2,2)      # some positive definite symmetric matrix
pdsInverse(Sigma)                   # get inv(Sigma)
pdsInverse(Sigma,returnUpper=TRUE)  # get inv(A), where Sigma=A'A, A is upper triangle
```

posterior *update the prior distribution with sufficient statistics*

Description

This is a generic function that will update the prior distribution of a "BayesianBrick" object by adding information of the observation's sufficient statistics. i.e. for the model structure:

$$theta|gamma \sim H(gamma)$$

$$x|theta \sim F(theta)$$

update gamma to gamma_posterior by adding the information of x to gamma.
For a given sample set x or it's sufficient statistics ss, and a Bayesian bricks object obj, `posterior()` will update the posterior parameters in obj for different model structures:

class(obj)="LinearGaussianGaussian":

$$x \sim Gaussian(Az + b, Sigma)$$

$$z \sim Gaussian(m, S)$$

`posterior()` will update m and S in obj. See `?posterior.LinearGaussianGaussian` for details.

class(obj)="GaussianGaussian": Where

$$x \sim Gaussian(mu, Sigma)$$

$$mu \sim Gaussian(m, S)$$

Sigma is known. `posterior()` will update m and S in obj. See `?posterior.GaussianGaussian` for details.

class(obj)="GaussianInvWishart": Where

$$x \sim Gaussian(mu, Sigma)$$

$$Sigma \sim InvWishart(v, S)$$

mu is known. `posterior()` will update v and S in obj. See `?posterior.GaussianInvWishart` for details.

class(obj)="GaussianNIW": Where

$$x \sim Gaussian(mu, Sigma)$$

$$Sigma \sim InvWishart(v, S)$$

$$mu \sim Gaussian(m, Sigma/k)$$

`posterior()` will update m, k, v and S in obj. See `?posterior.GaussianNIW` for details.

class(obj)="GaussianNIG": Where

$$x \sim Gaussian(Xbeta, sigma^2)$$

$$sigma^2 \sim InvGamma(a, b)$$

$$beta \sim Gaussian(m, sigma^2 V)$$

X is a row vector, or a design matrix where each row is an obervation. `posterior()` will update m, V, a and b in obj. See `?posterior.GaussianNIG` for details.

class(obj)="CatDirichlet": Where

$$x \sim Categorical(pi)$$

$$pi \sim Dirichlet(alpha)$$

`posterior()` will update alpha in obj. See `?posterior.CatDirichlet` for details.

class(obj)="CatDP": Where

$$x \sim Categorical(pi)$$

$$pi \sim DirichletProcess(alpha)$$

`posterior()` will update alpha in obj. See `?posterior.CatDP` for details.

class(obj)="DP": Where

$$pi|alpha \sim DP(alpha, U)$$

$$z|pi \sim Categorical(pi)$$

$$theta_z|psi \sim H0(psi)$$

$$x|theta_z, z \sim F(theta_z)$$

`posterior()` will update alpha and psi in obj. See `?posterior.DP` for details.

class(obj)="HDP": Where

$$G|gamma \sim DP(gamma, U)$$

$$pi_j|G, alpha \sim DP(alpha, G), j = 1 : J$$

$$z|pi_j \sim Categorical(pi_j)$$

$$k|z, G \sim Categorical(G), \text{ if z is a sample from the base measure G}$$

$$theta_k|psi \sim H0(psi)$$

`posterior()` will update gamma, alpha and psi in obj. See `?posterior.HDP` for details.

class(obj)="HDP2": Where

$$G|eta \sim DP(eta, U)$$

$$G_m|gamma, G \sim DP(gamma, G), m = 1 : M$$

$$pi_{mj}|G_m, alpha \sim DP(alpha, G_m), j = 1 : J_m$$

$$z|pi_{mj} \sim Categorical(pi_{mj})$$

$$k|z, G_m \sim Categorical(G_m), \text{ if z is a sample from the base measure } G_m$$

$$u|k, G \sim Categorical(G), \text{ if k is a sample from the base measure} G$$

$$theta_u|psi \sim H0(psi)$$

$$x|theta_u, u \sim F(theta_u)$$

`posterior()` will update eta, gamma, alpha and psi in obj. See `?posterior.HDP2` for details.

Usage

```
posterior(obj, ...)
```

Arguments

`obj`	A "BayesianBrick" object used to select a method.
`...`	further arguments passed to or from other methods.

Value

None, or an error message if the update fails.

See Also

posterior.LinearGaussianGaussian for Linear Gaussian and Gaussian conjugate structure, posterior.GaussianGaussi for Gaussian-Gaussian conjugate structure, posterior.GaussianInvWishart for Gaussian-Inverse-Wishart conjugate structure, posterior.GaussianNIW for Gaussian-NIW conjugate structure, posterior.GaussianNIG for Gaussian-NIG conjugate structure, posterior.CatDirichlet for Categorical-Dirichlet conjugate structure, posterior.CatDP for Categorical-DP conjugate structure ...

---

posterior.CatDirichlet

Update a "CatDirichlet" object with sample sufficient statistics

---

Description

For the model structure:

$$pi|alpha \sim Dir(alpha)$$

$$x|pi \sim Categorical(pi)$$

Where Dir() is the Dirichlet distribution, Categorical() is the Categorical distribution. See ?dDir and dCategorical for the definitions of these distribution.
update alpha by adding the information of newly observed samples x.
The model structure and prior parameters are stored in a "CatDirichlet" object, the prior parameters in this object will be updated after running this function.

Usage

```
## S3 method for class 'CatDirichlet'
posterior(obj, ss, w = NULL, ...)
```

Arguments

obj	A "CatDirichlet" object.
ss	Sufficient statistics of x. In Categorical-Dirichlet case the sufficient statistic of sample x can be either x itself, of an "ssCat" object generated by the function sufficientStatistics.CatDirichlet().
w	Sample weights, default NULL.
...	Additional arguments to be passed to other inherited types.

Value

None. the gamma stored in "obj" will be updated based on "ss".

References

Murphy, Kevin P. Machine learning: a probabilistic perspective. MIT press, 2012.

See Also

CatDirichlet,posteriorDiscard.CatDirichlet

Examples

```
obj <- CatDirichlet(gamma=list(alpha=rep(1,26),uniqueLabels = letters))
x <- sample(letters,size = 20,replace = TRUE)
w <- runif(20)
posterior(obj=obj,ss=x)
obj
posteriorDiscard(obj=obj,ss=x)
obj
## weighted sample
posterior(obj=obj,ss=x,w=w)
obj
posteriorDiscard(obj=obj,ss=x,w=w)
obj
```

---

posterior.CatDP *Update a "CatDP" object with sample sufficient statistics*

Description

For the model structure:

$$pi|alpha \sim DP(alpha, U)$$

$$x|pi \sim Categorical(pi)$$

where DP(alpha,U) is a Dirichlet Process on positive integers, alpha is the "concentration parameter" of the Dirichlet Process, U is the "base measure" of this Dirichlet process, it is an uniform distribution on all positive integers.Categorical() is the Categorical distribution. See dCategorical for the definition of the Categorical distribution.
In the case of CatDP, x can only be positive integers.
Update prior knowledge by adding the information of newly observed samples x. The model structure and prior parameters are stored in a "CatDP" object, the prior parameters in this object will be updated after running this function.

Usage

```
## S3 method for class 'CatDP'
posterior(obj, ss, w = NULL, ...)
```

Arguments

obj	A "CatDP" object.
ss	Sufficient statistics of x. In Categorical-DP case the sufficient statistic of sample x can either be an object of type "ssCatDP" generated by sufficientStatistics(), or x itself(if x is a integer vector with all positive values).
w	Sample weights, default NULL.
...	Additional arguments to be passed to other inherited types.

Value

None. the model stored in "obj" will be updated based on "ss".

References

Teh, Yee W., et al. "Sharing clusters among related groups: Hierarchical Dirichlet processes." Advances in neural information processing systems. 2005.

See Also

CatDP,posteriorDiscard.CatDP,sufficientStatistics.CatDP

Examples

```
## generate some integer samples
x <- sample(1L:10L,size = 40,replace = TRUE)
obj <- CatDP()
obj2 <- CatDP()
obj3 <- CatDP()
## update CatDP object with sufficient statistics
ss <- sufficientStatistics(obj=obj,x=x)
posterior(obj = obj,ss = ss)
## or, update with x itself
posterior(obj = obj2,ss = x)
## or, update with x itself, one by one
for(xx in x) posterior(obj = obj3,ss = xx)
## obj, obj2, obj3 should be the same:
obj
obj2
obj3
```

---

posterior.CatHDP *Update a "CatHDP" object with sample sufficient statistics*

Description

For the model structure:

$$G|gamma \sim DP(gamma, U)$$

$$pi_j|G, alpha \sim DP(alpha, G), j = 1 : J$$

$$z|pi_j \sim Categorical(pi_j)$$

$$k|z, G \sim Categorical(G), \text{ if z is a sample from the base measure G}$$

where DP(gamma,U) is a Dirichlet Process on positive integers, gamma is the "concentration parameter", U is the "base measure" of this Dirichlet process, U is an uniform distribution on all positive integers. DP(alpha,G) is a Dirichlet Process on integers with concentration parameter alpha and base measure G. Categorical() is the Categorical distribution. See `dCategorical` for the definition of the Categorical distribution.
In the case of CatHDP, z and k can only be positive integers.
Update the prior knowledge by adding the information of newly observed samples z and k. The model structure and prior parameters are stored in a "CatHDP" object, the prior parameters in this object will be updated after running this function.

Usage

```
## S3 method for class 'CatHDP'
posterior(obj, ss1, ss2, j, w = NULL, ...)
```

Arguments

`obj`	A "CatHDP" object.
`ss1`	Sufficient statistics of k. In CatHDP case the sufficient statistic of sample k is k itself(if k is a integer vector with all positive values).
`ss2`	Sufficient statistics of z. In CatHDP case the sufficient statistic of sample z is z itself(if z is a integer vector with all positive values).
`j`	integer, group label.
`w`	Sample weights, default NULL.
`...`	Additional arguments to be passed to other inherited types.

Value

None. the model stored in "obj" will be updated based on "ss1" and "ss2".

References

Teh, Yee W., et al. "Sharing clusters among related groups: Hierarchical Dirichlet processes." Advances in neural information processing systems. 2005.

See Also

`CatHDP`,`posteriorDiscard.CatHDP`

`posterior.CatHDP2` *Update a "CatHDP2" object with sample sufficient statistics*

Description

For the model structure:

$$G|eta \sim DP(eta, U)$$

$$G_m|gamma \sim DP(gamma, G), m = 1 : M$$

$$pi_{mj}|G_m, alpha \sim DP(alpha, G_m), j = 1 : J_m$$

$$z|pi_{mj} \sim Categorical(pi_{mj})$$

$$k|z, G_m \sim Categorical(G_m),\ \text{if z is a sample from the base measure } G_m$$

$$u|k, G \sim Categorical(G),\ \text{if k is a sample from the base measure G}$$

where DP(eta,U) is a Dirichlet Process on positive integers, eta is the "concentration parameter", U is the "base measure" of this Dirichlet process, U is an uniform distribution on all positive integers. DP(gamma,G) is a Dirichlet Process on integers with concentration parameter gamma and base measure G. DP(alpha,G_m) is a Dirichlet Process on integers with concentration parameter alpha and base measure G_m. Categorical() is the Categorical distribution. See `dCategorical` for the definition of the Categorical distribution.
In the case of CatHDP2, u, z and k can only be positive integers.
Update the prior knowledge by adding the information of newly observed samples u, z and k. The model structure and prior parameters are stored in a "CatHDP2" object, the prior parameters in this object will be updated after running this function.

Usage

```
## S3 method for class 'CatHDP2'
posterior(obj, ss1, ss2, ss3, m, j, w = NULL, ...)
```

Arguments

`obj`	A "CatHDP2" object.
`ss1`	Sufficient statistics of u. In CatHDP2 case the sufficient statistic of sample u is u itself(if u is a integer vector with all positive values).
`ss2`	Sufficient statistics of k. In CatHDP2 case the sufficient statistic of sample k is k itself(if k is a integer vector with all positive values).
`ss3`	Sufficient statistics of z. In CatHDP2 case the sufficient statistic of sample z is z itself(if z is a integer vector with all positive values).
`m`	integer, group label.
`j`	integer, subgroup label.
`w`	Sample weights, default NULL.
`...`	Additional arguments to be passed to other inherited types.

Value

None. the model stored in "obj" will be updated based on "ss1" and "ss2".

References

Teh, Yee W., et al. "Sharing clusters among related groups: Hierarchical Dirichlet processes." Advances in neural information processing systems. 2005.

See Also

CatHDP2,posteriorDiscard.CatHDP2

`posterior.DP` *Update a "DP" object with sample sufficient statistics*

Description

For the model structure:

$$pi|alpha \sim DP(alpha, U)$$

$$z|pi \sim Categorical(pi)$$

$$theta_z|psi \sim H0(psi)$$

$$x|theta_z, z \sim F(theta_z)$$

where DP(alpha,U) is a Dirichlet Process on positive integers, alpha is the "concentration parameter" of the Dirichlet Process, U is the "base measure" of this Dirichlet process. The choice of F() and H0() can be described by an arbitrary "BasicBayesian" object such as "GaussianGaussian","GaussianInvWishart","GaussianNIW", "GaussianNIG", "CatDirichlet", and "CatDP". See `?BasicBayesian` for definition of "BasicBayesian" objects, and see for example `?GaussianGaussian` for specific "BasicBayesian" instances. As a summary, An "DP" object is simply a combination of a "CatDP" object (see `?CatDP`) and an object of any "BasicBayesian" type.
This function will update the prior knowledge by adding the information of newly observed samples x. The model structure and prior parameters are stored in a "DP" object, the prior parameters in this object will be updated after running this function.

Usage

```
## S3 method for class 'DP'
posterior(obj, ss = NULL, z, w = NULL, ...)
```

Arguments

obj	A "DP" object.
ss	Sufficient statistics of x of the "BasicBayesian" object, must be a list of sufficient statistics for each of the observations. Use sufficientStatistics(...,foreach=TRUE) to generate ss. See examples.
z	integer, the partition label of the parameter space where the observation x is drawn from.
w	Sample weights, default NULL.
...	Additional arguments to be passed to other inherited types.

Value

None. the model stored in "obj" will be updated based on "ss".

References

Teh, Yee W., et al. "Sharing clusters among related groups: Hierarchical Dirichlet processes." Advances in neural information processing systems. 2005.

See Also

DP,posteriorDiscard.DP,sufficientStatistics.DP

Examples

```
x <- rnorm(40)
z <- sample(1L:10L,size = 40,replace = TRUE)
obj <- DP()
ss <- sufficientStatistics(obj = obj,x=x,foreach = TRUE) #must use foreach=TRUE
for(i in 1L:length(z)) posterior(obj = obj,ss = ss[[i]],z=z[i])
obj
```

---

posterior.GaussianGaussian

Update a "GaussianGaussian" object with sample sufficient statistics

---

Description

For the model structure:

$$x \sim Gaussian(mu, Sigma)$$

$$mu \sim Gaussian(m, S)$$

Where Sigma is known. Gaussian() is the Gaussian distribution. See ?dGaussian for the definition of Gaussian distribution.
Update (m,S) by adding the information of newly observed samples x. The model structure and prior parameters are stored in a "GaussianGaussian" object, the prior parameters in this object will be updated after running this function.

Usage

```
## S3 method for class 'GaussianGaussian'
posterior(obj, ss, ...)
```

Arguments

obj	A "GaussianGaussian" object.
ss	Sufficient statistics of x. In Gaussian-Gaussian case the sufficient statistic of sample x is a object of type "ssGaussianMean", it can be generated by the function sufficientStatistics().
...	Additional arguments to be passed to other inherited types.

Value

None. the gamma stored in "obj" will be updated based on "ss".

References

Gelman, Andrew, et al. Bayesian data analysis. CRC press, 2013.

See Also

GaussianGaussian,posteriorDiscard.GaussianGaussian,sufficientStatistics.GaussianGaussian

Examples

```
obj <- GaussianGaussian(gamma=list(Sigma=matrix(c(2,1,1,2),2,2),m=c(0.2,0.5),S=diag(2)))
obj
x <- rGaussian(100,c(0,0),Sigma = matrix(c(2,1,1,2),2,2))
ss <- sufficientStatistics(obj=obj,x=x,foreach = FALSE)
posterior(obj = obj,ss = ss)
obj
```

---

posterior.GaussianInvWishart

Update a "GaussianInvWishart" object with sample sufficient statistics

---

Description

For the model structure:

$$x \sim Gaussian(mu, Sigma)$$

$$Sigma \sim InvWishart(v, S)$$

mu is known. Gaussian() is the Gaussian distribution. See ?dGaussian and ?dInvWishart for the definition of the distributions.
Update (v,S) by adding the information of newly observed samples x. The model structure and prior parameters are stored in a "GaussianInvWishart" object, the prior parameters in this object will be updated after running this function.

Usage

```
## S3 method for class 'GaussianInvWishart'
posterior(obj, ss, ...)
```

Arguments

obj	A "GaussianInvWishart" object.
ss	Sufficient statistics of x. In Gaussian and Inverse-Wishart case the sufficient statistic of sample x is a object of type "ssGaussianVar", it can be generated by the function sufficientStatistics().
...	Additional arguments to be passed to other inherited types.

Value

None. the gamma stored in "obj" will be updated based on "ss".

References

Gelman, Andrew, et al. Bayesian data analysis. CRC press, 2013.

MARolA, K. V., JT KBNT, and J. M. Bibly. Multivariate analysis. AcadeInic Press, Londres, 1979.

See Also

GaussianInvWishart,posteriorDiscard.GaussianInvWishart,sufficientStatistics.GaussianInvWishart

Examples

```
obj <- GaussianInvWishart(gamma=list(mu=c(-1.5,1.5),v=3,S=diag(2)))
x <- rGaussian(10,mu = c(-1.5,1.5),Sigma = matrix(c(0.1,0.03,0.03,0.1),2,2))
ss <- sufficientStatistics(obj=obj,x=x,foreach = FALSE)
obj
posterior(obj,ss = ss)
obj
```

---

posterior.GaussianNIG *Update a "GaussianNIG" object with sample sufficient statistics*

Description

For the model structure:

$$x \sim Gaussian(Xbeta, sigma^2)$$

$$sigma^2 \sim InvGamma(a, b)$$

$$beta \sim Gaussian(m, sigma^2 V)$$

Where X is a row vector, or a design matrix where each row is an obervation. InvGamma() is the Inverse-Gamma distribution, Gaussian() is the Gaussian distribution. See ?dInvGamma and

dGaussian for the definitions of these distribution.
The model structure and prior parameters are stored in a "GaussianNIG" object.
Update (m,V,a,b) by adding the information of newly observed samples (x,X). The model structure and prior parameters are stored in a "GaussianNIG" object, the prior parameters in this object will be updated after running this function.

Usage

```
## S3 method for class 'GaussianNIG'
posterior(obj, ss, ...)
```

Arguments

obj	A "GaussianNIG" object.
ss	Sufficient statistics of (x,X). In Gaussian-NIG case the sufficient statistic of sample (x,X) is a object of type "ssGaussianLinear", it can be generated by the function sufficientStatistics().
...	Additional arguments to be passed to other inherited types.

Value

None. the gamma stored in "obj" will be updated based on "ss".

References

Banerjee, Sudipto. "Bayesian Linear Model: Gory Details." Downloaded from http://www. biostat. umn. edu/~ph7440 (2008).

See Also

GaussianNIG,posteriorDiscard.GaussianNIG,sufficientStatistics.GaussianNIG

Examples

```
obj <- GaussianNIG(gamma=list(m=0,V=1,a=1,b=0))
X <- 1:20
x <- rnorm(20)+ X*0.3
ss <- sufficientStatistics(obj = obj,X=X,x=x)
posterior(obj = obj,ss = ss)
obj
```

`posterior.GaussianNIW` *Update a "GaussianNIW" object with sample sufficient statistics*

Description

For the model structure:

$$mu, Sigma|m, k, v, S \sim NIW(m, k, v, S)$$

$$x|mu, Sigma \sim Gaussian(mu, Sigma)$$

Where NIW() is the Normal-Inverse-Wishart distribution, Gaussian() is the Gaussian distribution. See `?dNIW` and `dGaussian` for the definitions of these distribution.
Update (m,k,v,S) by adding the information of newly observed samples x. The model structure and prior parameters are stored in a "GaussianNIW" object, the prior parameters in this object will be updated after running this function.

Usage

```
## S3 method for class 'GaussianNIW'
posterior(obj, ss, ...)
```

Arguments

`obj`	A "GaussianNIW" object.
`ss`	Sufficient statistics of x. In Gaussian-NIW case the sufficient statistic of sample x is a object of type "ssGaussian", it can be generated by the function sufficientStatistics().
`...`	Additional arguments to be passed to other inherited types.

Value

None. the gamma stored in "obj" will be updated based on "ss".

References

Murphy, Kevin P. "Conjugate Bayesian analysis of the Gaussian distribution." def 1.22 (2007): 16.

Gelman, Andrew, et al. "Bayesian Data Analysis Chapman & Hall." CRC Texts in Statistical Science (2004).

See Also

`GaussianNIW`,`posteriorDiscard.GaussianNIW`,`sufficientStatistics.GaussianNIW`

Examples

