

Package ‘CNPBayes’

January 22, 2019

Type Package

Title Bayesian mixture models for copy number polymorphisms

Version 1.13.5

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, Jacob Carey, and Rob Scharpf

Depends R (>= 3.4.0), IRanges, GenomicRanges

Imports Rcpp (>= 0.12.1), S4Vectors, matrixStats, RColorBrewer, gtools, combinat, GenomeInfoDb (>= 1.11.6), methods, BiocGenerics, graphics, stats, coda, SummarizedExperiment, mclust, reshape2, ggplot2, magrittr, purrr, tidyr, dplyr, tibble, scales, data.table, RcppArmadillo

Suggests testthat, knitr, BiocStyle, rmarkdown, BiocCheck, MASS, VanillaICE, tidyverse

Collate 'help.R' 'AllGenerics.R' 'AllClasses.R' 'Cfunctions.R' 'Defunct-classes.R' 'MultiBatch.R' 'MultiBatchList.R' 'MultiBatchP.R' 'RcppExports.R' 'augment.R' 'coerce-methods.R' 'copynumber-models.R' 'data.R' 'defunct-functions.R' 'functions.R' 'genotype_cnps.R' 'gibbs.R' 'marginal_likelihood.R' 'methods-Hyperparameters.R' 'methods-McmcChains.R' 'methods-McmcParams.R' 'methods-MixtureModel.R' 'methods-MultiBatchModel.R' 'methods-MultiBatchPooled.R' 'methods-SingleBatchModel.R' 'methods-SingleBatchPooled.R' 'methods-SummarizedExperiment.R' 'methods-TrioBatchModel.R' 'model_initialization.R' 'plot-functions.R' 'posteriorSimulation-methods.R' 'relabeling.R' 'simulate_data.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

Encoding UTF-8

RoxygenNote 6.1.1

git_url <https://git.bioconductor.org/packages/CNPBayes>

git_branch master

git_last_commit 1e9cddb

git_last_commit_date 2019-01-03

Date/Publication 2019-01-21

R topics documented:

baflikelihood	4
batch	5
BatchModelExample	5
bayesFactor	6
bic	6
burnin	7
chains	7
chromosome	8
CNPBayes	9
collapseBatch	9
consensusCNP	10
copyNumber	12
CopyNumberModel	13
DensityModel-class	14
dfr	17
downSample	17
eta.0	18
findSurrogates,MultiBatch-method	19
ggChains	20
gibbs	21
Hyperparameters	23
Hyperparameters-class	24
HyperparametersBatch-class	24
HyperparametersMarginal-class	25
HyperparametersMultiBatch	25
HyperparametersMultiBatch-class	26
HyperparametersSingleBatch	27
HyperparametersSingleBatch-class	28
HyperparametersTrios	28
HyperparametersTrios-class	29
hyperParams	29
iter<-	30
k	31
label_switch	32
logBayesFactor	32
logPrior	33
log_lik	33

m2.0	34
mapCnProbability	35
mapParams	35
mapping	36
map_z	37
marginalLik	37
marginalLikelihood	38
MarginalModelExample	39
marginal_lik	39
MBP	40
McmcChains-class	41
McmcParams	41
mcmcParams	42
McmcParams-class	43
MixtureModel-class	44
mlParams	45
modelName,MixtureModel-method	46
modes	46
mu	47
muc	48
MultiBatch-class	48
MultiBatchModel-class	49
MultiBatchModel2	50
MultiBatchModelExample	51
MultiBatchPooledExample	52
muMean	52
names,McmcChains-method	53
nStarts	53
nu.0	54
numberObs	55
oned	55
orderModels	56
p,MixtureModel-method	56
pic	57
posteriorPredictive	57
posteriorSimulation	58
posterior_cases	59
pp	60
probCopyNumber	60
probz	61
probzpar	61
qInverseTau2	62
saveBatch	63
sigma	63
sigma2.0	64
sigma<-	64
sigmac	65
simulateBatchData	66
simulateData	67
SingleBatchCopyNumber-class	67
singleBatchGuided,MultiBatchList,MultiBatch-method	68
SingleBatchModel-class	68

SingleBatchModel2	69
SingleBatchModelExample	70
tau	70
tau2	71
tauc	72
tauMean	72
theta	73
thin	74
tileMedians	74
TrioBatchModel	75
triodata_1rr	76
upSample2	76
y	78
z	78
zFreq	79
zFreqPar	79
[,McmcChains,ANY,ANY,ANY-method	80

Index **81**

bafLikelihood	<i>Calculate the likelihood of the observed B allele frequencies for a given copy number model</i>
---------------	--

Description

Calculate the likelihood of the observed B allele frequencies for a given copy number model

Usage

```
bafLikelihood(cn.model, snpdata)
```

Arguments

cn.model	a copy number model
snpdata	a SummarizedExperiment with assay element 'GT' containing an integer coding (1, 2, and 3) for the generic genotypes AA, AB, and BB, respectively.

Value

a list. The first element is the log likelihood and the second element is the copy number model that maximized the likelihood.

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

Description

The batches are represented as a vector of integers.

Usage

```
batch(object)
```

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

Arguments

object see showMethods(batch)

Value

The batch of each data element.

Examples

```
batch(MultiBatchModelExample)
```

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

bayesFactor	<i>Compute the Bayes factor</i>
-------------	---------------------------------

Description

Calculated as $\log(\text{ML1}) - \log(\text{ML2}) + \log$ prior odds where ML1 is the marginal likelihood of the model with the most free parameters

Usage

```
bayesFactor(model.list, prior.odds = 1)
```

Arguments

model.list	list of models from gibbs
prior.odds	scalar

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

Description

Calculate BIC of a model

Usage

```
bic(object)

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

Arguments

object	see showMethods(bic)
--------	----------------------

Value

The BIC of the model.

Examples

```
mb <- MultiBatchModelExample
mcmcParams(mb) <- McmcParams(iter=100, burnin=50)
mb <- posteriorSimulation(mb)
bic(mb)
```

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,ANY'  
burnin(object) <- value
```

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

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

Arguments

object see showMethods(burnin)

value new number of burnin iterations

Value

The number of burnin simulations.

Examples

```
burnin(SingleBatchModelExample)  
mp <- mcmcParams(SingleBatchModelExample)  
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(MultiBatchModelExample))
dim(theta.chain)
plot.ts(theta.chain, plot.type="single",
        col=seq_len(k(SingleBatchModelExample)))
```

chromosome

Extract character vector of sequence names

Description

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

Usage

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

S4 method for signature 'GenomicRanges'

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

Arguments

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

Value

A character vector

Examples

```
g <- GRanges("chr1", IRanges(10, 15))
chromosome(g)
```


CNPBayes

*Bayesian mixture models for copy number estimation***Description**

Bayesian mixture models for copy number estimation

collapseBatch

Estimate batch from any sample-level surrogate variables that capture aspects of sample processing, such as the PCR experiment (e.g., the 96 well chemistry plate), laboratory, DNA source, or DNA extraction method.

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. For smaller values of THR, plates are more likely to be judged as similar and combined.

Usage

```
collapseBatch(object, provisional_batch, THR = 0.1, nchar = 8)
```

```
## S4 method for signature 'MultiBatchModel'
collapseBatch(object)
```

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

```
## S4 method for signature 'numeric'
collapseBatch(object, provisional_batch, THR = 0.1,
  nchar = 8)
```

Arguments

object	see showMethods(collapseBatch)
provisional_batch	a vector labelling from which batch each observation came from.
THR	p-value threshold below which the null hypothesis should be rejected and batches are collapsed
nchar	integer specifying the maximum number of characters in the batch labels

Value

The new batch value.

