

# Package ‘batchmix’

June 21, 2022

**Type** Package

**Title** Semi-Supervised Bayesian Mixture Models Incorporating Batch Correction

**License** GPL-3

**Version** 1.0.1

**Description** Semi-supervised and unsupervised Bayesian mixture models that simultaneously infer the cluster/class structure and a batch correction. Densities available are the multivariate normal and the multivariate t. The model sampler is implemented in C++. This package is aimed at analysis of low-dimensional data generated across several batches. See Coleman et al. (2022) <[doi:10.1101/2022.01.14.476352](https://doi.org/10.1101/2022.01.14.476352)> for details of the model.

**Imports** Rcpp (>= 1.0.5), tidyr, ggplot2

**LinkingTo** Rcpp, RcppArmadillo, testthat

**Suggests** xml2, knitr, rmarkdown

**SystemRequirements** GNU make

**Encoding** UTF-8

**URL** <https://github.com/stcolema/batchmix>

**BugReports** <https://github.com/stcolema/batchmix/issues>

**RoxygenNote** 7.2.0

**VignetteBuilder** knitr, rmarkdown

**NeedsCompilation** yes

**Author** Stephen Coleman [aut, cre],  
Paul Kirk [aut],  
Chris Wallace [aut]

**Maintainer** Stephen Coleman <[stephen.coleman@mrc-bsu.cam.ac.uk](mailto:stephen.coleman@mrc-bsu.cam.ac.uk)>

**Repository** CRAN

**Date/Publication** 2022-06-21 10:30:02 UTC

**R topics documented:**

batchmix-package . . . . .	3
batchSemiSupervisedMixtureModel . . . . .	4
calcAllocProb . . . . .	7
checkDataGenerationInputs . . . . .	8
checkProposalWindows . . . . .	10
collectAcceptanceRates . . . . .	11
continueChain . . . . .	12
continueChains . . . . .	13
createSimilarityMat . . . . .	15
gammaLogLikelihood . . . . .	16
generateBatchData . . . . .	16
generateBatchDataMVT . . . . .	18
generateBatchDataVaryingRepresentation . . . . .	19
generateGroupIDsInSimulator . . . . .	21
generateInitialLabels . . . . .	22
getLikelihood . . . . .	23
getSampledBatchScale . . . . .	24
getSampledBatchShift . . . . .	25
getSampledClusterMeans . . . . .	26
invGammaLogLikelihood . . . . .	27
invWishartLogLikelihood . . . . .	28
plotAcceptanceRates . . . . .	28
plotLikelihoods . . . . .	29
plotSampledBatchMeans . . . . .	30
plotSampledBatchScales . . . . .	31
plotSampledClusterMeans . . . . .	32
plotSampledParameter . . . . .	33
predictClass . . . . .	34
prepareInitialParameters . . . . .	35
processMCMCChain . . . . .	36
processMCMCChains . . . . .	38
rStickBreakingPrior . . . . .	40
runBatchMix . . . . .	40
runMCMCChains . . . . .	44
sampleMVN . . . . .	46
sampleMVT . . . . .	48
samplePriorLabels . . . . .	50
sampleSemisupervisedMVN . . . . .	51
sampleSemisupervisedMVT . . . . .	53
wishartLogLikelihood . . . . .	55

---

batchmix-package

*Bayesian Mixture Modelling for Joint Model-Based Clustering/Classification and Batch Correction*

---

## Description

Semi-supervised and unsupervised Bayesian mixture models that simultaneously infer the cluster/class structure and a batch correction. Densities available are the multivariate normal and the multivariate t. The model sampler is implemented in C++. This package is aimed at analysis of low-dimensional data generated across several batches. See (Coleman et al. (2022)) [<https://doi.org/10.1101/2022.01.14.476352>] for details of the model.

## Author(s)

Stephen Coleman <stcolema@tcd.ie>, Paul D.W. Kirk, Chris Wallace

## Examples

```
# Data in a matrix format
X <- matrix(c(rnorm(100, 0, 1), rnorm(100, 3, 1)), ncol = 2, byrow = TRUE)

# Initial labelling
labels <- c(rep(1, 10),
  sample(c(1,2), size = 40, replace = TRUE),
  rep(2, 10),
  sample(c(1,2), size = 40, replace = TRUE)
)

# Which labels are observed
fixed <- c(rep(1, 10), rep(0, 40), rep(1, 10), rep(0, 40))

# Batch
batch_vec <- sample(seq(1, 5), replace = TRUE, size = 100)

# Sampling parameters
R <- 1000
thin <- 50

# Classification
samples <- runBatchMix(X,
  R,
  thin,
  batch_vec,
  "MVN",
  initial_labels = labels,
  fixed = fixed,
)

# Clustering
```

```
samples <- runBatchMix(X, R, thin, batch_vec, "MVT")
```

---

```
batchSemiSupervisedMixtureModel
```

```
Batch semisupervised mixture model
```

---

## Description

A Bayesian mixture model with batch effects.

## Usage

```
batchSemiSupervisedMixtureModel(
  X,
  R,
  thin,
  initial_labels,
  fixed,
  batch_vec,
  type,
  K_max = length(unique(initial_labels)),
  alpha = 1,
  mu_proposal_window = 0.5^2,
  cov_proposal_window = 0.002,
  m_proposal_window = 0.3^2,
  S_proposal_window = 0.01,
  t_df_proposal_window = 0.015,
  m_scale = 0.1,
  rho = 11,
  theta = 5,
  initial_class_means = NULL,
  initial_class_covariance = NULL,
  initial_batch_shift = NULL,
  initial_batch_scale = NULL,
  initial_class_df = NULL,
  verbose = TRUE
)
```

## Arguments

X	Data to cluster as a matrix with the items to cluster held in rows.
R	The number of iterations in the sampler.
thin	The factor by which the samples generated are thinned, e.g. if “thin=50” only every 50th sample is kept.
initial_labels	Initial clustering.

fixed	Which items are fixed in their initial label.
batch_vec	Labels identifying which batch each item being clustered is from.
type	Character indicating density type to use. One of 'MVN' (multivariate normal distribution) or 'MVT' (multivariate t distribution).
K_max	The number of components to include (the upper bound on the number of clusters in each sample). Defaults to the number of unique labels in "initial_labels".
alpha	The concentration parameter for the stick-breaking prior and the weights in the model.
mu_proposal_window	The proposal window for the cluster mean proposal kernel. The proposal density is a Gaussian distribution, the window is the variance.
cov_proposal_window	The proposal window for the cluster covariance proposal kernel. The proposal density is a Wishart distribution, this argument is the reciprocal of the degree of freedom.
m_proposal_window	The proposal window for the batch mean proposal kernel. The proposal density is a Gaussian distribution, the window is the variance.
S_proposal_window	The proposal window for the batch standard deviation proposal kernel. The proposal density is a Gamma distribution, this argument is the reciprocal of the rate.
t_df_proposal_window	The proposal window for the degrees of freedom for the multivariate t distribution (not used if type is not 'MVT'). The proposal density is a Gamma distribution, this argument is the reciprocal of the rate.
m_scale	The scale hyperparameter for the batch shift prior distribution. This defines the scale of the batch effect upon the mean and should be in (0, 1].
rho	The shape of the prior distribution for the batch scale.
theta	The scale of the prior distribution for the batch scale.
initial_class_means	A $P \times K$ matrix of initial values for the class means. Defaults to draws from the prior distribution.
initial_class_covariance	A $P \times P \times K$ array of initial values for the class covariance matrices. Defaults to draws from the prior distribution.
initial_batch_shift	A $P \times B$ matrix of initial values for the batch shift effect Defaults to draws from the prior distribution.
initial_batch_scale	A $P \times B$ matrix of initial values for the batch scales Defaults to draws from the prior distribution.
initial_class_df	A $K$ vector of initial values for the class degrees of freedom. Defaults to draws from the prior distribution.
verbose	Logical indicating if warning about proposal windows should be printed.

**Value**

A named list containing the sampled partitions, cluster and batch parameters, model fit measures and some details on the model call.

**Examples**

```
# Data in a matrix format
X <- matrix(c(rnorm(100, 0, 1), rnorm(100, 3, 1)), ncol = 2, byrow = TRUE)

# Initial labelling
labels <- c(
  rep(1, 10),
  sample(c(1, 2), size = 40, replace = TRUE),
  rep(2, 10),
  sample(c(1, 2), size = 40, replace = TRUE)
)

fixed <- c(rep(1, 10), rep(0, 40), rep(1, 10), rep(0, 40))

# Batch
batch_vec <- sample(seq(1, 5), replace = TRUE, size = 100)

# Density choice
type <- "MVN"

# Sampling parameters
R <- 1000
thin <- 50

# MCMC samples and BIC vector
samples <- batchSemiSupervisedMixtureModel(X,
  R,
  thin,
  labels,
  fixed,
  batch_vec,
  type
)

# Given an initial value for the parameters
initial_class_means <- matrix(c(1, 1, 3, 4), nrow = 2)
initial_class_covariance <- array(c(1, 0, 0, 1, 1, 0, 0, 1),
  dim = c(2, 2, 2)
)

# We can use values from a previous chain
initial_batch_shift <- samples$batch_shift[, , R / thin]
initial_batch_scale <- matrix(
  c(1.2, 1.3, 1.7, 1.1, 1.4, 1.3, 1.2, 1.2, 1.1, 2.0),
  nrow = 2
)
)
```

```
samples <- batchSemiSupervisedMixtureModel(X,
  R,
  thin,
  labels,
  fixed,
  batch_vec,
  type,
  initial_class_means = initial_class_means,
  initial_class_covariance = initial_class_covariance,
  initial_batch_shift = initial_batch_shift,
  initial_batch_scale = initial_batch_scale
)
```

---

calcAllocProb	<i>Calculate allocation probabilities</i>
---------------	---

---

### Description

Calculate the empirical allocation probability for each class based on the sampled allocation probabilities.