```
x <- rGaussian(1000,mu = c(1,1),Sigma = matrix(c(1,0.5,0.5,3),2,2))
w <- runif(1000)
obj <- GaussianNIW(gamma=list(m=c(0,0),k=1,v=2,S=diag(2)))
obj
ss <- sufficientStatistics_Weighted(obj = obj,x=x,w=w,foreach = TRUE)
for(i in 1L:length(ss)) posterior(obj = obj,ss = ss[[i]])
obj
```

`posterior.HDP` *Update a "HDP" object with sample sufficient statistics*

Description

For the model structure:

$$G|gamma \sim DP(gamma, U)$$

$$pi_j|G, alpha \sim DP(alpha, G), j = 1 : J$$

$$z|pi_j \sim Categorical(pi_j)$$

$$k|z, G \sim Categorical(G),$$ if z is a sample from the base measure G

$$theta_k|psi \sim H0(psi)$$

$$x|theta_k, k \sim F(theta_k)$$

where DP(gamma,U) is a Dirichlet Process on positive integers, gamma is the "concentration parameter", U is the "base measure" of this Dirichlet process, U is an uniform distribution on all positive integers. DP(alpha,G) is a Dirichlet Process on integers with concentration parameter alpha and base measure G. The choice of F() and H0() can be described by an arbitrary "BasicBayesian" object such as "GaussianGaussian","GaussianInvWishart","GaussianNIW", "GaussianNIG", "CatDirichlet", and "CatDP". See `?BasicBayesian` for definition of "BasicBayesian" objects, and see for example `?GaussianGaussian` for specific "BasicBayesian" instances. As a summary, An "HDP" object is simply a combination of a "CatHDP" object (see `?CatHDP`) and an object of any "BasicBayesian" type.
In the case of HDP, z and k can only be positive integers.
This function will update the prior knowledge by adding the information of newly observed samples x, z and k. The model structure and prior parameters are stored in a "HDP" object, the prior parameters in this object will be updated after running this function.

Usage

```
## S3 method for class 'HDP'
posterior(obj, ss = NULL, ss1, ss2 = NULL, j, w = NULL, ...)
```

Arguments

obj	A "HDP" object.
ss	Sufficient statistics of x of the "BasicBayesian" object, must be a list of sufficient statistics for each of the observations. Use sufficientStatistics(...,foreach=TRUE) to generate ss.
ss1	Sufficient statistics of k. In HDP case the sufficient statistic of sample k is k itself(if k is a integer vector with all positive values).
ss2	Sufficient statistics of z. In HDP case the sufficient statistic of sample z is z itself(if z is a integer vector with all positive values).
j	integer, group label.
w	Sample weights, default NULL.
...	Additional arguments to be passed to other inherited types.

Value

None. the model stored in "obj" will be updated based on "ss", "ss1" and "ss2".

References

Teh, Yee W., et al. "Sharing clusters among related groups: Hierarchical Dirichlet processes." Advances in neural information processing systems. 2005.

See Also

HDP,posteriorDiscard.HDP,sufficientStatistics.HDP

---

posterior.HDP2 *Update a "HDP2" object with sample sufficient statistics*

Description

For the model structure:

$$G|eta \sim DP(eta, U)$$

$$G_m|gamma, G \sim DP(gamma, G), m = 1 : M$$

$$pi_{mj}|G_m, alpha \sim DP(alpha, G_m), j = 1 : J_m$$

$$z|pi_{mj} \sim Categorical(pi_{mj})$$

$k|z, G_m \sim Categorical(G_m)$, if z is a sample from the base measure G_{mj}

$u|k, G \sim Categorical(G)$, if k is a sample from the base measure G

$$theta_u|psi \sim H0(psi)$$

$$x|theta_u, u \sim F(theta_u)$$

where DP(eta,U) is a Dirichlet Process on positive integers, eta is the "concentration parameter", U is the "base measure" of this Dirichlet process, U is an uniform distribution on all positive integers. DP(gamma,G) is a Dirichlet Process on integers with concentration parameter gamma and base measure G. DP(alpha,G_m) is a Dirichlet Process on integers with concentration parameter alpha and base measure G_m. The choice of F() and H0() can be described by an arbitrary "BasicBayesian" object such as "GaussianGaussian","GaussianInvWishart","GaussianNIW", "GaussianNIG", "CatDirichlet", and "CatDP". See `?BasicBayesian` for definition of "BasicBayesian" objects, and see for example `?GaussianGaussian` for specific "BasicBayesian" instances. As a summary, An "HDP2" object is simply a combination of a "CatHDP2" object (see `?CatHDP2`) and an object of any "BasicBayesian" type.
In the case of HDP2, u, z and k can only be positive integers.
This function will update the prior knowledge by adding the information of newly observed samples x, z and k. The model structure and prior parameters are stored in a "HDP2" object, the prior parameters in this object will be updated after running this function.

Usage

```
## S3 method for class 'HDP2'
posterior(obj, ss = NULL, ss1, ss2, ss3, m, j, w = NULL, ...)
```

Arguments

`obj`	A "HDP2" object.
`ss`	Sufficient statistics of x of the "BasicBayesian" object, must be a list of sufficient statistics for each of the observations. Use sufficientStatistics(...,foreach=TRUE) to generate ss.
`ss1`	Sufficient statistics of u. In HDP2 case the sufficient statistic of sample u is u itself(if u is a integer vector with all positive values).
`ss2`	Sufficient statistics of k. In HDP2 case the sufficient statistic of sample k is k itself(if k is a integer vector with all positive values).
`ss3`	Sufficient statistics of z. In HDP2 case the sufficient statistic of sample z is z itself(if z is a integer vector with all positive values).
`m`	integer, group label.
`j`	integer, subgroup label.
`w`	Sample weights, default NULL.
`...`	Additional arguments to be passed to other inherited types.

Value

None. the model stored in "obj" will be updated based on "ss", "ss1", "ss2"and "ss3".

References

Teh, Yee W., et al. "Sharing clusters among related groups: Hierarchical Dirichlet processes." Advances in neural information processing systems. 2005.

See Also

HDP2,posteriorDiscard.HDP2,sufficientStatistics.HDP2

`posterior.LinearGaussianGaussian` *Update a "LinearGaussianGaussian" object with sample sufficient statistics*

Description

For following model structure:

$$x \sim Gaussian(Az + b, Sigma)$$

$$z \sim Gaussian(m, S)$$

Where Sigma is known. A is a $dimxxdimz$ matrix, x is a $dimxx1$ random vector, z is a $dimzx1$ random vector, b is a $dimmx1$ vector. Gaussian() is the Gaussian distribution. See `?dGaussian` for the definition of Gaussian distribution.
Update (m,S) by adding the information of newly observed samples x. The model structure and prior parameters are stored in a "LinearGaussianGaussian" object, the prior parameters in this object will be updated after running this function.

Usage

```
## S3 method for class 'LinearGaussianGaussian'
posterior(obj, ss, ...)
```

Arguments

`obj`	A "LinearGaussianGaussian" object.
`ss`	Sufficient statistics of x. In LinearGaussian-Gaussian case the sufficient statistic of sample x is a object of type "ssLinearGaussianGaussian", it can be generated by the function sufficientStatistics().
`...`	Additional arguments to be passed to other inherited types.

Value

None. the gamma stored in "obj" will be updated based on "ss".

References

Murphy, Kevin P. Machine learning: a probabilistic perspective. MIT press, 2012.

See Also

LinearGaussianGaussian,posteriorDiscard.LinearGaussianGaussian,sufficientStatistics.LinearGaussianGa

Examples

```
obj <- LinearGaussianGaussian(gamma=list(Sigma=matrix(c(2,1,1,2),2,2),
                              m=c(0.2,0.5,0.6),S=diag(3)))
x <- rGaussian(100,mu = runif(2),Sigma = diag(2))
A <- matrix(runif(6),2,3)
b <- runif(2)
ss <- sufficientStatistics(obj,x=x,A=A,b=b)
obj
posterior(obj=obj,ss=ss)
obj
```

`posteriorDiscard` *update the prior distribution with sufficient statistics*

Description

Contrary to posterior(), posteriorDiscard() a generic function that will update the prior distribution of a "BayesianBrick" object by removing the information provided by the observation's sufficient statistics. i.e. for the model structure:

$$theta|gamma \sim H(gamma)$$

$$x|theta \sim F(theta)$$

update gamma to gamma_posterior by removing the information of x from gamma. For a given sample set x or it's sufficient statistics ss, and a Bayesian bricks object obj, `posteriorDiscard()` will update the posterior parameters in obj for different model structures:

class(obj)="LinearGaussianGaussian":

$$x \sim Gaussian(Az + b, Sigma)$$

$$z \sim Gaussian(m, S)$$

`posteriorDiscard()` will update m and S in obj. See ?`posteriorDiscard.LinearGaussianGaussian` for details.

class(obj)="GaussianGaussian": Where

$$x \sim Gaussian(mu, Sigma)$$

$$mu \sim Gaussian(m, S)$$

Sigma is known. `posteriorDiscard()` will update m and S in obj. See ?`posteriorDiscard.GaussianGaussian` for details.

class(obj)="GaussianInvWishart": Where

$$x \sim Gaussian(mu, Sigma)$$

$$Sigma \sim InvWishart(v, S)$$

mu is known. `posteriorDiscard()` will update v and S in obj. See ?`posteriorDiscard.GaussianInvWishart` for details.

class(obj)="GaussianNIW": Where

$$x \sim Gaussian(mu, Sigma)$$

$$Sigma \sim InvWishart(v, S)$$

$$mu \sim Gaussian(m, Sigma/k)$$

`posteriorDiscard()` will update m, k, v and S in obj. See `?posteriorDiscard.GaussianNIW` for details.

class(obj)="GaussianNIG": Where

$$x \sim Gaussian(Xbeta, sigma^2)$$

$$sigma^2 \sim InvGamma(a, b)$$

$$beta \sim Gaussian(m, sigma^2 V)$$

`posteriorDiscard()` will update m, V, a and b in obj. See `?posteriorDiscard.GaussianNIG` for details.

class(obj)="CatDirichlet": Where

$$x \sim Categorical(pi)$$

$$pi \sim Dirichlet(alpha)$$

`posteriorDiscard()` will update alpha in obj. See `?posteriorDiscard.CatDirichlet` for details.

class(obj)="CatDP": Where

$$x \sim Categorical(pi)$$

$$pi \sim DirichletProcess(alpha)$$

`posteriorDiscard()` will update alpha in obj. See `?posteriorDiscard.CatDP` for details.

class(obj)="DP": Where

$$pi|alpha \sim DP(alpha, U)$$

$$z|pi \sim Categorical(pi)$$

$$theta_z|psi \sim H0(psi)$$

$$x|theta_z, z \sim F(theta_z)$$

`posteriorDiscard()` will update alpha and psi in obj. See `?posteriorDiscard.DP` for details.

class(obj)="HDP": Where

$$G|gamma \sim DP(gamma, U)$$

$$pi_j|G, alpha \sim DP(alpha, G), j = 1 : J$$

$$z|pi_j \sim Categorical(pi_j)$$

$$k|z, G \sim Categorical(G),$$ if z is a sample from the base measure G

$$theta_k|psi \sim H0(psi)$$

`posteriorDiscard()` will update gamma, alpha and psi in obj. See `?posteriorDiscard.HDP` for details.

class(obj)="HDP2": Where

$$G|eta \sim DP(eta, U)$$

$$G_m|gamma, G \sim DP(gamma, G), m = 1 : M$$

$$pi_{mj}|G_m, alpha \sim DP(alpha, G_m), j = 1 : J_m$$

$$z|pi_{mj} \sim Categorical(pi_{mj})$$

$$k|z, G_m \sim Categorical(G_m), \text{ if z is a sample from the base measure } G_m$$

$$u|k, G \sim Categorical(G), \text{ if k is a sample from the base measure } G$$

$$theta_u|psi \sim H0(psi)$$

$$x|theta_u, u \sim F(theta_u)$$

`posteriorDiscard()` will update eta, gamma, alpha and psi in obj. See `?posteriorDiscard.HDP2` for details.

Usage

```
posteriorDiscard(obj, ...)
```

Arguments

`obj`	A "BayesianBrick" object used to select a method.
`...`	further arguments passed to or from other methods.

Value

None, or an error message if the update fails.

See Also

`posteriorDiscard.LinearGaussianGaussian` for Linear Gaussian and Gaussian conjugate structure, `posteriorDiscard.GaussianGaussian` for Gaussian-Gaussian conjugate structure, `posteriorDiscard.GaussianIn` for Gaussian-Inverse-Wishart conjugate structure, `posteriorDiscard.GaussianNIW` for Gaussian-NIW conjugate structure, `posteriorDiscard.GaussianNIG` for Gaussian-NIG conjugate structure, `posteriorDiscard.CatDirichlet` for Categorical-Dirichlet conjugate structure, `posteriorDiscard.CatDP` for Categorical-DP conjugate structure ...

`posteriorDiscard.CatDirichlet` *Update a "CatDirichlet" object with sample sufficient statistics*

Description

Contrary to posterior(), this function will update alpha by removing the information of observed samples x for the model structure:

$$pi|alpha \sim Dir(alpha)$$

$$x|pi \sim Categorical(pi)$$

Where Dir() is the Dirichlet distribution, Categorical() is the Categorical distribution. See `?dDir` and `dCategorical` for the definitions of these distribution.
The model structure and prior parameters are stored in a "CatDirichlet" object, the prior parameters in this object will be updated after running this function.

Usage

```
## S3 method for class 'CatDirichlet'
posteriorDiscard(obj, ss, w = NULL, ...)
```

Arguments

`obj`	A "CatDirichlet" object.
`ss`	Sufficient statistics of x. In Categorical-Dirichlet case the sufficient statistic of sample x can be either x itself, of an "ssCat" object generated by the function sufficientStatistics.CatDirichlet().
`w`	Sample weights,default NULL.
`...`	Additional arguments to be passed to other inherited types.

Value

None. the prior parameters stored in "obj" will be updated with the information in "ss".

References

Murphy, Kevin P. Machine learning: a probabilistic perspective. MIT press, 2012.

See Also

`CatDirichlet`,`posterior.CatDirichlet`

Examples

```
obj <- CatDirichlet(gamma=list(alpha=rep(1,26),uniqueLabels = letters))
x <- sample(letters,size = 20,replace = TRUE)
w <- runif(20)
posterior(obj=obj,ss=x)
obj
posteriorDiscard(obj=obj,ss=x)
obj
## weighted sample
posterior(obj=obj,ss=x,w=w)
obj
posteriorDiscard(obj=obj,ss=x,w=w)
obj
```

`posteriorDiscard.CatDP` *Update a "CatDP" object with sample sufficient statistics*

Description

For the model structure:

$$pi|alpha \sim DP(alpha, U)$$

$$x|pi \sim Categorical(pi)$$

where DP(alpha,U) is a Dirichlet Process on positive integers, alpha is the "concentration parameter" of the Dirichlet Process, U is the "base measure" of this Dirichlet process, it is an uniform distribution on all positive integers.Categorical() is the Categorical distribution. See `dCategorical` for the definition of the Categorical distribution.
In the case of CatDP, x can only be positive integers.
Contrary to posterior(), this function will update the prior knowledge by removing the information of observed samples x. The model structure and prior parameters are stored in a "CatDP" object, the prior parameters in this object will be updated after running this function.

Usage

```
## S3 method for class 'CatDP'
posteriorDiscard(obj, ss, w = NULL, ...)
```

Arguments

`obj`	A "CatDP" object.
`ss`	Sufficient statistics of x. In Categorical-DP case the sufficient statistic of sample x can either be an object of type "ssCatDP" generated by sufficientStatistics(), or x itself(if x is a integer vector with all positive values).
`w`	Sample weights,default NULL.
`...`	Additional arguments to be passed to other inherited types.

Value

None. the model stored in "obj" will be updated with the information in "ss".

References

Teh, Yee W., et al. "Sharing clusters among related groups: Hierarchical Dirichlet processes." Advances in neural information processing systems. 2005.

See Also

CatDP,posterior.CatDP

Examples

```
## generate some integer samples
x <- sample(1L:10L,size = 40,replace = TRUE)
obj <- CatDP()
ss <- sufficientStatistics(obj=obj,x=x)
posterior(obj = obj,ss = ss)
obj2 <- CatDP(objCopy = obj)            #create obj2 contains the sames info as obj
obj3 <- CatDP(objCopy = obj)            #create obj3 contains the sames info as obj
## discard by samples
posteriorDiscard(obj = obj,ss = x)
## or discard by samples, one by one
for(xx in x) posteriorDiscard(obj = obj2,ss = xx)
## or discard by sufficient statistics
posteriorDiscard(obj = obj3,ss = ss)
## obj, obj2 and obj3 should be the same:
obj
obj2
obj3
```

---

posteriorDiscard.CatHDP

Update a "CatHDP" object with sample sufficient statistics

Description

For the model structure:

$$G|gamma \sim DP(gamma, U)$$

$$pi_j|G, alpha \sim DP(alpha, G), j = 1 : J$$

$$z|pi_j \sim Categorical(pi_j)$$

$$k|z, G \sim Categorical(G),$$ if z is a sample from the base measure G

where DP(gamma,U) is a Dirichlet Process on positive integers, gamma is the "concentration parameter", U is the "base measure" of this Dirichlet process, U is an uniform distribution on all

positive integers. DP(alpha,G) is a Dirichlet Process on integers with concentration parameter alpha and base measure G. Categorical() is the Categorical distribution. See `dCategorical` for the definition of the Categorical distribution.
In the case of CatHDP, z and k can only be positive integers.
Contrary to posterior(), this function will update the prior knowledge by removing the information of observed samples z and k. The model structure and prior parameters are stored in a "CatDP" object, the prior parameters in this object will be updated after running this function.

Usage

```
## S3 method for class 'CatHDP'
posteriorDiscard(obj, ss1, ss2, j, w = NULL, ...)
```

Arguments

`obj`	A "CatHDP" object.
`ss1`	Sufficient statistics of k. In CatHDP case the sufficient statistic of sample k is k itself(if k is a integer vector with all positive values).
`ss2`	Sufficient statistics of z. In CatHDP case the sufficient statistic of sample z is z itself(if z is a integer vector with all positive values).
`j`	integer, group label.
`w`	Sample weights, default NULL.
`...`	Additional arguments to be passed to other inherited types.

Value

None. the model stored in "obj" will be updated based on "ss1" and "ss2".

References

Teh, Yee W., et al. "Sharing clusters among related groups: Hierarchical Dirichlet processes." Advances in neural information processing systems. 2005.

See Also

`CatHDP`,`posteriorDiscard.CatHDP`

`posteriorDiscard.CatHDP2`

Update a "CatHDP2" object with sample sufficient statistics

Description

For the model structure:

$$G|eta \sim DP(eta, U)$$

$$G_m|gamma \sim DP(gamma, G), m = 1 : M$$

$$pi_{mj}|G_m, alpha \sim DP(alpha, G_m), j = 1 : J_m$$

$$z|pi_{mj} \sim Categorical(pi_{mj})$$

$$k|z, G_m \sim Categorical(G_m), \text{ if z is a sample from the base measure } G_m$$

$$u|k, G \sim Categorical(G), \text{ if k is a sample from the base measure G}$$

where DP(eta,U) is a Dirichlet Process on positive integers, eta is the "concentration parameter", U is the "base measure" of this Dirichlet process, U is an uniform distribution on all positive integers. DP(gamma,G) is a Dirichlet Process on integers with concentration parameter gamma and base measure G. DP(alpha,G_m) is a Dirichlet Process on integers with concentration parameter alpha and base measure G_m. Categorical() is the Categorical distribution. See `dCategorical` for the definition of the Categorical distribution.
In the case of CatHDP2, u, z and k can only be positive integers.
Contrary to posterior(), this function will update the prior knowledge by removing the information of observed samples u, z and k. The model structure and prior parameters are stored in a "CatDP" object, the prior parameters in this object will be updated after running this function.

Usage

