

Package ‘CNPBayes’

April 22, 2016

Type Package

Title Bayesian mixture models for copy number polymorphisms

Version 1.0.0

Maintainer Jacob Carey <jcarey15@jhu.edu>

Description Bayesian hierarchical mixture models for batch effects and copy number.

Date Tue Jan 20 20:41:00 EST 2015

Author Stephen Cristiano, Robert Scharpf, and Jacob Carey

Depends foreach, GenomicRanges

Imports Rcpp (>= 0.12.1), S4Vectors, matrixStats, RColorBrewer, gtools, oligoClasses, combinat, GenomeInfoDb, IRanges, methods, BiocGenerics

Suggests testthat, knitr, BiocStyle, VanillaICE (>= 1.31.3),

Collate 'help.R' 'AllGenerics.R' 'AllClasses.R' 'RcppExports.R' 'data.R' 'functions.R' 'marginal_likelihood.R' 'methods-BatchModel.R' 'methods-DensityModel.R' 'methods-Hyperparameters.R' 'methods-MarginalModel.R' 'methods-McmcChains.R' 'methods-McmcParams.R' 'methods-MixtureModel.R' 'methods-SummarizedExperiment.R' 'model_initialization.R' 'simulate_data.R' 'visualization.R'

VignetteBuilder knitr

License Artistic-2.0

LinkingTo Rcpp

biocViews CopyNumberVariation, Bayesian

Roxygen list(wrap=FALSE)

LazyData TRUE

URL <https://github.com/scristia/CNPBayes>

BugReports <https://github.com/scristia/CNPBayes/issues>

NeedsCompilation yes

R topics documented:

batch	3
BatchModel	4
BatchModel-class	5
BatchModelExample	6
bic	6
burnin	7
chains	8
chromosome	8
clusters	9
CNPBayes	10
collapseBatch	10
consensusCNP	11
DensityModel	13
DensityModel-class	14
downsample	15
downSampleEachBatch	16
eta.0	16
extract	17
Hyperparameters	18
Hyperparameters-class	18
HyperparametersBatch	19
HyperparametersBatch-class	20
HyperparametersMarginal	21
HyperparametersMarginal-class	22
hyperParams	22
iter<-	23
k	24
logBayesFactor	25
logPrior	25
log_lik	26
m2.0	27
map	27
mapCnProbability	28
marginalLikelihood	28
MarginalModel	29
MarginalModel-class	30
MarginalModelExample	31
McmcChains-class	31
McmcParams	32
mcmcParams	32
McmcParams-class	33
MixtureModel-class	34
modes	35
mu	36
muc	36
muMean	37

names,McmcChains-method	37
nStarts	38
nu.0	39
oned	39
p	40
pic	40
plot	41
posteriorSimulation	42
probz	43
qInverseTau2	43
saveBatch	44
sigma	45
sigma2	45
sigma2.0	46
sigmac	47
simulateBatchData	47
simulateData	48
tau	49
tau2	49
tauc	50
tauMean	51
theta	51
thin	52
tracePlot	53
y	54
z	54
zFreq	55

Index **56**

batch	<i>Retrieve batches from object.</i>
-------	--------------------------------------

Description

The batches are represented as a vector of integers.

Usage

```
batch(object)

## S4 method for signature 'DensityModel'
batch(object)

## S4 method for signature 'MixtureModel'
batch(object)
```

Arguments

object see showMethods(batch)

Value

The batch of each data element.

Examples

```
batch(BatchModelExample)
```

BatchModel

Create an object for running hierarchical MCMC simulations.

Description

Create an object for running hierarchical MCMC simulations.

Usage

```
BatchModel(data = numeric(), k = 2L, batch, hypp, mcmc.params)
```

Arguments

data the data for the simulation.

k An integer value specifying the number of latent classes.

batch a vector of the different batch numbers (must be sorted)

hypp An object of class 'Hyperparameters' used to specify the hyperparameters of the model.

mcmc.params An object of class 'McmcParams'

Value

An object of class 'BatchModel'

Examples

```
model <- BatchModel(rnorm(10), k=1, batch=rep(1:2, each=5))
```

BatchModel-class *An object for running MCMC simulations.*

Description

Run hierarchical MCMC for batch model.

Slots

`k` An integer value specifying the number of latent classes.

`hyperparams` An object of class 'Hyperparameters' used to specify the hyperparameters of the model.

`theta` the means of each component and batch

`sigma2` the variances of each component and batch

`nu.0` the shape parameter for `sigma2`

`sigma2.0` the rate parameter for `sigma2`

`pi` mixture probabilities which are assumed to be the same for all batches

`mu` means from batches, averaged across batches

`tau2` variances from batches, weighted by precisions

`data` the data for the simulation.

`data.mean` the empirical means of the components

`data.prec` the empirical precisions

`z` latent variables

`zfreq` table of latent variables

`probz` $n \times k$ matrix of probabilities

`logprior` log likelihood of prior: $\log(p(\text{sigma2.0})p(\text{nu.0})p(\text{mu}))$

`loglik` log likelihood: $\sum p_k \Phi(\theta_k, \sigma_k)$

`mcmc.chains` an object of class 'McmcChains' to store MCMC samples

`batch` a vector of the different batch numbers

`batchElements` a vector labeling from which batch each observation came from

`modes` the values of parameters from the iteration which maximizes log likelihood and log prior

`mcmc.params` An object of class 'McmcParams'

`.internal.constraint` Constraint on parameters. For internal use only.

BatchModelExample	<i>This data is a simulated example of Batch data</i>
-------------------	---

Description

This data is a simulated example of Batch data

Usage

```
BatchModelExample
```

Value

An example of a 'BatchModel' BatchModelExample

Author(s)

Jacob Carey

bic	<i>Calculate BIC of a model</i>
-----	---------------------------------

Description

Calculate BIC of a model

Usage

```
bic(object)

## S4 method for signature 'BatchModel'
bic(object)

## S4 method for signature 'MarginalModel'
bic(object)
```

Arguments

object see showMethods(bic)

Value

The BIC of the model.

Examples

```
bic(BatchModelExample)
```

burnin	<i>Number of burnin iterations.</i>
--------	-------------------------------------

Description

This function retrieves the number of burnin simulations to be discarded.

This function changes the number of burnin simulations to be discarded.

Usage

```
burnin(object)

burnin(object) <- value

## S4 method for signature 'McmcParams'
burnin(object)

## S4 replacement method for signature 'McmcParams'
burnin(object) <- value

## S4 method for signature 'MixtureModel'
burnin(object)

## S4 replacement method for signature 'MixtureModel'
burnin(object) <- value
```

Arguments

object	see showMethods(burnin)
value	new number of burnin iterations

Value

The number of burnin simulations.

Examples

```
burnin(MarginalModelExample)
mp <- mcmcParams(MarginalModelExample)
burnin(mp)
```

chains	<i>Retrieve simulated chains from model object.</i>
--------	---

Description

The method `chains` applied to a `MixtureModel`-derived class will return an object of class `McmcChains` that contains the chains for all simulated parameters. Typically, `chains` is called in conjunction with an accessor for one of these parameters.

Usage

```
chains(object)

## S4 method for signature 'MixtureModel'
chains(object)
```

Arguments

```
object          showMethods(chains)
```

Value

The simulated chains.

Examples

```
theta.chain <- theta(chains(MarginalModelExample))
dim(theta.chain)
plot.ts(theta.chain, plot.type="single",
        col=seq_len(k(MarginalModelExample)))
```

chromosome	<i>Extract character vector of sequence names</i>
------------	---

Description

Short cut for `as.character(seqnames(g))` where `g` is a `GRanges` object.

Usage

```
chromosome(object, ...)
```

Arguments

```
object          a GRanges instance
...             currently ignored
```


Value

A character vector

Examples

```
## Not run:
g <- GRanges("chr1", IRanges(10, 15))
chromosome(g)

## End(Not run)
```

clusters	<i>Accessor for extracting the kmeans clusters from a DensityModel instance</i>
----------	---

Description

Accessor for extracting the kmeans clusters from a DensityModel instance

Usage

```
clusters(object)

## S4 method for signature 'DensityModel'
clusters(object)
```

Arguments

object an instance of class 'DensityModel'

Value

k-means clustering of the component means using the modes as centers.