### Usage

```
calcAllocProb(mcmc_samples, burn = 0, method = "median")
```

### Arguments

mcmc_samples	Output from “batchSemiSupervisedMixtureModel”.
burn	The number of samples to discard.
method	The point estimate to use. “method = ‘mean’” or “method = ‘median’”. “median” is the default.

### Value

An  $N \times K$  matrix of class probabilities.

### Examples

```
# Data in matrix format
X <- matrix(c(rnorm(100, 0, 1), rnorm(100, 3, 1)), ncol = 2, byrow = TRUE)

# Initial labelling
labels <- c(rep(1, 10),
  sample(c(1,2), size = 40, replace = TRUE),
  rep(2, 10),
  sample(c(1,2), size = 40, replace = TRUE))
```

```
)  
  
fixed <- c(rep(1, 10), rep(0, 40), rep(1, 10), rep(0, 40))  
  
# Batch  
batch_vec <- sample(seq(1, 5), replace = TRUE, size = 100)  
  
# Sampling parameters  
R <- 1000  
thin <- 50  
  
# MCMC samples and BIC vector  
samples <- batchSemiSupervisedMixtureModel(X, R, thin, labels, fixed, batch_vec, "MVN")  
  
# Burn in  
burn <- 20  
eff_burn <- burn / thin  
  
# Probability across classes  
probs <- calcAllocProb(samples, burn = burn)
```

---

checkDataGenerationInputs

*Check data generation inputs*

---

### **Description**

Checks that the inputs for the “generateBatchData“ function are correct. For internal use only.

### **Usage**

```
checkDataGenerationInputs(  
  N,  
  P,  
  group_means,  
  group_std_devs,  
  batch_shift,  
  batch_scale,  
  group_weights,  
  batch_weights,  
  type,  
  group_dfs,  
  frac_known,  
  permute_variables,  
  scale_data  
)
```



**Arguments**

N	The number of items (rows) to generate.
P	The number of columns in the generated dataset.
group_means	A vector of the group means for a column.
group_std_devs	A vector of group standard deviations for a column.
batch_shift	A vector of batch means in a column.
batch_scale	A vector of batch standard deviations within a column.
group_weights	A K x B matrix of the expected proportion of N in each group in each batch.
batch_weights	A vector of the expected proportion of N in each batch.
type	A string indicating if data should be generated from multivariate normal ("MVN") or multivariate t ("MVT") densities.
group_dfs	A K-vector of the group specific degrees of freedom.
frac_known	The number of items with known labels.
permute_variables	Logical indicating if group and batch means and standard deviations should be permuted in each column or not.
scale_data	Logical indicating if data should be mean centred and standardised.

**Value**

No return value, called for side effects.

**Examples**

```

N <- 500
P <- 2
K <- 2
B <- 5
mean_dist <- 4
batch_dist <- 0.3
group_means <- seq(1, K) * mean_dist
batch_shift <- rnorm(B, mean = batch_dist, sd = batch_dist)
group_std_devs <- rep(2, K)
batch_scale <- rep(1.2, B)
group_weights <- rep(1 / K, K)
batch_weights <- rep(1 / B, B)
type <- "MVT"
group_dfs <- c(4, 7)
frac_known <- 0.3
permute_variables <- TRUE
scale_data <- FALSE

checkDataGenerationInputs(N,
  P,
  group_means,
  group_std_devs,
  batch_shift,

```

```

    batch_scale,
    group_weights,
    batch_weights,
    type,
    group_dfs,
    frac_known,
    permute_variables,
    scale_data
)

```

---

checkProposalWindows *Check proposal windows*

---

### Description

Checks the proposal windows are acceptable.

### Usage

```

checkProposalWindows(
  mu_proposal_window,
  cov_proposal_window,
  m_proposal_window,
  S_proposal_window,
  t_df_proposal_window,
  verbose = TRUE
)

```

### Arguments

`mu_proposal_window`  
The proposal window for the cluster mean proposal kernel. The proposal density is a Gaussian distribution, the window is the variance.

`cov_proposal_window`  
The proposal window for the cluster covariance proposal kernel. The proposal density is a Wishart distribution, this argument is the reciprocal of the degree of freedom.

`m_proposal_window`  
The proposal window for the batch mean proposal kernel. The proposal density is a Gaussian distribution, the window is the variance.

`S_proposal_window`  
The proposal window for the batch standard deviation proposal kernel. The proposal density is a Gamma distribution, this argument is the reciprocal of the rate.

`t_df_proposal_window`  
The proposal window for the degrees of freedom for the multivariate t distribution (not used if type is not 'MVT'). The proposal density is a Gamma distribution, this argument is the reciprocal of the rate.

verbose            Logical indicating if a warning should be printed if proposal windows are outside their expected scale.

**Value**

No return value, called for side effects

**Examples**

```
checkProposalWindows(0.1, 0.2, 0.3, 0.1, 0.4, 0.3)
```

---

collectAcceptanceRates

*Collect acceptance rate*

---

**Description**

Collects the acceptance rates for each parameter into a data.frame

**Usage**

```
collectAcceptanceRates(samples, type)
```

**Arguments**

samples            The output of either the “batchMixtureModel“ or “batchSemiSupervisedMixtureModel“ function.

type                The type of mixture model used; this changes which parameters the function expects to find.

**Value**

A wide data.frame of all the sampled parameters and the iteration.

**Examples**

```
# Data in a matrix format
X <- matrix(c(rnorm(100, 0, 1), rnorm(100, 3, 1)), ncol = 2, byrow = TRUE)

# Initial labelling
labels <- c(rep(1, 10),
  sample(c(1,2), size = 40, replace = TRUE),
  rep(2, 10),
  sample(c(1,2), size = 40, replace = TRUE)
)

fixed <- c(rep(1, 10), rep(0, 40), rep(1, 10), rep(0, 40))
```

```

# Batch
batch_vec <- sample(seq(1, 5), replace = TRUE, size = 100)

# Sampling parameters
R <- 1000
thin <- 50

# MCMC samples and BIC vector
samples <- batchSemiSupervisedMixtureModel(X, R, thin, labels, fixed, batch_vec, "MVN")

# Acceptance rates
collectAcceptanceRates(samples, "MVN")

```

---

continueChain

*Continue chain*

---

## Description

Continues sampling from a previous position for a given chain.

## Usage

```
continueChain(mcmc_output, X, fixed, batch_vec, R, keep_old_samples = TRUE)
```

## Arguments

mcmc_output	Chain to be continued.
X	Data to cluster as a matrix with the items to cluster held in rows.
fixed	The indicator vector for which labels are observed.
batch_vec	The vector of the batch labels for the data.
R	The number of iterations to run in this continuation (thinning factor is the same as initial chain).
keep_old_samples	Logical indicating if the original samples should be kept or only the new samples returned. Defaults to TRUE.

## Value

A named list containing the sampled partitions, cluster and batch parameters, model fit measures and some details on the model call.

**Examples**

```
# Data in a matrix format
X <- matrix(c(rnorm(100, 0, 1), rnorm(100, 3, 1)), ncol = 2, byrow = TRUE)

# Initial labelling
labels <- c(
  rep(1, 10),
  sample(c(1, 2), size = 40, replace = TRUE),
  rep(2, 10),
  sample(c(1, 2), size = 40, replace = TRUE)
)

fixed <- c(rep(1, 10), rep(0, 40), rep(1, 10), rep(0, 40))

# Batch
batch_vec <- sample(seq(1, 5), replace = TRUE, size = 100)

# Density choice
type <- "MVT"

# Sampling parameters
R <- 1000
thin <- 50

# MCMC samples and BIC vector
mcmc_output <- runBatchMix(
  X,
  R,
  thin,
  batch_vec,
  type,
  initial_labels = labels,
  fixed = fixed
)

# Given an initial value for the parameters
mcmc_output <- continueChain(
  mcmc_output,
  X,
  fixed,
  batch_vec,
  R,
)
```

**Description**

Continues sampling from a list of previous chains.

**Usage**

```
continueChains(mcmc_output, X, fixed, batch_vec, R, keep_old_samples = TRUE)
```

**Arguments**

mcmc_output	Chains to be continued.
X	Data to cluster as a matrix with the items to cluster held in rows.
fixed	The indicator vector for which labels are observed.
batch_vec	The vector of the batch labels for the data.
R	The number of iterations to run in this continuation (thinning factor is the same as initial chain).
keep_old_samples	Logical indicating if the original samples should be kept or only the new samples returned. Defaults to TRUE.

**Value**

A named list containing the sampled partitions, cluster and batch parameters, model fit measures and some details on the model call.