```
## S3 method for class 'CatHDP2'
posteriorDiscard(obj, ss1, ss2, ss3, m, j, w = NULL, ...)
```

Arguments

`obj`	A "CatHDP2" object.
`ss1`	Sufficient statistics of u. In CatHDP2 case the sufficient statistic of sample u is u itself(if u is a integer vector with all positive values).
`ss2`	Sufficient statistics of k. In CatHDP2 case the sufficient statistic of sample k is k itself(if k is a integer vector with all positive values).
`ss3`	Sufficient statistics of z. In CatHDP2 case the sufficient statistic of sample z is z itself(if z is a integer vector with all positive values).
`m`	integer, group label.
`j`	integer, subgroup label.
`w`	Sample weights, default NULL.
`...`	Additional arguments to be passed to other inherited types.

Value

None. the model stored in "obj" will be updated based on "ss1" and "ss2".

References

Teh, Yee W., et al. "Sharing clusters among related groups: Hierarchical Dirichlet processes." Advances in neural information processing systems. 2005.

See Also

CatHDP2,posteriorDiscard.CatHDP2

`posteriorDiscard.DP` *Update a "DP" object with sample sufficient statistics*

Description

For the model structure:

$$pi|alpha \sim DP(alpha, U)$$
$$z|pi \sim Categorical(pi)$$
$$theta_z|psi \sim H0(psi)$$
$$x|theta_z, z \sim F(theta_z)$$

where DP(alpha,U) is a Dirichlet Process on positive integers, alpha is the "concentration parameter" of the Dirichlet Process, U is the "base measure" of this Dirichlet process. The choice of F() and H0() can be described by an arbitrary "BasicBayesian" object such as "GaussianGaussian","GaussianInvWishart","GaussianNIW", "GaussianNIG", "CatDirichlet", and "CatDP". See `?BasicBayesian` for definition of "BasicBayesian" objects, and see for example `?GaussianGaussian` for specific "BasicBayesian" instances. As a summary, An "DP" object is simply a combination of a "CatDP" object (see `?CatDP`) and an object of any "BasicBayesian" type.
Contrary to posterior(), this function will update the prior knowledge by removing the information of observed samples x. The model structure and prior parameters are stored in a "CatDP" object, the prior parameters in this object will be updated after running this function.

Usage

```
## S3 method for class 'DP'
posteriorDiscard(obj, ss = NULL, z, w = NULL, ...)
```

Arguments

`obj`	A "DP" object.
`ss`	Sufficient statistics of x of the "BasicBayesian" object, must be a list of sufficient statistics for each of the observations. Use sufficientStatistics(...,foreach=TRUE) to generate ss. See examples.
`z`	integer, the partition label of the parameter space where the observation x is drawn from.
`w`	Sample weights, default NULL.
`...`	Additional arguments to be passed to other inherited types.

Value

None. the model stored in "obj" will be updated based on "ss".

References

Teh, Yee W., et al. "Sharing clusters among related groups: Hierarchical Dirichlet processes." Advances in neural information processing systems. 2005.

See Also

DP,posteriorDiscard.DP,sufficientStatistics.DP

Examples

```
x <- rnorm(40)
z <- sample(1L:10L,size = 40,replace = TRUE)
obj <- DP()
ss <- sufficientStatistics(obj = obj,x=x,foreach = TRUE) #must use foreach=TRUE
for(i in 1L:length(z)) posterior(obj = obj,ss = ss[[i]],z=z[i])
obj
for(i in 1L:length(z)) posteriorDiscard(obj = obj,ss = ss[[i]],z=z[i])
obj
```

---

posteriorDiscard.GaussianGaussian

Update a "GaussianGaussian" object with sample sufficient statistics

---

Description

For the model structure:

$$x \sim Gaussian(mu, Sigma)$$

$$mu \sim Gaussian(m, S)$$

Where Sigma is known. Gaussian() is the Gaussian distribution. See ?dGaussian for the definition of Gaussian distribution.
Contrary to posterior(), this function will update (m,S) by removing the information of observed samples x. The model structure and prior parameters are stored in a "GaussianGaussian" object, the prior parameters in this object will be updated after running this function.

Usage

```
## S3 method for class 'GaussianGaussian'
posteriorDiscard(obj, ss, w = NULL, ...)
```

Arguments

obj	A "GaussianGaussian" object.
ss	Sufficient statistics of x. In Gaussian-Gaussian case the sufficient statistic of sample x is a object of type "ssGaussianMean", it can be generated by the function sufficientStatistics().
w	Sample weights, default NULL.
...	Additional arguments to be passed to other inherited types.

Value

None. the gamma stored in "obj" will be updated based on "ss".

References

Gelman, Andrew, et al. Bayesian data analysis. CRC press, 2013.

See Also

GaussianGaussian,posterior.GaussianGaussian,sufficientStatistics.GaussianGaussian

Examples

```
obj <- GaussianGaussian(gamma=list(Sigma=matrix(c(2,1,1,2),2,2),m=c(0.2,0.5),S=diag(2)))
obj
x <- rGaussian(100,c(0,0),Sigma = matrix(c(2,1,1,2),2,2))
ss <- sufficientStatistics(obj=obj,x=x,foreach = FALSE)
## update prior into posterior
posterior(obj = obj,ss = ss)
obj
## remove the information, back to prior
posteriorDiscard(obj = obj,ss = ss)
obj
```

---

posteriorDiscard.GaussianInvWishart

Update a "GaussianInvWishart" object with sample sufficient statistics

---

Description

For the model structure:

$$x \sim Gaussian(mu, Sigma)$$

$$Sigma \sim InvWishart(v, S)$$

mu is known. Gaussian() is the Gaussian distribution. See ?dGaussian and ?dInvWishart for the definition of the distributions.
Contrary to posterior(), this function will update (v,S) by removing the information of observed samples x. The model structure and prior parameters are stored in a "GaussianInvWishart" object, the prior parameters in this object will be updated after running this function.

Usage

```
## S3 method for class 'GaussianInvWishart'
posteriorDiscard(obj, ss, w = NULL, ...)
```

Arguments

obj	A "GaussianInvWishart" object.
ss	Sufficient statistics of x. In Gaussian and Inverse-Wishart case the sufficient statistic of sample x is a object of type "ssGaussianVar", it can be generated by the function sufficientStatistics().
w	Sample weights, default NULL.
...	Additional arguments to be passed to other inherited types.

Value

None. the gamma stored in "obj" will be updated based on "ss".

References

Gelman, Andrew, et al. Bayesian data analysis. CRC press, 2013.

MARolA, K. V., JT KBNT, and J. M. Bibly. Multivariate analysis. AcadeInic Press, Londres, 1979.

See Also

GaussianInvWishart,posterior.GaussianInvWishart,sufficientStatistics.GaussianInvWishart

Examples

```
obj <- GaussianInvWishart(gamma=list(mu=c(-1.5,1.5),v=3,S=diag(2)))
x <- rGaussian(100,mu = c(-1.5,1.5),Sigma = matrix(c(0.1,0.03,0.03,0.1),2,2))
ss <- sufficientStatistics(obj=obj,x=x,foreach = FALSE)
obj
posterior(obj=obj,ss = ss)
obj
posteriorDiscard(obj=obj,ss=ss)
obj
```

---

posteriorDiscard.GaussianNIG

Update a "GaussianNIG" object with sample sufficient statistics

---

Description

For the model structure:

$$x \sim Gaussian(Xbeta, sigma^2)$$

$$sigma^2 \sim InvGamma(a, b)$$

$$beta \sim Gaussian(m, sigma^2 V)$$

Where X is a row vector, or a design matrix where each row is an obervation. InvGamma() is the Inverse-Gamma distribution, Gaussian() is the Gaussian distribution. See `?dInvGamma` and `dGaussian` for the definitions of these distribution.
The model structure and prior parameters are stored in a "GaussianNIG" object.
Contrary to posterior(), this function will update (m,V,a,b) by removing the information of observed samples (x,X). The model structure and prior parameters are stored in a "GaussianNIG" object, the prior parameters in this object will be updated after running this function.

Usage

```
## S3 method for class 'GaussianNIG'
posteriorDiscard(obj, ss, w = NULL, ...)
```

Arguments

`obj`	A "GaussianNIG" object.
`ss`	Sufficient statistics of (x,X). In Gaussian-NIG case the sufficient statistic of sample (x,X) is a object of type "ssGaussianLinear", it can be generated by the function sufficientStatistics().
`w`	Sample weights,default NULL.
`...`	Additional arguments to be passed to other inherited types.

Value

None. the gamma stored in "obj" will be updated with the information in "ss".

References

Banerjee, Sudipto. "Bayesian Linear Model: Gory Details." Downloaded from http://www. biostat. umn. edu/~ph7440 (2008).

See Also

`GaussianNIG`,`posterior.GaussianNIG`

Examples

```
X <- 1:20
x <- rnorm(20)+ X*0.3
## add information, then remove them
obj <- GaussianNIG(gamma=list(m=0,V=1,a=1,b=1))
ss <- sufficientStatistics(obj = obj,X=X,x=x)
```

```
posterior(obj = obj,ss = ss)
obj
posteriorDiscard(obj=obj,ss=ss)
obj
## or, add information, then remove them one by one
obj <- GaussianNIG(gamma=list(m=0,V=1,a=1,b=1))
ssEach <- sufficientStatistics(obj = obj,X=X,x=x,foreach = TRUE)
for(sss in ssEach) posterior(obj = obj,ss = sss)
obj
for(sss in ssEach) posteriorDiscard(obj = obj,ss = sss)
obj
```

`posteriorDiscard.GaussianNIW` *Update a "GaussianNIW" object with sample sufficient statistics*

Description

For the model structure:

$$mu, Sigma|m, k, v, S \sim NIW(m, k, v, S)$$

$$x|mu, Sigma \sim Gaussian(mu, Sigma)$$

Where NIW() is the Normal-Inverse-Wishart distribution, Gaussian() is the Gaussian distribution. See `?dNIW` and `dGaussian` for the definitions of these distribution.
Contrary to posterior(), this function will update (m,k,v,S) by removing the information of observed samples x. The model structure and prior parameters are stored in a "GaussianNIW" object, the prior parameters in this object will be updated after running this function.

Usage

```
## S3 method for class 'GaussianNIW'
posteriorDiscard(obj, ss, w = NULL, ...)
```

Arguments

`obj`	A "GaussianNIW" object.
`ss`	Sufficient statistics of x. In Gaussian-NIW case the sufficient statistic of sample x is a object of type "ssGaussian", it can be generated by the function sufficientStatistics().
`w`	Sample weights,default NULL.
`...`	Additional arguments to be passed to other inherited types.

Value

None. the gamma stored in "obj" will be updated with the information in "ss".

References

Murphy, Kevin P. "Conjugate Bayesian analysis of the Gaussian distribution." def 1.22 (2007): 16.

Gelman, Andrew, et al. "Bayesian Data Analysis Chapman & Hall." CRC Texts in Statistical Science (2004).

See Also

GaussianNIW,posterior.GaussianNIW

Examples

```
## generate some random Gaussian samples
x <- rGaussian(1000,mu = c(1,1),Sigma = matrix(c(1,0.5,0.5,3),2,2))
w <- runif(1000)
## add information to 'obj' then remove them one by one
obj <- GaussianNIW(gamma=list(m=c(0.2,3),k=1,v=2,S=diag(2)))
ss <- sufficientStatistics_Weighted(obj = obj,x=x,w=w,foreach = TRUE)
for(i in 1L:length(ss)) posterior(obj = obj,ss=ss[[i]])
obj
for(i in 1L:length(ss)) posteriorDiscard(obj = obj,ss=ss[[i]])
obj
## add information to 'obj' then remove them as a whole
obj <- GaussianNIW(gamma=list(m=c(0.2,3),k=1,v=2,S=diag(2)))
ssAll <- sufficientStatistics_Weighted(obj = obj,x=x,w=w,foreach = FALSE)
posterior(obj = obj,ss = ssAll)
obj
posteriorDiscard(obj = obj,ss = ssAll)
obj
```

---

posteriorDiscard.HDP *Update a "HDP" object with sample sufficient statistics*

Description

For the model structure:

$$G|gamma \sim DP(gamma, U)$$

$$pi_j|G, alpha \sim DP(alpha, G), j = 1:J$$

$$z|pi_j \sim Categorical(pi_j)$$

$$k|z, G \sim Categorical(G),$$ if z is a sample from the base measure G

$$theta_k|psi \sim H0(psi)$$

$$x|theta_k, k \sim F(theta_k)$$

where DP(gamma,U) is a Dirichlet Process on positive integers, gamma is the "concentration parameter", U is the "base measure" of this Dirichlet process, U is an uniform distribution on all positive integers. DP(alpha,G) is a Dirichlet Process on integers with concentration parameter alpha and base measure G. The choice of F() and H0() can be described by an arbitrary "BasicBayesian" object

such as "GaussianGaussian","GaussianInvWishart","GaussianNIW", "GaussianNIG", "CatDirichlet", and "CatDP". See ?BasicBayesian for definition of "BasicBayesian" objects, and see for example ?GaussianGaussian for specific "BasicBayesian" instances. As a summary, An "HDP" object is simply a combination of a "CatHDP" object (see ?CatHDP) and an object of any "BasicBayesian" type.
In the case of HDP, z and k can only be positive integers.
Contrary to posterior(), this function will update the prior knowledge by removing the information of observed samples x. The model structure and prior parameters are stored in a "CatDP" object, the prior parameters in this object will be updated after running this function.

Usage

```
## S3 method for class 'HDP'
posteriorDiscard(obj, ss = NULL, ss1, ss2 = NULL, j, w = NULL, ...)
```

Arguments

obj	A "HDP" object.
ss	Sufficient statistics of x of the "BasicBayesian" object, must be a list of sufficient statistics for each of the observations. Use sufficientStatistics(...,foreach=TRUE) to generate ss.
ss1	Sufficient statistics of k. In HDP case the sufficient statistic of sample k is k itself(if k is a integer vector with all positive values).
ss2	Sufficient statistics of z. In HDP case the sufficient statistic of sample z is z itself(if z is a integer vector with all positive values).
j	integer, group label.
w	Sample weights, default NULL.
...	Additional arguments to be passed to other inherited types.

Value

None. the model stored in "obj" will be updated based on "ss1" and "ss2".

References

Teh, Yee W., et al. "Sharing clusters among related groups: Hierarchical Dirichlet processes." Advances in neural information processing systems. 2005.

See Also

HDP,posteriorDiscard.HDP,sufficientStatistics.HDP

posteriorDiscard.HDP2 *Update a "HDP2" object with sample sufficient statistics*

Description

For the model structure:

$$G|eta \sim DP(eta, U)$$

$$G_m|gamma, G \sim DP(gamma, G), m = 1 : M$$

$$pi_{mj}|G_m, alpha \sim DP(alpha, G_m), j = 1 : J_m$$

$$z|pi_{mj} \sim Categorical(pi_{mj})$$

$$k|z, G_m \sim Categorical(G_m), \text{ if z is a sample from the base measure } G_{mj}$$

$$u|k, G \sim Categorical(G), \text{ if k is a sample from the base measure G}$$

$$theta_u|psi \sim H0(psi)$$

$$x|theta_u, u \sim F(theta_u)$$

where DP(eta,U) is a Dirichlet Process on positive integers, eta is the "concentration parameter", U is the "base measure" of this Dirichlet process, U is an uniform distribution on all positive integers. DP(gamma,G) is a Dirichlet Process on integers with concentration parameter gamma and base measure G. DP(alpha,G_m) is a Dirichlet Process on integers with concentration parameter alpha and base measure G_m. The choice of F() and H0() can be described by an arbitrary "BasicBayesian" object such as "GaussianGaussian","GaussianInvWishart","GaussianNIW", "GaussianNIG", "CatDirichlet", and "CatDP". See `?BasicBayesian` for definition of "BasicBayesian" objects, and see for example `?GaussianGaussian` for specific "BasicBayesian" instances. As a summary, An "HDP2" object is simply a combination of a "CatHDP2" object (see `?CatHDP2`) and an object of any "BasicBayesian" type.
In the case of HDP2, u, z and k can only be positive integers.
Contrary to posterior(), this function will update the prior knowledge by removing the information of observed samples x. The model structure and prior parameters are stored in a "CatDP" object, the prior parameters in this object will be updated after running this function.

Usage

```
## S3 method for class 'HDP2'
posteriorDiscard(obj, ss = NULL, ss1, ss2, ss3, m, j, w = NULL, ...)
```

Arguments

`obj`	A "HDP2" object.
`ss`	Sufficient statistics of x of the "BasicBayesian" object, must be a list of sufficient statistics for each of the observations. Use sufficientStatistics(...,foreach=TRUE) to generate ss.
`ss1`	Sufficient statistics of u. In HDP2 case the sufficient statistic of sample u is u itself(if u is a integer vector with all positive values).

ss2	Sufficient statistics of k. In HDP2 case the sufficient statistic of sample k is k itself(if k is a integer vector with all positive values).
ss3	Sufficient statistics of z. In HDP2 case the sufficient statistic of sample z is z itself(if z is a integer vector with all positive values).
m	integer, group label.
j	integer, subgroup label.
w	Sample weights, default NULL.
...	Additional arguments to be passed to other inherited types.

Value

None. the model stored in "obj" will be updated based on "ss", "ss1", "ss2"and "ss3".

References

Teh, Yee W., et al. "Sharing clusters among related groups: Hierarchical Dirichlet processes." Advances in neural information processing systems. 2005.

See Also

HDP2,posteriorDiscard.HDP2,sufficientStatistics.HDP2

---

posteriorDiscard.LinearGaussianGaussian

Update a "LinearGaussianGaussian" object with sample sufficient statistics

---

Description

For following model structure:

$$x \sim Gaussian(Az + b, Sigma)$$

$$z \sim Gaussian(m, S)$$

Where Sigma is known. A is a $dimxxdimz$ matrix, x is a $dimxx1$ random vector, z is a $dimzx1$ random vector, b is a $dimmx1$ vector. Gaussian() is the Gaussian distribution. See ?dGaussian for the definition of Gaussian distribution.
Contrary to posterior(), this function will update (m,S) by removing the information of observed samples x. The model structure and prior parameters are stored in a "LinearGaussianGaussian" object, the prior parameters in this object will be updated after running this function.

Usage

```
## S3 method for class 'LinearGaussianGaussian'
posteriorDiscard(obj, ss, ...)
```

Arguments

obj	A "LinearGaussianGaussian" object.
ss	Sufficient statistics of x. In Gaussian-Gaussian case the sufficient statistic of sample x is a object of type "ssLinearGaussianGaussian", it can be generated by the function sufficientStatistics().
...	Additional arguments to be passed to other inherited types.

Value

None. the gamma stored in "obj" will be updated based on "ss".

References

Murphy, Kevin P. Machine learning: a probabilistic perspective. MIT press, 2012.

See Also

LinearGaussianGaussian,posterior.LinearGaussianGaussian,sufficientStatistics.LinearGaussianGaussian

Examples

```
obj <- LinearGaussianGaussian(gamma=list(Sigma=matrix(c(2,1,1,2),2,2),
                              m=c(0.2,0.5,0.6),S=diag(3)))
x <- rGaussian(100,mu = runif(2),Sigma = diag(2))
A <- matrix(runif(6),2,3)
b <- runif(2)
ss <- sufficientStatistics(obj,x=x,A=A,b=b)
obj
## update prior into posterior
posterior(obj=obj,ss=ss)
obj
## remove the information, back to prior
posteriorDiscard(obj = obj,ss = ss)
obj
```

---

posteriorDiscard_bySufficientStatistics

update the prior distribution with sufficient statistics

Description

update the prior distribution with sufficient statistics

Usage

```
posteriorDiscard_bySufficientStatistics(obj, ...)
```

Arguments

obj	A "BayesianBrick" object used to select a method.
...	other parameters.

Value

None, or an error message if the update fails.

---

posteriorDiscard_bySufficientStatistics.CatDirichlet

Update the prior Dirichlet distribution with sample sufficient statistics

---

Description

Update the prior Dirichlet distribution with sample sufficient statistics

Usage

```
## S3 method for class 'CatDirichlet'
posteriorDiscard_bySufficientStatistics(obj, ss, ...)
```

Arguments

obj	A "CatDirichlet" object.
ss	Sufficient statistics of x. In Categorical-Dirichlet case the sufficient statistic of sample x can be either x itself, of an "ssCat" object generated by the function sufficientStatistics.CatDirichlet().
...	Additional arguments to be passed to other inherited types.

Value

None. the gamma stored in "obj" will be updated based on "ss".

References

Murphy, Kevin P. Machine learning: a probabilistic perspective. MIT press, 2012.

See Also

posteriorDiscard

posteriorDiscard_bySufficientStatistics.CatDP

Update a "CatDP" object with sample sufficient statistics

Description

Update a "CatDP" object with sample sufficient statistics

Usage

```
## S3 method for class 'CatDP'
posteriorDiscard_bySufficientStatistics(obj, ss, ...)
```

Arguments

obj	A "CatDP" object.
ss	Sufficient statistics of x. In Categorical-DP case the sufficient statistic of sample x can either be an object of type "ssCatDP" generated by sufficientStatistics(), or x itself(if x is a integer vector with all positive values).
...	Additional arguments to be passed to other inherited types.

Value

None. the model stored in "obj" will be updated based on "ss".

References

Teh, Yee W., et al. "Sharing clusters among related groups: Hierarchical Dirichlet processes." Advances in neural information processing systems. 2005.

posterior_bySufficientStatistics

update the prior distribution with sufficient statistics

Description

update the prior distribution with sufficient statistics

Usage

```
posterior_bySufficientStatistics(obj, ...)
```

Arguments

obj	A "BayesianBrick" object used to select a method.
...	further arguments passed to or from other methods.

Value

None, or an error message if the update fails.

`posterior_bySufficientStatistics.CatDirichlet`
Update a "CatDirichlet" object with sample sufficient statistics

---

Description

Update a "CatDirichlet" object with sample sufficient statistics

Usage

