Package ‘sams’

April 19, 2022

Type Package
Title Merge-Split Samplers for Conjugate Bayesian Nonparametric Models
Version 0.4.3
License GPL-3
Encoding UTF-8
LazyData true
RoxygenNote 7.1.2
Depends R (>= 3.5.0)
Imports graphics, stats
NeedsCompilation no
Author Spencer Newcomb [aut],
    David B. Dahl [ctb, cre] (<https://orcid.org/0000-0002-8173-1547>)
Maintainer David B. Dahl <dahl@stat.byu.edu>
Repository CRAN
Date/Publication 2022-04-19 19:20:02 UTC

R topics documented:

asCanonical ................................................................. 2
asClusterLabels .......................................................... 3
asSetPartition ............................................................. 3
clusterProportions ....................................................... 4
clusterTrace .............................................................. 5
asCanonical

Coerce a Vector of Cluster Labels to Canonical Form

Description

Coerce a Vector of Cluster Labels to Canonical Form

Usage

asCanonical(partition)

Arguments

partition A numeric vector representing a set partition of the integers 1, ..., n using cluster labels

Value

A numeric vector representing partition, but now in canonical form.
asClusterLabels

<table>
<thead>
<tr>
<th>Description</th>
<th>Coerce a Set Partition in List Structure to Numeric Vectors of Cluster Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>Usage</td>
<td>asClusterLabels(partition)</td>
</tr>
<tr>
<td>Arguments</td>
<td>partition A list representing a set partition of the integers 1, ..., n</td>
</tr>
<tr>
<td>Value</td>
<td>A numeric vector representing the set partition using cluster labels.</td>
</tr>
</tbody>
</table>

asSetPartition

<table>
<thead>
<tr>
<th>Description</th>
<th>Coerce a Set Partition as Numeric Vectors of Cluster Labels to a List Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Usage</td>
<td>asSetPartition(partition)</td>
</tr>
<tr>
<td>Arguments</td>
<td>partition A numeric vector representing a partition of the integers 1, ..., n using cluster labels</td>
</tr>
<tr>
<td>Value</td>
<td>The set partition in a list structure.</td>
</tr>
</tbody>
</table>
clusterProportions

*Compute the Proportion of Items in Each Cluster for All Partitions*

**Description**
Compute the Proportion of Items in Each Cluster for All Partitions

**Usage**

\[
\text{clusterProportions}(\text{partitions})
\]

**Arguments**

- **partitions**: A matrix, with each row representing a set partition of the integers \(1, ..., n\) as cluster labels

**Value**
A matrix whose columns represent the cumulative proportion of the data that correspond to that cluster.

**Examples**

```r
# Neal (2000) model and data
nealData <- c(-1.48, -1.40, -1.16, -1.08, -1.02, 0.14, 0.51, 0.53, 0.78)
mkLogPosteriorPredictiveDensity <- function(data = nealData, 
sigma2 = 0.1^2, 
mu0 = 0, 
sigma02 = 1) {

function(i, subset) {
    posteriorVariance <- 1 / ( 1/sigma02 + length(subset)/sigma2 )
    posteriorMean <- posteriorVariance * ( mu0/sigma02 + sum(data[subset])/sigma2 )
    posteriorPredictiveSD <- sqrt(posteriorVariance + sigma2)
    dnorm(data[i], posteriorMean, posteriorPredictiveSD, log=TRUE)
}

logPostPredict <- mkLogPosteriorPredictiveDensity()

nSamples <- 500L
partitions <- matrix(0, nrow=nSamples, ncol=length(nealData))
for (i in 2:nSamples) {
    partitions[i,] <- nealAlgorithm3(partitions[i-1,], logPostPredict, mass = 1.0, nUpdates = 2)
}
clusterProportions(partitions)
```
clusterTrace

Plot Traces of Cluster Sizes

Description
Plot Traces of Cluster Sizes

Usage
clusterTrace(
  partitions,
  plot.cols = rep("black", ncol(partitions)),
  plot.title = ""
)