See Also

[DensityModel-class](#)

Examples

```
truth <- simulateData(N=2500, p=rep(1/3, 3),
                     theta=c(-1, 0, 1),
                     sds=rep(0.1, 3))
dm <- DensityModel(truth)
clusters(dm)
```

 CNPBayes

Bayesian mixture models for copy number estimation

Description

Bayesian mixture models for copy number estimation

 collapseBatch

Estimate batch from a collection of chemistry plates or some other variable that captures the time in which the arrays were processed.

Description

In high-throughput assays, low-level summaries of copy number at copy number polymorphic loci (e.g., the mean log R ratio for each sample, or a principal-component derived summary) often differ between groups of samples due to technical sources of variation such as reagents, technician, or laboratory. Technical (as opposed to biological) differences between groups of samples are referred to as batch effects. A useful surrogate for batch is the chemistry plate on which the samples were hybridized. In large studies, a Bayesian hierarchical mixture model with plate-specific means and variances is computationally prohibitive. However, chemistry plates processed at similar times may be qualitatively similar in terms of the distribution of the copy number summary statistic. Further, we have observed that some copy number polymorphic loci exhibit very little evidence of a batch effect, while other loci are more prone to technical variation. We suggest combining plates that are qualitatively similar in terms of the Kolmogorov-Smirnov two-sample test of the distribution and to implement this test independently for each candidate copy number polymorphism identified in a study. The collapseBatch function is a wrapper to the ks.test implemented in the stats package that compares all pairwise combinations of plates. The ks.test is performed recursively on the batch variables defined for a given CNP until no batches can be combined.

Usage

```
collapseBatch(object, plate, THR = 0.1)

## S4 method for signature 'BatchModel'
collapseBatch(object)

## S4 method for signature 'SummarizedExperiment'
collapseBatch(object, plate, THR = 0.1)

## S4 method for signature 'numeric'
collapseBatch(object, plate, THR = 0.1)
```

Arguments

object	see showMethods(collapseBatch)
plate	a vector labelling from which batch each observation came from.
THR	threshold below which the null hypothesis should be rejected and batches are collapsed.

Value

The new batch value.

Examples

```
bt <- collapseBatch(y(BatchModelExample), batch(BatchModelExample))
newBatchModel <- BatchModel(y(BatchModelExample), k(BatchModelExample),
                             bt, hyperParams(BatchModelExample),
                             mcmcParams(BatchModelExample))
```

consensusCNP	<i>Identify consensus start and stop coordinates of a copy number polymorphism</i>
--------------	--

Description

The collection of copy number variants (CNVs) identified in a study can be encapsulated in a GRangesList, where each element is a GRanges of the CNVs identified for an individual. (For a study with 1000 subjects, the GRangesList object would have length 1000 if each individual had 1 or more CNVs.) For regions in which CNVs occur in more than 2 percent of study participants, the start and end boundaries of the CNVs may differ because of biological differences in the CNV size as well as due to technical noise of the assay and the uncertainty of the breakpoints identified by a segmentation of the genomic data. Among subjects with a CNV called at a given locus, the consensusCNP function identifies the largest region that is copy number variant in half of these subjects.

Usage

```
consensusCNP(grl, transcripts, min.width = 2000, max.width = 2e+05,
             min.prevalance = 0.02)
```

Arguments

grl	A GRangesList of all CNVs in a study – each element is the collection of CNVs for one individual.
transcripts	a GRanges object containing annotation of genes or transcripts (optional)
min.width	length-one integer vector specifying the minimum width of CNVs
max.width	length-one integer vector specifying the maximum width of CNVs

`min.prevalance` a length-one numeric vector specifying the minimum prevalence of a copy number polymorphism. Must be in the interval [0,1]. If less than 0, this function will return all CNV loci regardless of prevalence. If greater than 1, this function will return a length-zero GRanges object

Value

a GRanges object providing the intervals of all identified CNPs above a user-specified prevalence cutoff.

Examples

```
library(GenomicRanges)
##
## Simulate 2 loci at which CNVs are common
##
set.seed(100)
starts <- rpois(1000, 100) + 10e6L
ends <- rpois(1000, 100) + 10.1e6L
cnv1 <- GRanges("chr1", IRanges(starts, ends))
cnv1$id <- paste0("sample", seq_along(cnv1))

starts <- rpois(500, 1000) + 101e6L
ends <- rpois(500, 1000) + 101.4e6L
cnv2 <- GRanges("chr5", IRanges(starts, ends))
cnv2$id <- paste0("sample", seq_along(cnv2))

##
## Simulate a few other CNVs that are less common because they are
## very large, or because they occur in regions that in which copy
## number alterations are not common
##
cnv3 <- GRanges("chr1", IRanges(9e6L, 15e6L), id="sample1400")
starts <- seq(5e6L, 200e6L, 10e6L)
ends <- starts + rpois(length(starts), 25e3L)
cnv4 <- GRanges("chr1", IRanges(starts, ends),
                id=paste0("sample", sample(1000:1500, length(starts))))

all_cnvs <- suppressWarnings(c(cnv1, cnv2, cnv3, cnv4))
gr1 <- split(all_cnvs, all_cnvs$id)
cnps <- consensusCNP(gr1)

##
## 2nd CNP is filtered because of its size
##
truth <- GRanges("chr1", IRanges(10000100L, 10100100L))
seqinfo(truth) <- seqinfo(gr1)
identical(cnps, truth)

##
## Both CNVs identified
##
```

```
cnps <- consensusCNP(gr1, max.width=500e3)
truth <- GRanges(c("chr1", "chr5"),
                 IRanges(c(10000100L, 101000999L),
                        c(10100100L, 101400999L)))
seqlevels(truth, force=TRUE) <- seqlevels(gr1)
seqinfo(truth) <- seqinfo(gr1)
identical(cnps, truth)
```

DensityModel

Constructor for DensityModel class

Description

Instantiates an instance of 'DensityModel' (or 'DensityBatchModel') from a MarginalModel or BatchModel object. See the corresponding class for additional details and examples.

Usage

```
DensityModel(object, merge = FALSE)
```

Arguments

object	see showMethods(DensityModel)
merge	Logical. Whether to use kmeans clustering to cluster the component means using the estimated modes from the overall density as the centers for the kmeans function.

Value

An object of class 'DensityModel'

See Also

[DensityModel-class kmeans](#)

Examples

```
dm <- DensityModel(MarginalModelExample)
```

DensityModel-class *An object to store estimated mixture model densities*

Description

Instances of DensityModel store the estimated densities for each component and the overall (marginal) estimate of the density. The derived class DensityBatchModel additionally stores the density for each batch / component combination (i.e., if there are 3 components and 10 batches, there are 30 estimated densities). The intended use-case of the DensityModel class is to facilitate visualization of the estimated densities (see examples) as well as to provide an estimate of the number of modes in the overall density. If the number of estimated modes is smaller than the number of components of the best-fitting mixture model, post-hoc merging of components may be useful.

Slots

component The component densities.

overall The overall (marginal across batches and components) estimate of the density.

modes A numeric vector providing the estimated modes in the overall density. The modes are defined by a crude estimate of the first derivative of the overall density (see findModes).

data A numeric vector containing the data

clusters A vector providing the k-means clustering of the component means using the modes as centers. If an object of class DensityModel is instantiated with merge=FALSE, this slot takes values 1, ..., K, where K is the number of components.