**Examples**

```
# Data in a matrix format
X <- matrix(c(rnorm(100, 0, 1), rnorm(100, 3, 1)), ncol = 2, byrow = TRUE)

# Initial labelling
labels <- c(
  rep(1, 10),
  sample(c(1, 2), size = 40, replace = TRUE),
  rep(2, 10),
  sample(c(1, 2), size = 40, replace = TRUE)
)

fixed <- c(rep(1, 10), rep(0, 40), rep(1, 10), rep(0, 40))

# Batch
batch_vec <- sample(seq(1, 5), replace = TRUE, size = 100)

# Density choice
type <- "MVT"

# Sampling parameters
R <- 1000
thin <- 50
```

```
n_chains <- 4

# MCMC samples
mcmc_output <- runMCMCChains(
  X,
  n_chains,
  R,
  thin,
  batch_vec,
  type,
  initial_labels = labels,
  fixed = fixed
)

# Given an initial value for the parameters
new_output <- continueChains(
  mcmc_output,
  X,
  fixed,
  batch_vec,
  R,
  keep_old_samples = TRUE
)
```

---

createSimilarityMat    *Create Similarity Matrix*

---

### Description

Constructs a similarity matrix of the pairwise coclustering rate.

### Usage

```
createSimilarityMat(allocations)
```

### Arguments

**allocations**    Matrix of sampled partitions. Columns correspond to items/samples being clustered, each row is a sampled partition.//'

### Value

A symmetric  $n \times n$  matrix (for  $n$  rows in cluster record) describing the fraction of iterations for which each pairwise combination of points are assigned the same label.

---

gammaLogLikelihood	<i>Gamma log-likelihood</i>
--------------------	-----------------------------

---

**Description**

Used in calculating model probability in Metropolis-Hastings algorithm when proposals are from the Gamma distribution.

**Usage**

```
gammaLogLikelihood(x, shape, rate)
```

**Arguments**

x	- double; the value to calculate the unnormalised likelihood of.
shape	- double; the shape of the Gamma distribution.
rate	- double; the rate of the Gamma distribution

**Value**

the unnormalised log-likelihood of x in a Gamma with parameters shape and rate.

---

generateBatchData	<i>Generate batch data</i>
-------------------	----------------------------

---

**Description**

Generate data from K multivariate normal or multivariate t distributions with additional noise from batches. Assumes independence across columns. In each column the parameters are randomly permuted for both the groups and batches.

**Usage**

```
generateBatchData(
  N,
  P,
  group_means,
  group_std_devs,
  batch_shift,
  batch_scale,
  group_weights,
  batch_weights,
  type = "MVN",
  group_dfs = NULL,
  frac_known = 0.2,
```



```

    permute_variables = TRUE,
    scale_data = FALSE
  )

```

### Arguments

N	The number of items (rows) to generate.
P	The number of columns in the generated dataset.
group_means	A vector of the group means for a column.
group_std_devs	A vector of group standard deviations for a column.
batch_shift	A vector of batch means in a column.
batch_scale	A vector of batch standard deviations within a column.
group_weights	One of either a K x B matrix of the expected proportion of each batch in each group or a K-vector of the expected proportion of the entire dataset in each group.
batch_weights	A vector of the expected proportion of N in each batch.
type	A string indicating if data should be generated from multivariate normal ("MVN") or multivariate t ("MVT") densities (defaults to "MVN").
group_dfs	A K-vector of the group specific degrees of freedom.
frac_known	The number of items with known labels.
permute_variables	Logical indicating if group and batch means and standard deviations should be permuted in each column or not (defaults to "TRUE").
scale_data	Logical indicating if data should be mean centred and standardised (defaults to "FALSE").

### Value

A list of 5 objects; the data generated from the groups with and without batch effects, the label indicating the generating group, the batch label and the vector indicating training versus test.

### Examples

```

N <- 500
P <- 2
K <- 2
B <- 5
mean_dist <- 4
batch_dist <- 0.3
group_means <- seq(1, K) * mean_dist
batch_shift <- rnorm(B, mean = batch_dist, sd = batch_dist)
std_dev <- rep(2, K)
batch_var <- rep(1.2, B)
group_weights <- rep(1 / K, K)
batch_weights <- rep(1 / B, B)
dfs <- c(4, 7)
my_data <- generateBatchData(

```

```

N,
P,
group_means,
std_dev,
batch_shift,
batch_var,
group_weights,
batch_weights,
type = "MVT",
group_dfs = dfs
)

```

---

generateBatchDataMVT *Generate batch data from a multivariate t distribution*

---

### Description

Generate data from K multivariate t distributions with additional noise from batches. Assumes independence across columns. In each column the parameters are randomly permuted for both the groups and batches.

### Usage

```

generateBatchDataMVT(
  N,
  P,
  group_means,
  group_std_devs,
  batch_shift,
  batch_scale,
  group_weights,
  batch_weights,
  dfs,
  frac_known = 0.2
)

```

### Arguments

N	The number of items (rows) to generate.
P	The number of columns in the generated dataset.
group_means	A vector of the group means for a column.
group_std_devs	A vector of group standard deviations for a column.
batch_shift	A vector of batch means in a column.
batch_scale	A vector of batch standard deviations within a column.
group_weights	A K x B matrix of the expected proportion of N in each group in each batch.
batch_weights	A vector of the expected proportion of N in each batch.
dfs	A K-vector of the group specific degrees of freedom.
frac_known	The number of items with known labels.

**Value**

A list of 5 objects; the data generated from the groups with and without batch effects, the label indicating the generating group, the batch label and the vector indicating training versus test.

**Examples**

```
N <- 500
P <- 2
K <- 2
B <- 5
mean_dist <- 4
batch_dist <- 0.3
group_means <- seq(1, K)* mean_dist
batch_shift <- rnorm(B, mean = batch_dist, sd = batch_dist)
std_dev <- rep(2, K)
batch_var <- rep(1.2, B)
group_weights <- rep(1 / K, K)
batch_weights <- rep(1 / B, B)
dfs <- c(4, 7)
my_data <- generateBatchDataMVT(
  N,
  P,
  group_means,
  std_dev,
  batch_shift,
  batch_var,
  group_weights,
  batch_weights,
  dfs
)
```

---

generateBatchDataVaryingRepresentation  
*Generate batch data*

---

**Description**

Generate data from groups across batches. Assumes independence across columns. In each column the parameters are randomly permuted for both the groups and batches.

**Usage**

```
generateBatchDataVaryingRepresentation(
  N,
  P,
  group_means,
  group_std_dev,
  batch_shift,
  batch_scale,
```

```

    group_weights,
    batch_weights,
    frac_known = 0.2
  )

```

### Arguments

N	The number of items (rows) to generate.
P	The number of columns in the generated dataset.
group_means	A vector of the group means for a column.
group_std_dev	A vector of group standard deviations for a column.
batch_shift	A vector of batch means in a column.
batch_scale	A vector of batch standard deviations within a column.
group_weights	A K x B matrix of the expected proportion of N in each group in each batch.
batch_weights	A vector of the expected proportion of N in each batch.
frac_known	The expected fraction of observed labels. Used to generate a “fixed“ vector to feed into the “batchSemiSupervisedMixtureModel“ function.

### Value

A list of 4 objects; the data generated from the groups with and without batch effects, the label indicating the generating group and the batch label.

### Examples

```

N <- 500
P <- 2
K <- 2
B <- 5
mean_dist <- 4
batch_dist <- 0.3
group_means <- seq(1, K) * mean_dist
batch_shift <- rnorm(B, mean = batch_dist, sd = batch_dist)
std_dev <- rep(2, K)
batch_var <- rep(1.2, B)
group_weights <- matrix(c(
  0.8, 0.6, 0.4, 0.2, 0.2,
  0.2, 0.4, 0.6, 0.8, 0.8
),
nrow = K, ncol = B, byrow = TRUE
)
batch_weights <- rep(1 / B, B)

my_data <- generateBatchDataVaryingRepresentation(
  N,
  P,
  group_means,
  std_dev,
  batch_shift,

```

```

    batch_var,
    group_weights,
    batch_weights
)

```

---

```

generateGroupIDsInSimulator
    Generate group IDs

```

---

### Description

Generate group IDs within “generateBatchData“.

### Usage

```

generateGroupIDsInSimulator(
  N,
  K,
  B,
  batch_IDs,
  group_weights,
  varying_group_within_batch
)

```

### Arguments

N	The number of items (rows) to generate.
K	The number of groups to generate.
B	The number of batches present in “batch_IDs“.
batch_IDs	The batch membership of each item.
group_weights	One of either a K x B matrix of the expected proportion of each batch in each group or a K-vector of the expected proportion of the entire dataset in each group.
varying_group_within_batch	Flag indicating if the groups are varying across batches.

### Value

A N-vector of group membership.

### Examples

```
N <- 500
K <- 2
B <- 5
group_weights <- rep(1 / K, K)
batch_weights <- rep(1 / B, B)
batch_IDs <- sample(seq(1, B), N, replace = TRUE, prob = batch_weights)
varying_group_within_batch <- FALSE
group_IDs <- generateGroupIDsInSimulator(
  N,
  K,
  B,
  batch_IDs,
  group_weights,
  varying_group_within_batch
)
```

---

generateInitialLabels *Generate initial labels*

---

### Description

For simulated data, generates an initial labelling for sampling.

### Usage

```
generateInitialLabels(alpha, K, fixed, labels = NULL)
```

### Arguments

alpha	The mass in the stick breaking prior
K	The number of classes available.
fixed	The vector of 0s and 1s indicating which labels are to be held fixed.
labels	The initial labelling. Defaults to NULL.