Arguments
partitions A matrix, with each row a numeric vector cluster labels
plot.cols A character vector of valid color names, whose length represents the maximum
          number of stacked traces to be plotted
plot.title A character string to be used as the main title on the trace plot

Examples
# Neal (2000) model and data
nealData <- c(-1.48, -1.40, -1.16, -1.08, -1.02, 0.14, 0.51, 0.53, 0.78)
mkLogPosteriorPredictiveDensity <- function(data = nealData,
                                           sigma2 = 0.1^2,
                                           mu0 = 0,
                                           sigma02 = 1) {
  function(i, subset) {
    posteriorVariance <- 1 / ( 1/sigma02 + length(subset)/sigma2 )
    posteriorMean <- posteriorVariance * ( mu0/sigma02 + sum(data[subset])/sigma2 )
    posteriorPredictiveSD <- sqrt(posteriorVariance + sigma2)
    dnorm(data[i], posteriorMean, posteriorPredictiveSD, log=TRUE)
  }
}
logPostPredict <- mkLogPosteriorPredictiveDensity()

nSamples <- 500L
partitions <- matrix(0, nrow=nSamples, ncol=length(nealData))
for ( i in 2:nSamples ) {
  partitions[i,] <- nealAlgorithm3(partitions[i-1,, logPostPredict, mass = 1.0, nUpdates = 2)
}
clusterTrace(partitions, plot.title = "Neal (2000) Data")
**clusterWithItem**  
*Identify Which Cluster Contains a Given Item*

**Description**

Identify Which Cluster Contains a Given Item

**Usage**

```
clusterWithItem(i, partition)
```

**Arguments**

- **i**: Item index as an integer vector of length one
- **partition**: Set partition of the integers 1, ..., $n$ represented as either a numeric vector of cluster labels, or a list containing subsets of these integers

**Value**

A list consisting of

- **which**: An integer representing which cluster $i$ belongs to
- **cluster**: The subset of indices that correspond to the same cluster as $i$

---

**createNewCluster**  
*Create a New Cluster with Given Item*

**Description**

Create a New Cluster with Given Item

**Usage**

```
createNewCluster(i, partition)
```

**Arguments**

- **i**: Item index as an integer vector of length one
- **partition**: Set partition of the integers 1, ..., $n$ represented as either a numeric vector of cluster labels, or a list containing subsets of these integers

**Value**

Updated partition with a new cluster.
**dCRP**

*Compute Probability Mass of a Partition Under the Two Parameter Chinese Restaurant Process (CRP)*

**Description**

Compute Probability Mass of a Partition Under the Two Parameter Chinese Restaurant Process (CRP)

**Usage**

dCRP(partition, mass = 1, discount = 0, log = FALSE)

**Arguments**

- **partition**: A numeric vector of cluster labels, or a matrix whose rows are numeric vectors of cluster labels
- **mass**: A numeric value indicating the mass parameter in the CRP, which must be greater than the `-discount` argument
- **discount**: A numeric value on the interval [0,1), indicating the discount parameter of the two parameter CRP
- **log**: A logical value indicating whether results should be returned on the log scale

**Value**

A numeric vector of probabilities, or log probabilities if `log = TRUE`.

**Examples**

```r
partitions <- matrix(c(0,0,0,0,0,
                       0,0,0,1,1,
                       0,0,1,2,3,
                       0,1,2,3,4), ncol = 5, nrow = 5, byrow = TRUE)

# discount = 0 shows higher probability for lower quantity of components
dCRP(partitions, mass = 1, discount = 0, log = FALSE)

# discount = 0.5 shows higher probability for higher quantity of components
dCRP(partitions, mass = 1, discount = 0.5, log = FALSE)
```
getThetas  
Get theta Parameters from a Numeric Vector of Cluster Labels and Unique phi Values

Description
Get theta Parameters from a Numeric Vector of Cluster Labels and Unique phi Values

Usage
getThetas(partition, phi)