Examples

```
mb.ex <- MultiBatchModelExample
batches <- batch(mb.ex)
bt <- collapseBatch(y(mb.ex), batches)
batches <- as.integer(factor(bt))
hp <- hpList(k=k(mb.ex))["MB"]
model <- MB(dat=y(mb.ex),
            hp=hp,
            batch=batches,
            mp=mcmcParams(mb.ex))
```

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)
## Not run:
cnps <- consensusCNP(gr1)
##
## 2nd CNP is filtered because of its size
##
truth <- GRanges("chr1", IRanges(10000100L, 10100100L))
seqinfo(truth) <- seqinfo(gr1)
identical(cnps, truth)

## End(Not run)

##
## Both CNVs identified

```

```
##
## Not run:
  cnps <- consensusCNP(gr1, max.width=500e3)

## End(Not run)
truth <- GRanges(c("chr1", "chr5"),
                 IRanges(c(10000100L, 101000999L),
                         c(10100100L, 101400999L)))
seqlevels(truth, pruning.mode="coarse") <- seqlevels(gr1)
seqinfo(truth) <- seqinfo(gr1)
## Not run:
  identical(cnps, truth)

## End(Not run)
```

copyNumber

Extract copy number estimates from a 'CopyNumberModel'

Description

Extract copy number estimates from a 'CopyNumberModel'

Usage

```
copyNumber(object)

## S4 method for signature 'MultiBatchCopyNumber'
copyNumber(object)

## S4 method for signature 'MultiBatchCopyNumberPooled'
copyNumber(object)
```

Arguments

object a SingleBatchCopyNumber or MultiBatchCopyNumber object

See Also

[CopyNumberModel](#)

Examples

```
sb <- SingleBatchModelExample
cn.model <- CopyNumberModel(sb)
head(copyNumber(cn.model))

## here is an identity mapping
## Not run:
  mapping(cn.model) <- 1:3
  identical(copyNumber(cn.model), z(cn.model))
  table(copyNumber(cn.model))

## here, we map the first two mixture components to one copy number state
```

```

mapping(cn.model) <- c(1, 1, 2)
table(copyNumber(cn.model))

## End(Not run)

```

CopyNumberModel	<i>Constructs a CopyNumberModel from SB, SBP, MB, or MBP models</i>
-----------------	---

Description

The mixture components do not necessarily reflect distinct copy number states, possibly due to skewed (non-Gaussian) log R ratios. While easy to fit skewed data with a finite mixture of Gaussians, additional steps are needed to assess whether the components correspond to distinct copy number states. This accessor `copyNumber` returns the copy number states – i.e., the result after mapping mixture components to copy number states.

Usage

```

CopyNumberModel(model, params = mapParams())

SingleBatchCopyNumber(model)

MultiBatchCopyNumber(model)

MultiBatchCopyNumberPooled(model)

## S4 method for signature 'SingleBatchModel'
CopyNumberModel(model, params = mapParams())

## S4 method for signature 'MultiBatchModel'
CopyNumberModel(model, params = mapParams())

## S4 method for signature 'MultiBatchPooled'
CopyNumberModel(model, params = mapParams())

mapCopyNumber(model, params = mapParams())

```

Arguments

<code>model</code>	a SB, SBP, MB, or MBP model
<code>params</code>	a list of parameters used for mapping mixture components to copy number states.

Value

a MultiBatchCopyNumber instance
a MultiBatchCopyNumber instance

See Also

[copyNumber](#)

Examples

```
sb <- SingleBatchModelExample
cn.model <- CopyNumberModel(sb, mapParams())
```

DensityModel-class *Defunct classes in CNPBayes*

Description

The following classes in CNPBayes are deprecated and are provided only for compatibility.

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.

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'
`label_switch` length-one logical vector indicating whether label-switching occurs (possibly an overfit model)
`.internal.constraint` Constraint on parameters. For internal use only.
`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
`label_switch` length-one logical vector indicating whether label-switching occurs (possibly an overfit model)
`mcmc.params` An object of class 'McmcParams'
`.internal.constraint` Constraint on parameters. For internal use only.
`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: $\log(p(\sigma^2_0)p(\nu_0)p(\mu))$
`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
`label_switch` length-one logical vector indicating whether label-switching occurs (possibly an overfit model)
`mcmc.params` An object of class 'McmcParams'
`.internal.constraint` Constraint on parameters. For internal use only.
`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: $\log(p(\sigma^2_0)p(\nu_0)p(\mu))$
`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'
`label_switch` length-one logical vector indicating whether label-switching occurs (possibly an overfit model)
`.internal.constraint` Constraint on parameters. For internal use only.

dfr	<i>Accessor for degrees of freedom</i>
-----	--

Description

Accessor for degrees of freedom

Accessor for degrees of freedom

Usage

```
dfr(object)
```

```
dfr(object) <- value
```

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

```
dfr(object)
```

```
## S4 replacement method for signature 'Hyperparameters,numeric'
```

```
dfr(object) <- value
```

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

```
dfr(object)
```

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

```
dfr(object) <- value
```

Arguments

object a Hyperparameters- or MixtureModel-derived class

value scalar providing degrees of freedom for t-distribution

Examples

```
hp <- Hyperparameters()
dfr(hp)
dfr(hp) <- 10
dfr(hp)
```

downSample	<i>Down sample the observations in a mixture</i>
------------	--

Description

For large datasets (several thousand subjects), the computational burden for fitting Bayesian mixture models can be high. Downsampling can reduce the computational burden with little effect on inference. This function draws a random sample with replacement. Batches with few observations are combined with larger batches that have a similar median log R ratio.

Usage

```
downSample(dat, size = 1000, min.batchsize = 75)
```

Arguments

<code>dat</code>	data.frame with required columns medians, batches, and plate
<code>size</code>	the number of observations to sample with replacement
<code>min.batchsize</code>	the smallest number of observations allowed in a batch. Batches smaller than this size will be combined with other batches

Value

A tibble of the downsampled data (medians), the original batches, and the updated batches after downsampling

Examples

```
## TODO: this is more complicated than it needs to be
library(dplyr)
mb <- MultiBatchModelExample
mapping <- tibble(plate=letters[1:10],
                  batch_orig=sample(c("1", "2", "3"), 10, replace=TRUE))
full.data <- tibble(medians=y(mb),
                   batch_orig=as.character(batch(mb))) %>%
  left_join(mapping, by="batch_orig")
partial.data <- downSample(full.data, 200)
## map the original batches to the batches after down-sampling
mapping <- partial.data %>%
  select(c(plate, batch_index)) %>%
  group_by(plate) %>%
  summarize(batch_index=unique(batch_index))
mp <- McmcParams(iter=50, burnin=100)
## Not run:
mb2 <- MultiBatchModel2(dat=ds$medians,
                       batches=ds$batch_index, mp=mp)
mb2 <- posteriorSimulation(mb2)
if(FALSE) ggMixture(mb2)
full.dat2 <- full.data %>%
  left_join(mapping, by="plate")
## compute probabilities for the full dataset
mb.up <- upSample2(full.dat2, mb2)
if(FALSE) ggMixture(mb2)

## End(Not run)
```

 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

object see showMethods(eta.0)

Value

eta.0 of a 'MixtureModel'

Examples

```
eta.0(SingleBatchModelExample)
```

findSurrogates,MultiBatch-method

Estimate batch from any sample-level surrogate variables that capture aspects of sample processing, such as the PCR experiment (e.g., the 96 well chemistry plate), laboratory, DNA source, or DNA extraction method.

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. For smaller values of THR, plates are more likely to be judged as similar and combined.

Usage

```
## S4 method for signature 'MultiBatch'
findSurrogates(object, THR = 0.1)
```

Arguments

object	a MultiBatch instance
THR	scalar for the p-value cutoff from the K-S test. Two batches with p-value > THR will be combined to define a single new batch

Details

All pairwise comparisons of batches are performed. The two most similar batches are combined if the p-value exceeds THR. The process is repeated recursively until no two batches can be combined.

Value

MultiBatch object

ggChains

Trace plots of MCMC chains and mixture model densities

Description

The ggChains method provides a convenient wrapper for plotting the chains of all parameters in the various mixture model implementations. In addition to the estimated number of independent MCMC draws (effective sample size) and Gelman-Rubin convergence diagnostics implemented in gibbs, visualization of the chains is helpful for assessing convergence.

Usage

```
ggChains(model)

ggMixture(model, bins = 100)

## S4 method for signature 'MultiBatchCopyNumber'
ggMixture(model, bins = 100)

## S4 method for signature 'MultiBatchCopyNumberPooled'
ggMixture(model, bins = 100)

## S4 method for signature 'MultiBatchModel'
ggMixture(model, bins = 100)

## S4 method for signature 'MultiBatch'
ggMixture(model, bins = 100)

## S4 method for signature 'MultiBatchPooled'
ggMixture(model, bins = 100)

## S4 method for signature 'MultiBatchModel'
ggChains(model)

## S4 method for signature 'MultiBatchPooled'
ggChains(model)
```

Arguments

`model` A SB, MB, SBP, or MBP model
`bins` a length-one numeric vector indicating the number of bins – passed to `geom_hist`

Details

The `ggMixture` method overlays the density of the posterior predictive distribution of the Gaussian mixture on the empirical data. `ggMixture` assumes that you have already run the Gibbs sampler either by the `gibbs` function or by the `posteriorSimulation` function.

Value

A `gg` object

See Also

[gibbs](#)

Examples

```
sb <- SingleBatchModelExample
iter(sb) <- 1000
burnin(sb) <- 100
sb <- posteriorSimulation(sb)
fig.chains <- ggChains(sb)
## component-specific chains
fig.chains[["comp"]]
## single-parameter chains and log-likelihood
fig.chains[["single"]]