### Value

An N vector of labels.

### Examples

```
N <- 500
P <- 2
K <- 2
B <- 5
mean_dist <- 4
batch_dist <- 0.3
cluster_means <- seq(1, K) * mean_dist
```

```
batch_shift <- rnorm(B, mean = batch_dist, sd = batch_dist)
std_dev <- rep(2, K)
batch_var <- rep(1.2, B)
cluster_weights <- rep(1 / K, K)
batch_weights <- rep(1 / B, B)

my_data <- generateBatchData(
  N,
  P,
  cluster_means,
  std_dev,
  batch_shift,
  batch_var,
  cluster_weights,
  batch_weights
)

initial_labels <- generateInitialLabels(1, K, my_data$fixed)
```

---

getLikelihood

*Get likelihood*

---

### Description

Extracts the model fit score from the mixture model output.

### Usage

```
getLikelihood(mcmc_output, choice = "complete_likelihood")
```

### Arguments

mcmc_output	The output from the mixture model.
choice	The model fit score to use. Must be one of "observed_likelihood", "complete_likelihood" or "BIC". Defaults to "complete_likelihood".

### Value

A data.frame containing the model fit score of choice and the iteration.

### Examples

```
# Data in a matrix format
X <- matrix(c(rnorm(100, 0, 1), rnorm(100, 3, 1)), ncol = 2, byrow = TRUE)

# Batch
batch_vec <- sample(seq(1, 5), replace = TRUE, size = 100)
```

```
# Sampling parameters
R <- 100
thin <- 5

# MCMC samples and BIC vector
samples <- runBatchMix(X, R, thin, batch_vec, "MVN")

lkl_df <- getLikelihood(samples)
```

---

getSampledBatchScale *Get sampled batch shift*

---

### Description

Given an array of sampled batch scales from the “mixtureModel” function, acquire a tidy version ready for “ggplot2” use.

### Usage

```
getSampledBatchScale(
  sampled_batch_scale,
  B = dim(sampled_batch_scale)[2],
  P = dim(sampled_batch_scale)[1],
  R = dim(sampled_batch_scale)[3],
  thin = 1
)
```

### Arguments

sampled_batch_scale	A 3D array of sampled batch mean shifts.
B	The number of batches present. Defaults to the number of columns in the batch mean matrix from the first sample.
P	The dimension of the batch mean shifts. Defaults to the number of rows in the batch mean matrix from the first sample.
R	The number of iterations run. Defaults to the number of slices in the sampled batch mean array.
thin	The thinning factor of the sampler. Defaults to 1.

### Value

A data.frame of three columns; the parameter, the sampled value and the iteration.



**Examples**

```

# Data in matrix format
X <- matrix(c(rnorm(100, 0, 1), rnorm(100, 3, 1)), ncol = 2, byrow = TRUE)

# Observed batches represented by integers
batch_vec <- sample(seq(1, 5), size = 100, replace = TRUE)

# MCMC iterations (this is too low for real use)
R <- 100
thin <- 5

# MCMC samples
samples <- runBatchMix(X, R, thin, batch_vec, "MVN")

batch_scale_df <- getSampledBatchShift(samples$batch_scale, R = R, thin = thin)

```

---

```
getSampledBatchShift  Get sampled batch shift
```

---

**Description**

Given an array of sampled batch mean shifts from the “mixtureModel” function, acquire a tidy version ready for “ggplot2” use.

**Usage**

```

getSampledBatchShift(
  sampled_batch_shift,
  B = dim(sampled_batch_shift)[2],
  P = dim(sampled_batch_shift)[1],
  R = dim(sampled_batch_shift)[3],
  thin = 1
)

```

**Arguments**

sampled_batch_shift	A 3D array of sampled batch mean shifts.
B	The number of batches present. Defaults to the number of columns in the batch mean matrix from the first sample.
P	The dimension of the batch mean shifts. Defaults to the number of rows in the batch mean matrix from the first sample.
R	The number of iterations run. Defaults to the number of slices in the sampled batch mean array.
thin	The thinning factor of the sampler. Defaults to 1.

**Value**

A data.frame of three columns; the parameter, the sampled value and the iteration.

**Examples**

```
# Data in matrix format
X <- matrix(c(rnorm(100, 0, 1), rnorm(100, 3, 1)), ncol = 2, byrow = TRUE)

# Observed batches represented by integers
batch_vec <- sample(seq(1, 5), size = 100, replace = TRUE)

# MCMC iterations (this is too low for real use)
R <- 100
thin <- 5

# MCMC samples
samples <- runBatchMix(X, R, thin, batch_vec, "MVN")

batch_shift_df <- getSampledBatchShift(samples$batch_shift,
  R = R,
  thin = thin
)
```

---

```
getSampledClusterMeans
```

*Get sampled cluster means*

---

**Description**

Given an array of sampled cluster means from the “mixtureModel“ function, acquire a tidy version ready for “ggplot2“ use.

**Usage**

```
getSampledClusterMeans(
  sampled_cluster_means,
  K = dim(sampled_cluster_means)[2],
  P = dim(sampled_cluster_means)[1],
  R = dim(sampled_cluster_means)[3],
  thin = 1
)
```

**Arguments**

```
sampled_cluster_means
  A 3D array of sampled cluster means.
```

K	The number of clusters present. Defaults to the number of columns in the batch mean matrix from the first sample.
P	The dimension of the batch mean shifts. Defaults to the number of rows in the batch mean matrix from the first sample.
R	The number of iterations run. Defaults to the number of slices in the sampled batch mean array.
thin	The thinning factor of the sampler. Defaults to 1.

**Value**

A data.frame of three columns; the parameter, the sampled value and the iteration.

**Examples**

```
# Data in matrix format
X <- matrix(c(rnorm(100, 0, 1), rnorm(100, 3, 1)), ncol = 2, byrow = TRUE)

# Observed batches represented by integers
batch_vec <- sample(seq(1, 5), size = 100, replace = TRUE)

# MCMC iterations (this is too low for real use)
R <- 100
thin <- 5

# MCMC samples
samples <- runBatchMix(X, R, thin, batch_vec, "MVN")

batch_shift_df <- getSampledClusterMeans(samples$means, R = R, thin = thin)
```

---

invGammaLogLikelihood *Inverse gamma log-likelihood*

---

**Description**

Used in calculating model probability in Metropolis-Hastings algorithm when proposals are from the inverse-Gamma distribution.

**Usage**

```
invGammaLogLikelihood(x, shape, scale)
```

**Arguments**

x	- double; the value to calculate the likelihood of.
shape	- double; the shape of the inverse-Gamma distribution.
scale	- double; the scale of the inverse-Gamma distribution

**Value**

the unnormalised log-likelihood of  $x$  in a inverse-Gamma with parameters shape and scale.

---

invWishartLogLikelihood

*Inverse-Wishart log-likelihood*

---

**Description**

Used in calculating model probability in Metropolis-Hastings algorithm when proposals are from the Wishart distribution.

**Usage**

```
invWishartLogLikelihood(X, Psi, nu, P)
```

**Arguments**

X	- matrix; the matrix to calculate the likelihood of.
Psi	- matrix; the scale of the inverse-Wishart distribution.
nu	- double; the degrees of freedom for the inverse-Wishart distribution.
P	- unsigned integer; the dimension of X.

**Value**

the unnormalised log-likelihood of  $X$  in a inverse-Wishart with parameters Psi and nu.

---

plotAcceptanceRates *Plot acceptance rates*

---

**Description**

Plot the acceptance rates for the parameters sampled in a Metropolis-Hastings step. Aiming for acceptance rates in [0.2, 0.5] for the class means, the batch effect on location and scale is a good rule of thumb. The class covariance should be in [0.35, 0.8] based on the authors' experience. The class degree of freedom appears to be prone to high acceptance rates, but aim to keep this above 0.2 at a minimum.

**Usage**

```
plotAcceptanceRates(mcmc_lst)
```

**Arguments**

mcmc_lst	The output of the "runMCMCChains" function.
----------	---

**Value**

A ggplot object of the boxplots of acceptance rates for each parameter across chains.

**Examples**

```
# Data in a matrix format
X <- matrix(c(rnorm(100, 0, 1), rnorm(100, 3, 1)), ncol = 2, byrow = TRUE)

# Initial labelling
labels <- c(
  rep(1, 10),
  sample(c(1, 2), size = 40, replace = TRUE),
  rep(2, 10),
  sample(c(1, 2), size = 40, replace = TRUE)
)

fixed <- c(rep(1, 10), rep(0, 40), rep(1, 10), rep(0, 40))

# Batch
batch_vec <- sample(seq(1, 5), replace = TRUE, size = 100)

# Sampling parameters
R <- 500
thin <- 10
n_chains <- 4

# MCMC samples and BIC vector
mcmc_lst <- runMCMCchains(X, n_chains, R, thin, batch_vec, "MVN",
  initial_labels = labels,
  fixed = fixed
)

# Plot the acceptance rate of each parameter in the 4 chains
plotAcceptanceRates(mcmc_lst)
```

---

plotLikelihoods      *Plot likelihoods*

---

**Description**

Plots the model fit for multiple chains.

**Usage**

```
plotLikelihoods(
  mcmc_outputs,
  choice = "complete_likelihood",
  colour_by_chain = TRUE
)
```

**Arguments**

`mcmc_outputs` The output from “runMCMCChains”.