```
## S3 method for class 'CatDirichlet'
posterior_bySufficientStatistics(obj, ss, ...)
```

Arguments

`obj`	A "CatDirichlet" object.
`ss`	Sufficient statistics of x. In Categorical-Dirichlet case the sufficient statistic of sample x can be either x itself, of an "ssCat" object generated by the function sufficientStatistics.CatDirichlet().
`...`	Additional arguments to be passed to other inherited types.

Value

None. the gamma stored in "obj" will be updated based on "ss".

References

Murphy, Kevin P. Machine learning: a probabilistic perspective. MIT press, 2012.

`posterior_bySufficientStatistics.CatDP`
Update a "CatDP" object with sample sufficient statistics

Description

Update a "CatDP" object with sample sufficient statistics

Usage

```
## S3 method for class 'CatDP'
posterior_bySufficientStatistics(obj, ss, ...)
```

Arguments

obj	A "CatDP" object.
ss	Sufficient statistics of x. In Categorical-DP case the sufficient statistic of sample x can either be an object of type "ssCatDP" generated by sufficientStatistics(), or x itself(if x is a integer vector with all positive values).
...	Additional arguments to be passed to other inherited types.

Value

None. the model stored in "obj" will be updated based on "ss".

References

Teh, Yee W., et al. "Sharing clusters among related groups: Hierarchical Dirichlet processes." Advances in neural information processing systems. 2005.

---

print.BasicBayesian *Print the content of an BasicBayesian object*

Description

Print the content of an BasicBayesian object

Usage

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

Arguments

x	An object of the target type.
...	other parameters passed to print.

Value

None.

print.CatHDP *Print the content of an CatHDP object*

Description

Print the content of an CatHDP object

Usage

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

Arguments

x	An object of type "CatHDP".
...	Additional arguments to be passed to other inherited types.

Value

None.

print.CatHDP2 *Print the content of an CatHDP2 object*

Description

Print the content of an CatHDP2 object

Usage

```
## S3 method for class 'CatHDP2'
print(x, ...)
```

Arguments

x	An object of type "CatHDP2".
...	Additional arguments to be passed to other inherited types.

Value

None.

print.DP *print the content of a "DP" object*

Description

print the content of a "DP" object

Usage

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

Arguments

x	An object of type "DP".
...	Additional arguments to be passed to other inherited types.

Value

None.

print.HDP *print the content of a "HDP" object*

Description

print the content of a "HDP" object

Usage

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

Arguments

x	An object of type "HDP".
...	Additional arguments to be passed to other inherited types.

Value

None.

print.HDP2 *print the content of a "HDP2" object*

Description

print the content of a "HDP2" object

Usage

```
## S3 method for class 'HDP2'
print(x, ...)
```

Arguments

x	An object of type "HDP2".
...	Additional arguments to be passed to other inherited types.

Value

None.

rCategorical *Random generation for Categorical distribution*

Description

Generate random integer samples from a Categorical distribution. For a random variable x, the density function of categorical distribution is defined as

$$prod_{kin1:K} p_k^{I(x=k)}$$

Where K is the number of unique values.

Usage

```
rCategorical(n, p)
```

Arguments

n	integer, number of samples.
p	numeric, probabilities. length(p)=K.

Value

An integer vector of length n.

See Also

dCategorical

Examples

```
rCategorical(n=20,p=c(1,2))
```

---

rDir *Random generation for Dirichelt distribution*

Description

Generate random samples from Dirichlet distribution. For a random vector x, the density function is Dirichlet distribution is defined as: 1/Beta(alpha) prod_i=1:p x_i^alpha_i -1 Where Beta() is the beta function. p is the dimension of x.

Usage

```
rDir(n, alpha)
```

Arguments

n	integer, number of samples.
alpha	numeric, Dirichlet parameter.

Value

A numeric matrix of n rows and length(alpha) columns.

See Also

dDir

Examples

```
rDir(5,c(1,2,3)) #generate 5 samples with parameters c(1,2,3)
```

---

release_questions *additional release questions*

Description

This is a list of additional questions you want devtools::release() to ask when releasing your package.

Usage

```
release_questions()
```

rGaussian *Random generation for Gaussian distribution*

Description

Generate random samples from a Gaussian distribution. For a random vector x, the density function of a (multivariate) Gaussian distribution is defined as:

$$sqrt(2pi^p|Sigma|)^{-1}exp(-1/2(x-mu)^T Sigma^{-1}(x-mu))$$

where p is the dimension of x.

Usage

```
rGaussian(n, mu, Sigma = NULL, A = NULL)
```

Arguments

n	integer, number of samples.
mu	numeric, mean vector.
Sigma	matrix, covariance matrix, one of Sigma and A should be non-NULL.
A	matrix, the Cholesky decomposition of Sigma, an upper triangular matrix, one of Sigma and A should be non-NULL.

Value

A matrix of n rows and length(mu) columns.

See Also

dGaussian

Examples

```
x <- rGaussian(1000,mu = c(1,1),Sigma = matrix(c(1,0.5,0.5,3),2,2))
plot(x)
```

rInvGamma *Random number generation of Inverse-Gamma distribution*

Description

Generation random samples from Inverse-Gamma distribution. For a random variable x, the density function is defined as:

$$(rate^shape)/Gamma(shape)x^{-shape-1}exp(-rate/x)$$

Where Gamma() is the Gamma function.

Usage

```
rInvGamma(n, shape, scale)
```

Arguments

n	integer, number of samples to be generated.
shape	numeric, the shape parameter of gamma distribution.
scale	numeric, the scale, or inverse-scale parameter of gamma distribution. The 'rate' parameter in Gamma is the 'scale' parameter in InvGamma.

Value

A numeric vector, samples of Inverse-Gamma distribution.

rInvWishart *Random generation for Inverse-Wishart distribution*

Description

Generate random samples from Inverse-Wishart distribution. For a random matrix x, the density function of Inverse-Wishart is defined as:

$$(2^{(dfp)/2}Gamma_p(df/2)|scale|^{-df/2})^{-1}|x|^{(-df-p-1)/2}exp(-1/2tr(x^{-1}scale))$$

Where x is a pxp symmetric positive definite matrix, Gamma_p() is the multivariate Gamma function of dimension p.

Usage

```
rInvWishart(df, scale)
```

Arguments

df	numeric, the degree of freedom.
scale	matrix, a symmetric positive-definite matrix, the 'scale' parameter. The 'rate' parameter in Wishart is the 'scale' parameter in InvWishart.

Value

A symmetric positive-definite matrix.

References

Hoff, Peter D. A first course in Bayesian statistical methods. Vol. 580. New York: Springer, 2009.

Examples

```
scale <- crossprod(matrix(rnorm(15),5,3)) # the prior scale
m <- matrix(0,3,3)
## get 1000 samples and calculate the sample mean
for(i in 1:1000){
    m <- m+rInvWishart(df=5,scale=scale)/1000
}
## m should roughly equal scale/(df-p-1), p is the dimension.
m
scale/(5-3-1)
```

---

rNIW *Random number generation for Normal-Inverse-Wishart (NIW) distribution.*

Description

Generate a NIW sample.For a random vector mu, and a random matrix Sigma, the density function is defined as:

$sqrt(2pi^p|Sigma/k|)^{-1}exp(-1/2(mu-m)^T(Sigma/k)^{-1}(mu-m))(2^{(vp)/2}Gamma_p(v/2)|S|^{-v/2})^{-1}|Sigma|^{(-v-p-$

Where p is the dimension of mu and Sigma.

Usage

```
rNIW(m, k, v, S)
```

Arguments

m	numeric, mean of mu.
k	numeric, precision of mu.
v	numeric, degree of freedom of Sigma.
S	numeric, a symmetric positive definite scale matrix of Sigma, S is proportional to E(Sigma).

Value

A list of two list(mu,Sigma), where 'mu' is a numeric vector, the Gaussian sample; 'Sigma' is a symmetric positive definite matrix, the Inverse-Wishart sample.

References

O'Hagan, Anthony, and Jonathan J. Forster. Kendall's advanced theory of statistics, volume 2B: Bayesian inference. Vol. 2. Arnold, 2004.

MARolA, K. V., JT KBNT, and J. M. Bibly. Multivariate analysis. AcadeInic Press, Londres, 1979.

See Also

dNIW

Examples

```
rNIW(m=runif(3),k=0.001,v=5,S=crossprod(matrix(rnorm(15),5,3)))
```

`rPosterior` *Generate random samples from the posterior distribution*

Description

This is a generic function that will generate random samples from the posterior distribution. i.e. for the model structure:

$$theta|gamma \sim H(gamma)$$

$$x|theta \sim F(theta)$$

generate random sampels of theta from the distribution $theta \sim H(gamma)$. For a given Bayesian bricks object obj, `rPosterior()` will generate random samples for different model structures:

class(obj)="LinearGaussianGaussian": Where

$$x \sim Gaussian(Az + b, Sigma)$$

$$z \sim Gaussian(m, S)$$

`rPosterior()` will generate random samples from Gaussian(m,S) See `?rPosterior.LinearGaussianGaussian` for details.

class(obj)="GaussianGaussian": Where

$$x \sim Gaussian(mu, Sigma)$$

$$mu \sim Gaussian(m, S)$$

Sigma is known. `rPosterior()` will generate random samples from Gaussian(m,S) See `?rPosterior.GaussianGaussian` for details.

class(obj)="GaussianInvWishart": Where

$$x \sim Gaussian(mu, Sigma)$$

$$Sigma \sim InvWishart(v, S)$$

mu is known. `rPosterior()` will generate random samples from InvWishart(v,S) See `?rPosterior.GaussianInvWishart` for details.

class(obj)="GaussianNIW": Where

$$x \sim Gaussian(mu, Sigma)$$

$$Sigma \sim InvWishart(v, S)$$

$$mu \sim Gaussian(m, Sigma/k)$$

`rPosterior()` will generate random samples from NIW(m,k,v,S) See `?rPosterior.GaussianNIW` for details.

class(obj)="GaussianNIG": Where

$$x \sim Gaussian(Xbeta, sigma^2)$$

$$sigma^2 \sim InvGamma(a, b)$$

$$beta \sim Gaussian(m, sigma^2 V)$$

X is a row vector, or a design matrix where each row is an obervation. `rPosterior()` will generate random samples from NIG(m,V,a,b) See `?rPosterior.GaussianNIG` for details.

class(obj)="CatDirichlet": Where

$$x \sim Categorical(pi)$$

$$pi \sim Dirichlet(alpha)$$

`rPosterior()` will generate random samples from Dirichlet(alpha) See `?rPosterior.CatDirichlet` for details.

Usage

```
rPosterior(obj, ...)
```

Arguments

`obj`	A "BayesianBrick" object used to select a method.
`...`	further arguments passed to or from other methods.

Value

numeric, the density value

See Also

rPosterior.LinearGaussianGaussian for Linear Gaussian and Gaussian conjugate structure, rPosterior.GaussianGaussian for Gaussian-Gaussian conjugate structure, rPosterior.GaussianInvWishart for Gaussian-Inverse-Wishart conjugate structure, rPosterior.GaussianNIW for Gaussian-NIW conjugate structure, rPosterior.GaussianNIG for Gaussian-NIG conjugate structure, rPosterior.CatDirichlet for Categorical-Dirichlet conjugate structure ...

---

rPosterior.CatDirichlet

Generate ramdom samples from the posterior distribution of a "Cat-Dirichlet" object

---

Description

Generate random samples from the posterior distribution of the following structure:

$$pi|alpha \sim Dir(alpha)$$

$$x|pi \sim Categorical(pi)$$

Where Dir() is the Dirichlet distribution, Categorical() is the Categorical distribution. See ?dDir and dCategorical for the definitions of these distribution.
The model structure and prior parameters are stored in a "CatDirichlet" object.
Posterior distribution is Dir(pi|alpha).

Usage

```
## S3 method for class 'CatDirichlet'
rPosterior(obj, n = 1, ...)
```

Arguments

obj	A "CatDirichlet" object.
n	integer, the number of samples to be generated.
...	Additional arguments to be passed to other inherited types.

Value

A matrix, each row is a sample of pi.

See Also

CatDirichlet, dPosterior.CatDirichlet

Examples

```
obj <- CatDirichlet(gamma=list(alpha=rep(1,26),uniqueLabels = letters))
rPosterior(obj = obj,n=3)
```

`rPosterior.GaussianGaussian` *Generate ramdom samples from the posterior distribution of a "GaussianGaussian" object*

Description

Generate random samples from the posterior distribution of the following structure:

$$x \sim Gaussian(mu, Sigma)$$

$$mu \sim Gaussian(m, S)$$

Where Sigma is known. Gaussian() is the Gaussian distribution. See `?dGaussian` for the definition of Gaussian distribution.
The model structure and prior parameters are stored in a "GaussianGaussian" object.
Posterior distribution is Gaussian(mulm,S).

Usage

```
## S3 method for class 'GaussianGaussian'
rPosterior(obj, n = 1, ...)
```

Arguments

`obj`	A "GaussianGaussian" object.
`n`	integer, number of samples.
`...`	Additional arguments to be passed to other inherited types.

Value

A matrix of n rows, each row is a sample of mu.

See Also

`GaussianGaussian`, `dPosterior.GaussianGaussian`

Examples

```
obj <- GaussianGaussian(gamma=list(Sigma=matrix(c(2,1,1,2),2,2),m=c(0.2,0.5),S=diag(2)))
rPosterior(obj=obj,n=20)
```

`rPosterior.GaussianInvWishart` *Generate one ramdom sample from the posterior distribution of a "GaussianInvWishart" object*

Description

Generate random samples from the posterior distribution of the following structure:

$$x \sim Gaussian(mu, Sigma)$$

$$Sigma \sim InvWishart(v, S)$$

mu is known. Gaussian() is the Gaussian distribution. See `?dGaussian` and `?dInvWishart` for the definition of the distributions.
The model structure and prior parameters are stored in a "GaussianInvWishart" object.
Posterior distribution is InvWishart(Sigmalv,S).

Usage

```
## S3 method for class 'GaussianInvWishart'
rPosterior(obj, ...)
```

Arguments

`obj`	A "GaussianInvWishart" object.
`...`	Additional arguments to be passed to other inherited types.

Value

matrix, a sample of Sigma.

See Also

`GaussianInvWishart`, `dPosterior.GaussianInvWishart`

Examples

```
obj <- GaussianInvWishart(gamma=list(mu=c(-1.5,1.5),v=3,S=diag(2)))
rPosterior(obj = obj)
```

`rPosterior.GaussianNIG` *Generate ramdom samples from the posterior distribution of a "GaussianNIG" object*

Description

Generate random samples from the posterior distribution of the following structure: Generate the the density value of the posterior distribution of the following structure:

$$x \sim Gaussian(Xbeta, sigma^2)$$

$$sigma^2 \sim InvGamma(a, b)$$

$$beta \sim Gaussian(m, sigma^2 V)$$

Where X is a row vector, or a design matrix where each row is an obervation. InvGamma() is the Inverse-Gamma distribution, Gaussian() is the Gaussian distribution. See `?dInvGamma` and `dGaussian` for the definitions of these distribution.
The model structure and prior parameters are stored in a "GaussianNIG" object.
Posterior distribution is the distribution of beta,sigma^2|m,V,a,b.

Usage

```
## S3 method for class 'GaussianNIG'
rPosterior(obj, ...)
```

Arguments

`obj`	A "GaussianNIG" object.
`...`	Additional arguments to be passed to other inherited types.

Value

list(beta,sigma2), where beta is a numeric vector, sigma is a scalar value.

See Also

`GaussianNIG`, `dPosterior.GaussianNIG`

Examples

```
obj <- GaussianNIG(gamma=list(m=c(0,0),V=diag(2),a=1,b=1))
rPosterior(obj = obj)
```

`rPosterior.GaussianNIW` *Generate ramdom samples from the posterior distribution of a "GaussianNIW" object*

Description

Generate random samples from the posterior distribution of the following structure:

$$mu, Sigma|m, k, v, S \sim NIW(m, k, v, S)$$

$$x|mu, Sigma \sim Gaussian(mu, Sigma)$$

Where NIW() is the Normal-Inverse-Wishart distribution, Gaussian() is the Gaussian distribution. See `?dNIW` and `dGaussian` for the definitions of these distribution.
The model structure and prior parameters are stored in a "GaussianNIW" object.
Posterior distribution is NIW(mu,Sigma|m,k,v,S).

Usage

```
## S3 method for class 'GaussianNIW'
rPosterior(obj, ...)
```

Arguments

`obj`	A "GaussianNIW" object.
`...`	Additional arguments to be passed to other inherited types.

Value

list(mu,Sigma), where mu is a numeric vector, Sigma is a symmetric positive definite matrix.

See Also

`GaussianNIW`, `dPosterior.GaussianNIW`

Examples

```
obj <- GaussianNIW(gamma=list(m=c(0,0),k=1,v=2,S=diag(2)))
rPosterior(obj = obj)
```

`rPosterior.LinearGaussianGaussian` *Posterior random generation of a "LinearGaussianGaussian" object*

Description

Generate random samples from the posterior distribution of the following structure:

$$x \sim Gaussian(Az + b, Sigma)$$

$$z \sim Gaussian(m, S)$$

Where Sigma is known. A is a $dimxxdimz$ matrix, x is a $dimxx1$ random vector, z is a $dimzx1$ random vector, b is a $dimmx1$ vector. Gaussian() is the Gaussian distribution. See `?dGaussian` for the definition of Gaussian distribution.
The model structure and prior parameters are stored in a "LinearGaussianGaussian" object.
Posterior distribution is Gaussian(z|m,S).

Usage

```
## S3 method for class 'LinearGaussianGaussian'
rPosterior(obj, n = 1, ...)
```

Arguments

`obj`	A "LinearGaussianGaussian" object.
`n`	integer, number of samples.
`...`	Additional arguments to be passed to other inherited types.

Value

A matrix of n rows, each row is a sample of z.

See Also

`LinearGaussianGaussian`, `dPosterior.LinearGaussianGaussian`

Examples

```
obj <- LinearGaussianGaussian(gamma=list(Sigma=matrix(c(2,1,1,2),2,2),
                                         m=c(0.2,0.5,0.6),S=diag(3)))
rPosterior(obj = obj,n=20)
```

rPosteriorPredictive *Generate random samples from the posterior predictive distribution*

Description

This is a generic function that will generate random samples from the posterior predictive distribution. i.e. for the model structure:

$$theta|gamma \sim H(gamma)$$

$$x|theta \sim F(theta)$$

generate x_new from the posterior predictive distribution of x|gamma. For a given Bayesian bricks object obj, `rPosteriorPredictive()` will generate random samples from different model structures:

class(obj)="LinearGaussianGaussian": Where

$$x \sim Gaussian(Az + b, Sigma)$$

$$z \sim Gaussian(m, S)$$

`rPosteriorPredictive()` will generate samples from the distribution of x|m,S,A,b,Sigma See `?rPosteriorPredictive.LinearGaussianGaussian` for details.

class(obj)="GaussianGaussian": Where

$$x \sim Gaussian(mu, Sigma)$$

$$mu \sim Gaussian(m, S)$$

Sigma is known. `rPosteriorPredictive()` will generate samples from the distribution of x|m,S,Sigma See `?rPosteriorPredictive.GaussianGaussian` for details.

class(obj)="GaussianInvWishart": Where

$$x \sim Gaussian(mu, Sigma)$$

$$Sigma \sim InvWishart(v, S)$$

mu is known. `rPosteriorPredictive()` will generate samples from the distribution of x|mu,v,S See `?rPosteriorPredictive.GaussianInvWishart` for details.

class(obj)="GaussianNIW": Where

$$x \sim Gaussian(mu, Sigma)$$

$$Sigma \sim InvWishart(v, S)$$

$$mu \sim Gaussian(m, Sigma/k)$$

`rPosteriorPredictive()` will generate samples from the distribution of x|m,k,v,S See `?rPosteriorPredictive.Gaussi` for details.

class(obj)="GaussianNIG": Where

$$x \sim Gaussian(Xbeta, sigma^2)$$

$$sigma^2 \sim InvGamma(a, b)$$

$$beta \sim Gaussian(m, sigma^2 V)$$

X is a row vector, or a design matrix where each row is an obervation. `rPosteriorPredictive()` will generate samples from the distribution of x,X|m,V,a,b See `?rPosteriorPredictive.GaussianNIG` for details.

class(obj)="CatDirichlet": Where

$$x \sim Categorical(pi)$$

$$pi \sim Dirichlet(alpha)$$

`rPosteriorPredictive()` will generate samples from the distribution of x|alpha See `?rPosteriorPredictive.CatDiric` for details.

class(obj)="CatDP": Where

$$x \sim Categorical(pi)$$

$$pi \sim DirichletProcess(alpha)$$

`rPosteriorPredictive()` will generate samples from the distribution of x|alpha See `?rPosteriorPredictive.CatDP` for details.

Usage