See Also

[DensityModel](#)

Examples

```
## marginal model
truth <- simulateData(N=2500, p=rep(1/3, 3),
                     theta=c(-1, 0, 1),
                     sds=rep(0.1, 3))

dm <- DensityModel(truth)
print(dm)
dm.merged <- DensityModel(truth, merge=TRUE)
print(dm.merged)
## here, because there are 3 distinct modes, specifying merge=TRUE
## does not change the resulting clusters
identical(clusters(dm), clusters(dm.merged))
## These objects can be plotted
plot(dm)
## Note that calling plot on a MixtureModel-derived object returns
## a density object as a side-effect of the plotting
dm2 <- CNPBayes::plot(truth)
identical(dm, dm2)
```

```

## batch model
k <- 3
nbatch <- 3
means <- matrix(c(-1.2, -1.0, -0.8,
                  -0.2, 0, 0.2,
                  0.8, 1, 1.2), nbatch, k, byrow=FALSE)
sds <- matrix(0.1, nbatch, k)
N <- 1500
truth <- simulateBatchData(N=N,
                           batch=rep(letters[1:3], length.out=N),
                           theta=means,
                           sds=sds,
                           p=c(1/5, 1/3, 1-1/3-1/5))

dm <- DensityModel(truth)
dm.merged <- DensityModel(truth, merge=TRUE)
print(dm)
dm2 <- CNPBayes::plot(truth)
identical(dm, dm2)
## suppress plotting of the batch-specific densities
CNPBayes::plot(dm2, show.batch=FALSE)

```

downsample

Create tile labels for each observation

Description

A wrapper for function `downSampleEachBatch`. Batches are automatically merged as needed.

Usage

```
downsample(batch.file, plate, y, ntiles = 250, THR = 0.1)
```

Arguments

<code>batch.file</code>	the name of a file containing RDS data to be read in.
<code>plate</code>	a vector containing the labels from which batch each observation came from.
<code>y</code>	in memory data
<code>ntiles</code>	number of tiles in a batch
<code>THR</code>	threshold above which to merge batches in Kolmogorov-Smirnov test.

Value

Tile labels for each observation

`downSampleEachBatch` *Create tile labels for each observation*

Description

Create tile labels for each observation

Usage

```
downSampleEachBatch(y, nt, batch)
```

Arguments

<code>y</code>	vector containing data
<code>nt</code>	the number of tiles in a batch
<code>batch</code>	a vector containing the labels from which batch each observation came from.

Value

Tile labels for each observation

`eta.0` *Retrieve the rate parameter for the tau2 distribution.*

Description

Retrieve the rate parameter for the tau2 distribution.

Usage

```
eta.0(object)

## S4 method for signature 'MixtureModel'
eta.0(object)

## S4 method for signature 'Hyperparameters'
eta.0(object)
```

Arguments

<code>object</code>	see <code>showMethods(eta.0)</code>
---------------------	-------------------------------------

Value

`eta.0` of a 'MixtureModel'

Examples

```
eta.0(MarginalModelExample)
```

extract	<i>extract data, latent variable, and batch for given observation</i>
---------	---

Description

extract data, latent variable, and batch for given observation

extract estimated parameters at particular iteration of simulation.

Allows a user to pass a vector for burnin, thin, and iter.

Usage

```
## S4 method for signature 'BatchModel,ANY'
x[i, j, ..., drop = FALSE]
```

```
## S4 method for signature 'McmcChains,ANY'
x[i, j, ..., drop = FALSE]
```

```
## S4 method for signature 'McmcParams,ANY'
x[i, j, ..., drop = FALSE]
```

Arguments

x	An object of class <code>BatchModel</code> , <code>McmcChains</code> , or <code>McmcParams</code>
i	An element of the instance to be extracted.
j	Not used.
...	Not used.
drop	Not used.

Value

An object of class `'BatchModel'`

An object of class `'McmcChains'`

An object of class `'McmcParams'`

Hyperparameters *Create an object of class 'Hyperparameters'*

Description

Create an object of class 'Hyperparameters'

Usage

```
Hyperparameters(type = "batch", k = 2L, ...)
```

Arguments

type	specifies 'marginal' or 'batch'
k	number of components
...	optional parameters. See details

Details

Additional hyperparameters can be passed to the HyperparametersMarginal and HyperparametersBatch models.

Value

An object of class HyperparametersMarginal or HyperparametersBatch

Examples

```
hypp <- Hyperparameters("marginal", k=2)
```

Hyperparameters-class *An object to specify the hyperparameters of a model.*

Description

An object to specify the hyperparameters of a model.

Slots

k Number of components
 mu.0 Prior mean for mu.
 tau2.0 prior variance on mu
 eta.0 rate parameter for tau2
 m2.0 shape parameter for tau2
 alpha mixture probabilities
 beta parameter for nu.0 distribution
 a shape for sigma2.0
 b rate for sigma2.0

HyperparametersBatch *Create an object of class 'HyperparametersBatch' for the batch mixture model*

Description

Create an object of class 'HyperparametersBatch' for the batch mixture model

Usage

```
HyperparametersBatch(k = 0L, mu.0 = 0, tau2.0 = 100, eta.0 = 1800,
  m2.0 = 1/60, alpha, beta = 0.1, a = 1.8, b = 6)
```

Arguments

k	length-one integer vector specifying number of components (typically $1 \leq k \leq 4$)
mu.0	length-one numeric vector of the of the normal prior for the component means.
tau2.0	length-one numeric vector of the variance for the normal prior of the component means
eta.0	length-one numeric vector of the shape parameter for the Inverse Gamma prior of the component variances, tau2_h. The shape parameter is parameterized as $1/2 * eta.0$. In the batch model, tau2_h describes the inter-batch heterogeneity of means for component h.
m2.0	length-one numeric vector of the rate parameter for the Inverse Gamma prior of the component variances, tau2_h. The rate parameter is parameterized as $1/2 * eta.0 * m2.0$. In the batch model, tau2_h describes the inter-batch heterogeneity of means for component h.
alpha	length-k numeric vector of the shape parameters for the dirichlet prior on the mixture probabilities

beta	length-one numeric vector for the parameter of the geometric prior for nu.0 (nu.0 is the shape parameter of the Inverse Gamma sampling distribution for the component-specific variances. Together, nu.0 and sigma2.0 model inter-component heterogeneity in variances.). beta is a probability and must be in the interval [0,1].
a	length-one numeric vector of the shape parameter for the Gamma prior used for sigma2.0 (sigma2.0 is the shape parameter of the Inverse Gamma sampling distribution for the component-specific variances).
b	a length-one numeric vector of the rate parameter for the Gamma prior used for sigma2.0 (sigma2.0 is the rate parameter of the Inverse Gamma sampling distribution for the component-specific variances)

Value

An object of class HyperparametersBatch

Examples

```
HyperparametersBatch(k=3)
```

HyperparametersBatch-class

An object to specify the hyperparameters of a batch effect model.

Description

This class inherits from the Hyperparameters class. This class is for hyperparameters which are hierachical over the batches.

Slots

k Number of components
mu.0 Prior mean for mu.
tau2.0 prior variance on mu
eta.0 rate paramater for tau2
m2.0 shape parameter for tau2
alpha mixture probabilities
beta parameter for nu.0 distribution
a shape for sigma2.0
b rate for sigma2.0

HyperparametersMarginal

Create an object of class 'HyperparametersMarginal' for the marginal mixture model

Description

Create an object of class 'HyperparametersMarginal' for the marginal mixture model

Usage

```
HyperparametersMarginal(k = 0L, mu.0 = 0, tau2.0 = 100, eta.0 = 1,
  m2.0 = 0.1, alpha, beta = 0.1, a = 1.8, b = 6)
```

Arguments

k	length-one integer vector specifying number of components (typically $1 \leq k \leq 4$)
mu.0	length-one numeric vector of the mean for the normal prior of the component means
tau2.0	length-one numeric vector of the variance for the normal prior of the component means
eta.0	length-one numeric vector of the shape parameter for the Inverse Gamma prior of the component variances. The shape parameter is parameterized as $1/2 * \eta.0$.
m2.0	length-one numeric vector of the rate parameter for the Inverse Gamma prior of the component variances. The rate parameter is parameterized as $1/2 * \eta.0 * m2.0$.
alpha	length-k numeric vector of the shape parameters for the dirichlet prior on the mixture probabilities
beta	length-one numeric vector for the parameter of the geometric prior for nu.0 (nu.0 is the shape parameter of the Inverse Gamma sampling distribution for the component-specific variances). beta is a probability and must be in the interval [0,1].
a	length-one numeric vector of the shape parameter for the Gamma prior used for sigma2.0 (sigma2.0 is the shape parameter of the Inverse Gamma sampling distribution for the component-specific variances)
b	a length-one numeric vector of the rate parameter for the Gamma prior used for sigma2.0 (sigma2.0 is the rate parameter of the Inverse Gamma sampling distribution for the component-specific variances)

Value

An object of class HyperparametersMarginal

Examples

```
HyperparametersMarginal(k=3)
```

```
HyperparametersMarginal-class
```

An object to specify the hyperparameters of a marginal model.