`choice` The model fit score to use. Must be one of “observed\_likelihood”, “complete\_likelihood” or “BIC”. Defaults to “complete\_likelihood”.

`colour_by_chain` Logical indicating if plots should be coloured by chain or all the same colour. Defaults to “TRUE”.

**Value**

A ggplot2 object. Line plot of likelihood across iteration.

**Examples**

```
# Data in a matrix format
X <- matrix(c(rnorm(100, 0, 1), rnorm(100, 3, 1)), ncol = 2, byrow = TRUE)

# Initial labelling
labels <- c(
  rep(1, 10),
  sample(c(1, 2), size = 40, replace = TRUE),
  rep(2, 10),
  sample(c(1, 2), size = 40, replace = TRUE)
)

fixed <- c(rep(1, 10), rep(0, 40), rep(1, 10), rep(0, 40))

# Batch
batch_vec <- sample(seq(1, 5), replace = TRUE, size = 100)

# Sampling parameters
R <- 1000
thin <- 50
n_chains <- 4

# MCMC samples
samples <- runMCMCChains(X, n_chains, R, thin, batch_vec, "MVN",
  initial_labels = labels,
  fixed = fixed
)

p <- plotLikelihoods(samples)
```

**Description**

Plot the sampled values for the batch mean shifts in each dimension from the output of the mixture model functions. Not recommended for large B or P.

**Usage**

```
plotSampledBatchMeans(samples, burn_in = 0)
```

**Arguments**

samples	The output of the “batchUnsupervisedMixtureModel” or “batchSemiSupervisedMixtureModel” functions.
burn_in	The samples at the beginning of the chain to drop. Defaults to 0.

**Value**

A ggplot object of the values in each sampled batch mean per iteration.

**Examples**

```
# Data in matrix format
X <- matrix(c(rnorm(100, 0, 1), rnorm(100, 3, 1)), ncol = 2, byrow = TRUE)

# Observed batches represented by integers
batch_vec <- sample(seq(1, 5), size = 100, replace = TRUE)

# MCMC iterations (this is too low for real use)
R <- 100
thin <- 5

# MCMC samples and BIC vector
samples <- runBatchMix(X, R, thin, batch_vec, "MVN")

# Plot the sampled value of the batch mean shift against MCMC iteration
plotSampledBatchMeans(samples)
```

---

plotSampledBatchScales

*Plot sampled batch scales*

---

**Description**

Plot the sampled values for the batch scale in each dimension from the output of the mixture model functions. Not recommended for large B or P.

**Usage**

```
plotSampledBatchScales(samples, burn_in = 0)
```

**Arguments**

`samples`            The output of the “batchUnsupervisedMixtureModel” or “batchSemiSupervised-MixtureModel” functions.

`burn_in`            The samples at the beginning of the chain to drop. Defaults to 0.

**Value**

A ggplot object of the values in each sampled batch mean per iteration.

**Examples**

```
# Data in matrix format
X <- matrix(c(rnorm(100, 0, 1), rnorm(100, 3, 1)), ncol = 2, byrow = TRUE)

# Observed batches represented by integers
batch_vec <- sample(seq(1, 5), size = 100, replace = TRUE)

# MCMC iterations (this is too low for real use)
R <- 100
thin <- 5

# MCMC samples and BIC vector
samples <- runBatchMix(X, R, thin, batch_vec, "MVN")

# Plot the sampled value of the batch scales against MCMC iteration
plotSampledBatchScales(samples)
```

---

```
plotSampledClusterMeans
```

*Plot sampled cluster means*

---

**Description**

Plot the sampled values for the cluster means in each dimension from the output of the mixture model functions. Not recommended for large K or P.

**Usage**

```
plotSampledClusterMeans(samples, burn_in = 0)
```

**Arguments**

`samples`            The output of the “batchUnsupervisedMixtureModel” or “batchSemiSupervised-MixtureModel” functions.

`burn_in`            The samples at the beginning of the chain to drop. Defaults to 0.



**Value**

A ggplot object of the values in each sampled cluster mean per iteration.

**Examples**

```
# Data in matrix format
X <- matrix(c(rnorm(100, 0, 1), rnorm(100, 3, 1)), ncol = 2, byrow = TRUE)

# Observed batches represented by integers
batch_vec <- sample(seq(1, 5), size = 100, replace = TRUE)

# MCMC iterations (this is too low for real use)
R <- 100
thin <- 5

# MCMC samples and BIC vector
samples <- runBatchMix(X, R, thin, batch_vec, "MVN")

# Plot the sampled value of the cluster means against MCMC iteration
plotSampledClusterMeans(samples)
```

---

plotSampledParameter *Plot sampled vector parameter*

---

**Description**

Plot the sampled values for a sampled vector from the output of the “mixtureModel“ function. Not recommended for large B or P.

**Usage**

```
plotSampledParameter(samples, parameter, R = NULL, thin = 1, burn_in = 0)
```

**Arguments**

samples	The output of the “mixtureModel“ function.
parameter	The name of the parameter to be plotted (a string).
R	The number of iterations run. Defaults to the number of samples for the cluster membership.
thin	The thinning factor of the sampler. Defaults to 1.
burn_in	The samples at the beginning of the chain to drop. Defaults to 0.

**Value**

A ggplot object of the values in each sampled batch mean per iteration.

**Examples**

```

# Data in matrix format
X <- matrix(c(rnorm(100, 0, 1), rnorm(100, 3, 1)), ncol = 2, byrow = TRUE)

# Observed batches represented by integers
batch_vec <- sample(seq(1, 3), size = 100, replace = TRUE)

# MCMC iterations (this is too low for real use)
R <- 50
thin <- 1

# MCMC samples and BIC vector
samples <- runBatchMix(X, R, thin, batch_vec, "MVN", K_max = 8)

# Plot the sampled value of the cluster means against MCMC iteration
parameter <- "means"
plotSampledParameter(samples, parameter, R, thin)

```

---

predictClass	<i>Predict class</i>
--------------	----------------------

---

**Description**

Predicts a final class for each item given a matrix of allocation probabilities.

**Usage**

```
predictClass(prob)
```

**Arguments**

prob                      Output from the “calcAllocProb” function, a N x K matrix of allocation probabilities.

**Value**

An N vector of class allocations.

**Examples**

```

# Data in a matrix format
X <- matrix(c(rnorm(100, 0, 1), rnorm(100, 3, 1)), ncol = 2, byrow = TRUE)

# Initial labelling
labels <- c(rep(1, 10),
  sample(c(1,2), size = 40, replace = TRUE),
  rep(2, 10),
  sample(c(1,2), size = 40, replace = TRUE))

```

```

)

fixed <- c(rep(1, 10), rep(0, 40), rep(1, 10), rep(0, 40))

# Batch
batch_vec <- sample(seq(1, 5), replace = TRUE, size = 100)

# Sampling parameters
R <- 1000
thin <- 50

# MCMC samples and BIC vector
samples <- batchSemiSupervisedMixtureModel(X,
  R,
  thin,
  labels,
  fixed,
  batch_vec,
  "MVN"
)

# Burn in
burn <- 200
eff_burn <- burn / thin

# Probability across classes
probs <- calcAllocProb(samples, burn = burn)

# Predict the class
preds <- predictClass(probs)

```

---

```
prepareInitialParameters
```

*Prepare initial values*

---

## Description

Prepares user given values for input into the C++ function.

## Usage

```

prepareInitialParameters(
  initial_class_means,
  initial_class_covariance,
  initial_batch_shift,
  initial_batch_scale,
  initial_class_df,
  P,

```

```

    K,
    B,
    type
)

```

### Arguments

`initial_class_means`  
A  $P \times K$  matrix of initial values for the class means. Defaults to draws from the prior distribution.

`initial_class_covariance`  
A  $P \times P \times K$  array of initial values for the class covariance matrices. Defaults to draws from the prior distribution.

`initial_batch_shift`  
A  $P \times B$  matrix of initial values for the batch shift effect Defaults to draws from the prior distribution.

`initial_batch_scale`  
A  $P \times B$  matrix of initial values for the batch scales Defaults to draws from the prior distribution.

`initial_class_df`  
A  $K$  vector of initial values for the class degrees of freedom. Defaults to draws from the prior distribution.

`P`  
Integer. The number of measurements for each sample in the dataset being modelled.

`K`  
The number of classes/clusters being modelled.

`B`  
The number of batches being modelled.

`type`  
The type of mixture model used; one of "MVN" or "MVT".

### Value

A named list containing the different parameters.

---

processMCMCChain	<i>Process MCMC chain</i>
------------------	---------------------------

---

### Description

Applies a burn in to and finds a point estimate for the output of “batchSemiSupervisedMixture-Model”.

### Usage

```
processMCMCChain(mcmc_output, burn, point_estimate_method = "median")
```

**Arguments**

mcmc_output	Output from “batchSemiSupervisedMixtureModel”
burn	The number of MCMC samples to drop as part of a burn in.
point_estimate_method	Summary statistic used to define the point estimate. Must be “mean” or “median”. “median” is the default.