```
rPosteriorPredictive(obj, n, ...)
```

Arguments

`obj`	A "BayesianBrick" object used to select a method.
`n`	integer, specify the number of samples to be generated.
`...`	further arguments passed to or from other methods.

Value

a matrix or vector or list of random samples, depends on the type of 'obj'.

See Also

`rPosteriorPredictive.GaussianNIW` for Gaussian-NIW conjugate structure, `rPosteriorPredictive.GaussianNIG` for Gaussian-NIG conjugate structure, `rPosteriorPredictive.CatDirichlet` for Categorical-Dirichlet conjugate structure, `rPosteriorPredictive.CatDP` for Categorical-DP conjugate structure ...

`rPosteriorPredictive.CatDirichlet` *Generate random samples from the posterior predictive distribution of a "CatDirichlet" object*

Description

Generate random samples from the posterior predictive distribution of the following structure:

$$pi|alpha \sim Dir(alpha)$$

$$x|pi \sim Categorical(pi)$$

Where Dir() is the Dirichlet distribution, Categorical() is the Categorical distribution. See `?dDir` and `dCategorical` for the definitions of these distribution.
The model structure and prior parameters are stored in a "CatDirichlet" object
posterior predictive is a distribution of x|alpha

Usage

```
## S3 method for class 'CatDirichlet'
rPosteriorPredictive(obj, n, ...)
```

Arguments

`obj`	A "CatDirichlet" object.
`n`	integer, number of samples.
`...`	Additional arguments to be passed to other inherited types.

Value

A vector of the same type as obj$gamma$uniqueLabels.

References

Murphy, Kevin P. Machine learning: a probabilistic perspective. MIT press, 2012.

See Also

`CatDirichlet`, `dPosteriorPredictive.CatDirichlet`

Examples

```
obj <- CatDirichlet(gamma=list(alpha=runif(26,1,2),uniqueLabels = letters))
rPosteriorPredictive(obj=obj,n=200)
```

`rPosteriorPredictive.CatDP` *Generate random samples from the posterior predictive distribution of a "CatDP" object*

Description

Generate random samples from the posterior predictive distribution of the following structure:

$$pi|alpha \sim DP(alpha, U)$$

$$x|pi \sim Categorical(pi)$$

where DP(alpha,U) is a Dirichlet Process on positive integers, alpha is the "concentration parameter" of the Dirichlet Process, U is the "base measure" of this Dirichlet process, it is an uniform distribution on all positive integers.Categorical() is the Categorical distribution. See `dCategorical` for the definition of the Categorical distribution.
In the case of CatDP, x can only be positive integers.
The model structure and prior parameters are stored in a "CatDP" object.
Posterior predictive distribution is the distribution of x|alpha.

Usage

```
## S3 method for class 'CatDP'
rPosteriorPredictive(obj, n = 1L, ...)
```

Arguments

`obj`	A "CatDP" object.
`n`	integer, number of samples.
`...`	Additional arguments to be passed to other inherited types.

Value

integer, the categorical samples.

References

Teh, Yee W., et al. "Sharing clusters among related groups: Hierarchical Dirichlet processes." Advances in neural information processing systems. 2005.

See Also

`CatDP`, `dPosteriorPredictive.CatDP`

Examples

```
x <- sample(1L:10L,size = 40,replace = TRUE)
obj <- CatDP()
ss <- sufficientStatistics(obj=obj,x=x)
posterior(obj = obj,ss = ss)
rPosteriorPredictive(obj = obj,n=200L)
```

`rPosteriorPredictive.CatHDP` *Generate random samples from the posterior predictive distribution of a "CatHDP" object*

Description

Generate random samples from the posterior predictive distribution of the following structure:

$$G|gamma \sim DP(gamma, U)$$

$$pi_j|G, alpha \sim DP(alpha, G), j = 1 : J$$

$$z|pi_j \sim Categorical(pi_j)$$

$$k|z, G \sim Categorical(G), \text{ if z is a sample from the base measure G}$$

where DP(gamma,U) is a Dirichlet Process on positive integers, gamma is the "concentration parameter", U is the "base measure" of this Dirichlet process, U is an uniform distribution on all positive integers. DP(alpha,G) is a Dirichlet Process on integers with concentration parameter alpha and base measure G. Categorical() is the Categorical distribution. See `dCategorical` for the definition of the Categorical distribution.
In the case of CatHDP, z and k can only be positive integers.
The model structure and prior parameters are stored in a "CatHDP" object.
Posterior predictive is a distribution of z,k|alpha,gamma,U.

Usage

```
## S3 method for class 'CatHDP'
rPosteriorPredictive(obj, n = 1L, j, ...)
```

Arguments

`obj`	A "CatHDP" object.
`n`	integer, number of samples.
`j`	integer, group label.
`...`	Additional arguments to be passed to other inherited types.

Value

integer, the categorical samples.

References

Teh, Yee W., et al. "Sharing clusters among related groups: Hierarchical Dirichlet processes." Advances in neural information processing systems. 2005.

See Also

CatHDP, dPosteriorPredictive.CatHDP

`rPosteriorPredictive.CatHDP2` *Generate random samples from the posterior predictive distribution of a "CatHDP2" object*

Description

Generate random samples from the posterior predictive distribution of the following structure:

$$G|eta \sim DP(eta, U)$$

$$G_m|gamma \sim DP(gamma, G), m = 1:M$$

$$pi_{mj}|G_m, alpha \sim DP(alpha, G_m), j = 1:J_m$$

$$z|pi_{mj} \sim Categorical(pi_{mj})$$

$$k|z, G_m \sim Categorical(G_m),\ \text{if z is a sample from the base measure } G_m$$

$$u|k, G \sim Categorical(G),\ \text{if k is a sample from the base measure G}$$

where DP(eta,U) is a Dirichlet Process on positive integers, eta is the "concentration parameter", U is the "base measure" of this Dirichlet process, U is an uniform distribution on all positive integers. DP(gamma,G) is a Dirichlet Process on integers with concentration parameter gamma and base measure G. DP(alpha,G_m) is a Dirichlet Process on integers with concentration parameter alpha and base measure G_m. Categorical() is the Categorical distribution. See `dCategorical` for the definition of the Categorical distribution.
In the case of CatHDP2, u, z and k can only be positive integers.
The model structure and prior parameters are stored in a "CatHDP2" object.
Posterior predictive is a distribution of u,z,klalpha,gamm,eta,U.

Usage

```
## S3 method for class 'CatHDP2'
rPosteriorPredictive(obj, n = 1L, m, j, ...)
```

Arguments

`obj`	A "CatHDP2" object.
`n`	integer, number of samples.
`m`	integer, group label.
`j`	integer, subgroup label.
`...`	Additional arguments to be passed to other inherited types.

Value

integer, the categorical samples.

References

Teh, Yee W., et al. "Sharing clusters among related groups: Hierarchical Dirichlet processes." Advances in neural information processing systems. 2005.

See Also

CatHDP2, dPosteriorPredictive.CatHDP2

`rPosteriorPredictive.DP` *Generate random samples from the posterior predictive distribution of a "DP" object*

Description

Generate random samples from the posterior predictive distribution of the following structure:

$$pi|alpha \sim DP(alpha, U)$$

$$z|pi \sim Categorical(pi)$$

$$theta_z|psi \sim H0(psi)$$

$$x|theta_z, z \sim F(theta_z)$$

where DP(alpha,U) is a Dirichlet Process on positive integers, alpha is the "concentration parameter" of the Dirichlet Process, U is the "base measure" of this Dirichlet process. The choice of F() and H0() can be described by an arbitrary "BasicBayesian" object such as "GaussianGaussian","GaussianInvWishart","GaussianNIW", "GaussianNIG", "CatDirichlet", and "CatDP". See `?BasicBayesian` for definition of "BasicBayesian" objects, and see for example `?GaussianGaussian` for specific "BasicBayesian" instances. As a summary, An "DP" object is simply a combination of a "CatDP" object (see `?CatDP`) and an object of any "BasicBayesian" type.
The model structure and prior parameters are stored in a "DP" object.
This function will generate random samples from the distribution z|alpha,psi,x.

Usage

```
## S3 method for class 'DP'
rPosteriorPredictive(obj, n = 1, x, ...)
```

Arguments

`obj`	A "DP" object.
`n`	integer, number of samples.
`x`	Random samples of the "BasicBayesian" object.
`...`	Additional arguments to be passed to other inherited types.

Value

integer, the categorical samples.

References

Teh, Yee W., et al. "Sharing clusters among related groups: Hierarchical Dirichlet processes." Advances in neural information processing systems. 2005.

See Also

DP, dPosteriorPredictive.DP

Examples

```
x <- rnorm(4)
z <- sample(1L:10L,size = 4,replace = TRUE)
obj <- DP()
ss <- sufficientStatistics(obj = obj,x=x,foreach = TRUE)
for(i in 1L:length(x)) posterior(obj = obj,ss=ss[[i]],z=z[i])
xnew <- rnorm(10)
znew <- sample(1L:10L,size = 10,replace = TRUE)
rPosteriorPredictive(obj = obj,n=1,x=xnew[5])
```

---

rPosteriorPredictive.GaussianGaussian

Generate random samples from the posterior predictive distribution of a "GaussianGaussian" object

---

Description

Generate random samples from the posterior predictive distribution of the following structure:

$$x \sim Gaussian(mu, Sigma)$$

$$mu \sim Gaussian(m, S)$$

Where Sigma is known. Gaussian() is the Gaussian distribution. See ?dGaussian for the definition of Gaussian distribution.
The model structure and prior parameters are stored in a "GaussianGaussian" object.
Posterior predictive is a distribution of x|m,S,Sigma.

Usage

```
## S3 method for class 'GaussianGaussian'
rPosteriorPredictive(obj, n = 1, ...)
```

Arguments

obj	A "GaussianGaussian" object.
n	integer, number of samples.
...	Additional arguments to be passed to other inherited types.

Value

A matrix of n rows, each row is a sample.

References

Gelman, Andrew, et al. Bayesian data analysis. CRC press, 2013.

See Also

GaussianGaussian, dPosteriorPredictive.GaussianGaussian

Examples

```
obj <- GaussianGaussian(gamma=list(Sigma=matrix(c(2,1,1,2),2,2),m=c(0.2,0.5),S=diag(2)))
rPosteriorPredictive(obj=obj,20)
```

---

rPosteriorPredictive.GaussianInvWishart

Generate random samples from the posterior predictive distribution of a "GaussianInvWishart" object

---

Description

Generate random samples from the posterior predictive distribution of the following structure:

$$x \sim Gaussian(mu, Sigma)$$

$$Sigma \sim InvWishart(v, S)$$

mu is known. Gaussian() is the Gaussian distribution. See ?dGaussian and ?dInvWishart for the definition of the distributions.
The model structure and prior parameters are stored in a "GaussianInvWishart" object.
Posterior predictive is a distribution of x|v,S,mu.

Usage

```
## S3 method for class 'GaussianInvWishart'
rPosteriorPredictive(obj, n, ...)
```

Arguments

obj	A "GaussianInvWishart" object.
n	integer, number of samples.
...	Additional arguments to be passed to other inherited types.

Value

A matrix of n rows, each row is a sample.

References

Gelman, Andrew, et al. Bayesian data analysis. CRC press, 2013.

MARolA, K. V., JT KBNT, and J. M. Bibly. Multivariate analysis. AcadeInic Press, Londres, 1979.

See Also

GaussianInvWishart, dPosteriorPredictive.GaussianInvWishart

Examples

```
obj <- GaussianInvWishart(gamma=list(mu=c(-1.5,1.5),v=3,S=diag(2)))
x <- rGaussian(100,mu = c(-1.5,1.5),Sigma = matrix(c(0.1,0.03,0.03,0.1),2,2))
ss <- sufficientStatistics(obj=obj,x=x,foreach = FALSE)
## use x to update the prior informatoin
posterior(obj=obj,ss = ss)
## use the posterior to generate new samples
rPosteriorPredictive(obj = obj,n=20)
```

---

rPosteriorPredictive.GaussianNIG

Generate random samples from the posterior predictive distribution of a "GaussianNIG" object

---

Description

Generate random samples from the posterior predictive distribution of the following structure:

$$x \sim Gaussian(Xbeta, sigma^2)$$

$$sigma^2 \sim InvGamma(a, b)$$

$$beta \sim Gaussian(m, sigma^2 V)$$

Where X is a row vector, or a design matrix where each row is an obervation. InvGamma() is the Inverse-Gamma distribution, Gaussian() is the Gaussian distribution. See ?dInvGamma and dGaussian for the definitions of these distribution.
The model structure and prior parameters are stored in a "GaussianNIG" object.
Posterior predictive is a distribution of x|m,V,a,b,X

Usage

```
## S3 method for class 'GaussianNIG'
rPosteriorPredictive(obj, n, X, ...)
```

Arguments

obj	A "GaussianNIG" object.
n	integer, number of samples.
X	matrix, the location of the prediction, each row is a location.
...	Additional arguments to be passed to other inherited types.

Value

A matrix of n rows and nrow(X) columns, each row is a sample.

References

Banerjee, Sudipto. "Bayesian Linear Model: Gory Details." Downloaded from http://www. biostat. umn. edu/ ph7440 (2008).

See Also

GaussianNIG, dPosteriorPredictive.GaussianNIG

Examples

```
obj <- GaussianNIG(gamma=list(m=c(1,1),V=diag(2),a=1,b=1))
X <- matrix(runif(20),ncol=2)
rPosteriorPredictive(obj=obj,n=3,X=X)
```

---

rPosteriorPredictive.GaussianNIW

Generate random samples from the posterior predictive distribution of a "GaussianNIW" object

---

Description

Generate random samples from the posterior predictive distribution of the following structure:

$$mu, Sigma|m, k, v, S \sim NIW(m, k, v, S)$$

$$x|mu, Sigma \sim Gaussian(mu, Sigma)$$

Where NIW() is the Normal-Inverse-Wishart distribution, Gaussian() is the Gaussian distribution. See ?dNIW and dGaussian for the definitions of these distribution.
The model structure and prior parameters are stored in a "GaussianNIW" object.
Posterior predictive is a distribution of x|m,k,v,S.

Usage

```
## S3 method for class 'GaussianNIW'
rPosteriorPredictive(obj, n, ...)
```

Arguments

obj	A "GaussianNIW" object.
n	integer, number of samples.
...	Additional arguments to be passed to other inherited types.

Value

A matrix of n rows, each row is a sample.

References

Murphy, Kevin P. "Conjugate Bayesian analysis of the Gaussian distribution." def 1.22 (2007): 16.

Gelman, Andrew, et al. "Bayesian Data Analysis Chapman & Hall." CRC Texts in Statistical Science (2004).

See Also

GaussianNIW, dPosteriorPredictive.GaussianNIW

Examples

```
obj <- GaussianNIW(gamma=list(m=c(0,0),k=1,v=2,S=diag(2)))
rPosteriorPredictive(obj=obj,20)
```

---

rPosteriorPredictive.HDP

Generate random samples from the posterior predictive distribution of a "HDP" object

---

Description

Generate random samples from the posterior predictive distribution of the following structure:

$$G|gamma \sim DP(gamma, U)$$

$$pi_j|G, alpha \sim DP(alpha, G), j = 1 : J$$

$$z|pi_j \sim Categorical(pi_j)$$

$k|z, G \sim Categorical(G)$, if z is a sample from the base measure G

$$theta_k|psi \sim H0(psi)$$

$$x|theta_k, k \sim F(theta_k)$$

where DP(gamma,U) is a Dirichlet Process on positive integers, gamma is the "concentration parameter", U is the "base measure" of this Dirichlet process, U is an uniform distribution on all positive integers. DP(alpha,G) is a Dirichlet Process on integers with concentration parameter alpha and base measure G. The choice of F() and H0() can be described by an arbitrary "BasicBayesian" object such as "GaussianGaussian","GaussianInvWishart","GaussianNIW", "GaussianNIG", "CatDirichlet", and "CatDP". See `?BasicBayesian` for definition of "BasicBayesian" objects, and see for example `?GaussianGaussian` for specific "BasicBayesian" instances. As a summary, An "HDP" object is simply a combination of a "CatHDP" object (see `?CatHDP`) and an object of any "BasicBayesian" type.
In the case of HDP, z and k can only be positive integers.
The model structure and prior parameters are stored in a "HDP" object.
This function will generate random samples from the distribution z,k|gamma,alpha,psi,x.

Usage

```
## S3 method for class 'HDP'
rPosteriorPredictive(obj, n = 1, x, j, ...)
```

Arguments

`obj`	A "HDP" object.
`n`	integer, number of samples.
`x`	Random samples of the "BasicBayesian" object.
`j`	integer, group label.
`...`	Additional arguments to be passed to other inherited types.

Value

integer, the categorical samples.

References

Teh, Yee W., et al. "Sharing clusters among related groups: Hierarchical Dirichlet processes." Advances in neural information processing systems. 2005.

See Also

`HDP`, `dPosteriorPredictive.HDP`

`rPosteriorPredictive.HDP2` *Generate random samples from the posterior predictive distribution of a "HDP2" object*

Description

Generate random samples from the posterior predictive distribution of the following structure:

$$G|eta \sim DP(eta, U)$$

$$G_m|gamma, G \sim DP(gamma, G), m = 1 : M$$

$$pi_{mj}|G_m, alpha \sim DP(alpha, G_m), j = 1 : J_m$$

$$z|pi_{mj} \sim Categorical(pi_{mj})$$

$$k|z, G_m \sim Categorical(G_m), \text{ if z is a sample from the base measure } G_{mj}$$

$$u|k, G \sim Categorical(G), \text{ if k is a sample from the base measure G}$$

$$theta_u|psi \sim H0(psi)$$

$$x|theta_u, u \sim F(theta_u)$$

where DP(eta,U) is a Dirichlet Process on positive integers, eta is the "concentration parameter", U is the "base measure" of this Dirichlet process, U is an uniform distribution on all positive integers. DP(gamma,G) is a Dirichlet Process on integers with concentration parameter gamma and base measure G. DP(alpha,G_m) is a Dirichlet Process on integers with concentration parameter alpha and base measure G_m. The choice of F() and H0() can be described by an arbitrary "BasicBayesian" object such as "GaussianGaussian","GaussianInvWishart","GaussianNIW", "GaussianNIG", "CatDirichlet", and "CatDP". See `?BasicBayesian` for definition of "BasicBayesian" objects, and see for example `?GaussianGaussian` for specific "BasicBayesian" instances. As a summary, An "HDP2" object is simply a combination of a "CatHDP2" object (see `?CatHDP2`) and an object of any "BasicBayesian" type.
In the case of HDP2, u, z and k can only be positive integers.
The model structure and prior parameters are stored in a "HDP2" object.
This function will generate random samples from the distribution u,z,kleta,gamma,alpha,psi,x.

Usage

```
## S3 method for class 'HDP2'
rPosteriorPredictive(obj, n = 1, x, m, j, ...)
```

Arguments

`obj`	A "HDP2" object.
`n`	integer, number of samples.
`x`	Random samples of the "BasicBayesian" object.
`m`	integer, group label.
`j`	integer, subgroup label.
`...`	Additional arguments to be passed to other inherited types.

Value

integer, the categorical samples.

References

Teh, Yee W., et al. "Sharing clusters among related groups: Hierarchical Dirichlet processes." Advances in neural information processing systems. 2005.

See Also

HDP2, dPosteriorPredictive.HDP2

---

rPosteriorPredictive.LinearGaussianGaussian

Generate random samples from the posterior predictive distribution of a "LinearGaussianGaussian" object

---

Description

Generate random samples from the posterior predictive distribution of the following structure:

$$x \sim Gaussian(Az + b, Sigma)$$

$$z \sim Gaussian(m, S)$$

Where Sigma is known. A is a $dimxxdimz$ matrix, x is a $dimxx1$ random vector, z is a $dimzx1$ random vector, b is a $dimmx1$ vector. Gaussian() is the Gaussian distribution. See ?dGaussian for the definition of Gaussian distribution.
The model structure and prior parameters are stored in a "LinearGaussianGaussian" object.
Posterior predictive is a distribution of x|m,S,A,b,Sigma.

Usage

```
## S3 method for class 'LinearGaussianGaussian'
rPosteriorPredictive(obj, n = 1, A, b = NULL, ...)
```

Arguments

obj	A "LinearGaussianGaussian" object.
n	integer, number of samples.
A	matrix or list. when you want the random samples x to be a $nx1$ matrix, A must be a matrix of $nxdimz$, dimz is the dimension of z; When you want the random samples x to be a $Nxdimx$ matrix, where $dimx > 1$, A can be either a list or a matrix. When A is a list, $A = A_1, A_2, ...A_N$ is a list of $dimxxdimz$ matrices. If A is a single $dimxxdimz$ matrix, it will be replicated N times into a length N list.

b — matrix, when you want the random samples x to be a $Nx1$ matrix, b must also be a $Nx1$ matrix or length N vector; When you want x to be a $Nxdimx$ matrix, where $dimx > 1$, b can be either a matrix or a vector. When b is a matrix, $b = b_1^T, ..., b_N^T$ is a $Nxdimx$ matrix, each row is a transposed vector. When b is a length $dimx$ vector, it will be transposed into a row vector and replicated N times into a $Nxdimx$ matrix. When b = NULL, it will be treated as a vector of zeros. Default NULL.

... — Additional arguments to be passed to other inherited types.

Value

A matrix of n rows, each row is a sample.

References

Murphy, Kevin P. Machine learning: a probabilistic perspective. MIT press, 2012.

See Also

LinearGaussianGaussian, dPosteriorPredictive.LinearGaussianGaussian

Examples