Description

This class inherits from the Hyperparameters class. This class is for hyperparameters which are marginal over the batches.

Slots

```
k Number of components
mu.0 Prior mean for mu.
tau2.0 prior variance on mu
eta.0 rate parameter for tau2
m2.0 shape parameter for tau2
alpha mixture probabilities
beta parameter for nu.0 distribution
a shape for sigma2.0
b rate for sigma2.0
```

```
hyperParams
```

Accessor for Hyperparameters object for a MixtureModel-derived object

Description

Accessor for Hyperparameters object for a MixtureModel-derived object
 Replace the hyperparameters for a MixtureModel-derived object

Usage

```
hyperParams(object)
```

```
hyperParams(object) <- value
```

```
## S4 method for signature 'MixtureModel'
hyperParams(object)
```

```
## S4 replacement method for signature 'MixtureModel,Hyperparameters'
hyperParams(object) <- value
```

Arguments

object see showMethods(hyperParams)
 value an object of class 'Hyperparameters'

Value

The Hyperparameters of a MixtureModel

Examples

```
## Not run:
  hyperParams(MarginalModelExample)

## End(Not run)
hypp <- Hyperparameters(type="marginal",
                        k=k(MarginalModelExample),
                        alpha=c(9, 9, 10))
hyperParams(MarginalModelExample) <- hypp
```

iter<- *Reset number of iterations.*

Description

This function changes the number of simulations.

This function retrieves the number of iterations of an MCMC simulation.

Usage

```
iter(object, force = FALSE) <- value

iter(object)

## S4 method for signature 'McmcParams'
iter(object)

## S4 replacement method for signature 'McmcParams'
iter(object, force = FALSE) <- value

## S4 method for signature 'MixtureModel'
iter(object)

## S4 replacement method for signature 'MixtureModel'
iter(object, force = FALSE) <- value
```

Arguments

object	see showMethods(iter)
force	Allow changing of the size of the elements?
value	new number of iterations

Value

The number of MCMC iterations

Examples

```
iter(MarginalModelExample)
```

k	<i>Number of components.</i>
---	------------------------------

Description

This function retrieves the number of a priori components.

Updates the number of components and erases chains from a previous posteriorSimulation (if one was performed). Draws from prior to guess new starting values.

Usage

```
k(object)
```

```
k(object) <- value
```

```
## S4 method for signature 'DensityModel'
k(object)
```

```
## S4 replacement method for signature 'Hyperparameters'
k(object) <- value
```

```
## S4 method for signature 'MixtureModel'
k(object)
```

```
## S4 replacement method for signature 'MixtureModel'
k(object) <- value
```

Arguments

object	see showMethods(k)
value	An integer for the new number of components.

Value

The number of components

Examples

```
k(MarginalModelExample) <- 2
```

logBayesFactor	<i>Compute the log bayes factor between models.</i>
----------------	---

Description

Models of varying component sizes are compared. The log bayes factor is calculated comparing the two models with the highest marginal likelihood, as computed by `marginalLikelihood`.

Usage

```
logBayesFactor(x)
```

Arguments

`x` the result of a call to `computeMarginalLik`.

Value

Log Bayes factor comparing the two models with highest likelihood.

logPrior	<i>Calculate log likelihood of prior for model</i>
----------	--

Description

Calculate log likelihood of prior for model

Usage

```
logPrior(object)
```

```
## S4 method for signature 'McmcChains'
logPrior(object)
```

```
## S4 method for signature 'MixtureModel'
logPrior(object)
```

Arguments

object see showMethods(logPrior)

Value

log likelihood of the prior.

Examples

```
logPrior(MarginalModelExample)
```

log_lik	<i>Retrieve log likelihood.</i>
---------	---------------------------------

Description

Retrieve log likelihood.

Usage

```
log_lik(object)

## S4 method for signature 'McmcChains'
log_lik(object)

## S4 method for signature 'MixtureModel'
log_lik(object)
```

Arguments

object see showMethods(log_lik)

Value

The log likelihood

Examples

```
## retrieve log likelihood at each MCMC iteration
log_lik(chains(MarginalModelExample))
## retrieve log likelihood at last MCMC iteration
log_lik(MarginalModelExample)
```

`m2.0`*Retrieve the shape parameter for the tau2 distribution.*

Description

Retrieve the shape parameter for the tau2 distribution.

Usage

```
m2.0(object)
```

```
## S4 method for signature 'MixtureModel'
```

```
m2.0(object)
```

```
## S4 method for signature 'Hyperparameters'
```

```
m2.0(object)
```

Arguments

object see showMethods(m2.0)

Value

m2.0 for a model

Examples

```
m2.0(MarginalModelExample)
```

`map`*Calculate the maximum a posteriori estimate of latent variable assignment.*

Description

Calculate the maximum a posteriori estimate of latent variable assignment.

Usage

```
map(object)
```

Arguments

object an object of class MixtureModel.

Value

map estimate of latent variable assignment for each observation

Examples

```
map(MarginalModelExample)
```

```
mapCnProbability      Probabilistic copy number assignments.
```

Description

Calculate probabilistic copy number assignments using Bayes Rule applied at the MAP estimates of the cluster mean, variance, and class proportion parameters

Usage

```
mapCnProbability(model)
```

Arguments

```
model          An object of class MixtureModel.
```

Value

A matrix of size $N \times K$ where N is number of observations and K is the number of components.

```
marginalLikelihood  Compute the marginal likelihood of a converged model.
```

Description

Compute the marginal likelihood of a converged model.

Usage

```
marginalLikelihood(model, niter)

## S4 method for signature 'MarginalModel,integer'
marginalLikelihood(model, niter)

## S4 method for signature 'SingleBatchPooledVar,integer'
marginalLikelihood(model, niter)

## S4 method for signature 'MarginalModel,ANY'
marginalLikelihood(model)
```

```
## S4 method for signature 'BatchModel,integer'
marginalLikelihood(model, niter)
```

```
## S4 method for signature 'BatchModel,ANY'
marginalLikelihood(model)
```

```
## S4 method for signature 'list,integer'
marginalLikelihood(model, niter)
```

```
## S4 method for signature 'list,ANY'
marginalLikelihood(model, niter)
```

Arguments

model	An object of class <code>MarginalModel</code> , or a list of <code>MarginalModel</code> 's. Can also be an object of <code>BatchModel</code> or a list of such models.
niter	The number of iterations for the reduced Gibb's sampler.

Value

A vector of the marginal likelihood of the model(s)

Examples

```
marginalLikelihood(MarginalModelExample)
```

<code>MarginalModel</code>	<i>Create an object for running marginal MCMC simulations.</i>
----------------------------	--

Description

Create an object for running marginal MCMC simulations.