## plot the mixture
fig.mix <- ggMixture(sb)
data(MultiBatchModelExample)
fig <- ggMixture(MultiBatchModelExample)
```

<code>gibbs</code>	<i>Run a Gibbs sampler on one or multiple types of Bayesian Gaussian mixture models</i>
--------------------	---

Description

Model types: SB (SingleBatchModel): hierarchical model with mixture component-specific means and variances; MB (MultiBatchModel): hierarchical model with mixture component- and batch-specific means and variances; SBP (SingleBatchPooled): similar to SB model but with a pooled MBP (MultiBatchPooled): similar to MB model but with a pooled estimate of the variance (across mixture components) for each batch.

Usage

```
gibbs(model = c("SB", "MB", "SBP", "MBP", "TBM"), dat, mp, hp.list,
      batches, k_range = c(1, 4), max_burnin = 32000, min_GR = 1.2,
      top = 2, df = 100, min_effsize = 500, maplabel, mprob)
```

Arguments

<code>model</code>	a character vector indicating which models to fit (any combination of 'SB', 'MB', 'SBP', and 'MBP')
<code>dat</code>	numeric vector of the summary copy number data for each sample at a single CNP (e.g., the median log R ratio for each sample)
<code>mp</code>	an object of class <code>McmcParams</code>
<code>hp.list</code>	a list of hyperparameters for each of the different models. If missing, this list will be generated automatically with default hyperparameters that work well for copy number data
<code>batches</code>	an integer vector of the same length as <code>dat</code> indicating the batch in which the sample was processed
<code>k_range</code>	a length-two numeric vector providing the minimum and maximum number of components to model. For example, <code>c(1, 3)</code> will fit mixture models with 1, 2, and 3 components.
<code>max_burnin</code>	the maximum number of burnin iterations. See details.
<code>min_GR</code>	length-1 numeric vector specifying the maximum Gelman-Rubin (GR) statistic. If the GR statistic is above this value, the marginal likelihood will not be estimated.
<code>top</code>	a length-one numeric vector indicating how many of the top models to return.
<code>df</code>	length-1 numeric vector for t-distribution degrees of freedom
<code>min_effsize</code>	length-1 numeric vector specifying the minimum effective size of the MCMC simulations. If below this value, the marginal likelihood will not be estimated.

Details

For each model specified, a Gibbs sampler will be initiated

The number of mixture models fit depends on `k_range` and

Value

A list of models of length `top` sorted by decreasing

See Also

[gelman.diag](#) [effectiveSize](#) [marginalLikelihood](#) See [ggChains](#)

Examples

```
set.seed(100)
nbatch <- 3
k <- 3
means <- matrix(c(-2.1, -2, -1.95, -0.41, -0.4, -0.395, -0.1,
  0, 0.05), nbatch, k, byrow = FALSE)
sds <- matrix(0.15, nbatch, k)
sds[, 1] <- 0.3
N <- 1000
truth <- simulateBatchData(N = N, batch = rep(letters[1:3],
  length.out = N),
  p = c(1/10, 1/5, 1 - 0.1 - 0.2),
  theta = means,
```

```

                                sds = sds)
## clearly not enough simulations
mp <- McmcParams(iter=10, burnin=5)
## this should be much higher > 1,000
max_burnin <- 10
## suppress the warnings from the short chains
suppressWarnings(gibbs(model=c("SB", "SBP", "MB", "MBP"),
  dat=y(truth),
  mp=mp,
  batches=batch(truth),
  k_range=c(1, 4),
  max_burnin=max_burnin,
  min_GR=1.2,
  top=2,
  df=100))

```

Hyperparameters *Create an object of class 'Hyperparameters'*

Description

The elements of the list are named by the type of model (SB, MB, SBP, MBP).

Usage

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

```
hpList(...)
```

Arguments

type	specifies 'marginal' or 'batch'
k	number of components
...	additional arguments passed to hyperparameter constructors

Details

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

Value

An object of class `HyperparametersMarginal` or `HyperparametersBatch`
a list of hyperparameter objects

See Also

[HyperparametersMultiBatch](#) [Hyperparameters](#) [HyperparametersTrios](#)

Examples

```

hypp <- Hyperparameters("marginal", k=2)
hp.list <- hpList(k=3)
hp.list[["SB"]]

```

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
dfn positive number for t-distribution degrees of freedom

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 hierarchical 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

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

HyperparametersMultiBatch

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

Description

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

Usage

```
HyperparametersMultiBatch(k = 3L, mu.0 = 0, tau2.0 = 0.4,
  eta.0 = 32, m2.0 = 0.5, alpha, beta = 0.1, a = 1.8, b = 6,
  dfr = 100)
```

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 $1/2 * \text{eta}.0 * \text{m2}.0$. 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)
df r	length-one numeric vector for t-distribution degrees of freedom

Value

An object of class HyperparametersBatch

See Also

[hpList](#)

Examples

```
HyperparametersMultiBatch(k=3)
```

HyperparametersMultiBatch-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
 dfr positive number for t-distribution degrees of freedom

HyperparametersSingleBatch

Create an object of class 'HyperparametersSingleBatch' for the single batch mixture model

Description

Create an object of class 'HyperparametersSingleBatch' for the single batch mixture model

Usage

```
HyperparametersSingleBatch(k = 0L, mu.0 = 0, tau2.0 = 0.4,
  eta.0 = 32, m2.0 = 0.5, alpha, beta = 0.1, a = 1.8, b = 6,
  dfr = 100)
```

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)
dfr	length-one numeric vector for t-distribution degrees of freedom

Value

An object of class `HyperparametersSingleBatch`

Examples

```
HyperparametersSingleBatch(k=3)
```

`HyperparametersSingleBatch`-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`
`dfr` positive number for t-distribution degrees of freedom

`HyperparametersTrios` *Create an object of class 'Hyperparameters' with additional parameters for Trios*

Description

Create an object of class 'Hyperparameters' with additional parameters for Trios

Usage

```
HyperparametersTrios(k = 3, mu.0 = 0, tau2.0 = 0.4, eta.0 = 32,
  m2.0 = 0.5, alpha, beta = 0.1, a = 1.8, b = 6, dfr = 100)
```

Value

An object of class `HyperparameterTrios`

Examples

```
hyp.trio <- HyperparametersTrios(k=3)
```

HyperparametersTrios-class

An object to specify the hyperparameters of a model with additional parameters for trios

Description

This class inherits from the hyperparameters class.

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
 dfn positive number for t-distribution degrees of freedom

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

```
hyperParams(SingleBatchModelExample)
hypp <- Hyperparameters(type="marginal",
                        k=k(SingleBatchModelExample),
                        alpha=c(9, 9, 10))
hyperParams(SingleBatchModelExample) <- 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) <- value

iter(object)

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

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

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

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

Arguments

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

Value

The number of MCMC iterations

Examples

```
iter(SingleBatchModelExample)
```

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

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

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

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

## S4 method for signature 'TrioBatchModel'
k(object)

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

Arguments

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

Value

The number of components

Examples

```
k(SingleBatchModelExample) <- 2
```

label_switch	<i>Accessor for determining whether label switching occurred during MCMC</i>
--------------	--

Description

Accessor for determining whether label switching occurred during MCMC

Usage

```
label_switch(object)

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

Arguments

object MixtureModel-derived class

Examples

```
label_switch(SingleBatchModelExample)
```

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 each set of two models by 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(SingleBatchModelExample)
```

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
head(log_lik(chains(SingleBatchModelExample)))
## retrieve log likelihood at last MCMC iteration
log_lik(SingleBatchModelExample)
```

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(SingleBatchModelExample)
```

mapCnProbability	<i>Probabilistic copy number assignments.</i>
------------------	---

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 x K where N is number of observations and K is the number of components.

mapParams	<i>Parameters for mapping mixture components to distinct copy number states</i>
-----------	---

Description

Parameters for mapping mixture components to distinct copy number states

Usage

```
mapParams(threshold = 0.1, proportion.subjects = 0.5,
  outlier.variance.ratio = 5, max.homozygous = c(-1.5, -0.5),
  min.foldchange = 1.5)
```

Arguments

threshold	numeric value in [0, 0.5]. For a given observation (sample), mixture component probabilities > threshold and less than 1-threshold are combined.
proportion.subjects	numeric value in [0, 1]. Two components are combined if the fraction of subjects with component probabilities in the range [threshold, 1-threshold] exceeds this value.
outlier.variance.ratio	if the ratio of the component variance to the median variance of the other component exceeds the value of this argument, the component is considered to correspond to outliers.
max.homozygous	length-2 numeric vector of cutoffs used for establishing a homozygous deletion component. The first element is the cutoff for the mean log R ratios when there are 2 or more states. The second element is the cutoff for the mean log R ratio when there are 3 or more states.

`min.foldchange` a length-one numeric vector. When there are 3 or more states, we compute the ratio of the distance between the means of the sorted components 1 and 2 (the 2 components with lowest means) and the distance between the means of components 2 and 3. If the ratio (i) exceeds the value specified by this parameter, (ii) there are 3 or more states, and (iii) the first component has a mean less than `max_homozygous[2]`, we infer that the first component is a homozygous deletion.