```
obj <- LinearGaussianGaussian(gamma=list(Sigma=matrix(c(2,1,1,2),2,2),
                                         m=c(0.2,0.5,0.6),S=diag(3)))
A <- matrix(runif(6),2,3)
b <- runif(2)
rPosteriorPredictive(obj = obj,n=20,A=A,b=b)
```

---

rT *Random Generation for (multivariate) t distribution*

Description

Generate random samples from a (multivariate) t distribution. For a random vector x, the density function is defined as:

$$Gamma((df+p)/2)/(Gamma(df/2)df^{p/2}pi^{p/2}|Sigma|^{1/2})[1+1/df(x-df)^T Sigma^{-1}(x-df)]^{-(df+p)/2}$$

Where p is the dimension of x.

Usage

```
rT(n, mu, Sigma = NULL, A = NULL, df = 1)
```

Arguments

n	integer, number of samples.
mu	numeric, mean vector.
Sigma	matrix, Sigma is proportional to the covariance matrix of x, one of Sigma and A should be non-NULL.
A	matrix, the Cholesky decomposition of Sigma, an upper triangular matrix, one of Sigma and A should be non-NULL.
df	numeric, degrees of freedom.

Value

A matrix of n rows and length(mu) columns, each row is a sample.

See Also

dT

Examples

```
x <- rT(1000,mu = c(1,1),Sigma = matrix(c(1,0.5,0.5,3),2,2))
plot(x)
```

---

rWishart *Random generation for Wishart distribution*

Description

Generate random samples from Wishart distribution. For a random matrix x, the density function of Wishart distribution is defined as:

$$(2^{(dfp)/2}Gamma_p(df/2)|rate|^{-df/2})^{-1}|x|^{(df-p-1)/2}exp(-1/2tr(xrate))$$

Where x is a pxp symmetric positive definite matrix, Gamma_p() is the multivariate Gamma function of dimension p.

Usage

```
rWishart(df, rate = NULL, scale = NULL)
```

Arguments

df	numeric, the degree of freedom.
rate	matrix, a symmetric positive-definite matrix, the 'rate', or 'inverse-scale' parameter. The 'rate' parameter in Wishart is the 'scale' parameter in InvWishart
scale,	matrix, the inverse of rate. Only one of 'rate' and 'scale' should be non-NULL.

Value

A symmetric positive-definite matrix.

References

Smith, W. B., and R. R. Hocking. "Algorithm as 53: Wishart variate generator." Journal of the Royal Statistical Society. Series C (Applied Statistics) 21.3 (1972): 341-345.

Examples

```
rate <- crossprod(matrix(rnorm(15),5,3)) #the prior inverse-scale
m <- matrix(0,3,3)
## get 1000 samples and calculate the sample mean
for(i in 1:100){
    m <- m+rWishart(df=5,rate=rate)/100
}
## m should roughly equal to df*inverse(rate):
m
pdsInverse(rate)*5
## try generating samples with 'rate' parameter:
scale <- pdsInverse(rate)
m2 <- matrix(0,3,3)
for(i in 1:100){
    m2 <- m2+rWishart(df=5,scale=scale)/100
}
## m2 should roughly equal df*scale:
m2
5*scale
```

---

sufficientStatistics *Get sample sufficient statistics*

Description

This is a generic function that will generate the sufficient statistics of a given Bayesian bricks object. i.e.
for the model structure:

$$theta|gamma \sim H(gamma)$$

$$x|theta \sim F(theta)$$

get the sufficient statistics T(x).
For a given sample set x, each row of x is an observation, and a Bayesian bricks object obj. `sufficientStatistics()` return the sufficient statistics for different model structures:

class(obj)="LinearGaussianGaussian":

$$x \sim Gaussian(Az + b, Sigma)$$

$$z \sim Gaussian(m, S)$$

The sufficient statistics are:

- SA = $sum_{i=1:N} A_i^T Sigma^{-1} A_i$
- SAx = $sum_{i=1:N} A_i^T Sigma^{-1}(x_i - b_i)$

See `?sufficientStatistics.LinearGaussianGaussian` for details.

class(obj)="GaussianGaussian": Where

$$x \sim Gaussian(mu, Sigma)$$

$$mu \sim Gaussian(m, S)$$

Sigma is known. The sufficient statistics are:

- N: the effective number of samples.
- xsum: the row sums of the samples.

See `?sufficientStatistics.GaussianGaussian` for details.

class(obj)="GaussianInvWishart": Where

$$x \sim Gaussian(mu, Sigma)$$

$$Sigma \sim InvWishart(v, S)$$

mu is known.
The sufficient statistics are:

- N: the effective number of samples.
- xsum: the sample scatter matrix centered on the mean vector.

See `?sufficientStatistics.GaussianInvWishart` for details.

class(obj)="GaussianNIW": Where

$$x \sim Gaussian(mu, Sigma)$$

$$Sigma \sim InvWishart(v, S)$$

$$mu \sim Gaussian(m, Sigma/k)$$

The sufficient statistics are:

- N: the effective number of samples.
- xsum: the row sums of the samples.
- S: the uncentered sample scatter matrix.

See `?sufficientStatistics.GaussianNIW` for details.

class(obj)="GaussianNIG": Where

$$x \sim Gaussian(Xbeta, sigma^2)$$

$$sigma^2 \sim InvGamma(a, b)$$

$$beta \sim Gaussian(m, sigma^2 V)$$

X is a row vector, or a design matrix where each row is an obervation. The sufficient statistics are:

- N: the effective number of samples.

- SXx: covariance of X and x
- SX: the uncentered sample scatter matrix.
- Sx: the variance of x

See `?sufficientStatistics.GaussianNIG` for details.

class(obj)="CatDirichlet": Where

$$x \sim Categorical(pi)$$

$$pi \sim Dirichlet(alpha)$$

The sufficient statistics of CatDirichlet object can either be x itself, or the counts of the unique labels in x.
See `?sufficientStatistics.CatDirichlet` for details.

class(obj)="CatDP": Where

$$x \sim Categorical(pi)$$

$$pi \sim DirichletProcess(alpha)$$

The sufficient statistics of CatDP object can either be x itself, or the counts of the unique labels in x.
See `?sufficientStatistics.CatDP` for details.

class(obj)="DP": Where

$$pi|alpha \sim DP(alpha, U)$$

$$z|pi \sim Categorical(pi)$$

$$theta_z|psi \sim H0(psi)$$

$$x|theta_z, z \sim F(theta_z)$$

The sufficient statistics of "DP" object is the same sufficient statistics of the "BasicBayesian" inside the "DP". See `?sufficientStatistics.DP` for details.

class(obj)="HDP": Where

$$G|gamma \sim DP(gamma, U)$$

$$pi_j|G, alpha \sim DP(alpha, G), j = 1 : J$$

$$z|pi_j \sim Categorical(pi_j)$$

$$k|z, G \sim Categorical(G), \text{ if z is a sample from the base measure G}$$

$$theta_k|psi \sim H0(psi)$$

The sufficient statistics of "HDP" object is the same sufficient statistics of the "BasicBayesian" inside the "HDP". See `?sufficientStatistics.HDP` for details.

class(obj)="HDP2": Where

$$G|eta \sim DP(eta, U)$$

$$G_m|gamma, G \sim DP(gamma, G), m = 1 : M$$

$$pi_{mj}|G_m, alpha \sim DP(alpha, G_m), j = 1 : J_m$$

$$z|pi_{mj} \sim Categorical(pi_{mj})$$

$$k|z, G_m \sim Categorical(G_m), \text{ if z is a sample from the base measure} G_m$$

$$u|k, G \sim Categorical(G), \text{ if k is a sample from the base measure} G$$

$$theta_u|psi \sim H0(psi)$$

$$x|theta_u, u \sim F(theta_u)$$

The sufficient statistics of "HDP2" object is the same sufficient statistics of the "BasicBayesian" inside the "HDP2". See `?sufficientStatistics.HDP2` for details.

Usage

```
sufficientStatistics(obj, x, ...)
```

Arguments

`obj`	a "BayesianBrick" object used to select a method.
`x`	a set of samples.
`...`	further arguments passed to or from other methods.

Value

An object of corresponding sufficient statistics class, such as "ssGaussian"

See Also

`sufficientStatistics.LinearGaussianGaussian` for Linear Gaussian and Gaussian conjugate structure, `sufficientStatistics.GaussianGaussian` for Gaussian-Gaussian conjugate structure, `sufficientStatistics.GaussianInvWishart` for Gaussian-Inverse-Wishart conjugate structure, `sufficientStatistics.GaussianNIW` for Gaussian-NIW conjugate structure, `sufficientStatistics.GaussianNIG` for Gaussian-NIG conjugate structure, `sufficientStatistics.CatDirichlet` for Categorical-Dirichlet conjugate structure, `sufficientStatistics.CatDP` for Categorical-DP conjugate structure ...

Examples

```
x <- rGaussian(10,mu = 1,Sigma = 1)
obj <- GaussianNIW()                    #an GaussianNIW object
sufficientStatistics(obj=obj,x=x)
```

`sufficientStatistics.CatDirichlet` *Sufficient statistics of a "CatDirichlet" object*

Description

For following Categorical-Dirichlet model structure:

$$pi|alpha \sim Dir(alpha)$$

$$x|pi \sim Categorical(pi)$$

Where Dir() is the Dirichlet distribution, Categorical() is the Categorical distribution. See `?dDir` and `dCategorical` for the definitions of these distribution.
The sufficient statistics of a set of samples x is:

- the effective counts of each unique label in x. i.e. T(x)[i] = sum(uniqueLabels[i]

Usage

```
## S3 method for class 'CatDirichlet'
sufficientStatistics(obj, x, foreach = FALSE, ...)
```

Arguments

`obj`	A "CatDirichlet" object.
`x`	numeric,integer or character, samples of the Categorical distribution.
`foreach`	logical, specifying whether to return the sufficient statistics for each observation. Default FALSE.
`...`	Additional arguments to be passed to other inherited types.

Value

An object of class "ssCat", the sufficient statistics of a set of categorical samples. Or an object of the same class as x if foreach=TRUE.

References

Murphy, Kevin P. Machine learning: a probabilistic perspective. MIT press, 2012.

See Also

`CatDirichlet`, `sufficientStatistics_Weighted.CatDirichlet`

Examples

```
obj <- CatDirichlet(gamma=list(alpha=runif(26,1,2),uniqueLabels = letters))
x <- sample(letters,size = 20,replace = TRUE)
w <- runif(20)
sufficientStatistics(obj=obj,x=x)       #return the counts of each unique label
sufficientStatistics_Weighted(obj=obj,x=x,w=w) #return the weighted counts of each unique lable
sufficientStatistics(obj=obj,x=x,foreach = TRUE) #return the sample itself
```

---

sufficientStatistics.CatDP

Sufficient statistics of a "CatDP" object

Description

For following model structure:

$$pi|alpha \sim DP(alpha, U)$$

$$x|pi \sim Categorical(pi)$$

where DP(alpha,U) is a Dirichlet Process on positive integers, alpha is the "concentration parameter" of the Dirichlet Process, U is the "base measure" of this Dirichlet process, it is an uniform distribution on all positive integers.Categorical() is the Categorical distribution. See `dCategorical` for the definition of the Categorical distribution.
In the case of CatDP, x can only be positive integers.
The model structure and prior parameters are stored in a "CatDP" object.
The sufficient statistics of a set of samples x is:

- unique positive integer values
- effective counts of the unique positive integers

Usage

```
## S3 method for class 'CatDP'
sufficientStatistics(obj, x, foreach = FALSE, ...)
```

Arguments

`obj`	A "CatDP" object.
`x`	integer, the elements of the vector must all greater than 0, the samples of a Categorical distribution.
`foreach`	logical, specifying whether to return the sufficient statistics for each observation. Default FALSE.
`...`	Additional arguments to be passed to other inherited types.

Value

An object of class "ssCatDP", the sufficient statistics of a set of categorical samples. Or an integer vector same as x if foreach=TRUE.

References

Teh, Yee W., et al. "Sharing clusters among related groups: Hierarchical Dirichlet processes." Advances in neural information processing systems. 2005.

See Also

CatDP, sufficientStatistics_Weighted.CatDP

Examples

```
x <- sample(1L:10L,size = 4,replace = TRUE)
obj <- CatDP()
## an object of class "ssCatDP", which contains the counts of each unique integer
sufficientStatistics(obj=obj,x=x)
## will return x itself
sufficientStatistics(obj=obj,x=x,foreach = TRUE)
```

`sufficientStatistics.DP`

Sufficient statistics of a "DP" object

Description

For following model structure:

$$pi|alpha \sim DP(alpha, U)$$

$$z|pi \sim Categorical(pi)$$

$$theta_z|psi \sim H0(psi)$$

$$x|theta_z, z \sim F(theta_z)$$

where DP(alpha,U) is a Dirichlet Process on positive integers, alpha is the "concentration parameter" of the Dirichlet Process, U is the "base measure" of this Dirichlet process. The choice of F() and H0() can be described by an arbitrary "BasicBayesian" object such as "GaussianGaussian","GaussianInvWishart","GaussianNIW", "GaussianNIG", "CatDirichlet", and "CatDP". See `?BasicBayesian` for definition of "BasicBayesian" objects, and see for example `?GaussianGaussian` for specific "BasicBayesian" instances. As a summary, An "DP" object is simply a combination of a "CatDP" object (see `?CatDP`) and an object of any "BasicBayesian" type.
The sufficient statistics of a set of samples x in a "DP" object is the same sufficient statistics of the "BasicBayesian" inside the "DP", see examples.

Usage

```
## S3 method for class 'DP'
sufficientStatistics(obj, x, ...)
```

Arguments

obj	A "DP" object.
x	Random samples of the "BasicBayesian" object.
...	Additional arguments to be passed to other inherited types.

Value

Return the sufficient statistics of the corresponding BasicBayesian type, see examples.

References

Teh, Yee W., et al. "Sharing clusters among related groups: Hierarchical Dirichlet processes." Advances in neural information processing systems. 2005.

See Also

DP, sufficientStatistics_Weighted.DP

Examples

```
obj1 <- DP(gamma=list(alpha=1,H0aF="GaussianNIW",parH0=list(m=1,k=1,v=1,S=1)))
obj2 <- DP(gamma=list(alpha=1,H0aF="CatDirichlet",parH0=list(alpha=1,uniqueLabels=letters)))
x1 <- rnorm(100)
x2 <- sample(letters,100,replace = TRUE)
sufficientStatistics(obj = obj1,x=x1)
sufficientStatistics(obj = obj2,x=x2)
sufficientStatistics(obj = obj1,x=x1,foreach = TRUE)
```

---

sufficientStatistics.GaussianGaussian

Sufficient statistics of a "GaussianGaussian" object

Description

For following model structure:

$$x \sim Gaussian(mu, Sigma)$$

$$mu \sim Gaussian(m, S)$$

Where Sigma is known. Gaussian() is the Gaussian distribution. See ?dGaussian for the definition of Gaussian distribution.
The sufficient statistics of a set of samples x (each row of x is a sample) are:

- the effective number of samples N=nrow(x)
- the sample sum xsum = colSums(x)

Usage

```
## S3 method for class 'GaussianGaussian'
sufficientStatistics(obj, x, foreach = FALSE, ...)
```

Arguments

obj	A "GaussianGaussian" object.
x	matrix, Gaussian samples, when x is a matrix, each row is a sample of dimension ncol(x). when x is a vector, x is length(x) samples of dimension 1.
foreach	logical, specifying whether to return the sufficient statistics for each observation. Default FALSE.
...	Additional arguments to be passed to other inherited types.

Value

If foreach=TRUE, will return a list of sufficient statistics for each row of x, otherwise will return the sufficient statistics of x as a whole.

References

Gelman, Andrew, et al. Bayesian data analysis. CRC press, 2013.

See Also

GaussianGaussian, sufficientStatistics_Weighted.GaussianGaussian

Examples

```
obj <- GaussianGaussian(gamma=list(Sigma=matrix(c(2,1,1,2),2,2),m=c(0.2,0.5),S=diag(2)))
x <- rGaussian(100,c(0,0),Sigma = matrix(c(2,1,1,2),2,2))
sufficientStatistics(obj = obj,x=x)
sufficientStatistics(obj = obj,x=x,foreach=TRUE)
```

---

sufficientStatistics.GaussianInvWishart

Sufficient statistics of a "GaussianInvWishart" object

Description

For following model structure:

$$x \sim Gaussian(mu, Sigma)$$

$$Sigma \sim InvWishart(v, S)$$

mu is known. Gaussian() is the Gaussian distribution. See ?dGaussian and ?dInvWishart for the definition of the distributions.
The sufficient statistics of a set of samples x (each row of x is a sample) are:

- the effective number of samples N=nrow(x)
- the centered sample scatter matrix S = (t(x)-mu)^T

Usage

```
## S3 method for class 'GaussianInvWishart'
sufficientStatistics(obj, x, foreach = FALSE, ...)
```

Arguments

obj	A "GaussianInvWishart" object.
x	matrix, Gaussian samples, when x is a matrix, each row is a sample of dimension ncol(x). when x is a vector, x is length(x) samples of dimension 1.
foreach	logical, specifying whether to return the sufficient statistics for each observation. Default FALSE.
...	Additional arguments to be passed to other inherited types.

Value

If foreach=TRUE, will return a list of sufficient statistics for each row of x, otherwise will return the sufficient statistics of x as a whole.

References

Gelman, Andrew, et al. Bayesian data analysis. CRC press, 2013.

MARolA, K. V., JT KBNT, and J. M. Bibly. Multivariate analysis. AcadeInic Press, Londres, 1979.

See Also

GaussianInvWishart, sufficientStatistics_Weighted.GaussianInvWishart

Examples

```
obj <- GaussianInvWishart(gamma=list(mu=c(-1.5,1.5),v=3,S=diag(2)))
x <- rGaussian(10,mu = c(-1.5,1.5),Sigma = matrix(c(0.1,0.03,0.03,0.1),2,2))
sufficientStatistics(obj=obj,x=x,foreach = FALSE)
sufficientStatistics(obj=obj,x=x,foreach = TRUE)
```

`sufficientStatistics.GaussianNIG` *Sufficient statistics of a "GaussianNIG" object*

Description

For following Gaussian-NIG model structure:

$$x \sim Gaussian(Xbeta, sigma^2)$$

$$sigma^2 \sim InvGamma(a, b)$$

$$beta \sim Gaussian(m, sigma^2 V)$$

Where X is a row vector, or a design matrix where each row is an obervation. InvGamma() is the Inverse-Gamma distribution, Gaussian() is the Gaussian distribution. See `?dInvGamma` and `dGaussian` for the definitions of these distribution.
The model structure and prior parameters are stored in a "GaussianNIG" object.
The sufficient statistics of a set of samples (x,X) are:

- the effective number of samples N=nrow(X) or length(x)
- the covariance of X and x SXx=t(X)
- the covariance of X SX=t(X)
- the covariance of x Sx=t(x)

Usage

```
## S3 method for class 'GaussianNIG'
sufficientStatistics(obj, x, X, foreach = FALSE, ...)
```

Arguments

`obj`	A "GaussianNIG" object.
`x`	numeric, must satisfy length(x) = nrow(X)
`X`	matrix, must satisfy length(x) = nrow(X)
`foreach`	logical, if foreach=TRUE, will return a list of sufficient statistics for each (x,X), otherwise will return the sufficient statistics as a whole.
`...`	Additional arguments to be passed to other inherited types.

Value

If foreach=TRUE, will return a list of sufficient statistics for each row of (x,X), otherwise will return the sufficient statistics of (x,X) as a whole.

References

Banerjee, Sudipto. "Bayesian Linear Model: Gory Details." Downloaded from http://www. biostat. umn. edu/~ph7440 (2008).

See Also

GaussianNIG, sufficientStatistics_Weighted.GaussianNIG

Examples

```
obj <- GaussianNIG(gamma=list(m=0,V=1,a=1,b=0))
X <- 1:20
x <- rnorm(20)+ X*0.3
sufficientStatistics(obj = obj,X=X,x=x)
sufficientStatistics(obj = obj,X=X,x=x,foreach = TRUE)
```

`sufficientStatistics.GaussianNIW`
Sufficient statistics of a "GaussianNIW" object

Description

For following Gaussian-NIW model structure:

$$mu, Sigma|m, k, v, S \sim NIW(m, k, v, S)$$

$$x|mu, Sigma \sim Gaussian(mu, Sigma)$$

Where NIW() is the Normal-Inverse-Wishart distribution, Gaussian() is the Gaussian distribution. See `?dNIW` and `dGaussian` for the definitions of these distribution.
The sufficient statistics of a set of samples x (each row of x is a sample) are:

- the effective number of samples N=nrow(x)
- the sample sum xsum = colSums(x)
- the uncentered scatter matrix S = t(x)

Usage