Usage

```
MarginalModel(data = numeric(), k = 2, hypp, mcmc.params)
```

Arguments

data	the data for the simulation.
k	An integer value specifying the number of latent classes.
hypp	An object of class 'Hyperparameters' used to specify the hyperparameters of the model.
mcmc.params	An object of class 'McmcParams'

Value

An object of class 'MarginalModel'

Examples

```
model <- MarginalModel(data=rnorm(10), k=1)
```

MarginalModel-class *The 'MarginalModel' class*

Description

Run marginal MCMC simulation

Slots

k An integer value specifying the number of latent classes.

hyperparams An object of class 'Hyperparameters' used to specify the hyperparameters of the model.

theta the means of each component and batch

sigma2 the variances of each component and batch

nu.0 the shape parameter for sigma2

sigma2.0 the rate parameter for sigma2

pi mixture probabilities which are assumed to be the same for all batches

mu overall mean

tau2 overall variance

data the data for the simulation.

data.mean the empirical means of the components

data.prec the empirical precisions

z latent variables

zfreq table of latent variables

probz n x k matrix of probabilities

logprior log likelihood of prior: $\log(p(\text{sigma2.0})p(\text{nu.0})p(\text{mu}))$

loglik log likelihood: $\sum p_k \Phi(\theta_k, \sigma_k)$

mcmc.chains an object of class 'McmcChains' to store MCMC samples

batch a vector of the different batch numbers

batchElements a vector labeling from which batch each observation came from

modes the values of parameters from the iteration which maximizes log likelihood and log prior

mcmc.params An object of class 'McmcParams'

.internal.constraint Constraint on parameters. For internal use only.

MarginalModelExample *This data is a simulated example of Marginal data*

Description

This data is a simulated example of Marginal data

Usage

MarginalModelExample

Value

An example of a ‘MarginalModel‘ MarginalModelExample

Author(s)

Jacob Carey

McmcChains-class *An object to hold estimated parameters.*

Description

An object of this class holds estimates of each parameter at each iteration of the MCMC simulation.

Slots

theta means of each batch and component

sigma2 variances of each batch and component

pi mixture probabilities

mu overall mean in a marginal. In batch model, averaged across batches

tau2 overall variance in a marginal model. In a batch model, weighted average by precision across batches.

nu.0 shape parameter for sigma.2 distribution

sigma2.0 rate parameter for sigma.2 distribution

logprior log likelihood of prior.

loglik log likelihood.

zfreq table of z.

z latent variables

McmcParams	<i>Create an object of class 'McmcParams' to specify iterations, burnin, etc.</i>
------------	---

Description

Create an object of class 'McmcParams' to specify iterations, burnin, etc.

Usage

```
McmcParams(iter = 1000L, burnin = 0L, thin, nStarts = 1,
  param_updates = .param_updates())
```

Arguments

iter	number of iterations
burnin	number of burnin iterations
thin	thinning interval
nStarts	number of chains to run
param_updates	labeled vector specifying whether each parameter is to be updated (1) or not (0).

Value

An object of class 'McmcParams'

Examples

```
mp <- McmcParams(iter=100, burnin=10)
```

mcmcParams	<i>Retrieve MCMC parameters from model.</i>
------------	---

Description

View number of iterations, burnin, etc.

Replace number of iterations, burnin, etc. Any update of the MCMC parameters will trigger an update of the chains. However, if iter (the number of MCMC iterations) is set to a nonpositive value, the chains will not be updated and kept as is.

Usage

```

mcmcParams(object)

mcmcParams(object, force = FALSE) <- value

## S4 method for signature 'MixtureModel'
mcmcParams(object)

## S4 replacement method for signature 'MixtureModel'
mcmcParams(object, force = FALSE) <- value

```

Arguments

object see showMethods(mcmcParams)
force logical value. If false (default) the update will not proceed.
value an object of class 'McmcParams' containing the new number of iterations, etc.

Value

An object of class 'McmcParams'

Examples

```
mcmcParams(MarginalModelExample)
```

<code>McmcParams-class</code>	<i>An object to specify MCMC options for a later simulation</i>
-------------------------------	---

Description

An object to specify MCMC options for a later simulation

Slots

`thin` A one length numeric to specify thinning. A value of `n` indicates that every `n`th sample should be saved. Thinning helps to reduce autocorrelation.
`iter` A one length numeric to specify how many MCMC iterations should be sampled.
`burnin` A one length numeric to specify burnin. The first `n` samples will be discarded.
`nstarts` A one length numeric to specify the number of chains in a simulation.
`param_updates` Indicates whether each parameter should be updated (1) or fixed (0).

Examples

```

McmcParams()
McmcParams(iter=1000)
mp <- McmcParams()
iter(mp)

```

MixtureModel-class *An object for running MCMC simulations.*

Description

BatchModel and MarginalModel both inherit from this class.

Slots

`k` An integer value specifying the number of latent classes.

`hyperparams` An object of class 'Hyperparameters' used to specify the hyperparameters of the model.

`theta` the means of each component and batch

`sigma2` the variances of each component and batch

`nu.0` the shape parameter for `sigma2`

`sigma2.0` the rate parameter for `sigma2`

`pi` mixture probabilities which are assumed to be the same for all batches

`mu` overall mean

`tau2` overall variance

`data` the data for the simulation.

`data.mean` the empirical means of the components

`data.prec` the empirical precisions

`z` latent variables

`zfreq` table of latent variables

`probz` $n \times k$ matrix of probabilities

`logprior` log likelihood of prior: $\log(p(\text{sigma2.0})p(\text{nu.0})p(\text{mu}))$

`loglik` log likelihood: $\sum p_k \Phi(\theta_k, \sigma_k)$

`mcmc.chains` an object of class 'McmcChains' to store MCMC samples

`batch` a vector of the different batch numbers

`batchElements` a vector labeling from which batch each observation came from

`modes` the values of parameters from the iteration which maximizes log likelihood and log prior

`mcmc.params` An object of class 'McmcParams'

`.internal.constraint` Constraint on parameters. For internal use only.

modes	<i>Retrieve the modes from a model.</i>
-------	---

Description

The iteration which maximizes log likelihood and log prior is found. The estimates for each parameter at this iteration are retrieved.

For a mixture model with K components, there are K! possible modes. One can permute the ordering of the modes and assign the permuted order to a MixtureModel derived class by this method.

Usage

```
modes(object)

modes(object) <- value

## S4 method for signature 'DensityModel'
modes(object)

## S4 method for signature 'MixtureModel'
modes(object)

## S4 replacement method for signature 'MixtureModel'
modes(object) <- value
```

Arguments

object	a MixtureModel-derived class
value	a list of the modes. See mode(object) to obtain the correct format of the list.

Value

A list of the modes of each parameter

Examples

```
modes(MarginalModelExample)
```

mu	<i>Retrieve overall mean</i>
----	------------------------------

Description

Retrieve overall mean

Usage

```
mu(object)

## S4 method for signature 'BatchModel'
mu(object)

## S4 method for signature 'MarginalModel'
mu(object)

## S4 method for signature 'McmcChains'
mu(object)
```

Arguments

object see showMethods(mu)

Value

A vector containing 'mu'

Examples

```
mu(MarginalModelExample)
```

muc	<i>Retrieve overall mean at each iteration of the MCMC.</i>
-----	---

Description

Retrieve overall mean at each iteration of the MCMC.

Usage

```
muc(object)
```

Arguments

object an object of class MarginalModel or BatchModel

Value

A vector of length N or matrix of size N x B, where N is the number of observations and B is the number of unique batches.

Examples

```
muc(MarginalModelExample)
```

muMean	<i>Retrieve overall mean averaged across MCMC simulations.</i>
--------	--

Description

Retrieve overall mean averaged across MCMC simulations.

Usage

```
muMean(object)
```

Arguments

object an object of class MarginalModel or BatchModel

Value

A vector of size 1 or number of batches

Examples

```
muMean(MarginalModelExample)
```

names,McmcChains-method	<i>Retrieve the names of the parameters estimated in the MCMC chain.</i>
-------------------------	--

Description

Retrieve the names of the parameters estimated in the MCMC chain.

Usage

```
## S4 method for signature 'McmcChains'  
names(x)
```

Arguments

x an object of class 'McmcChains'

Value

A vector of strings containing the names of each parameter

nStarts	<i>Number of MCMC chains.</i>
---------	-------------------------------

Description

This function retrieves the number of chains used for an MCMC simulation.