Examples

```
mapParams()
```

mapping	<i>Map mixture components to copy number states</i>
---------	---

Description

Map mixture components to copy number states

Usage

```
mapping(object)

mapping(object) <- value

## S4 method for signature 'MultiBatchCopyNumber'
mapping(object)

## S4 method for signature 'MultiBatchCopyNumberPooled'
mapping(object)

## S4 replacement method for signature 'MultiBatchCopyNumber,character'
mapping(object) <- value

## S4 replacement method for signature 'MultiBatchCopyNumberPooled,character'
mapping(object) <- value
```

Arguments

object	a SB, SBP, MB, or MBP model
value	a k-length numeric vector with values in 1, 2, ..., k, where k is the number of mixture components

See Also

[CopyNumberModel](#)

Examples

```

cn.model <- CopyNumberModel(SingleBatchModelExample)
## manually remap first two components to the same copy number state
## Not run:
mapping(cn.model) <- c(1, 1, 2)
ggMixture(cn.model)

## End(Not run)

```

map_z	<i>Calculate the maximum a posteriori estimate of latent variable assignment.</i>
-------	---

Description

Calculate the maximum a posteriori estimate of latent variable assignment.

Usage

```
map_z(object)
```

Arguments

object an object of class MixtureModel.

Value

map estimate of latent variable assignment for each observation

Examples

```
head(map_z(SingleBatchModelExample))
```

marginalLik	<i>Extract marginal likelihoods from a list of models</i>
-------------	---

Description

Extract marginal likelihoods from a list of models

Usage

```
marginalLik(models)
```

Arguments

models list of models

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

Description

The recommended function for fitting mixture models and evaluating convergence is through the ‘gibbs’ function. This function will return a list of models ordered by the marginal likelihood. The marginal likelihood is computed using the Chib’s estimator (JASA, Volume 90 (435), 1995).

Usage

```
marginalLikelihood(model, params = mlParams())

## S4 method for signature 'MultiBatchModel'
marginalLikelihood(model,
  params = mlParams())

## S4 method for signature 'MultiBatchPooled'
marginalLikelihood(model,
  params = mlParams())

## S4 method for signature 'list'
marginalLikelihood(model, params = mlParams(warnings =
  FALSE))
```

Arguments

model	An object of class MarginalModel, or a list of MarginalModel’s. Can also be an object of BatchModel or a list of such models.
params	A list containing parameters for marginalLikelihood computation. See mlParams for details.

Value

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

See Also

See [mlParams](#) for parameters related to computing the log marginal likelihood via Chib’s estimator. See [gibbs](#) for fitting multiple mixture models and returning a list sorted by the marginal likelihood. See [marginal_lik](#) for the accessor.

Note: currently thinning of the reduced MCMC chains is not allowed.

Examples

```
## In practice, run a much longer burnin and increase the number of
## iterations to save after burnin
mm <- SingleBatchModelExample
mcmcParams(mm) <- McmcParams(iter=50, burnin=0, nStarts=0)
mm <- posteriorSimulation(mm)
marginalLikelihood(mm)
```

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

marginal_lik *Accessor for the log marginal likelihood of a SB, SBP, MB, or MBP model*

Description

The marginal likelihood is computed by Chib's estimator (JASA, Volume 90 (435), 1995).

Usage

```
marginal_lik(object)

marginal_lik(object) <- value

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

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

Arguments

object	a SB, SBP, MB, or MBP model
value	scalar for marginal likelihood

See Also

See [marginalLikelihood](#) for computing the marginal likelihood of a mixture model.

Examples

```
sb <- SingleBatchModelExample
marginal_lik(sb)
```

MBP

*Constructor for MultiBatchPooled model***Description**

Initializes a MultiBatchPooled model, a container for storing data, parameters, and MCMC output for mixture models with batch- and component-specific means. The variance is assumed to be the same for all components, but allowed to differ by batch.

Usage

```
MBP(dat = numeric(), hp = HyperparametersMultiBatch(),
    mp = McmcParams(iter = 1000, burnin = 1000, thin = 10, nStarts = 4),
    batches = integer())
```

Arguments

dat	the data for the simulation.
hp	An object of class 'Hyperparameters' used to specify the hyperparameters of the model.
mp	An object of class 'McmcParams'
batches	an integer-vector of the different batches

Value

An object of class 'MultiBatchPooled'

Examples

```
model <- MBP(rnorm(10), batch=rep(1L, 10))
set.seed(100)
nbatch <- 3
k <- 3
means <- matrix(c(-2.1, -2, -1.95, -0.41, -0.4, -0.395, -0.1,
  0, 0.05), nbatch, k, byrow = FALSE)
sds <- matrix(0.15, nbatch, k)
sds[, 1] <- 0.3
N <- 1000
truth <- simulateBatchData(N = 2500,
  batch = rep(letters[1:3],
    length.out = 2500),
  theta = means, sds = sds,
  p = c(1/5, 1/3, 1 - 1/3 - 1/5))
MBP(dat=y(truth), batches=batch(truth),
  hp=hpList(k=3)[["MB"]])
```

McmcChains-class	<i>An object to hold estimated parameters.</i>
------------------	--

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.
 predictive posterior predictive distribution
 zstar needed for plotting posterior predictive distribution
 k integer specifying number of components
 iter integer specifying number of MCMC simulations
 B integer specifying number of batches

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 = 1L, nStarts = 1L,
  param_updates = .param_updates(), min_GR = 1.2,
  min_effsize = round(1/3 * iter, 0), max_burnin = 32000,
  min_chains = 1)
```

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) <- value

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

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

## S4 replacement method for signature 'list,ANY'
mcmcParams(object) <- value

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

## S4 replacement method for signature 'TrioBatchModel,ANY'
mcmcParams(object) <- value
```

Arguments

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

Value

An object of class 'McmcParams'

Examples

```
mcmcParams(SingleBatchModelExample)
```

<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).

`min_GR` minimum value of multivariate Gelman Rubin statistic for diagnosing convergence. Default is 1.2.

`min_effsize` the minimum mean effective size of the chains. Default is $1/3 * iter$.

`max_burnin` The maximum number of burnin iterations before we give up and return the existing model.

`min_chains` minimum number of independence MCMC chains used for assessing convergence. Default is 3.

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

`u` chi-square draws for controlling t-distribution

`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` an integer-vector numbering the different batches. Must the same length as `data`.

`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'

`label_switch` length-one logical indicating problems with label switching

`marginal_lik` the marginal likelihood of the model

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

`.internal.counter` For internal use only.

`marginal_lik` scalar for marginal likelihood

mlParams

*Parameters for evaluating marginal likelihood***Description**

Parameters for evaluating marginal likelihood

Usage

```
mlParams(root = 1/10, reject.threshold = exp(-10),
  prop.threshold = 0.5, prop.effective.size = 0.05,
  ignore.effective.size = FALSE, ignore.small.pstar = FALSE,
  warnings = TRUE)
```

Arguments

root	length-one numeric vector. We exponentiate $p(\theta^* \dots)$ by the value of root. Values less than one reduce the influence of extreme observations.
reject.threshold	length-one numeric vector between 0 and 1. Probabilities in the reduced Gibbs model for the thetas that are less than this threshold are flagged.
prop.threshold	length-one numeric vector between 0 and 1. If more than prop.threshold are flagged, the marginal likelihood is not evaluated.
prop.effective.size	Logical. If the effective size / total iterations is less than prop.effective.size, the marginal likelihood is not evaluated (unless ignore.effective.size is TRUE).
ignore.effective.size	Logical. By default, if the effective size of any theta chain is less than 0.02, the marginal likelihood is not calculated. If this parameter is set to TRUE, the effective size is ignored. Occasionally, the effective size is misleading. See details.
ignore.small.pstar	Logical. Flags from the reject.threshold parameter are ignored and the marginal likelihood is calculated.
warnings	Logical. If FALSE, warnings are not issued. This is FALSE by default for the marginalLikelihood-list method, and TRUE otherwise.

Details

For mixture models, a low effective size of one or more theta chains can occur for the following reasons:

- A. the model has not yet converged
- B. the model is overfit and there is lots of mixing (label swapping)between some of the chains
- C. the model is not overfit but there is a lot of mixing of the thetas

For both (A) and (B) it is desirable to return NAs. While (C) can also occur, it can be easily diagnosed by visual inspection of the chains. To the extent that (C) occurs, the correction factor may not be needed.

Value

a list of parameters to be passed to `marginalLikelihood`.

See Also

[effectiveSize](#) [marginalLikelihood](#)

Examples

```
m1Params()
```

```
modelName, MixtureModel-method  
    Abbreviated model name
```

Description

Abbreviated model name

Usage

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

Arguments

`object` a `SingleBatchModel`, `MultiBatchModel`, etc.

Examples

```
modelName(SingleBatchModelExample)
```

```
modes  
    Retrieve the modes from a model.
```

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 '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(SingleBatchModelExample)

```

mu

Retrieve overall mean

Description

Retrieve overall mean

Usage

```

mu(object)

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

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

```

Arguments

object see showMethods(mu)

Value

A vector containing 'mu'

Examples

```

mu(SingleBatchModelExample)

```

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

```
head(muc(SingleBatchModelExample))
```

MultiBatch-class	<i>An object for running MCMC simulations.</i>
------------------	--

Description

BatchModel and MarginalModel both inherit from this class.