```
## S3 method for class 'GaussianNIW'
sufficientStatistics(obj, x, foreach = FALSE, ...)
```

Arguments

`obj`	A "GaussianNIW" object.
`x`	matrix, Gaussian samples, when x is a matrix, each row is a sample of dimension ncol(x). when x is a vector, x is length(x) samples of dimension 1.
`foreach`	logical, if foreach=TRUE, will return a list of sufficient statistics for each row of x, otherwise will return the sufficient statistics of x as a whole.
`...`	Additional arguments to be passed to other inherited types.

Value

If foreach=TRUE, will return a list of sufficient statistics for each row of x, otherwise will return the sufficient statistics of x as a whole.

References

Murphy, Kevin P. "Conjugate Bayesian analysis of the Gaussian distribution." def 1.22 (2007): 16.

Gelman, Andrew, et al. "Bayesian Data Analysis Chapman & Hall." CRC Texts in Statistical Science (2004).

See Also

GaussianNIW, sufficientStatistics_Weighted.GaussianNIW

Examples

```
x <- rGaussian(10,mu = c(-1.5,1.5),Sigma = matrix(c(0.1,0.03,0.03,0.1),2,2))
obj <- GaussianNIW()                    #an GaussianNIW object
sufficientStatistics(obj=obj,x=x,foreach = FALSE)
sufficientStatistics(obj=obj,x=x,foreach = TRUE)
```

---

sufficientStatistics.HDP

Sufficient statistics of a "HDP" object

Description

For following model structure:

$$G|gamma \sim DP(gamma, U)$$
$$pi_j|G, alpha \sim DP(alpha, G), j = 1:J$$
$$z|pi_j \sim Categorical(pi_j)$$
$$k|z, G \sim Categorical(G),\ \text{if z is a sample from the base measure G}$$
$$theta_k|psi \sim H0(psi)$$
$$x|theta_k, k \sim F(theta_k)$$

where DP(gamma,U) is a Dirichlet Process on positive integers, gamma is the "concentration parameter", U is the "base measure" of this Dirichlet process, U is an uniform distribution on all positive integers. DP(alpha,G) is a Dirichlet Process on integers with concentration parameter alpha and base measure G. The choice of F() and H0() can be described by an arbitrary "BasicBayesian" object such as "GaussianGaussian","GaussianInvWishart","GaussianNIW", "GaussianNIG", "CatDirichlet", and "CatDP". See ?BasicBayesian for definition of "BasicBayesian" objects, and see for example ?GaussianGaussian for specific "BasicBayesian" instances. As a summary, An "HDP" object is simply a combination of a "CatHDP" object (see ?CatHDP) and an object of any "BasicBayesian" type.
In the case of HDP, z and k can only be positive integers.
The sufficient statistics of a set of samples x in a "HDP" object is the same sufficient statistics of the "BasicBayesian" inside the "HDP", see examples.

Usage

```
## S3 method for class 'HDP'
sufficientStatistics(obj, x, ...)
```

Arguments

obj	A "HDP" object.
x	Random samples of the "BasicBayesian" object.
...	further arguments passed to the corresponding sufficientStatistics method of the "BasicBayesian" object.

Value

Return the sufficient statistics of the corresponding BasicBayesian type, see examples.

References

Teh, Yee W., et al. "Sharing clusters among related groups: Hierarchical Dirichlet processes." Advances in neural information processing systems. 2005.

See Also

HDP, sufficientStatistics_Weighted.HDP

Examples

```
## a HDP with Gaussian NIW observations
obj1 <- HDP(gamma=list(gamma=1,alpha=1,j=2,
                       H0aF="GaussianNIW",
                       parH0=list(m=0,k=1,v=2,S=1)))
## a HDP with Categorical-Dirichlet observations
obj2 <- HDP(gamma=list(gamma=1,alpha=1,j=2,
            H0aF="CatDirichlet",
            parH0=list(alpha=1,uniqueLabels=letters[1:3])))
x1 <- rnorm(100)
x2 <- sample(letters[1:3],100,replace = TRUE)
sufficientStatistics(obj = obj1,x=x1,foreach = TRUE)
sufficientStatistics(obj = obj1,x=x1,foreach = FALSE)
sufficientStatistics(obj = obj2,x=x2,foreach = FALSE)
```

---

sufficientStatistics.HDP2

Sufficient statistics of a "HDP2" object

Description

For following model structure:

$$G|eta \sim DP(eta, U)$$

$$G_m|gamma, G \sim DP(gamma, G), m = 1 : M$$

$$pi_{mj}|G_m, alpha \sim DP(alpha, G_m), j = 1 : J_m$$

$$z|pi_{mj} \sim Categorical(pi_{mj})$$

$$k|z, G_m \sim Categorical(G_m), \text{ if z is a sample from the base measure } G_{mj}$$

$$u|k, G \sim Categorical(G), \text{ if k is a sample from the base measure G}$$

$$theta_u|psi \sim H0(psi)$$

$$x|theta_u, u \sim F(theta_u)$$

where DP(eta,U) is a Dirichlet Process on positive integers, eta is the "concentration parameter", U is the "base measure" of this Dirichlet process, U is an uniform distribution on all positive integers. DP(gamma,G) is a Dirichlet Process on integers with concentration parameter gamma and base measure G. DP(alpha,G_m) is a Dirichlet Process on integers with concentration parameter alpha and base measure G_m. The choice of F() and H0() can be described by an arbitrary "BasicBayesian" object such as "GaussianGaussian","GaussianInvWishart","GaussianNIW", "GaussianNIG", "CatDirichlet", and "CatDP". See `?BasicBayesian` for definition of "BasicBayesian" objects, and see for example `?GaussianGaussian` for specific "BasicBayesian" instances. As a summary, An "HDP2" object is simply a combination of a "CatHDP2" object (see `?CatHDP2`) and an object of any "BasicBayesian" type.
In the case of HDP2, u, z and k can only be positive integers.
The sufficient statistics of a set of samples x in a "HDP2" object is the same sufficient statistics of the "BasicBayesian" inside the "HDP2", see examples.

Usage

```
## S3 method for class 'HDP2'
sufficientStatistics(obj, x, ...)
```

Arguments

`obj`	A "HDP2" object.
`x`	Random samples of the "BasicBayesian" object.
`...`	further arguments passed to the corresponding sufficientStatistics method of the "BasicBayesian" object.

Value

Return the sufficient statistics of the corresponding BasicBayesian type, see examples.

References

Teh, Yee W., et al. "Sharing clusters among related groups: Hierarchical Dirichlet processes." Advances in neural information processing systems. 2005.

See Also

HDP2, sufficientStatistics_Weighted.HDP2

Examples

```
## a HDP2 with Gaussian NIW observations
obj1 <- HDP2(gamma=list(gamma=1,alpha=1,j=2,m=2,
                        H0aF="GaussianNIW",
                        parH0=list(m=0,k=1,v=2,S=1)))
## a HDP2 with Categorical-Dirichlet observations
obj2 <- HDP2(gamma=list(gamma=1,alpha=1,j=2,m=2,
                        H0aF="CatDirichlet",
                        parH0=list(alpha=1,uniqueLabels=letters[1:3])))
x1 <- rnorm(100)
x2 <- sample(letters[1:3],100,replace = TRUE)
sufficientStatistics(obj = obj1,x=x1,foreach = FALSE)
sufficientStatistics(obj = obj2,x=x2,foreach = FALSE)
sufficientStatistics(obj = obj1,x=x1,foreach = TRUE)
```

---

sufficientStatistics.LinearGaussianGaussian

Sufficient statistics of a "LinearGaussianGaussian" object

Description

For following model structure:

$$x \sim Gaussian(Az + b, Sigma)$$

$$z \sim Gaussian(m, S)$$

Where Sigma is known. A is a $dimxxdimz$ matrix, x is a $dimxx1$ random vector, z is a $dimzx1$ random vector, b is a $dimmx1$ vector. Gaussian() is the Gaussian distribution. See ?dGaussian for the definition of Gaussian distribution.
For one dimensional observations: x is a vector of length N, or a $Nx1$ matrix, each row is an observation; A is a $Nxdimz$ matrix; b is a length N vector. The sufficient statistics are:

- SA = $A^T A/Sigma$
- SAx = $A^T(x-b)/Sigma$

For $dimx$ dimensional observations: x must be a Nxm matrix, each row is an observation; A can be either a list or a matrix. When A is a list, $A = A_1, A_2, ...A_N$ is a list of $dimxxdimz$ matrices. If A is a single $dimxxdimz$ matrix, it will be replicated N times into a length N list; b can be either a matrix or a vector. When b is a matrix, $b = b_1^T, ..., b_N^T$ is a $Nxdimx$ matrix, each row is a transposed vector. When b is a length $dimx$ vector, it will be transposed into a row vector and replicated N times into a $Nxdimx$ matrix. The sufficient statistics are:

- SA = $sum_{i=1:N} A_i^T Sigma^{-1} A_i$
- SAx = $sum_{i=1:N} A_i^T Sigma^{-1}(x_i - b_i)$

Usage

```
## S3 method for class 'LinearGaussianGaussian'
sufficientStatistics(obj, x, A, b = NULL, foreach = FALSE, ...)
```

Arguments

obj	A "LinearGaussianGaussian" object.
x	matrix, Gaussian samples, when x is a matrix, each row is a sample of dimension ncol(x). when x is a vector, x is length(x) samples of dimension 1.
A	matrix or list. when x is a $Nx1$ matrix, A must be a matrix of $Nxdimz$, dimz is the dimension of z; When x is a $Nxdimx$ matrix, where $dimx > 1$, A can be either a list or a matrix. When A is a list, $A = A_1, A_2, ...A_N$ is a list of $dimxxdimz$ matrices. If A is a single $dimxxdimz$ matrix, it will be replicated N times into a length N list
b	matrix, when x is a $Nx1$ matrix, b must also be a $Nx1$ matrix or length N vector; When x is a $Nxdimx$ matrix, where $dimx > 1$, b can be either a matrix or a vector. When b is a matrix, $b = b_1^T, ..., b_N^T$ is a $Nxdimx$ matrix, each row is a transposed vector. When b is a length $dimx$ vector, it will be transposed into a row vector and replicated N times into a $Nxdimx$ matrix. When b = NULL, it will be treated as a vector of zeros. Default NULL.
foreach	logical, specifying whether to return the sufficient statistics for each observation. Default FALSE.
...	Additional arguments to be passed to other inherited types.

Value

If foreach=TRUE, will return a list of sufficient statistics for each row of x, otherwise will return the sufficient statistics of x as a whole.

References

Murphy, Kevin P. Machine learning: a probabilistic perspective. MIT press, 2012.

See Also

LinearGaussianGaussian, sufficientStatistics_Weighted.LinearGaussianGaussian

Examples

```
## create a LinearGaussianGaussian object
## where x is 2 dimensional, z is 3 dimensional
obj <- LinearGaussianGaussian(gamma=list(Sigma=matrix(c(2,1,1,2),2,2),
                              m=c(0.2,0.5,0.6),S=diag(3)))
x <- rGaussian(100,mu = runif(2),Sigma = diag(2))
A <- matrix(runif(6),2,3)
b <- runif(2)
sufficientStatistics(obj,x=x,A=A,b=b)
Alist <- replicate(100,A,simplify=FALSE)
```

```
## should print the same thing as above:
sufficientStatistics(obj,x=x,A=Alist,b=b)
```

`sufficientStatistics_Weighted`
Get weighted sample sufficient statistics

Description

This is a generic function that will generate the weighted sufficient statistics of a given "Bayesian-Brick" object. That is, for the model structure:

$$theta|gamma \sim H(gamma)$$

$$x|theta \sim F(theta)$$

get the weighted sufficient statistics T(x). For a given sample set x, each row of x is an observation, the sample weights w, and a Bayesian bricks object obj. `sufficientStatistics_Weighted()` return the weighted sufficient statistics for different model structures:

class(obj)="LinearGaussianGaussian":

$$x \sim Gaussian(Az + b, Sigma)$$

$$z \sim Gaussian(m, S)$$

The sufficient statistics are:

- SA = $sum_{i=1:N} w_i A_i^T Sigma^{-1} A_i$
- SAx = $sum_{i=1:N} w_i A_i^T Sigma^{-1} (x_i - b_i)$

See `?sufficientStatistics.LinearGaussianGaussian` for details.

class(obj)="GaussianGaussian": Where

$$x \sim Gaussian(mu, Sigma)$$

$$mu \sim Gaussian(m, S)$$

Sigma is known. The sufficient statistics are:

- N: the effective number of samples.
- xsum: the row sums of the samples.

See `?sufficientStatistics_Weighted.GaussianGaussian` for details.

class(obj)="GaussianInvWishart": Where

$$x \sim Gaussian(mu, Sigma)$$

$$Sigma \sim InvWishart(v, S)$$

mu is known.
The sufficient statistics are:

- N: the effective number of samples.
- xsum: the sample scatter matrix centered on the mean vector.

See ?sufficientStatistics_Weighted.GaussianInvWishart for details.

class(obj)="GaussianNIW": Where

$$x \sim Gaussian(mu, Sigma)$$

$$Sigma \sim InvWishart(v, S)$$

$$mu \sim Gaussian(m, Sigma/k)$$

The sufficient statistics are:

- N: the effective number of samples.
- xsum: the row sums of the samples.
- S: the uncentered sample scatter matrix.

See ?sufficientStatistics_Weighted.GaussianNIW for details.

class(obj)="GaussianNIG": Where

$$x \sim Gaussian(Xbeta, sigma^2)$$

$$sigma^2 \sim InvGamma(a, b)$$

$$beta \sim Gaussian(m, sigma^2 V)$$

X is a row vector, or a design matrix where each row is an obervation. The sufficient statistics are:

- N: the effective number of samples.
- SXx: covariance of X and x
- SX: the uncentered sample scatter matrix.
- Sx: the variance of x

See ?sufficientStatistics_Weighted.GaussianNIG for details.

class(obj)="CatDirichlet": Where

$$x \sim Categorical(pi)$$

$$pi \sim Dirichlet(alpha)$$

The sufficient statistics of CatDirichlet object can either be x itself, or the counts of the unique labels in x.
See ?sufficientStatistics_Weighted.CatDirichlet for details.

class(obj)="CatDP": Where

$$x \sim Categorical(pi)$$

$$pi \sim DirichletProcess(alpha)$$

The sufficient statistics of CatDP object can either be x itself, or the counts of the unique labels in x.
See ?sufficientStatistics_Weighted.CatDP for details.

class(obj)="DP": Where

$$pi|alpha \sim DP(alpha, U)$$

$$z|pi \sim Categorical(pi)$$

$$theta_z|psi \sim H0(psi)$$

$$x|theta_z, z \sim F(theta_z)$$

The sufficient statistics of "DP" object is the same sufficient statistics of the "BasicBayesian" inside the "DP". See `?sufficientStatistics_Weighted.DP` for details.

class(obj)="HDP": Where

$$G|gamma \sim DP(gamma, U)$$

$$pi_j|G, alpha \sim DP(alpha, G), j = 1 : J$$

$$z|pi_j \sim Categorical(pi_j)$$

$$k|z, G \sim Categorical(G), \text{ if z is a sample from the base measure G}$$

$$theta_k|psi \sim H0(psi)$$

The sufficient statistics of "HDP" object is the same sufficient statistics of the "BasicBayesian" inside the "HDP". See `?sufficientStatistics_Weighted.HDP` for details.

class(obj)="HDP2": Where

$$G|eta \sim DP(eta, U)$$

$$G_m|gamma, G \sim DP(gamma, G), m = 1 : M$$

$$pi_{mj}|G_m, alpha \sim DP(alpha, G_m), j = 1 : J_m$$

$$z|pi_{mj} \sim Categorical(pi_{mj})$$

$$k|z, G_m \sim Categorical(G_m), \text{ if z is a sample from the base measure } G_m$$

$$u|k, G \sim Categorical(G), \text{ if k is a sample from the base measure } G_m$$

$$theta_u|psi \sim H0(psi)$$

$$x|theta_u, u \sim F(theta_u)$$

The sufficient statistics of "HDP2" object is the same sufficient statistics of the "BasicBayesian" inside the "HDP2". See `?sufficientStatistics_Weighted.HDP2` for details.

Usage

```
sufficientStatistics_Weighted(obj, x, w, ...)
```

Arguments

`obj`	a "BayesianBrick" object used to select a method.
`x`	a set of samples.
`w`	numeric, sample weights.
`...`	further arguments passed to or from other methods.

Value

An object of corresponding sufficient statistics class, such as "ssGaussian"

See Also

sufficientStatistics_Weighted.LinearGaussianGaussian for Linear Gaussian and Gaussian conjugate structure, sufficientStatistics_Weighted.GaussianGaussian for Gaussian-Gaussian conjugate structure, sufficientStatistics_Weighted.GaussianInvWishart for Gaussian-Inverse-Wishart conjugate structure, sufficientStatistics_Weighted.GaussianNIW for Gaussian-NIW conjugate structure, sufficientStatistics_Weighted.GaussianNIG for Gaussian-NIG conjugate structure, sufficientStatistics_Weighted.CatDirichlet for Categorical-Dirichlet conjugate structure, sufficientStatistics_Weighted.CatDP for Categorical-DP conjugate structure ...

Examples

```
x <- rGaussian(10,mu = 1,Sigma = 1)
w <- runif(10)
obj <- GaussianNIW()                    #an GaussianNIW object
sufficientStatistics_Weighted(obj=obj,x=x,w=w)
```

---

sufficientStatistics_Weighted.CatDirichlet

Weighted sufficient statistics of a "CatDirichlet" object

Description

For following Categorical-Dirichlet model structure:

$$pi|alpha \sim Dir(alpha)$$

$$x|pi \sim Categorical(pi)$$

Where Dir() is the Dirichlet distribution, Categorical() is the Categorical distribution. See ?dDir and dCategorical for the definitions of these distribution.
the sufficient statistics of a set of samples x and weights w are:
the effective counts (in this case the sum of the weight w) of each unique label in x
Unique values of x must be in obj$gamma$uniqueLabels, where "obj" is a "CatDirichlet" object, see examples below.

Usage

```
## S3 method for class 'CatDirichlet'
sufficientStatistics_Weighted(obj, x, w, foreach = FALSE, ...)
```

Arguments

obj	A "CatDirichlet" object.
x	numeric,integer or character, samples of the Categorical distribution.
w	numeric, sample weights.
foreach	logical, specifying whether to return the sufficient statistics for each observation. Default FALSE.
...	Additional arguments to be passed to other inherited types.

Value

An object of class "ssCat", the sufficient statistics of a set of categorical samples. Or an object of the same class as x if foreach=TRUE.

References

Murphy, Kevin P. Machine learning: a probabilistic perspective. MIT press, 2012.

See Also

sufficientStatistics.CatDirichlet CatDirichlet

Examples

```
obj <- CatDirichlet(gamma=list(alpha=runif(26,1,2),uniqueLabels = letters))
x <- sample(letters,size = 20,replace = TRUE)
w <- runif(20)
sufficientStatistics(obj=obj,x=x)       #return the counts of each unique label
sufficientStatistics_Weighted(obj=obj,x=x,w=w) #return the weighted counts of each unique lable
```

---

sufficientStatistics_Weighted.CatDP

Weighted sufficient statistics of a "CatDP" object

Description

For following model structure:

$$pi|alpha \sim DP(alpha, U)$$

$$x|pi \sim Categorical(pi)$$

where DP(alpha,U) is a Dirichlet Process on positive integers, alpha is the "concentration parameter" of the Dirichlet Process, U is the "base measure" of this Dirichlet process, it is an uniform distribution on all positive integers.Categorical() is the Categorical distribution. See dCategorical for the definition of the Categorical distribution.
In the case of CatDP, x can only be positive integers.
The model structure and prior parameters are stored in a "CatDP" object.
The sufficient statistics of a set of samples x is:

- unique positive integer values
- effective counts of the unique positive integers

Usage

```
## S3 method for class 'CatDP'
sufficientStatistics_Weighted(obj, x, w, foreach = FALSE, ...)
```

Arguments

obj	A "CatDP" object.
x	integer, the elements of the vector must all greater than 0, the samples of a Categorical distribution.
w	numeric, sample weights
foreach	logical, specifying whether to return the sufficient statistics for each observation. Default FALSE.
...	Additional arguments to be passed to other inherited types.

Value

An object of class "ssCatDP", the sufficient statistics of a set of categorical samples. Or an integer vector same as x if foreach=TRUE.

References

Teh, Yee W., et al. "Sharing clusters among related groups: Hierarchical Dirichlet processes." Advances in neural information processing systems. 2005.

See Also

CatDP, sufficientStatistics.CatDP

Examples