This function changes the number of chains used for an MCMC simulation.

Usage

```
nStarts(object)
```

```
nStarts(object) <- value
```

```
## S4 method for signature 'McmcParams'
```

```
nStarts(object)
```

```
## S4 replacement method for signature 'McmcParams'
```

```
nStarts(object) <- value
```

```
## S4 method for signature 'MixtureModel'
```

```
nStarts(object)
```

```
## S4 replacement method for signature 'MixtureModel'
```

```
nStarts(object) <- value
```

Arguments

object see showMethods(nStarts)

value new number of chains

Value

An integer of the number of different starts.

Examples

```
number_of_chains <- nStarts(MarginalModelExample)
```

```
number_of_chains <- 3
```

```
nStarts(MarginalModelExample) <- number_of_chains
```

nu.0	<i>Retrieve the shape parameter for the sigma.2 distribution.</i>
------	---

Description

Retrieve the shape parameter for the sigma.2 distribution.

Usage

```
nu.0(object)
```

```
## S4 method for signature 'McmcChains'  
nu.0(object)
```

```
## S4 method for signature 'MixtureModel'  
nu.0(object)
```

Arguments

object see showMethods(nu.0)

Value

An integer

Examples

```
nu.0(MarginalModelExample)
```

oned	<i>Retrieve data.</i>
------	-----------------------

Description

Retrieve data.

Usage

```
oned(object)
```

```
## S4 method for signature 'MixtureModel'  
oned(object)
```

Arguments

object see showMethods(oned)

Value

A vector the length of the data

p *Retrieve mixture proportions.*

Description

Retrieve mixture proportions.

Usage

p(object)

Arguments

object an object of class MarginalModel or BatchModel

Value

A vector of length the number of components

Examples

p(MarginalModelExample)

pic *Retrieve mixture proportions at each iteration of the MCMC.*

Description

Retrieve mixture proportions at each iteration of the MCMC.

Usage

pic(object)

Arguments

object an object of class MarginalModel or BatchModel

Value

A matrix of size MCMC iterations x Number of components

Examples

pic(MarginalModelExample)

plot	<i>Plot the densities estimated from a mixture model for a copy number polymorphism</i>
------	---

Description

Plot estimates of the posterior density for each component and the overall, marginal density. For batch models, one can additionally plot batch-specific density estimates.

Usage

```
plot(x, y, ...)

## S4 method for signature 'DensityModel,ANY'
plot(x, y, ...)

## S4 method for signature 'MarginalModel,ANY'
plot(x, y, ...)

## S4 method for signature 'BatchModel,ANY'
plot(x, y, show.batch = TRUE, ...)

## S4 method for signature 'DensityBatchModel,ANY'
plot(x, show.batch = TRUE, ...)
```

Arguments

x	a DensityModel-derived object, or a MixtureModel-derived object.
y	If x is a DensityModel, y is a numeric vector of the one-dimensional summaries for a given copy number polymorphism. If x is a MixtureModel, y is ignored.
...	Additional arguments passed to hist.
show.batch	a logical. If true, batch specific densities will be plotted.

Value

A plot showing the density estimate

Examples

```
set.seed(100)
truth <- simulateData(N=2500,
  theta=c(-2, -0.4, 0),
  sds=c(0.3, 0.15, 0.15),
  p=c(0.05, 0.1, 0.8))

mcmcp <- McmcParams(iter=500, burnin=500, thin=2)
model <- MarginalModel(y(truth), k=3, mcmc.params=mcmcp)
```

```

model <- CNPBayes:::startAtTrueValues(model, truth)
model <- posteriorSimulation(model)
par(mfrow=c(1,2), las=1)
plot(truth)
plot(model)

```

posteriorSimulation *Run the MCMC simulation.*

Description

nStarts chains are run. b burnin iterations are run and then discarded. Next, s iterations are run in each train. The user can also specify an alternative number of components. The mode of the MCMC simulation is also calculated.

Usage

```

posteriorSimulation(object, k)

## S4 method for signature 'MixtureModel,ANY'
posteriorSimulation(object)

## S4 method for signature 'MixtureModel,integer'
posteriorSimulation(object, k)

## S4 method for signature 'MixtureModel,numeric'
posteriorSimulation(object, k)

```

Arguments

object	see showMethods(posteriorSimulation)
k	The number of a priori components. This is optional and if not specified, the stored k model components are used. This parameters is useful for running multiple models of varying components.

Value

An object of class 'MarginalModel' or 'BatchModel'

probz	<i>Retrieve the probability of latent variable membership by observation.</i>
-------	---

Description

Retrieve the probability of latent variable membership by observation.

Usage

```
probz(object)

## S4 method for signature 'MixtureModel'
probz(object)
```

Arguments

object see showMethods(probz)

Value

A matrix of size number of observations x number of components

Examples

```
probz(MarginalModelExample)
```

qInverseTau2	<i>Quantiles, shape, and rate of the prior for the inverse of tau2 (the precision)</i>
--------------	--

Description

The precision prior for tau2 in the hierarchical model is given by gamma(shape, rate). The shape and rate are a function of the hyperparameters eta.0 and m2.0. Specifically, shape=1/2*eta.0 and the rate=1/2*eta.0*m2.0. Quantiles for this distribution and the shape and rate can be obtained by specifying the hyperparameters eta.0 and m2.0, or alternatively by specifying the desired mean and standard deviation of the precisions.

Usage

```
qInverseTau2(eta.0 = 1800, m2.0 = 100, mn, sd)
```

Arguments

eta.0	hyperparameter for precision
m2.0	hyperparameter for precision
mn	mean of precision
sd	standard deviation of precision

Value

a list with elements 'quantiles', 'eta.0', 'm2.0', 'mean', and 'sd'

Examples

```

results <- qInverseTau2(mn=100, sd=1)
precision.quantiles <- results$quantiles
sd.quantiles <- sqrt(1/precision.quantiles)
results$mean
results$sd
results$eta.0
results$m2.0

results2 <- qInverseTau2(eta.0=1800, m2.0=100)

## Find quantiles from the default set of hyperparameters
hypp <- Hyperparameters(type="batch")
results3 <- qInverseTau2(eta.0(hypp), m2.0(hypp))
default.precision.quantiles <- results3$quantiles

```

saveBatch

Save se data

Description

Batches drawn from the same distribution as identified by Kolmogorov-Smirnov test are combined.

Usage

```
saveBatch(se, batch.file, THR = 0.1)
```

Arguments

se	a SummarizedExperiment object
batch.file	the file name to which to save the data
THR	threshold below which the null hypothesis should be rejected and batches are collapsed.

Value

A vector of collapsed batch labels

sigma	<i>Retrieve standard deviations of each component/batch mean.</i>
-------	---

Description

Retrieve standard deviations of each component/batch mean.

Usage

```
sigma(object)
```

Arguments

object an object of class `MarginalModel` or `BatchModel`

Value

A vector of length K , or a matrix of size $B \times K$, where K is the number of components and B is the number of batches

Examples

```
sigma(MarginalModelExample)
```

sigma2	<i>Retrieve the variances of each component and batch distribution</i>
--------	--

Description

For a `MarginalModel`, this function returns a vector of variances. For a `BatchModel`, returns a matrix of size number of batches by number of components.

Usage

```
sigma2(object)

## S4 method for signature 'BatchModel'
sigma2(object)

## S4 method for signature 'MarginalModel'
sigma2(object)

## S4 method for signature 'McmcChains'
sigma2(object)
```

Arguments

object see showMethods(sigma2)

Value

A vector of length number of components or a matrix of size number of batches x number of components

Examples

```
sigma2(MarginalModelExample)
```

sigma2.0

Retrieve the rate parameter for the sigma.2 distribution.

Description

Retrieve the rate parameter for the sigma.2 distribution.