**Value**

A named list similar to the output of “batchSemiSupervisedMixtureModel” with some additional entries:

- \* “mean\_est”:  $(P \times K)$  matrix. The point estimate of the cluster means with columns corresponding to clusters.
- \* “cov\_est”:  $(P \times P \times K)$  array. The point estimate of the cluster covariance matrices with slices corresponding to clusters.
- \* “shift\_est”:  $(P \times B)$  matrix. The point estimate of the batch shift effect with columns corresponding to batches.
- \* “scale\_est”:  $(P \times B)$  matrix. The point estimate of the batch scale effects. The  $b$ th column contains the diagonal entries of the scaling matrix for the  $b$ th batch.
- \* “mean\_sum\_est”:  $(P \times K \times B)$  array. The point estimate of the sum of the cluster means and the batch shift effect with columns corresponding to clusters and slices to batches.
- \* “cov\_comb\_est”: List of length  $B$ , with each entry being a  $(P \times P \times K)$  array. The point estimate of the combination of the cluster covariance matrices and the batch scale effect with list entries corresponding to batches and slices of each array corresponding to clusters.
- \* “inferred\_dataset”:  $(N \times P)$  matrix. The inferred “batch-free” dataset.
- \* “allocation\_probability”:  $(N \times K)$  matrix. The point estimate of the allocation probabilities for each data point to each class.
- \* “prob”:  $N$  vector. The point estimate of the probability of being allocated to the class with the highest probability.
- \* “pred”:  $N$  vector. The predicted class for each sample.

**Examples**

```
# Data in a matrix format
X <- matrix(c(rnorm(100, 0, 1), rnorm(100, 3, 1)), ncol = 2, byrow = TRUE)

# Initial labelling
labels <- c(rep(1, 10),
  sample(c(1,2), size = 40, replace = TRUE),
  rep(2, 10),
  sample(c(1,2), size = 40, replace = TRUE)
)

fixed <- c(rep(1, 10), rep(0, 40), rep(1, 10), rep(0, 40))
```

```

# Batch
batch_vec <- sample(seq(1, 5), replace = TRUE, size = 100)

# Sampling parameters
R <- 1000
burn <- 250
thin <- 50

# MCMC samples
samples <- runBatchMix(X, R, thin, batch_vec, "MVN",
  initial_labels = labels,
  fixed = fixed
)

# Process the MCMC samples
processed_samples <- processMCMCChain(samples, burn)

```

---

processMCMCChains      *Process MCMC chains*

---

### Description

Applies a burn in to and finds a point estimate for each of the chains outputted from “runMCMC-Chains”.

### Usage

```
processMCMCChains(mcmc_lst, burn)
```

### Arguments

mcmc_lst	Output from “runMCMCChains”
burn	The number of MCMC samples to drop as part of a burn in.

### Value

A named list similar to the output of “batchSemiSupervisedMixtureModel” with some additional entries:

\* “mean\_est”:  $(P \times K)$  matrix. The point estimate of the cluster means with columns corresponding to clusters.

\* “cov\_est”:  $(P \times P \times K)$  array. The point estimate of the cluster covariance matrices with slices corresponding to clusters.

\* “shift\_est”:  $(P \times B)$  matrix. The point estimate of the batch shift effect with columns corresponding to batches.

\* “scale\_est”:  $(P \times B)$  matrix. The point estimate of the batch scale effects. The  $b$ th column contains the diagonal entries of the scaling matrix for the  $b$ th batch.

- \* “mean\_sum\_est“:  $(P \times K \times B)$  array. The point estimate of the sum of the cluster means and the batch shift effect with columns corresponding to clusters and slices to batches.
- \* “cov\_comb\_est“: List of length  $B$ , with each entry being a  $(P \times P \times K)$  array. The point estimate of the combination of the cluster covariance matrices and the batch scale effect with list entries corresponding to batches and slices of each array corresponding to clusters.
- \* “inferred\_dataset“:  $(N \times P)$  matrix. The inferred “batch-free” dataset.
- \* “allocation\_probability“:  $(N \times K)$  matrix. The point estimate of the allocation probabilities for each data point to each class.
- \* “prob“:  $N$  vector. The point estimate of the probability of being allocated to the class with the highest probability.
- \* “pred“:  $N$  vector. The predicted class for each sample.

## Examples

```
# Data in a matrix format
X <- matrix(c(rnorm(100, 0, 1), rnorm(100, 3, 1)), ncol = 2, byrow = TRUE)

# Initial labelling
labels <- c(rep(1, 10),
  sample(c(1,2), size = 40, replace = TRUE),
  rep(2, 10),
  sample(c(1,2), size = 40, replace = TRUE)
)

fixed <- c(rep(1, 10), rep(0, 40), rep(1, 10), rep(0, 40))

# Batch
batch_vec <- sample(seq(1, 5), replace = TRUE, size = 100)

# Sampling parameters
R <- 1000
burn <- 250
thin <- 50
n_chains <- 4

# MCMC samples
samples <- runMCMCChains(X, n_chains, R, thin, batch_vec, "MVN",
  initial_labels = labels,
  fixed = fixed
)

# Process the MCMC samples
processed_samples <- processMCMCChains(samples, burn)
```

---

rStickBreakingPrior      *Random Draw From Stick Breaking Prior*

---

**Description**

Draw weights from the stick-breaking prior.

**Usage**

```
rStickBreakingPrior(alpha, K)
```

**Arguments**

alpha	The concentration parameter.
K	The number of weights to generate.

**Value**

A vector of component weights.

**Examples**

```
weights <- rStickBreakingPrior(1, 50)
```

---

runBatchMix      *Run Batch Mixture Model*

---

**Description**

Runs a MCMC chain for a Bayesian mixture model which models both batch effects and class/cluster structure.

**Usage**

```
runBatchMix(  
  X,  
  R,  
  thin,  
  batch_vec,  
  type,  
  K_max = NULL,  
  initial_labels = NULL,  
  fixed = NULL,  
  alpha = 1,  
  mu_proposal_window = 0.5^2,  
  cov_proposal_window = 0.002,
```



```

    m_proposal_window = 0.3^2,
    S_proposal_window = 0.01,
    t_df_proposal_window = 0.015,
    m_scale = 0.01,
    rho = 3,
    theta = 1,
    initial_class_means = NULL,
    initial_class_covariance = NULL,
    initial_batch_shift = NULL,
    initial_batch_scale = NULL,
    initial_class_df = NULL,
    verbose = TRUE
)

```

### Arguments

X	Data to cluster as a matrix with the items to cluster held in rows.
R	The number of iterations in the sampler.
thin	The factor by which the samples generated are thinned, e.g. if “thin=50” only every 50th sample is kept.
batch_vec	Labels identifying which batch each item being clustered is from.
type	Character indicating density type to use. One of ‘MVN’ (multivariate normal distribution) or ‘MVT’ (multivariate t distribution).
K_max	The number of components to include (the upper bound on the number of clusters in each sample). Defaults to the number of unique labels in “initial_labels”.
initial_labels	Initial clustering.
fixed	Which items are fixed in their initial label. If not given, defaults to a vector of 0 meaning the model is run unsupervised.
alpha	The concentration parameter for the stick-breaking prior and the weights in the model.
mu_proposal_window	The proposal window for the cluster mean proposal kernel. Making this smaller will normally increase the acceptance rate for the proposed values in the Metropolis-Hastings sampler. The proposal density is a Gaussian distribution, the window is the variance.
cov_proposal_window	The proposal window for the cluster covariance proposal kernel. The proposal density is a Wishart distribution, this argument is the reciprocal of the degree of freedom. It is recommended to set this aiming for acceptance rates of greater than 0.5 for the covariance matrices (e.g., between 2e-03 and 1e-04 is a good range to consider initially). As the entire covariance matrix is sampled at once exploration is difficult.
m_proposal_window	The proposal window for the batch mean proposal kernel. The proposal density is a Gaussian distribution, the window is the variance.

<code>S_proposal_window</code>	The proposal window for the batch standard deviation proposal kernel. The proposal density is a Gamma distribution, this argument is the reciprocal of the rate. Recommended range to initially consider is 0.015 to 2e-03, though smaller values might be necessary particularly in higher dimensional data.
<code>t_df_proposal_window</code>	The proposal window for the degrees of freedom for the multivariate t distribution (not used if type is not 'MVT'). The proposal density is a Gamma distribution, this argument is the reciprocal of the rate. If the data is more Gaussian than the degrees of freedom might have high acceptance rates regardless of the value chosen.
<code>m_scale</code>	The scale hyperparameter for the batch shift prior distribution. This defines the scale of the batch effect upon the mean and should be in (0, 1].
<code>rho</code>	The shape of the prior distribution for the batch scale.
<code>theta</code>	The scale of the prior distribution for the batch scale.
<code>initial_class_means</code>	A $P \times K$ matrix of initial values for the class means. Defaults to draws from the prior distribution.
<code>initial_class_covariance</code>	A $P \times P \times K$ array of initial values for the class covariance matrices. Defaults to draws from the prior distribution.
<code>initial_batch_shift</code>	A $P \times B$ matrix of initial values for the batch shift effect Defaults to draws from the prior distribution.
<code>initial_batch_scale</code>	A $P \times B$ matrix of initial values for the batch scales Defaults to draws from the prior distribution.
<code>initial_class_df</code>	A $K$ vector of initial values for the class degrees of freedom. Defaults to draws from the prior distribution.
<code>verbose</code>	Logical indicating if warning about proposal windows should be printed.

### Value

A named list containing the sampled partitions, cluster and batch parameters, model fit measures and some details on the model call.