Slots

data a tibble with one-dimensional summaries (oned), id, and batch

parameters list of parameters

chains object of class McmcChains

current_values current value of each chain

summaries list of empirical data and model summaries

flags list of model flags

down_sample integer vector for downsampling the full data

MultiBatchModel-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

`label_switch` length-one logical vector indicating whether label-switching occurs (possibly an overfit model)

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

`is_mendelian` integer vector equal in length to the number of trios

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

MultiBatchModel2

*Constructor for MultiBatchModel***Description**

Initializes a MultiBatchModel, a container for storing data, parameters, and MCMC output for mixture models with batch- and component-specific means and variances.

Initializes a MultiBatchModel, a container for storing data, parameters, and MCMC output for mixture models with batch- and component-specific means and variances.

Usage

```
MultiBatchModel2(dat = numeric(), hp = HyperparametersMultiBatch(),
  mp = McmcParams(iter = 1000, thin = 10, burnin = 1000, nStarts = 4),
  batches = integer())
```

```
MB(dat = numeric(), hp = HyperparametersMultiBatch(),
  mp = McmcParams(iter = 1000, thin = 10, burnin = 1000, nStarts = 4),
  batches = integer())
```

Arguments

dat	the data for the simulation.
hp	An object of class 'Hyperparameters' used to specify the hyperparameters of the model.
mp	An object of class 'McmcParams'
batches	an integer-vector of the different batches
dat	the data for the simulation.
batches	an integer-vector of the different batches
hp	An object of class 'Hyperparameters' used to specify the hyperparameters of the model.
mp	An object of class 'McmcParams'

Value

An object of class 'MultiBatchModel'

An object of class 'MultiBatchModel'

Examples

```
model <- MultiBatchModel2(rnorm(10), batch=rep(1:2, each=5))
set.seed(100)
nbatch <- 3
k <- 3
means <- matrix(c(-2.1, -2, -1.95, -0.41, -0.4, -0.395, -0.1,
  0, 0.05), nbatch, k, byrow = FALSE)
sds <- matrix(0.15, nbatch, k)
sds[, 1] <- 0.3
N <- 1000
```

```

truth <- simulateBatchData(N = N, batch = rep(letters[1:3],
                                             length.out = N),
                          p = c(1/10, 1/5, 1 - 0.1 - 0.2),
                          theta = means,
                          sds = sds)

truth <- simulateBatchData(N = 2500,
                          batch = rep(letters[1:3], length.out = 2500),
                          theta = means, sds = sds,
                          p = c(1/5, 1/3, 1 - 1/3 - 1/5))
MultiBatchModel2(dat=y(truth), batches=batch(truth),
                 hp=hpList(k=3)[["MB"]])
model <- MultiBatchModel2(rnorm(10), batch=rep(1:2, each=5))
set.seed(100)
nbatch <- 3
k <- 3
means <- matrix(c(-2.1, -2, -1.95, -0.41, -0.4, -0.395, -0.1,
                  0, 0.05), nbatch, k, byrow = FALSE)
sds <- matrix(0.15, nbatch, k)
sds[, 1] <- 0.3
N <- 1000
truth <- simulateBatchData(N = N, batch = rep(letters[1:3],
                                             length.out = N),
                          p = c(1/10, 1/5, 1 - 0.1 - 0.2),
                          theta = means,
                          sds = sds)
MB(dat=y(truth), batches=batch(truth),
   hp=hpList(k=3)[["MB"]])

```

MultiBatchModelExample

This data is an instance of MultiBatchModel

Description

This data is an instance of MultiBatchModel

Usage

```
MultiBatchModelExample
```

Value

An example of a ‘MultiBatchModel’ MultiBatchModelExample

Author(s)

Jacob Carey

MultiBatchPooledExample

This data is an instance of MultiBatchPooled

Description

This data is an instance of MultiBatchPooled

Usage

MultiBatchPooledExample

Value

An example of a 'MultiBatchPooled' MultiBatchPooledExample

Author(s)

Jacob Carey

muMean

Retrieve overall mean averaged across MCMC simulations.

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(SingleBatchModelExample)
```

 names,McmcChains-method

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

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

Number of MCMC chains.

Description

This function retrieves 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,ANY'
nStarts(object) <- value
```

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

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

Arguments

object see showMethods(nStarts)

value new number of chains

Details

Simulating starting values from the priors makes it imperative to run a large number of simulations for burnin and to carefully evaluate the chains following burning for convergence. The adequacy of the burnin is difficult to assess in high-dimensional settings with a large number of CNPs. To avoid starting in regions of low posterior probability, we use existing EM-based methods in the package {mclust} to select starting values from N bootstrap samples of the observed data, where N is specified as in the example below. For each bootstrap sample, starting values for the model are estimated. For each set of simulated starting values, the log likelihood of the full data is evaluated. The starting values with the largest log likelihood are used as initial values for the MCMC simulations.

Value

An integer of the number of different starts.

Examples

```
number_of_chains <- nStarts(SingleBatchModelExample)
number_of_chains <- 10
nStarts(SingleBatchModelExample) <- number_of_chains
```

nu.0

Retrieve the shape parameter for the sigma.2 distribution.

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(SingleBatchModelExample)
```

numberObs	<i>Number of observations</i>
-----------	-------------------------------

Description

Number of observations

Usage

```
numberObs(model)
```

```
## S4 method for signature 'MixtureModel'  
numberObs(model)
```

Arguments

model a MixtureModel-derived object

Examples

```
numberObs(SingleBatchModelExample)
```

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

orderModels	<i>Order models by Bayes factor</i>
-------------	-------------------------------------

Description

Order models by Bayes factor

Usage

```
orderModels(models, bf.thr = 10)
```

Arguments

models	list of MixtureModel-derived objects
bf.thr	scalar: minimal bayes factor for selecting a model with more parameters over a more parsimonious model

p,MixtureModel-method *Retrieve mixture proportions.*

Description

Retrieve mixture proportions.

Usage

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

Arguments

object	an object of class MarginalModel or BatchModel
--------	--

Value

A vector of length the number of components

Examples

```
p(SingleBatchModelExample)
```

pic	<i>Retrieve mixture proportions at each iteration of the MCMC.</i>
-----	--

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

```
head(pic(MultiBatchModelExample))
```

posteriorPredictive	<i>Simulate data from the posterior predictive distribution</i>
---------------------	---

Description

Simulating from the posterior predictive distribution can be helpful for assessing the adequacy of the mixture model.

Usage

```
posteriorPredictive(model)
```

Arguments

model a `SingleBatchModel` or `MultiBatchModel`

Examples

```
## Not run:
model <- SingleBatchModelExample
mp <- McmcParams(iter=200, burnin=50)
mcmcParams(model) <- mp
model <- posteriorSimulation(model)
pd <- posteriorPredictive(model)
if(FALSE) qqplot(pd, y(model))

bmodel <- MultiBatchModelExample
```

```

mp <- McmcParams(iter=500, burnin=150, nStarts=20)
mcmcParams(bmodel) <- mp
bmodel <- posteriorSimulation(bmodel)
batchy <- posteriorPredictive(bmodel)

## End(Not run)

```

posteriorSimulation *Run 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'
posteriorSimulation(object)

## S4 method for signature 'TrioBatchModel'
posteriorSimulation(object)

```

Arguments

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

Value

An object of class 'MarginalModel' or 'BatchModel'

See Also

[ggChains](#) for diagnosing convergence. See [ggMixture](#) for plotting the model-based densities.

Examples

```

# Fit model with pre-specified number of components (k=3)
set.seed(123)
## specify small number of iterations so that the example runs quickly
mp <- McmcParams(iter=2, burnin=0, nStarts=3)
sb <- SingleBatchModelExample
mcmcParams(sb) <- mp
posteriorSimulation(sb)

# Run additional iterations, but set nStart = 0 so that the last value of the

```

pp *Retrieve mixture proportions of parents.*

Description

Retrieve mixture proportions of parents.