Usage

```
sigma2.0(object)

## S4 method for signature 'McmcChains'
sigma2.0(object)

## S4 method for signature 'MixtureModel'
sigma2.0(object)
```

Arguments

object see showMethods(sigma2.0)

Value

A length 1 numeric

Examples

```
sigma2.0(MarginalModelExample)
```

sigmac	<i>Retrieve standard deviation of each component/batch mean at each iteration of the MCMC.</i>
--------	--

Description

Retrieve standard deviation of each component/batch mean at each iteration of the MCMC.

Usage

```
sigmac(object)
```

Arguments

object an object of class MarginalModel or BatchModel

Value

A matrix of size N x K where N is the number of observations and K is the number of components

Examples

```
sigmac(MarginalModelExample)
```

simulateBatchData	<i>Create simulated batch data for testing.</i>
-------------------	---

Description

Create simulated batch data for testing.

Usage

```
simulateBatchData(N = 2500, p, theta, sds, batch, zz)
```

Arguments

N number of observations

p a vector indicating probability of membership to each component

theta a vector of means, one per component/batch

sds a vector of standard deviations, one per component/batch

batch a vector of labels indication from which batch each simulation should come from

zz a vector indicating latent variable membership. Can be omitted.

Value

An object of class 'BatchModel'

Examples

```
k <- 3
nbatch <- 3
means <- matrix(c(-1.2, -1.0, -0.8,
                 -0.2, 0, 0.2,
                 0.8, 1, 1.2), nbatch, k, byrow=FALSE)
sds <- matrix(0.1, nbatch, k)
N <- 1500
truth <- simulateBatchData(N=N,
                          batch=rep(letters[1:3], length.out=N),
                          theta=means,
                          sds=sds,
                          p=c(1/5, 1/3, 1-1/3-1/5))
```

simulateData

Create simulated data for testing.

Description

Create simulated data for testing.

Usage

```
simulateData(N, p, theta, sds)
```

Arguments

N	number of observations
p	a vector indicating probability of membership to each component
theta	a vector of means, one per component
sds	a vector of standard deviations, one per component

Value

An object of class 'MarginalModel'

Examples

```
truth <- simulateData(N=2500, p=rep(1/3, 3),
                    theta=c(-1, 0, 1),
                    sds=rep(0.1, 3))
```

tau	<i>Retrieve overall standard deviation.</i>
-----	---

Description

Retrieve overall standard deviation.

Usage

```
tau(object)
```

Arguments

object an object of class `MarginalModel` or `BatchModel`

Value

A vector of standard deviations

Examples

```
tau(MarginalModelExample)
```

tau2	<i>Accessor for the tau2 parameter in the hierarchical mixture model</i>
------	--

Description

The interpretation of `tau2` depends on whether `object` is a `MarginalModel` or a `BatchModel`. For `BatchModel`, `tau2` is a vector with length equal to the number of components. Each element of the `tau2` vector can be interpreted as the within-component variance of the batch means (`theta`). For objects of class `MarginalModel` (assumes no batch effect), `tau2` is a length-one vector that describes the variance of the component means between batches. The hyperparameters of `tau2` are `eta.0` and `m2.0`. See the following examples for setting the hyperparameters, accessing the current value of `tau2` from a `MixtureModel`-derived object, and for plotting the chain of `tau2` values.

Usage

```
tau2(object)

## S4 method for signature 'BatchModel'
tau2(object)

## S4 method for signature 'MarginalModel'
tau2(object)

## S4 method for signature 'McmcChains'
tau2(object)
```

Arguments

object see showMethods(tau2)

Value

A vector of variances

See Also

Hyperparameters

Examples

```
k(BatchModelExample)
tau2(BatchModelExample)
plot.ts(tau2(chains(BatchModelExample)))
```

tauc

Retrieve overall standard deviation at each iteration of the MCMC.

Description

Retrieve overall standard deviation at each iteration of the MCMC.

Usage

```
tauc(object)
```

Arguments

object an object of class MarginalModel or BatchModel

Value

A vector of length N or matrix of size N x B, where N is the number of observations and B is the number of unique batches.

Examples

```
tauc(MarginalModelExample)
```

tauMean	<i>Retrieve overall standard deviation averaged across MCMC simulations.</i>
---------	--

Description

Retrieve overall standard deviation averaged across MCMC simulations.

Usage

```
tauMean(object)
```

Arguments

object an object of class MarginalModel or BatchModel

Value

A vector of size 1 or number of batches

Examples

```
tauMean(MarginalModelExample)
```

theta	<i>Accessor for the theta parameter in the hierarchical mixture model</i>
-------	---

Description

The interpretation of theta depends on whether object is a MarginalModel or a BatchModel. For BatchModel, theta is a matrix of size B x K, where B is the number of batches and K is the number of components. Each column of the theta matrix can be interpreted as the batch means for a particular component. For objects of class MarginalModel (assumes no batch effect), theta is a vector of length K. Each element of theta can be interpreted as the mean for a component. See the following examples for accessing the current value of theta from a MixtureModel-derived object, and for plotting the chain of theta values.

Usage

```
theta(object)

## S4 method for signature 'BatchModel'
theta(object)

## S4 method for signature 'MarginalModel'
theta(object)
```

```
## S4 method for signature 'McmcChains'
theta(object)
```

Arguments

object see showMethods(theta)

Value

A vector of length number of components or a matrix of size number of batches x number of components

Examples

```
## MarginalModel
k(MarginalModelExample)
theta(MarginalModelExample)
plot.ts(theta(chains(MarginalModelExample)))
## BatchModel
k(BatchModelExample)
length(unique(batch(BatchModelExample)))
theta(BatchModelExample)
## Plot means for batches in one component
plot.ts(theta(chains(BatchModelExample))[, 1:3])
```

thin	<i>Number of thinning intervals.</i>
------	--------------------------------------

Description

This function retrieves the number of thinning intervals used for an MCMC simulation.

Usage

```
thin(object)

## S4 method for signature 'McmcParams'
thin(object)

## S4 method for signature 'MixtureModel'
thin(object)
```

Arguments

object see showMethods(thin)

Value

An integer of the number of thinning intervals

Examples

```
thin(MarginalModelExample)
```

tracePlot

Create a trace plot of a parameter estimated by MCMC.

Description

Create a trace plot of a parameter estimated by MCMC.

Usage

```
tracePlot(object, name, ...)
```

```
## S4 method for signature 'BatchModel'
tracePlot(object, name, ...)
```

Arguments

object	see showMethods(tracePlot)
name	the name of the parameter for which to plot values. Can be 'theta', 'sigma', 'p', 'mu', or 'tau'.
...	Other argument to pass to plot.

Value

A traceplot of a parameter value

Examples

```
tracePlot(BatchModelExample, "theta")
tracePlot(BatchModelExample, "sigma")
```

y *Retrieve data.*

Description

Retrieve data.

Usage

```
y(object)
```

```
## S4 method for signature 'DensityModel'  
y(object)
```

```
## S4 method for signature 'MixtureModel'  
y(object)
```

Arguments

object see showMethods(y)

Value

A vector containing the data

Examples

```
y(MarginalModelExample)
```

z *Retrieve latent variable assignments.*

Description

Retrieves the simulated latent variable assignments of each observation at each MCMC simulation.

Usage

```
z(object)
```

```
## S4 method for signature 'McmcChains'  
z(object)
```

```
## S4 method for signature 'MixtureModel'  
z(object)
```

Arguments

object see showMethods(z)

Value

A vector the length of the data

Examples

```
z(MarginalModelExample)
```

zFreq	<i>Calculates a frequency table of latent variable assignments by observation.</i>
-------	--

Description

Calculates a frequency table of latent variable assignments by observation.