### Examples

```
# Data in a matrix format
X <- matrix(c(rnorm(100, 0, 1), rnorm(100, 3, 1)), ncol = 2, byrow = TRUE)

# Initial labelling
labels <- c(
  rep(1, 10),
  sample(c(1, 2), size = 40, replace = TRUE),
```

```
    rep(2, 10),
    sample(c(1, 2), size = 40, replace = TRUE)
  )

fixed <- c(rep(1, 10), rep(0, 40), rep(1, 10), rep(0, 40))

# Batch
batch_vec <- sample(seq(1, 5), replace = TRUE, size = 100)

# Density choice
type <- "MVN"

# Sampling parameters
R <- 1000
thin <- 50

# MCMC samples
mcmc_out <- runBatchMix(
  X,
  R,
  thin,
  batch_vec,
  type,
  initial_labels = labels,
  fixed = fixed
)

# Given an initial value for the parameters
initial_class_means <- matrix(c(1, 1, 3, 4), nrow = 2)
initial_class_covariance <- array(c(1, 0, 0, 1, 1, 0, 0, 1),
  dim = c(2, 2, 2)
)

# We can use values from a previous chain
initial_batch_shift <- mcmc_out$batch_shift[, , R / thin]
initial_batch_scale <- matrix(
  c(1.2, 1.3, 1.7, 1.1, 1.4, 1.3, 1.2, 1.2, 1.1, 2.0),
  nrow = 2
)

mcmc_out <- runBatchMix(X,
  R,
  thin,
  batch_vec,
  type,
  initial_labels = labels,
  fixed = fixed,
  initial_class_means = initial_class_means,
  initial_class_covariance = initial_class_covariance,
  initial_batch_shift = initial_batch_shift,
  initial_batch_scale = initial_batch_scale
)
```

---

runMCMCChains                      *Run MCMC Chains*

---

### Description

Run multiple chains of the batch mixture model of the same type.

### Usage

```
runMCMCChains(
  X,
  n_chains,
  R,
  thin,
  batch_vec,
  type,
  K_max = NULL,
  initial_labels = NULL,
  fixed = NULL,
  alpha = 1,
  mu_proposal_window = 0.5^2,
  cov_proposal_window = 0.002,
  m_proposal_window = 0.3^2,
  S_proposal_window = 0.01,
  t_df_proposal_window = 0.015,
  m_scale = 0.01,
  rho = 3,
  theta = 1,
  initial_class_means = NULL,
  initial_class_covariance = NULL,
  initial_batch_shift = NULL,
  initial_batch_scale = NULL,
  initial_class_df = NULL,
  verbose = TRUE
)
```

### Arguments

X	Data to cluster as a matrix with the items to cluster held in rows.
n_chains	Integer. Number of MCMC chains to run.
R	The number of iterations in the sampler.
thin	The factor by which the samples generated are thinned, e.g. if “thin=50” only every 50th sample is kept.
batch_vec	Labels identifying which batch each item being clustered is from.
type	Character indicating density type to use. One of 'MVN' (multivariate normal distribution) or 'MVT' (multivariate t distribution).

K_max	The number of components to include (the upper bound on the number of clusters in each sample). Defaults to the number of unique labels in “initial_labels”.
initial_labels	Initial clustering, if none given defaults to a random draw.
fixed	Which items are fixed in their initial label. If not given, defaults to a vector of 0 meaning the model is run unsupervised.
alpha	The concentration parameter for the stick-breaking prior and the weights in the model.
mu_proposal_window	The proposal window for the cluster mean proposal kernel. The proposal density is a Gaussian distribution, the window is the variance.
cov_proposal_window	The proposal window for the cluster covariance proposal kernel. The proposal density is a Wishart distribution, this argument is the reciprocal of the degree of freedom.
m_proposal_window	The proposal window for the batch mean proposal kernel. The proposal density is a Gaussian distribution, the window is the variance.
S_proposal_window	The proposal window for the batch standard deviation proposal kernel. The proposal density is a Gamma distribution, this argument is the reciprocal of the rate.
t_df_proposal_window	The proposal window for the degrees of freedom for the multivariate t distribution (not used if type is not 'MVT'). The proposal density is a Gamma distribution, this argument is the reciprocal of the rate.
m_scale	The scale hyperparameter for the batch shift prior distribution. This defines the scale of the batch effect upon the mean and should be in (0, 1].
rho	The shape of the prior distribution for the batch scale.
theta	The scale of the prior distribution for the batch scale.
initial_class_means	A $P \times K$ matrix of initial values for the class means. Defaults to draws from the prior distribution.
initial_class_covariance	A $P \times P \times K$ array of initial values for the class covariance matrices. Defaults to draws from the prior distribution.
initial_batch_shift	A $P \times B$ matrix of initial values for the batch shift effect Defaults to draws from the prior distribution.
initial_batch_scale	A $P \times B$ matrix of initial values for the batch scales Defaults to draws from the prior distribution.
initial_class_df	A $K$ vector of initial values for the class degrees of freedom. Defaults to draws from the prior distribution.
verbose	Logical indicating if warning about proposal windows should be printed.

**Value**

A list of named lists. Each entry is the output of “runBatchMix”.

**Examples**

```
# Data in a matrix format
X <- matrix(c(rnorm(100, 0, 1), rnorm(100, 3, 1)), ncol = 2, byrow = TRUE)

# Initial labelling
labels <- c(
  rep(1, 10),
  sample(c(1, 2), size = 40, replace = TRUE),
  rep(2, 10),
  sample(c(1, 2), size = 40, replace = TRUE)
)

fixed <- c(rep(1, 10), rep(0, 40), rep(1, 10), rep(0, 40))

# Batch
batch_vec <- sample(seq(1, 5), replace = TRUE, size = 100)

# Sampling parameters
R <- 1000
thin <- 50
n_chains <- 4

# MCMC samples
samples <- runMCMCchains(X, n_chains, R, thin, batch_vec, "MVN",
  initial_labels = labels,
  fixed = fixed
)
```

---

sampleMVN

*Sample mixture of multivariate normal distributions with batch effects*

---

**Description**

Performs MCMC sampling for a mixture model with batch effects.

**Usage**

```
sampleMVN(
  X,
  K,
  B,
  labels,
  batch_vec,
```

```

    mu_proposal_window,
    cov_proposal_window,
    m_proposal_window,
    S_proposal_window,
    R,
    thin,
    concentration,
    m_scale,
    rho,
    theta,
    initial_mu,
    initial_cov,
    initial_m,
    initial_S,
    mu_initialised,
    cov_initialised,
    m_initialised,
    S_initialised
)

```

### Arguments

X	The data matrix to perform clustering upon (items to cluster in rows).
K	The number of components to model (upper limit on the number of clusters found).
B	The number of batches to model.
labels	Vector item labels to initialise from.
batch_vec	Observed batch labels.
mu_proposal_window	The standard deviation for the Gaussian proposal density of the cluster means.
cov_proposal_window	The degrees of freedom for the Wishart proposal density of the cluster covariances.
m_proposal_window	The standard deviation for the Gaussian proposal density of the batch mean effects.
S_proposal_window	The rate for the Gamma proposal density of the batch scale.
R	The number of iterations to run for.
thin	thinning factor for samples recorded.
concentration	Vector of concentrations for mixture weights (recommended to be symmetric).
m_scale	The scale hyperparameter for the batch shift prior distribution.
rho	The shape of the prior distribution for the batch scale.
theta	The scale of the prior distribution for the batch scale.
initial_mu	A $P \times K$ matrix of initial values for the class means.

initial_cov	A $P \times P \times K$ cube of initial values for the class covariance matrices.
initial_m	A $P \times B$ matrix of initial values for the batch shift effects.
initial_S	A $P \times B$ matrix of initial values for the batch scales.
mu_initialised	Bool indicating if the class means are initialised by the user. If “false“ then initial values are drawn from the prior distribution.
cov_initialised	Bool indicating if the class covariance matrices are initialised by the user. If “false“ then initial values are drawn from the prior distribution.
m_initialised	Bool indicating if the batch shift effects are initialised by the user. If “false“ then initial values are drawn from the prior distribution.
S_initialised	Bool indicating if the batch scales are initialised by the user. If “false“ then initial values are drawn from the prior distribution.

**Value**

Named list of the different quantities drawn by the sampler.

---

sampleMVT

*Sample mixture of multivariate t-distributions with batch effects*

---

**Description**

Performs MCMC sampling for a MVT mixture model with batch effects.