Arguments
  partition  A numeric vector representing a partition of the integers 1, ..., n using cluster labels
  phi  A list of unique model parameters whose length must equal the number of unique cluster labels in partition

Value
A numeric vector of model parameters $\theta_1, ..., \theta_n$.

isCanonical  
Check if a Vector of Cluster Labels is in Canonical Form

Description
Check if a Vector of Cluster Labels is in Canonical Form

Usage
isCanonical(partition)

Arguments
  partition  A numeric vector representing a partition of the integers 1, ..., n using cluster labels

Value
Logical, indicating whether partition is in canonical form.
**joinExistingCluster**  
*Join Item to an Existing Cluster*

**Description**
Join Item to an Existing Cluster

**Usage**
```
joinExistingCluster(i, join, partition)
```

**Arguments**
- `i`: Item index as an integer vector of length one
- `join`: Label or index of cluster that `i` must join
- `partition`: Set partition of the integers 1, ..., `n` represented as either a numeric vector of cluster labels, or a list containing subsets of these integers

**Value**
Updated partition.

---

**nClusters**  
*Count the Number of Clusters in a Set Partition*

**Description**
Count the Number of Clusters in a Set Partition

**Usage**
```
nClusters(partition)
```

**Arguments**
- `partition`: A numeric vector representing a partition of the integers 1, ..., `n` using cluster labels

**Value**
The number of clusters in the given set partition as a numeric vector of length one.

**Examples**
```
p <- c(0,1,1,2,3,2,4,4,2)
nClusters(p)
```
nealAlgorithm3  
Conjugate Gibbs Sampler for a Partition

Description

Algorithm 3 from Neal (2000) to update the state of a partition based on the "Chinese Restaurant Process" (CRP) prior and a user-supplied log posterior predictive density function, with additional functionality for the two parameter CRP prior.

Usage

```r
ealAlgorithm3(
  partition,
  logPosteriorPredictiveDensity = function(i, subset) 0,
  mass = 1,
  discount = 0,
  nUpdates = 1L
)
```

Arguments

- `partition` A numeric vector of cluster labels representing the current partition.
- `logPosteriorPredictiveDensity` A function taking an index `i` (as a numeric vector of length one) and a subset of integers `subset`, and returning the natural logarithm of \( p(y_i | y_{\text{subset}}) \), i.e., that item's contribution to the log integrated likelihood given a subset of the other items. The default value "turns off" the likelihood, resulting in prior simulation (rather than posterior simulation).
- `mass` A specification of the mass (concentration) parameter in the CRP prior. Must be greater than the `discount` argument.
- `discount` A numeric value on the interval \([0,1)\) corresponding to the discount parameter in the two parameter CRP prior. Set to zero for the usual, one parameter CRP prior.
- `nUpdates` An integer giving the number of Gibbs scans before returning. This has the effect of thinning the Markov chain.

Value

A numeric vector giving the updated partition encoded using cluster labels.

References

Examples

```r
nealData <- c(-1.48, -1.4, -1.16, -1.08, -1.02, 0.14, 0.51, 0.53, 0.78)
mkLogPosteriorPredictiveDensity <- function(data = nealData,

 sigma2 = 0.1^2,

 mu0 = 0,

 sigma02 = 1) {
 function(i, subset) {
 posteriorVariance <- 1 / ( 1/sigma02 + length(subset)/sigma2 )
 posteriorMean <- posteriorVariance * ( mu0/sigma02 + sum(data[subset])/sigma2 )
 posteriorPredictiveSD <- sqrt(posteriorVariance + sigma2)
 dnorm(data[i], posteriorMean, posteriorPredictiveSD, log=TRUE)
 }
 }
)

logPostPredict <- mkLogPosteriorPredictiveDensity()

nSamples <- 1000L
partitions <- matrix(0, nrow = nSamples, ncol = length(nealData))
for (i in 2:nSamples) {
    partitions[i,] <- nealAlgorithm3(partitions[i-1,], logPostPredict, mass = 1.0, nUpdates = 1)
}

# convergence and mixing diagnostics
nSubsets <- apply(partitions, 1, function(x) length(unique(x)))
mean(nSubsets)
sum(acf(nSubsets)$acf) - 1  # Autocorrelation time

entropy <- apply(partitions, 1, partitionEntropy)
plot.ts(entropy)
```

**p18_bern**

**Multivariate Independent Bernoulli Data (p = 18)**

Description

Multivariate categorical data, generated as five clusters of independent variables with different success probabilities.