Usage

```
pp(object)
```

Arguments

object an object of class TrioBatchModel

Value

A vector of length the number of components

Examples

```
## Not run:
  pp(TrioBatchModelExample)

## End(Not run)
```

probCopyNumber *Posterior probabilities for copy number states*

Description

In contrast to posterior probabilities for mixture components, this function returns posterior probabilities for distinct copy number states. a SingleBatchCopyNumber or MultiBatchCopyNumber instance

Usage

```
probCopyNumber(model)

## S4 method for signature 'MultiBatchCopyNumber'
probCopyNumber(model)

## S4 method for signature 'MultiBatchCopyNumberPooled'
probCopyNumber(model)
```

Arguments

model a SB, SBP, MB, or MBP model

See Also

[CopyNumberModel](#)

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

```
head(probz(SingleBatchModelExample))
```

probzpar	<i>Retrieve the probability of latent variable membership by observation for parents.</i>
----------	---

Description

Retrieve the probability of latent variable membership by observation for parents.

Usage

```
probzpar(object)

## S4 method for signature 'TrioBatchModel'
probzpar(object)
```

Arguments

object see showMethods(probzpar)

Value

A matrix of size number of observations x number of components

Examples

```
## Not run:
  probzpar(TrioBatchModelExample)

## End(Not run)
```

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	<i>Save se data</i>
-----------	---------------------

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 x K, where K is the number of components and B is the number of batches

Examples

```
sigma(SingleBatchModelExample)
```

sigma2.0	<i>Retrieve the rate parameter for the sigma.2 distribution.</i>
----------	--

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(SingleBatchModelExample)
```

sigma<-	<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

```
sigma(object) <- value

sigma2(object)

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

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

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



```

sigma(object)

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

## S4 replacement method for signature 'MultiBatchPooled,ANY'
sigma2(object) <- value

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

## S4 method for signature 'MultiBatchCopyNumberPooled'
sigma(object)

## S4 replacement method for signature 'MultiBatchCopyNumberPooled,ANY'
sigma2(object) <- value

## S4 replacement method for signature 'MultiBatchCopyNumberPooled'
sigma(object) <- value

## S4 method for signature 'MultiBatchCopyNumberPooled'
sigma_(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(SingleBatchModelExample)
```

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 $N \times K$ where N is the number of observations and K is the number of components

Examples

```
sigmac(SingleBatchModelExample)
```

```
simulateBatchData      Create simulated batch data for testing.
```

Description

Create simulated batch data for testing.

Usage

```
simulateBatchData(N = 2500, p, theta, sds, batch, zz, df = 10)
```

Arguments

<code>N</code>	number of observations
<code>p</code>	a vector indicating probability of membership to each component
<code>theta</code>	a matrix of means. Columns are components and rows are batches.
<code>sds</code>	a matrix of standard deviations. Columns are components and rows are batches.
<code>batch</code>	a vector of labels indication from which batch each simulation should come from
<code>zz</code>	a vector indicating latent variable membership. Can be omitted.
<code>df</code>	length-1 numeric vector for the t-distribution degrees of freedom

Value

An object of class 'MultiBatchModel'

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

Description

Create simulated data for testing.

Usage

```
simulateData(N, p, theta, sds, df = 10)
```

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
df	length-1 numeric vector for the t-distribution degrees of freedom

Value

An object of class 'SingleBatchModel'

Examples

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

SingleBatchCopyNumber-class

Mixture model container where mixture components have been genotyped

Description

The components in a mixture model need not correspond to distinct copy number states. For example, when batch or copy number does not explain skewness or heavy-tails.

Details

Suppose a mixture model with four components is selected, where the 3rd and 4th components both correspond to the diploid state. The mapping slot will be the vector "0", "1", "2", and "2".

Slots

mapping character string vector indicating the copy number states. Typically '0', '1', '2', '3', or '4'.

singleBatchGuided,MultiBatchList,MultiBatch-method

Use values from SingleBatch model to simulate reasonable starting values for the SingleBatch-pooled and MultiBatch models of the same number of components

Description

Use values from SingleBatch model to simulate reasonable starting values for the SingleBatch-pooled and MultiBatch models of the same number of components

Usage

```
## S4 method for signature 'MultiBatchList,MultiBatch'
singleBatchGuided(x, guide)
```

SingleBatchModel-class

The 'SingleBatchModel' 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
u latent chi square variable for t-distribution
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: $\log(p(\sigma^2_0)p(\nu_0)p(\mu))$

`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'
`label_switch` length-one logical vector indicating whether label-switching occurs (possibly an overfit model)
`.internal.constraint` Constraint on parameters. For internal use only.

SingleBatchModel2 *Constructors for SB and SBP models*

Description

This creates a MultiBatchModel object with a single batch.

Usage

```

SingleBatchModel2(dat = numeric(), hp = Hyperparameters(),
  mp = McmcParams(iter = 1000, burnin = 1000, thin = 10, nStarts = 4))

SB(dat = numeric(), hp = Hyperparameters(), mp = McmcParams(iter =
  1000, burnin = 1000, thin = 10, nStarts = 4))

SingleBatchPooled(dat = numeric(), hp = Hyperparameters(),
  mp = McmcParams(iter = 1000, burnin = 1000, thin = 10, nStarts = 4))

SBP(dat = numeric(), hp = Hyperparameters(), mp = McmcParams(iter =
  1000, burnin = 1000, thin = 10, nStarts = 4))
  
```

Arguments

<code>dat</code>	numeric vector of average log R ratios
<code>hp</code>	an object of class Hyperparameters
<code>mp</code>	an object of class McmcParams

Value

An instance of MultiBatchModel
 An instance of MultiBatchModel

See Also

[MultiBatchModel2](#)
[MultiBatchModel2](#)

Examples

```

SB()
SB(dat=rnorm(100), hpList(k=2)[["SB"]])
## pooled variance model
SingleBatchPooled()
## or, equivalently
SBP()

```

SingleBatchModelExample

This data is an instance of SingleBatchModel

Description

This data is an instance of SingleBatchModel

Usage

```
SingleBatchModelExample
```

Value

An example of a ‘SingleBatchModel’ SingleBatchModelExample

Author(s)

Jacob Carey

tau

Retrieve overall standard deviation.

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(SingleBatchModelExample)
```

`tau2`*Accessor for the tau2 parameter in the hierarchical mixture model*

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 'McmcChains'
tau2(object)

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

Arguments

`object` see `showMethods(tau2)`

Value

A vector of variances

See Also

Hyperparameters

Examples

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

tauc	<i>Retrieve overall standard deviation at each iteration of the MCMC.</i>
------	---

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(SingleBatchModelExample)
```

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(SingleBatchModelExample)
```

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 $B \times K$, 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)

theta(object) <- value

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

## S4 method for signature 'MixtureModel'
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
## Not run:
k(SingleBatchModelExample)
theta(SingleBatchModelExample)
plot.ts(theta(chains(SingleBatchModelExample)))
## BatchModel
k(MultiBatchModelExample)
length(unique(batch(MultiBatchModelExample)))
theta(MultiBatchModelExample)
## Plot means for batches in one component
plot.ts(theta(chains(MultiBatchModelExample))[, 1:3])

## End(Not run)
```

thin	<i>Get or set the number of thinning intervals.</i>
------	---

Description

This function gets or sets the number of thinning intervals used for an MCMC simulation.

Usage

```
thin(object)

thin(object) <- value

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

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

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

Arguments

object	see showMethods(thin)
value	a length-one numeric vector indicating how often to save MCMC iterations to the chain. For example, a thin of 10 means that every 10th MCMC simulation is saved to the chain.

Value

An integer of the number of thinning intervals

Examples

```
thin(SingleBatchModelExample)
thin(SingleBatchModelExample) <- 10L
```

tileMedians	<i>Create tile labels for each observation</i>
-------------	--

Description

For large datasets (several thousand subjects), the computational burden for fitting Bayesian mixture models can be high. Downsampling can reduce the computational burden with little effect on inference. The function tileMedians is useful for putting the median log R ratios for each subject in a bucket. The observations in each bucket are averaged. This is done independently for each batch and the range of median log R ratios within each bucket is guaranteed to be less than 0.05. Note this function requires specification of a batch variable. If the study was small enough such

that all the samples were processed in a single batch, then downsampling would not be needed. By summarizing the observations in each bucket by batch, the SingleBatchModels (SB or SBP) and MultiBatchModels (MB or MBP) will be fit to the same data and are still comparable by marginal likelihoods or Bayes Factors.

Usage

```
tileMedians(y, nt, batch)

tileSummaries(tiles)
```

Arguments

y	vector containing data
nt	the number of observations per batch
batch	a vector containing the labels from which batch each observation came from.
tiles	a tibble as constructed by <code>tileMedians</code>

Value

A tibble with a tile assigned to each log R ratio

See Also

[ntile](#)

TrioBatchModel	<i>Constructor for TrioBatchModel</i>
----------------	---------------------------------------

Description

Initializes a TrioBatchModel, a container for storing data, parameters, and MCMC output for mixture models with batch- and component-specific means and variances.

Usage