**Usage**

```
sampleMVT(
  X,
  K,
  B,
  labels,
  batch_vec,
  mu_proposal_window,
  cov_proposal_window,
  m_proposal_window,
  S_proposal_window,
  t_df_proposal_window,
  R,
  thin,
  concentration,
  m_scale,
  rho,
  theta,
  initial_mu,
  initial_cov,
```



```

    initial_df,
    initial_m,
    initial_S,
    mu_initialised,
    cov_initialised,
    df_initialised,
    m_initialised,
    S_initialised
)

```

### Arguments

X	The data matrix to perform clustering upon (items to cluster in rows).
K	The number of components to model (upper limit on the number of clusters found).
B	The number of batches to model.
labels	Vector item labels to initialise from.
batch_vec	Observed batch labels.
mu_proposal_window	The standard deviation for the Gaussian proposal density of the cluster means.
cov_proposal_window	The degrees of freedom for the Wishart proposal density of the cluster covariances.
m_proposal_window	The standard deviation for the Gaussian proposal density of the batch mean effects.
S_proposal_window	The rate for the Gamma proposal density of the batch scale.
t_df_proposal_window	The rate for the Gamma proposal density of the cluster degrees of freedom.
R	The number of iterations to run for.
thin	thinning factor for samples recorded.
concentration	Vector of concentrations for mixture weights (recommended to be symmetric).
m_scale	The scale hyperparameter for the batch shift prior distribution.
rho	The shape of the prior distribution for the batch scale.
theta	The scale of the prior distribution for the batch scale.
initial_mu	A $P \times K$ matrix of initial values for the class means.
initial_cov	A $P \times P \times K$ cube of initial values for the class covariance matrices.
initial_df	A $K$ vector of initial values for the class degrees of freedom.
initial_m	A $P \times B$ matrix of initial values for the batch shift effects.
initial_S	A $P \times B$ matrix of initial values for the batch scales.
mu_initialised	Bool indicating if the class means are initialised by the user. If “false“ then initial values are drawn from the prior distribution.

cov_initialised	Bool indicating if the class covariance matrices are initialised by the user. If “false“ then initial values are drawn from the prior distribution.
df_initialised	Bool indicating if the class degrees of freedom are initialised by the user. If “false“ then initial values are drawn from the prior distribution.
m_initialised	Bool indicating if the batch shift effects are initialised by the user. If “false“ then initial values are drawn from the prior distribution.
S_initialised	Bool indicating if the batch scales are initialised by the user. If “false“ then initial values are drawn from the prior distribution.

**Value**

Named list of the different quantities drawn by the sampler.

---

samplePriorLabels	<i>Sample prior labels</i>
-------------------	----------------------------

---

**Description**

Generate labels from the stick-breaking prior.

**Usage**

```
samplePriorLabels(alpha, K, N)
```

**Arguments**

alpha	The concentration parameter for the stick-breaking prior.
K	The number of components to include (the upper bound on the number of unique labels generated).
N	The number of labels to generate.

**Value**

A vector of labels.

**Examples**

```
initial_labels <- samplePriorLabels(1, 50, 100)
```

---

`sampleSemisupervisedMVN`*Sample semi-supervised MVN Mixture model*

---

**Description**

Performs MCMC sampling for a mixture model.

**Usage**

```
sampleSemisupervisedMVN(  
  X,  
  K,  
  B,  
  labels,  
  batch_vec,  
  fixed,  
  mu_proposal_window,  
  cov_proposal_window,  
  m_proposal_window,  
  S_proposal_window,  
  R,  
  thin,  
  concentration,  
  m_scale,  
  rho,  
  theta,  
  initial_mu,  
  initial_cov,  
  initial_m,  
  initial_S,  
  mu_initialised,  
  cov_initialised,  
  m_initialised,  
  S_initialised  
)
```

**Arguments**

X	The data matrix to perform clustering upon (items to cluster in rows).
K	The number of components to model (upper limit on the number of clusters found).
B	The number of batches to model.
labels	Vector item labels to initialise from.
batch_vec	Observed batch labels.

fixed	Binary vector of the items that are fixed in their initial label.
mu_proposal_window	The standard deviation for the Gaussian proposal density of the cluster means.
cov_proposal_window	The degrees of freedom for the Wishart proposal density of the cluster covariances.
m_proposal_window	The standard deviation for the Gaussian proposal density of the batch mean effects.
S_proposal_window	The rate for the Gamma proposal density of the batch scale.
R	The number of iterations to run for.
thin	thinning factor for samples recorded.
concentration	Vector of concentrations for mixture weights (recommended to be symmetric).
m_scale	The scale hyperparameter for the batch shift prior distribution.
rho	The shape of the prior distribution for the batch scale.
theta	The scale of the prior distribution for the batch scale.
initial_mu	A $P \times K$ matrix of initial values for the class means.
initial_cov	A $P \times P \times K$ cube of initial values for the class covariance matrices.
initial_m	A $P \times B$ matrix of initial values for the batch shift effects.
initial_S	A $P \times B$ matrix of initial values for the batch scales.
mu_initialised	Bool indicating if the class means are initialised by the user. If “false“ then initial values are drawn from the prior distribution.
cov_initialised	Bool indicating if the class covariance matrices are initialised by the user. If “false“ then initial values are drawn from the prior distribution.
m_initialised	Bool indicating if the batch shift effects are initialised by the user. If “false“ then initial values are drawn from the prior distribution.
S_initialised	Bool indicating if the batch scales are initialised by the user. If “false“ then initial values are drawn from the prior distribution.

### Value

Named list of the different quantities drawn by the sampler.

---

`sampleSemisupervisedMVT`*Sample semi-supervised MVT Mixture model*

---

**Description**

Performs MCMC sampling for a mixture model.

**Usage**

```
sampleSemisupervisedMVT(  
  X,  
  K,  
  B,  
  labels,  
  batch_vec,  
  fixed,  
  mu_proposal_window,  
  cov_proposal_window,  
  m_proposal_window,  
  S_proposal_window,  
  t_df_proposal_window,  
  R,  
  thin,  
  concentration,  
  m_scale,  
  rho,  
  theta,  
  initial_mu,  
  initial_cov,  
  initial_df,  
  initial_m,  
  initial_S,  
  mu_initialised,  
  cov_initialised,  
  df_initialised,  
  m_initialised,  
  S_initialised  
)
```

**Arguments**

X	The data matrix to perform clustering upon (items to cluster in rows).
K	The number of components to model (upper limit on the number of clusters found).
B	The number of batches to model.

labels	Vector item labels to initialise from.
batch_vec	Observed batch labels.
fixed	Binary vector of the items that are fixed in their initial label.
mu_proposal_window	The standard deviation for the Gaussian proposal density of the cluster means.
cov_proposal_window	The degrees of freedom for the Wishart proposal density of the cluster covariances.
m_proposal_window	The standard deviation for the Gaussian proposal density of the batch mean effects.
S_proposal_window	The rate for the Gamma proposal density of the batch scale.
t_df_proposal_window	The rate for the Gamma proposal density of the cluster degrees of freedom.
R	The number of iterations to run for.
thin	thinning factor for samples recorded.
concentration	Vector of concentrations for mixture weights (recommended to be symmetric).
m_scale	The scale hyperparameter for the batch shift prior distribution.
rho	The shape of the prior distribution for the batch scale.
theta	The scale of the prior distribution for the batch scale.
initial_mu	A $P \times K$ matrix of initial values for the class means.
initial_cov	A $P \times P \times K$ cube of initial values for the class covariance matrices.
initial_df	A $K$ vector of initial values for the class degrees of freedom.
initial_m	A $P \times B$ matrix of initial values for the batch shift effects.
initial_S	A $P \times B$ matrix of initial values for the batch scales.
mu_initialised	Bool indicating if the class means are initialised by the user. If “false“ then initial values are drawn from the prior distribution.
cov_initialised	Bool indicating if the class covariance matrices are initialised by the user. If “false“ then initial values are drawn from the prior distribution.
df_initialised	Bool indicating if the class degrees of freedom are initialised by the user. If “false“ then initial values are drawn from the prior distribution.
m_initialised	Bool indicating if the batch shift effects are initialised by the user. If “false“ then initial values are drawn from the prior distribution.
S_initialised	Bool indicating if the batch scales are initialised by the user. If “false“ then initial values are drawn from the prior distribution.

### Value

Named list of the different quantities drawn by the sampler.

---

wishartLogLikelihood *Wishart log-likelihood*

---

**Description**

Used in calculating model probability in Metropolis-Hastings algorithm when proposals are from the Wishart distribution.

**Usage**

```
wishartLogLikelihood(X, V, n, P)
```

**Arguments**

X	- matrix; the matrix to calculate the likelihood of.
V	- matrix; the scale of the Wishart distribution.
n	- double; the degrees of freedom for the Wishart distribution.
P	- unsigned integer; the dimension of X.

**Value**

the unnormalised log-likelihood of X in a Wishart with parameters V and n.

# Index

## \* package

- batchmix-package, 3
- batchmix, (batchmix-package), 3
- batchmix-package, 3
- BatchMixtureModel (batchmix-package), 3
- batchSemiSupervisedMixtureModel, 4
- calcAllocProb, 7
- checkDataGenerationInputs, 8
- checkProposalWindows, 10
- collectAcceptanceRates, 11
- continueChain, 12
- continueChains, 13
- createSimilarityMat, 15
- gammaLogLikelihood, 16
- generateBatchData, 16
- generateBatchDataMVT, 18
- generateBatchDataVaryingRepresentation, 19
- generateGroupIDsInSimulator, 21
- generateInitialLabels, 22
- getLikelihood, 23
- getSampledBatchScale, 24
- getSampledBatchShift, 25
- getSampledClusterMeans, 26
- invGammaLogLikelihood, 27
- invWishartLogLikelihood, 28
- plotAcceptanceRates, 28
- plotLikelihoods, 29
- plotSampledBatchMeans, 30
- plotSampledBatchScales, 31
- plotSampledClusterMeans, 32
- plotSampledParameter, 33
- predictClass, 34
- prepareInitialParameters, 35
- processMCMCChain, 36
- processMCMCChains, 38
- rStickBreakingPrior, 40
- runBatchMix, 40
- runMCMCChains, 44
- sampleMVN, 46
- sampleMVT, 48
- samplePriorLabels, 50
- sampleSemiSupervisedMVN, 51
- sampleSemiSupervisedMVT, 53
- wishartLogLikelihood, 55