Usage

```
zFreq(object)

## S4 method for signature 'McmcChains'
zFreq(object)

## S4 method for signature 'MixtureModel'
zFreq(object)
```

Arguments

object see showMethods(zfreq)

Value

An integer vector of length the number of components

Examples

```
zFreq(MarginalModelExample)
```

Index

- [,BatchModel,ANY,ANY,ANY-method (extract), 17
- [,BatchModel,ANY,ANY-method (extract), 17
- [,BatchModel,ANY-method (extract), 17
- [,BatchModel-method (extract), 17
- [,McmcChains,ANY-method (extract), 17
- [,McmcChains-method (extract), 17
- [,McmcParams,ANY-method (extract), 17
- [,McmcParams-method (extract), 17

- batch, 3
- batch,DensityModel-method (batch), 3
- batch,MixtureModel-method (batch), 3
- BatchModel, 4
- BatchModel-class, 5
- BatchModelExample, 6
- bic, 6
- bic,BatchModel-method (bic), 6
- bic,MarginalModel-method (bic), 6
- burnin, 7
- burnin,McmcParams-method (burnin), 7
- burnin,MixtureModel-method (burnin), 7
- burnin<- (burnin), 7
- burnin<-,McmcParams-method (burnin), 7
- burnin<-,MixtureModel-method (burnin), 7

- chains, 8
- chains,MixtureModel-method (chains), 8
- chromosome, 8
- clusters, 9
- clusters,DensityModel-method (clusters), 9
- CNPBayes, 10
- CNPBayes-package (CNPBayes), 10
- collapseBatch, 10
- collapseBatch,BatchModel-method (collapseBatch), 10
- collapseBatch,numeric-method (collapseBatch), 10

- collapseBatch,SummarizedExperiment-method (collapseBatch), 10
- consensusCNP, 11

- DensityBatchModel-class (DensityModel-class), 14
- DensityModel, 13, 14
- DensityModel-class, 14
- downsample, 15
- downSampleEachBatch, 16

- eta.0, 16
- eta.0,Hyperparameters-method (eta.0), 16
- eta.0,MixtureModel-method (eta.0), 16
- extract, 17

- Hyperparameters, 18
- Hyperparameters-class, 18
- HyperparametersBatch, 19
- HyperparametersBatch-class, 20
- HyperparametersMarginal, 21
- HyperparametersMarginal-class, 22
- hyperParams, 22
- hyperParams,MixtureModel-method (hyperParams), 22
- hyperParams<- (hyperParams), 22
- hyperParams<-,MixtureModel,Hyperparameters-method (hyperParams), 22
- hyperParams<-,MixtureModel-method (hyperParams), 22

- iter (iter<-), 23
- iter,burnin,nStarts,McmcParams-method (McmcParams-class), 33
- iter,McmcParams-method (iter<-), 23
- iter,MixtureModel-method (iter<-), 23
- iter<-, 23
- iter<-,McmcParams-method (iter<-), 23
- iter<-,MixtureModel-method (iter<-), 23

- k, 24

- k, DensityModel-method (k), 24
- k, Hyperparameters-method (Hyperparameters-class), 18
- k, MixtureModel-method (k), 24
- k<- (k), 24
- k<-, Hyperparameters-method (k), 24
- k<-, Hyperparameters-method (k), 24
- k<-, MixtureModel-method (k), 24
- kmeans, 13
- log_lik, 26
- log_lik, McmcChains-method (log_lik), 26
- log_lik, MixtureModel-method (log_lik), 26
- logBayesFactor, 25
- logPrior, 25
- logPrior, McmcChains-method (logPrior), 25
- logPrior, MixtureModel-method (logPrior), 25
- m2.0, 27
- m2.0, Hyperparameters-method (m2.0), 27
- m2.0, MixtureModel-method (m2.0), 27
- map, 27
- mapCnProbability, 28
- marginalLikelihood, 28
- marginalLikelihood, BatchModel, ANY-method (marginalLikelihood), 28
- marginalLikelihood, BatchModel, integer-method (marginalLikelihood), 28
- marginalLikelihood, BatchModel-method (marginalLikelihood), 28
- marginalLikelihood, list, ANY-method (marginalLikelihood), 28
- marginalLikelihood, list, integer-method (marginalLikelihood), 28
- marginalLikelihood, list-method (marginalLikelihood), 28
- marginalLikelihood, MarginalModel, ANY-method (marginalLikelihood), 28
- marginalLikelihood, MarginalModel, integer-method (marginalLikelihood), 28
- marginalLikelihood, MarginalModel-method (marginalLikelihood), 28
- marginalLikelihood, SingleBatchPooledVar, integer-method (marginalLikelihood), 28
- MarginalModel, 29
- MarginalModel-class, 30
- MarginalModelExample, 31
- McmcChains-class, 31
- McmcParams, 32
- mcmcParams, 32
- mcmcParams, MixtureModel-method (mcmcParams), 32
- McmcParams-class, 33
- mcmcParams<- (mcmcParams), 32
- mcmcParams<-, MixtureModel-method (mcmcParams), 32
- MixtureModel-class, 34
- modes, 35
- modes, DensityModel-method (modes), 35
- modes, MixtureModel-method (modes), 35
- modes<- (modes), 35
- modes<-, MixtureModel-method (modes), 35
- mu, 36
- mu, BatchModel-method (mu), 36
- mu, MarginalModel-method (mu), 36
- mu, McmcChains-method (mu), 36
- muc, 36
- muMean, 37
- names, McmcChains-method, 37
- nStarts, 38
- nStarts, McmcParams-method (nStarts), 38
- nStarts, MixtureModel-method (nStarts), 38
- nStarts<- (nStarts), 38
- nStarts<-, McmcParams-method (nStarts), 38
- nStarts<-, MixtureModel-method (nStarts), 38
- nu.0, 39
- nu.0, McmcChains-method (nu.0), 39
- nu.0, MixtureModel-method (nu.0), 39
- oned, 39
- oned, MixtureModel-method (oned), 39
- p, 40
- pic, 40
- plot, 41
- plot, BatchModel, ANY-method (plot), 41
- plot, DensityBatchModel, ANY-method (plot), 41
- plot, DensityModel, ANY-method (plot), 41
- plot, DensityModel, numeric-method (plot), 41

- plot, MarginalModel, ANY-method (plot), 41
- posteriorSimulation, 42
- posteriorSimulation, MixtureModel, ANY-method (posteriorSimulation), 42
- posteriorSimulation, MixtureModel, integer-method (posteriorSimulation), 42
- posteriorSimulation, MixtureModel, numeric-method (posteriorSimulation), 42
- posteriorSimulation, MixtureModel-method (posteriorSimulation), 42
- probz, 43
- probz, MixtureModel-method (probz), 43
- qInverseTau2, 43
- saveBatch, 44
- sigma, 45
- sigma2, 45
- sigma2, BatchModel-method (sigma2), 45
- sigma2, MarginalModel-method (sigma2), 45
- sigma2, McmcChains-method (sigma2), 45
- sigma2, missing-method (sigma2), 45
- sigma2.0, 46
- sigma2.0, McmcChains-method (sigma2.0), 46
- sigma2.0, MixtureModel-method (sigma2.0), 46
- sigmac, 47
- simulateBatchData, 47
- simulateData, 48
- tau, 49
- tau2, 49
- tau2, BatchModel-method (tau2), 49
- tau2, MarginalModel-method (tau2), 49
- tau2, McmcChains-method (tau2), 49
- tauc, 50
- tauMean, 51
- theta, 51
- theta, BatchModel-method (theta), 51
- theta, MarginalModel-method (theta), 51
- theta, McmcChains-method (theta), 51
- thin, 52
- thin, McmcParams-method (thin), 52
- thin, MixtureModel-method (thin), 52
- tracePlot, 53
- tracePlot, BatchModel-method (tracePlot), 53
- y, 54
- y, DensityModel-method (y), 54
- y, MixtureModel-method (y), 54
- z, 54
- z, McmcChains-method (z), 54
- z, MixtureModel-method (z), 54
- zFreq, 55
- zFreq, McmcChains-method (zFreq), 55
- zfreq, McmcChains-method (zFreq), 55
- zFreq, MixtureModel-method (zFreq), 55
- zfreq, MixtureModel-method (zFreq), 55