```
TrioBatchModel(triodata = tibble(), hp = HyperparametersTrios(),
  mp = McmcParams(iter = 1000, thin = 10, burnin = 1000, nStarts = 4),
  mprob = mprob, maplabel = maplabel)
```

Arguments

triodata	the data for the simulation.
hp	An object of class 'Hyperparameters' used to specify the hyperparameters of the model.
mp	An object of class 'McmcParams'
batches	an integer-vector of the different batches

Value

An object of class 'TrioBatchModel'

triodata_lrr	<i>Retrieves intensity data from trios</i>
--------------	--

Description

Retrieves intensity data from trios

Usage

```
triodata_lrr(object)
```

Arguments

object see showMethods(triodata_lrr)

Value

An integer vector of length the number of components

Examples

```
## Not run:
  triodata_lrr(TrioBatchModelExample)

## End(Not run)
```

upSample2	<i>Restore model of down-sampled to original dimension.</i>
-----------	---

Description

Restore model of down-sampled to original dimension.

Usage

```
upSample2(orig.data, model, up_sample = TRUE)
```

Arguments

orig.data a data.frame containing the original data (not downsampled) with the batch labels stored with colname batch_orig and the median-summarized data stored in column medians

model model fit to the down-sampled data

up_sample logical. If TRUE, model is restored to the original dimension.

Examples

```

library(tidyverse)
library(dplyr)
set.seed(123)
k <- 3
nbatch <- 3
means <- matrix(c(-1.2, -1, -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))

##
## Make a tibble: required plate, plate.index, batch_orig
##
full.data <- tibble(medians=y(truth),
  plate=batch(truth),
  batch_orig=as.character(batch(truth))) %>%
  mutate(plate.index=as.integer(factor(plate, levels=unique(plate))))

## Below, we down-sample to 500 observations
## Required: batch_orig, batch_index
partial.data <- downSample(full.data, size=500)

##
## Required: a mapping from plate to batch
##
select <- dplyr::select
summarize <- dplyr::summarize
plate.mapping <- partial.data %>%
  select(c(plate, batch_index)) %>%
  group_by(plate) %>%
  summarize(batch_index=unique(batch_index))

## Fit a model as per usual to the down-sampled data
mp <- McmcParams(iter=200, burnin=10)
hp <- HyperparametersMultiBatch(k=3)
model <- MultiBatchModel2(dat=partial.data$medians,
  batches=partial.data$batch_index,
  mp=mp,
  hp=hp)
model <- posteriorSimulation(model)
##
## Add the batching used for the down-sampled data to the full data
##
full.data2 <- left_join(full.data, plate.mapping, by="plate")
##
## Estimate probabilities for each individual in the full data
##
model.full <- upSample2(full.data2, model)

```

y *Retrieve data.*

Description

Retrieve data.

Usage

```
y(object)
```

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

Arguments

object see showMethods(y)

Value

A vector containing the data

Examples

```
head(y(SingleBatchModelExample))
```

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 'MixtureModel'  
z(object)
```

Arguments

object see showMethods(z)

Value

A vector the length of the data

Examples

```
head(z(SingleBatchModelExample))
```

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(SingleBatchModelExample)
```

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

Description

Calculates a frequency table of latent variable assignments for parents by observation.

Usage

```
zFreqPar(object)

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

Arguments

object see showMethods(zfreqpar)

Value

An integer vector of length the number of components

Examples

```
## Not run:
  zfreqpar(TrioBatchModelExample)

## End(Not run)
```

```
[,McmcChains,ANY,ANY,ANY-method
```

```
  extract estimated parameters at particular iteration of simulation.
```

Description

extract estimated parameters at particular iteration of simulation.

extract estimated parameters at particular iteration of simulation.

extract data, latent variable, and batch for given observation

Usage

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

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

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

Arguments

x	An object of class <code>MultiBatchModel</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 `'McmcChains'`

An object of class `'McmcChainsTrios'`

An object of class `'MultiBatchModel'`

Index

- [,McmcChains,ANY,ANY,ANY-method, 80
- [,McmcChains,ANY-method
 - ([,McmcChains,ANY,ANY,ANY-method), 80
- [,McmcChains-method
 - ([,McmcChains,ANY,ANY,ANY-method), 80
- [,McmcChainsTrios,ANY,ANY,ANY-method
 - ([,McmcChains,ANY,ANY,ANY-method), 80
- [,McmcChainsTrios,ANY-method
 - ([,McmcChains,ANY,ANY,ANY-method), 80
- [,McmcChainsTrios-method
 - ([,McmcChains,ANY,ANY,ANY-method), 80
- [,MultiBatchModel,ANY,ANY,ANY-method
 - ([,McmcChains,ANY,ANY,ANY-method), 80
- [,MultiBatchModel,ANY,ANY-method
 - ([,McmcChains,ANY,ANY,ANY-method), 80
- [,MultiBatchModel,ANY-method
 - ([,McmcChains,ANY,ANY,ANY-method), 80
- [,MultiBatchModel-method
 - ([,McmcChains,ANY,ANY,ANY-method), 80

- bafLikelihood, 4
- batch, 5
- batch,MixtureModel-method (batch), 5
- BatchModel-class (DensityModel-class), 14
- BatchModelExample, 5
- bayesFactor, 6
- bic, 6
- bic,MultiBatchModel-method (bic), 6
- burnin, 7
- burnin,McmcParams-method (burnin), 7
- burnin,MixtureModel-method (burnin), 7
- burnin<- (burnin), 7
- burnin<-,McmcParams,ANY-method (burnin), 7

- burnin<-,McmcParams-method (burnin), 7
- burnin<-,MixtureModel,ANY-method (burnin), 7
- burnin<-,MixtureModel-method (burnin), 7

- chains, 7
- chains,MixtureModel-method (chains), 7
- chromosome, 8
- chromosome,GenomicRanges-method (chromosome), 8
- CNPBayes, 9
- CNPBayes-package (CNPBayes), 9
- collapseBatch, 9
- collapseBatch,MultiBatchModel-method (collapseBatch), 9
- collapseBatch,numeric-method (collapseBatch), 9
- collapseBatch,SummarizedExperiment-method (collapseBatch), 9
- consensusCNP, 10
- copyNumber, 12, 13
- copyNumber,MultiBatchCopyNumber-method (copyNumber), 12
- copyNumber,MultiBatchCopyNumberPooled-method (copyNumber), 12
- CopyNumberModel, 12, 13, 36, 60
- CopyNumberModel,MultiBatchModel-method (CopyNumberModel), 13
- CopyNumberModel,MultiBatchPooled-method (CopyNumberModel), 13
- CopyNumberModel,SingleBatchModel-method (CopyNumberModel), 13

- DensityBatchModel-class (DensityModel-class), 14
- DensityModel-class, 14
- dfr, 17
- dfr,Hyperparameters-method (dfr), 17
- dfr,MixtureModel,numeric-method (dfr), 17
- dfr,MixtureModel-method (dfr), 17
- dfr<- (dfr), 17
- dfr<-,Hyperparameters,numeric-method (dfr), 17