```
x <- sample(1L:10L,size = 4,replace = TRUE)
obj <- CatDP()
w <- runif(4)
## return an object of class "ssCatDP" contains the weighted counts of each unique integer
sufficientStatistics_Weighted(obj=obj,x=x,w=w)
## return x itself, no matter what w is
sufficientStatistics_Weighted(obj=obj,x=x,w=w,foreach = TRUE)
```

sufficientStatistics_Weighted.DP
Weighted sufficient statistics of a "DP" object

Description

For following model structure:

$$pi|alpha \sim DP(alpha, U)$$

$$z|pi \sim Categorical(pi)$$

$$theta_z|psi \sim H0(psi)$$

$$x|theta_z, z \sim F(theta_z)$$

where DP(alpha,U) is a Dirichlet Process on positive integers, alpha is the "concentration parameter" of the Dirichlet Process, U is the "base measure" of this Dirichlet process. The choice of F() and H0() can be described by an arbitrary "BasicBayesian" object such as "GaussianGaussian","GaussianInvWishart","GaussianNIW", "GaussianNIG", "CatDirichlet", and "CatDP". See `?BasicBayesian` for definition of "BasicBayesian" objects, and see for example `?GaussianGaussian` for specific "BasicBayesian" instances. As a summary, An "DP" object is simply a combination of a "CatDP" object (see `?CatDP`) and an object of any "BasicBayesian" type.
The sufficient statistics of a set of samples x in a "DP" object is the same sufficient statistics of the "BasicBayesian" inside the "DP", see examples.

Usage

```
## S3 method for class 'DP'
sufficientStatistics_Weighted(obj, x, w, ...)
```

Arguments

`obj`	A "DP" object.
`x`	Random samples of the "BasicBayesian" object.
`w`	numeric, sample weights.
`...`	Additional arguments to be passed to other inherited types.

Value

Return the sufficient statistics of the corresponding BasicBayesian type, see examples.

References

Teh, Yee W., et al. "Sharing clusters among related groups: Hierarchical Dirichlet processes." Advances in neural information processing systems. 2005.

See Also

DP, sufficientStatistics.DP

Examples

```
obj1 <- DP(gamma=list(alpha=1,H0aF="GaussianNIW",parH0=list(m=1,k=1,v=1,S=1)))
obj2 <- DP(gamma=list(alpha=1,H0aF="CatDirichlet",parH0=list(alpha=1,uniqueLabels=letters)))
x1 <- rnorm(100)
x2 <- sample(letters,100,replace = TRUE)
w <- runif(100)
sufficientStatistics_Weighted(obj = obj1,x=x1,w=w)
sufficientStatistics_Weighted(obj = obj2,x=x2,w=w)
sufficientStatistics_Weighted(obj = obj1,x=x1,w=w,foreach = TRUE)
```

---

sufficientStatistics_Weighted.GaussianGaussian

Weighted sufficient statistics of a "GaussianGaussian" object

---

Description

For following model structure:

$$x \sim Gaussian(mu, Sigma)$$

$$mu \sim Gaussian(m, S)$$

Where Sigma is known. Gaussian() is the Gaussian distribution. See ?dGaussian for the definition of Gaussian distribution.
The sufficient statistics of a set of samples x (each row of x is a sample) and weights w are:

- the effective number of samples N=sum(w)
- the sample sum xsum = colSums(x*w)

Usage

```
## S3 method for class 'GaussianGaussian'
sufficientStatistics_Weighted(obj, x, w, foreach = FALSE, ...)
```

Arguments

obj	A "GaussianGaussian" object.
x	matrix, Gaussian samples, when x is a matrix, each row is a sample of dimension ncol(x). when x is a vector, x is length(x) samples of dimension 1.
w	numeric, sample weights.
foreach	logical, specifying whether to return the sufficient statistics for each observation. Default FALSE.
...	Additional arguments to be passed to other inherited types.

Value

If foreach=TRUE, will return a list of sufficient statistics for each row of x, otherwise will return the sufficient statistics of x as a whole.

References

Gelman, Andrew, et al. Bayesian data analysis. CRC press, 2013.

See Also

GaussianGaussian, sufficientStatistics.GaussianGaussian

Examples

```
obj <- GaussianGaussian(gamma=list(Sigma=matrix(c(2,1,1,2),2,2),m=c(0.2,0.5),S=diag(2)))
x <- rGaussian(100,c(0,0),Sigma = matrix(c(2,1,1,2),2,2))
w <- runif(100)
sufficientStatistics_Weighted(obj=obj,x=x,w=w,foreach = FALSE)
sufficientStatistics_Weighted(obj=obj,x=x,w=w,foreach = TRUE)
```

---

sufficientStatistics_Weighted.GaussianInvWishart

Weighted sufficient statistics of a "GaussianInvWishart" object

---

Description

For following model structure:

$$x \sim Gaussian(mu, Sigma)$$

$$Sigma \sim InvWishart(v, S)$$

mu is known. Gaussian() is the Gaussian distribution. See ?dGaussian and ?dInvWishart for the definition of the distributions.
The sufficient statistics of a set of samples x (each row of x is a sample) and weights w are:

- the effective number of samples N=sum(w)
- the centered sample scatter matrix S = (w*(t(x)-mu))^T

Usage

```
## S3 method for class 'GaussianInvWishart'
sufficientStatistics_Weighted(obj, x, w, foreach = FALSE, ...)
```

Arguments

obj	A "GaussianInvWishart" object.
x	matrix, Gaussian samples, when x is a matrix, each row is a sample of dimension ncol(x). when x is a vector, x is length(x) samples of dimension 1.
w	numeric, sample weights.
foreach	logical, specifying whether to return the sufficient statistics for each observation. Default FALSE.
...	Additional arguments to be passed to other inherited types.

Value

If foreach=TRUE, will return a list of sufficient statistics for each row of x, otherwise will return the sufficient statistics of x as a whole.

References

Gelman, Andrew, et al. Bayesian data analysis. CRC press, 2013.

MARolA, K. V., JT KBNT, and J. M. Bibly. Multivariate analysis. AcadeInic Press, Londres, 1979.

See Also

GaussianInvWishart, sufficientStatistics.GaussianInvWishart

Examples

```
obj <- GaussianInvWishart(gamma=list(mu=c(-1.5,1.5),v=4,S=diag(2)))
x <- rGaussian(10,mu = c(-1.5,1.5),Sigma = matrix(c(0.1,0.03,0.03,0.1),2,2))
w <- runif(10)
sufficientStatistics_Weighted(obj=obj,x=x,w=w,foreach = FALSE)
sufficientStatistics_Weighted(obj=obj,x=x,w=w,foreach = TRUE)
```

---

sufficientStatistics_Weighted.GaussianNIG

Weighted sufficient statistics of a "GaussianNIG" object

Description

For following Gaussian-NIG model structure:

$$x \sim Gaussian(Xbeta, sigma^2)$$

$$sigma^2 \sim InvGamma(a, b)$$

$$beta \sim Gaussian(m, sigma^2 V)$$

Where X is a row vector, or a design matrix where each row is an obervation. InvGamma() is the Inverse-Gamma distribution, Gaussian() is the Gaussian distribution. See ?dInvGamma and

dGaussian for the definitions of these distribution.
The model structure and prior parameters are stored in a "GaussianNIG" object.
This object will be used as a place for recording and accumulating information in the related inference/sampling functions such as posterior(), posteriorDiscard(), MAP(), marginalLikelihood(), dPosteriorPredictive(), rPosteriorPredictive() and so on.
The sufficient statistics of a set of samples (x,X) and weights ware:

- the effective number of samples N=sum(w);
- the covariance of X and x SXx=t(w*X)
- the covariance of X SX=t(w*X)
- the covariance of x Sx=t(w*x)

Usage

```
## S3 method for class 'GaussianNIG'
sufficientStatistics_Weighted(obj, x, w, X, foreach = FALSE, ...)
```

Arguments

obj	A "GaussianNIG" object.
x	numeric, must satisfy length(x) = nrow(X).
w	numeric, sample weights.
X	matrix, must satisfy length(x) = nrow(X).
foreach	logical, if foreach=TRUE, will return a list of sufficient statistics for each (x,X), otherwise will return the sufficient statistics as a whole.
...	Additional arguments to be passed to other inherited types.

Value

If foreach=TRUE, will return a list of sufficient statistics for each row of (x,X), otherwise will return the sufficient statistics of (x,X) as a whole.

References

Banerjee, Sudipto. "Bayesian Linear Model: Gory Details." Downloaded from http://www. biostat. umn. edu/~ph7440 (2008).

See Also

GaussianNIG, sufficientStatistics.GaussianNIG

Examples

```
obj <- GaussianNIG(gamma=list(m=0,V=1,a=1,b=0))
X <- 1:20
x <- rnorm(20)+ X*0.3
w <- runif(20)
sufficientStatistics_Weighted(obj = obj,X=X,x=x,w=w)
sufficientStatistics_Weighted(obj = obj,X=X,x=x,w=w,foreach = TRUE)
```

`sufficientStatistics_Weighted.GaussianNIW`
Weighted sufficient statistics for a "GaussianNIW" object

Description

For following Gaussian-NIW model structure:

$$mu, Sigma|m, k, v, S \sim NIW(m, k, v, S)$$

$$x|mu, Sigma \sim Gaussian(mu, Sigma)$$

Where NIW() is the Normal-Inverse-Wishart distribution, Gaussian() is the Gaussian distribution. See `?dNIW` and `dGaussian` for the definitions of these distribution.
The sufficient statistics of a set of samples x (each row of x is a sample) and weights w are:

- the effective number of samples N=sum(w)
- the sample sum xsum = colSums(x*w)
- the uncentered scatter matrix S = t(w*x)

Usage

```
## S3 method for class 'GaussianNIW'
sufficientStatistics_Weighted(obj, x, w, foreach = FALSE, ...)
```

Arguments

`obj`	A "GaussianNIW" object.
`x,`	matrix, Gaussian samples, when x is a matrix, each row is a sample of dimension ncol(x). when x is a vector, x is length(x) samples of dimension 1.
`w`	numeric, sample weights.
`foreach`	logical, if foreach=TRUE, will return a list of sufficient statistics for each row of x, otherwise will return the sufficient statistics of x as a whole.
`...`	Additional arguments to be passed to other inherited types.

Value

If foreach=TRUE, will return a list of sufficient statistics for each row of x, otherwise will return the sufficient statistics of x as a whole.

References

Murphy, Kevin P. "Conjugate Bayesian analysis of the Gaussian distribution." def 1.22 (2007): 16.

Gelman, Andrew, et al. "Bayesian Data Analysis Chapman & Hall." CRC Texts in Statistical Science (2004).

See Also

GaussianNIW, sufficientStatistics.GaussianNIW

Examples

```
x <- rGaussian(10,mu = c(-1.5,1.5),Sigma = matrix(c(0.1,0.03,0.03,0.1),2,2))
obj <- GaussianNIW()                    #an GaussianNIW object
w <- runif(10)
sufficientStatistics_Weighted(obj=obj,x=x,w=w,foreach = FALSE)
sufficientStatistics_Weighted(obj=obj,x=x,w=w,foreach = TRUE)
```

`sufficientStatistics_Weighted.HDP`

Weighted sufficient statistics of a "HDP" object

Description

For following model structure:

$$G|gamma \sim DP(gamma, U)$$

$$pi_j|G, alpha \sim DP(alpha, G), j = 1:J$$

$$z|pi_j \sim Categorical(pi_j)$$

$$k|z, G \sim Categorical(G),$$ if z is a sample from the base measure G

$$theta_k|psi \sim H0(psi)$$

$$x|theta_k, k \sim F(theta_k)$$

where DP(gamma,U) is a Dirichlet Process on positive integers, gamma is the "concentration parameter", U is the "base measure" of this Dirichlet process, U is an uniform distribution on all positive integers. DP(alpha,G) is a Dirichlet Process on integers with concentration parameter alpha and base measure G. The choice of F() and H0() can be described by an arbitrary "BasicBayesian" object such as "GaussianGaussian","GaussianInvWishart","GaussianNIW", "GaussianNIG", "CatDirichlet", and "CatDP". See `?BasicBayesian` for definition of "BasicBayesian" objects, and see for example `?GaussianGaussian` for specific "BasicBayesian" instances. As a summary, An "HDP" object is simply a combination of a "CatHDP" object (see `?CatHDP`) and an object of any "BasicBayesian" type.
In the case of HDP, z and k can only be positive integers.
The sufficient statistics of a set of samples x in a "HDP" object is the same sufficient statistics of the "BasicBayesian" inside the "HDP", see examples.

Usage

```
## S3 method for class 'HDP'
sufficientStatistics_Weighted(obj, x, w, ...)
```

Arguments

obj	A "HDP" object.
x	Random samples of the "BasicBayesian" object.
w	numeric, sample weights.
...	further arguments passed to the corresponding sufficientStatistics method of the "BasicBayesian" object.

Value

Return the sufficient statistics of the corresponding BasicBayesian type, see examples.

References

Teh, Yee W., et al. "Sharing clusters among related groups: Hierarchical Dirichlet processes." Advances in neural information processing systems. 2005.

See Also

HDP, sufficientStatistics.HDP

Examples

```
## a HDP with Gaussian NIW observations
obj1 <- HDP(gamma=list(gamma=1,alpha=1,j=2,
                       H0aF="GaussianNIW",
                       parH0=list(m=0,k=1,v=2,S=1)))
## a HDP with Categorical-Dirichlet observations
obj2 <- HDP(gamma=list(gamma=1,alpha=1,j=2,
            H0aF="CatDirichlet",
            parH0=list(alpha=1,uniqueLabels=letters[1:3])))
x1 <- rnorm(100)
x2 <- sample(letters[1:3],100,replace = TRUE)
w <- runif(100)
sufficientStatistics_Weighted(obj = obj1,x=x1,w=w,foreach = FALSE)
sufficientStatistics_Weighted(obj = obj2,x=x2,w=w,foreach = FALSE)
sufficientStatistics_Weighted(obj = obj1,x=x1,w=w,foreach = TRUE)
```

`sufficientStatistics_Weighted.HDP2` *Weighted sufficient statistics of a "HDP2" object*

Description

For following model structure:

$$G|eta \sim DP(eta, U)$$

$$G_m|gamma, G \sim DP(gamma, G), m = 1 : M$$

$$pi_{mj}|G_m, alpha \sim DP(alpha, G_m), j = 1 : J_m$$

$$z|pi_{mj} \sim Categorical(pi_{mj})$$

$$k|z, G_m \sim Categorical(G_m),$$ if z is a sample from the base measure G_{mj}

$$u|k, G \sim Categorical(G),$$ if k is a sample from the base measure G

$$theta_u|psi \sim H0(psi)$$

$$x|theta_u, u \sim F(theta_u)$$

where DP(eta,U) is a Dirichlet Process on positive integers, eta is the "concentration parameter", U is the "base measure" of this Dirichlet process, U is an uniform distribution on all positive integers. DP(gamma,G) is a Dirichlet Process on integers with concentration parameter gamma and base measure G. DP(alpha,G_m) is a Dirichlet Process on integers with concentration parameter alpha and base measure G_m. The choice of F() and H0() can be described by an arbitrary "BasicBayesian" object such as "GaussianGaussian","GaussianInvWishart","GaussianNIW", "GaussianNIG", "CatDirichlet", and "CatDP". See `?BasicBayesian` for definition of "BasicBayesian" objects, and see for example `?GaussianGaussian` for specific "BasicBayesian" instances. As a summary, An "HDP2" object is simply a combination of a "CatHDP2" object (see `?CatHDP2`) and an object of any "BasicBayesian" type.
In the case of HDP2, u, z and k can only be positive integers.
The sufficient statistics of a set of samples x in a "HDP2" object is the same sufficient statistics of the "BasicBayesian" inside the "HDP2", see examples.

Usage

```
## S3 method for class 'HDP2'
sufficientStatistics_Weighted(obj, x, w, ...)
```

Arguments

`obj`	A "HDP2" object.
`x`	Random samples of the "BasicBayesian" object.
`w`	numeric, sample weights.
`...`	Additional arguments to be passed to other inherited types.

Value

Return the sufficient statistics of the corresponding BasicBayesian type, see examples.

References

Teh, Yee W., et al. "Sharing clusters among related groups: Hierarchical Dirichlet processes." Advances in neural information processing systems. 2005.

See Also

HDP2, sufficientStatistics.HDP2

Examples

```
## a HDP2 with Gaussian NIW observations
obj1 <- HDP2(gamma=list(gamma=1,alpha=1,j=2,m=2,
                        H0aF="GaussianNIW",
                        parH0=list(m=0,k=1,v=2,S=1)))
## a HDP2 with Categorical-Dirichlet observations
obj2 <- HDP2(gamma=list(gamma=1,alpha=1,j=2,m=2,
                        H0aF="CatDirichlet",
                        parH0=list(alpha=1,uniqueLabels=letters[1:3])))
x1 <- rnorm(100)
x2 <- sample(letters[1:3],100,replace = TRUE)
w <- runif(100)
sufficientStatistics_Weighted(obj = obj1,x=x1,w=w,foreach = FALSE)
sufficientStatistics_Weighted(obj = obj2,x=x2,w=w,foreach = FALSE)
sufficientStatistics_Weighted(obj = obj1,x=x1,w=w,foreach = TRUE)
```

---

sufficientStatistics_Weighted.LinearGaussianGaussian

Weighted sufficient statistics of a "LinearGaussianGaussian" object

---

Description

For following model structure:

$$x \sim Gaussian(Az + b, Sigma)$$

$$z \sim Gaussian(m, S)$$

Where Sigma is known. A is a $dimxxdimz$ matrix, x is a $dimxx1$ random vector, z is a $dimzx1$ random vector, b is a $dimmx1$ vector. Gaussian() is the Gaussian distribution. See ?dGaussian for the definition of Gaussian distribution.
For weight vector w and one dimensional observations: x is a vector of length N, or a $Nx1$ matrix, each row is an observation, must satisfy $nrow(x) = length(w)$; A is a $Nxdimz$ matrix; b is a length N vector. The sufficient statistics are:

- SA = $A^T(Aw)/Sigma$

- SAx = $A^T((x-b)w)/Sigma$

For weight vector w and $dimx$ dimensional observations: x must be a Nxm matrix, each row is an observation, must satisfy $nrow(x) = length(w)$; A can be either a list or a matrix. When A is a list, $A = A_1, A_2, ...A_N$ is a list of $dimxxdimz$ matrices. If A is a single $dimxxdimz$ matrix, it will be replicated N times into a length N list; b can be either a matrix or a vector. When b is a matrix, $b = b_1^T, ..., b_N^T$ is a $Nxdimx$ matrix, each row is a transposed vector. When b is a length $dimx$ vector, it will be transposed into a row vector and replicated N times into a $Nxdimx$ matrix. The sufficient statistics are:

- SA = $sum_{i=1:N} w_i A_i^T Sigma^{-1} A_i$
- SAx = $sum_{i=1:N} w_i A_i^T Sigma^{-1}(x_i - b_i)$

Usage

```
## S3 method for class 'LinearGaussianGaussian'
sufficientStatistics_Weighted(obj, x, w, A, b = NULL, foreach = FALSE, ...)
```

Arguments

obj	A "LinearGaussianGaussian" object.
x	matrix, Gaussian samples, when x is a matrix, each row is a sample of dimension ncol(x). when x is a vector, x is length(x) samples of dimension 1.
w	numeric, sample weights.
A	matrix or list. when x is a $Nx1$ matrix, A must be a matrix of $Nxdimz$, dimz is the dimension of z; When x is a $Nxdimx$ matrix, where $dimx > 1$, A can be either a list or a matrix. When A is a list, $A = A_1, A_2, ...A_N$ is a list of $dimxxdimz$ matrices. If A is a single $dimxxdimz$ matrix, it will be replicated N times into a length N list
b	matrix, when x is a $Nx1$ matrix, b must also be a $Nx1$ matrix or length N vector; When x is a $Nxdimx$ matrix, where $dimx > 1$, b can be either a matrix or a vector. When b is a matrix, $b = b_1^T, ..., b_N^T$ is a $Nxdimx$ matrix, each row is a transposed vector. When b is a length $dimx$ vector, it will be transposed into a row vector and replicated N times into a $Nxdimx$ matrix. When b = NULL, it will be treated as a vector of zeros. Default NULL.
foreach	logical, specifying whether to return the sufficient statistics for each observation. Default FALSE.
...	Additional arguments to be passed to other inherited types.

Value

If foreach=TRUE, will return a list of sufficient statistics for each row of x, otherwise will return the sufficient statistics of x as a whole.

References

Murphy, Kevin P. Machine learning: a probabilistic perspective. MIT press, 2012.

See Also

LinearGaussianGaussian, sufficientStatistics.LinearGaussianGaussian

Examples

```
obj <- LinearGaussianGaussian(gamma=list(Sigma=matrix(c(2,1,1,2),2,2),m=c(0.2,0.5,0.6),S=diag(3)))
x <- rGaussian(100,mu = runif(2),Sigma = diag(2))
w <- runif(100)
A <- matrix(runif(6),2,3)
b <- runif(2)
sufficientStatistics_Weighted(obj,x=x,w=w,A=A,b=b)
sufficientStatistics_Weighted(obj,x=x,w=w,A=A,b=b,foreach = TRUE)
```

%plus% *a plus b with NA values*

Description

make NA+NA=0 and NA+(not NA)=0 instead of NA+NA=NA and NA+(not NA)=NA

Usage

```
e1 %plus% e2
```

Arguments

e1	numeric
e2	numeric

Value

numeric, the sum of "e1" and "e2". The length of the returned vector is the same length as the longest of "e1" or "e2".

Index