- dfr<- ,MixtureModel,numeric-method (dfr), 17
- downSample, 17
- effectiveSize, 22, 46
- eta.0, 18
- eta.0,Hyperparameters-method (eta.0), 18
- eta.0,MixtureModel-method (eta.0), 18
- extract
 - ([,McmcChains,ANY,ANY,ANY-method), 80
- findSurrogates,MultiBatch-method, 19
- gelman.diag, 22
- ggChains, 20, 22, 58
- ggChains,MultiBatchModel-method (ggChains), 20
- ggChains,MultiBatchPooled-method (ggChains), 20
- ggMixture, 58
- ggMixture (ggChains), 20
- ggMixture,MultiBatch-method (ggChains), 20
- ggMixture,MultiBatchCopyNumber-method (ggChains), 20
- ggMixture,MultiBatchCopyNumberPooled-method (ggChains), 20
- ggMixture,MultiBatchModel-method (ggChains), 20
- ggMixture,MultiBatchPooled-method (ggChains), 20
- gibbs, 21, 21, 38
- hpList, 26
- hpList (Hyperparameters), 23
- Hyperparameters, 23, 23
- Hyperparameters-class, 24
- HyperparametersBatch-class, 24
- HyperparametersMarginal-class, 25
- HyperparametersMultiBatch, 23, 25
- HyperparametersMultiBatch-class, 26
- HyperparametersSingleBatch, 27
- HyperparametersSingleBatch-class, 28
- HyperparametersTrios, 23, 28
- HyperparametersTrios-class, 29
- hyperParams, 29
- hyperParams,MixtureModel-method (hyperParams), 29
- hyperParams<- (hyperParams), 29
- hyperParams<- ,MixtureModel,Hyperparameters-method (hyperParams), 29
- hyperParams<- ,MixtureModel-method (hyperParams), 29
- iter (iter<-), 30
- iter,burnin,nStarts,McmcParams-method (McmcParams-class), 43
- iter,McmcParams-method (iter<-), 30
- iter,MixtureModel-method (iter<-), 30
- iter<- , 30
- iter<- ,McmcParams,ANY-method (iter<-), 30
- iter<- ,McmcParams-method (iter<-), 30
- iter<- ,MixtureModel,ANY-method (iter<-), 30
- iter<- ,MixtureModel-method (iter<-), 30
- k, 31
- k,Hyperparameters-method (Hyperparameters-class), 24
- k,MixtureModel-method (k), 31
- k,TrioBatchModel-method (k), 31
- k<- (k), 31
- k<- ,Hyperparameters-method (k), 31
- k<- ,HyperparametersTrios-method (k), 31
- k<- ,MixtureModel-method (k), 31
- k<- ,TrioBatchModel-method (k), 31
- label_switch, 32
- label_switch,MixtureModel-method (label_switch), 32
- log_lik, 33
- log_lik,McmcChains-method (log_lik), 33
- log_lik,MixtureModel-method (log_lik), 33
- logBayesFactor, 32
- logPrior, 33
- logPrior,McmcChains-method (logPrior), 33
- logPrior,MixtureModel-method (logPrior), 33
- m2.0, 34
- m2.0,Hyperparameters-method (m2.0), 34
- m2.0,MixtureModel-method (m2.0), 34
- map_z, 37
- mapCnProbability, 35
- mapCopyNumber (CopyNumberModel), 13
- mapParams, 35
- mapping, 36
- mapping,MultiBatchCopyNumber,numeric-method (mapping), 36
- mapping,MultiBatchCopyNumber-method (mapping), 36
- mapping,MultiBatchCopyNumberPooled,numeric-method (mapping), 36

- mapping, MultiBatchCopyNumberPooled-method (mapping), 36
- mapping<- (mapping), 36
- mapping<- , MultiBatchCopyNumber, character-method (mapping), 36
- mapping<- , MultiBatchCopyNumberPooled, character-method (mapping), 36
- marginal_lik, 38, 39
- marginal_lik, MixtureModel-method (marginal_lik), 39
- marginal_lik<- (marginal_lik), 39
- marginal_lik<- , MixtureModel, numeric-method (marginal_lik), 39
- marginalLik, 37
- marginalLikelihood, 22, 38, 39, 46
- marginalLikelihood, list, ANY-method (marginalLikelihood), 38
- marginalLikelihood, list-method (marginalLikelihood), 38
- marginalLikelihood, MultiBatchModel, ANY-method (marginalLikelihood), 38
- marginalLikelihood, MultiBatchModel-method (marginalLikelihood), 38
- marginalLikelihood, MultiBatchPooled-method (marginalLikelihood), 38
- MarginalModel-class (DensityModel-class), 14
- MarginalModelExample, 39
- MB (MultiBatchModel2), 50
- MBP, 40
- McmcChains-class, 41
- McmcParams, 41
- mcmcParams, 42
- mcmcParams, list-method (mcmcParams), 42
- mcmcParams, MixtureModel-method (mcmcParams), 42
- McmcParams-class, 43
- mcmcParams<- (mcmcParams), 42
- mcmcParams<- , list, ANY-method (mcmcParams), 42
- mcmcParams<- , MixtureModel, ANY-method (mcmcParams), 42
- mcmcParams<- , TrioBatchModel, ANY-method (mcmcParams), 42
- MixtureModel-class, 44
- mlParams, 38, 45
- modelName, MixtureModel-method, 46
- modes, 46
- modes, MixtureModel-method (modes), 46
- modes<- (modes), 46
- modes<- , MixtureModel-method (modes), 46
- mu, 47
- mu, McmcChains-method (mu), 47
- mu, MixtureModel-method (mu), 47
- mu, MultiBatchModel-method (mu), 47
- mu, 48
- MultiBatch-class, 48
- MultiBatchCopyNumber (CopyNumberModel), 13
- MultiBatchCopyNumber-class (SingleBatchCopyNumber-class), 67
- MultiBatchCopyNumberPooled (CopyNumberModel), 13
- MultiBatchCopyNumberPooled-class (SingleBatchCopyNumber-class), 67
- MultiBatchModel-class, 49
- MultiBatchModel2, 50, 69
- MultiBatchModelExample, 51
- MultiBatchPooledExample, 52
- muMean, 52
- names, McmcChains-method, 53
- nStarts, 53
- nStarts, McmcParams-method (nStarts), 53
- nStarts, MixtureModel-method (nStarts), 53
- nStarts<- (nStarts), 53
- nStarts<- , McmcParams, ANY-method (nStarts), 53
- nStarts<- , McmcParams-method (nStarts), 53
- nStarts<- , MixtureModel, ANY-method (nStarts), 53
- nStarts<- , MixtureModel-method (nStarts), 53
- ntile, 75
- nu.0, 54
- nu.0, McmcChains-method (nu.0), 54
- nu.0, MixtureModel-method (nu.0), 54
- numberObs, 55
- numberObs, MixtureModel-method (numberObs), 55
- oned, 55
- oned, MixtureModel-method (oned), 55
- orderModels, 56
- p, MixtureModel-method, 56
- pic, 57
- posterior_cases, 59
- posteriorPredictive, 57
- posteriorSimulation, 58

- posteriorSimulation,MixtureModel-method (posteriorSimulation), 58
- posteriorSimulation,TrioBatchModel-method (posteriorSimulation), 58
- pp, 60
- probCopyNumber, 60
- probCopyNumber,MultiBatchCopyNumber-method (probCopyNumber), 60
- probCopyNumber,MultiBatchCopyNumberPooled-method (probCopyNumber), 60
- prozb, 61
- prozb,MixtureModel-method (prozb), 61
- prozbpar, 61
- prozbpar,TrioBatchModel-method (prozbpar), 61

- qInverseTau2, 62

- saveBatch, 63
- SB (SingleBatchModel2), 69
- SBP (SingleBatchModel2), 69
- sigma, 63
- sigma,MixtureModel-method (sigma<-), 64
- sigma,MultiBatchCopyNumberPooled-method (sigma<-), 64
- sigma2 (sigma<-), 64
- sigma2,McmcChains-method (sigma<-), 64
- sigma2,missing-method (sigma<-), 64
- sigma2,MixtureModel-method (sigma<-), 64
- sigma2,MultiBatchCopyNumberPooled-method (sigma<-), 64
- sigma2,MultiBatchPooled-method (sigma<-), 64
- sigma2.0, 64
- sigma2.0,McmcChains-method (sigma2.0), 64
- sigma2.0,MixtureModel-method (sigma2.0), 64
- sigma2<-,MultiBatchCopyNumberPooled,ANY-method (sigma<-), 64
- sigma2<-,MultiBatchCopyNumberPooled-method (sigma<-), 64
- sigma2<-,MultiBatchPooled,ANY-method (sigma<-), 64
- sigma<-, 64
- sigma<-,MultiBatchCopyNumberPooled-method (sigma<-), 64
- sigma_,MultiBatchCopyNumberPooled-method (sigma<-), 64
- sigmac, 65
- simulateBatchData, 66
- simulateData, 67
- SingleBatchCopyNumber (CopyNumberModel), 13
- SingleBatchCopyNumber-class, 67
- singleBatchGuided,MultiBatchList,MultiBatch-method, 68
- SingleBatchModel-class, 68
- SingleBatchModel2, 69
- SingleBatchModelExample, 70
- SingleBatchPooled (SingleBatchModel2), 69
- tau, 70
- tau2, 71
- tau2,McmcChains-method (tau2), 71
- tau2,MixtureModel-method (tau2), 71
- tauc, 72
- tauMean, 72
- theta, 73
- theta,McmcChains-method (theta), 73
- theta,MixtureModel-method (theta), 73
- theta<- (theta), 73
- thin, 74
- thin,McmcParams-method (thin), 74
- thin,MixtureModel-method (thin), 74
- thin<- (thin), 74
- thin<-,MixtureModel,numeric-method (thin), 74
- tileMedians, 74
- tileSummaries (tileMedians), 74
- TrioBatchModel, 75
- triodata_lrr, 76
- upSample2, 76
- y, 78
- y,MixtureModel-method (y), 78
- z, 78
- z,MixtureModel-method (z), 78
- zFreq, 79
- zFreq,McmcChains-method (zFreq), 79
- zfreq,McmcChains-method (zFreq), 79
- zFreq,MixtureModel-method (zFreq), 79
- zfreq,MixtureModel-method (zFreq), 79
- zFreqPar, 79
- zFreqPar,McmcChains-method (zFreqPar), 79
- zfreqpar,McmcChains-method (zFreqPar), 79