Usage

- p18_bern_1
- p18_bern_2
- p18_bern_3
Format

An object of class `data.frame` with 100 rows and 18 columns.
An object of class `data.frame` with 100 rows and 18 columns.
An object of class `data.frame` with 100 rows and 18 columns.

---

**p18_corr_mvn**  
Correlated Multivariate Normal Data ($p = 18$)

Description

Multivariate continuous data, generated as five clusters with an AR(1) correlation structure determining the common covariance matrix for each observation.

Usage

p18_corr_mvn_1
p18_corr_mvn_2
p18_corr_mvn_3

Format

An object of class `matrix` (inherits from `array`) with 100 rows and 18 columns.
An object of class `matrix` (inherits from `array`) with 100 rows and 18 columns.
An object of class `matrix` (inherits from `array`) with 100 rows and 18 columns.

---

**p18_mvn**  
Independent Multivariate Normal Data ($p = 18$)

Description

Multivariate continuous data, generated as five clusters of independent variables with different means.

Usage

p18_mvn_1
p18_mvn_2
p18_mvn_3
**p6_bern**

**Format**

An object of class `matrix` (inherits from `array`) with 100 rows and 18 columns.

**Description**

Multivariate categorical data, generated as five clusters of independent variables with different success probabilities.

**Usage**

```
p6_bern_1
p6_bern_2
p6_bern_3
```

**Format**

An object of class `data.frame` with 100 rows and 6 columns.

**Description**

Multivariate categorical data, generated as five clusters of independent variables with different success probabilities.

**Usage**

```
p6_big_bern_1
p6_big_bern_2
p6_big_bern_3
```
Format

An object of class `data.frame` with 1000 rows and 6 columns.
An object of class `data.frame` with 1000 rows and 6 columns.
An object of class `data.frame` with 1000 rows and 6 columns.

| p6_mvn      | Independent Multivariate Normal Data ($p = 6$) |

Description

Multivariate continuous data, generated as five clusters of independent variables with different means.

Usage

```r
p6_mvn_1
p6_mvn_2
p6_mvn_3
```

Format

An object of class `matrix` (inherits from `array`) with 100 rows and 6 columns.
An object of class `matrix` (inherits from `array`) with 100 rows and 6 columns.
An object of class `matrix` (inherits from `array`) with 100 rows and 6 columns.

| partitionEntropy | Calculate the Entropy of a Set Partition |

Description

Calculate the Entropy of a Set Partition

Usage

```r
partitionEntropy(partition)
```

Arguments

| partition | A numeric vector representing a partition of the integers $1, ..., n$ using cluster labels |
Value

Calculated partition entropy as a numeric vector of length one

Examples

```r
p <- c(0,0,0,1,1,2) # n = 6, 3 unique clusters
partitionEntropy(p)
```

---

**poch**

Compute the Pochhammer Symbol (Rising Factorials) With Increment

**Description**

Compute the Pochhammer Symbol (Rising Factorials) With Increment

**Usage**

```r
poch(x, y = NULL, n = 1, log = FALSE)
```

**Arguments**

- **x**: Non-negative numeric value
- **y**: Non-negative real value representing increment parameter for Pochhammer function. If NULL, there is no increment (i.e. y=1).
- **n**: Non-negative integer representing subscript in Pochhammer symbol
- **log**: Logical value indicating whether to return results on log scale

**Value**

A numeric value indicating the result of Pochhammer function.

**Examples**

```r
# effect of increment parameter
poch(5, y = NULL, n = 3, log = FALSE)
poch(5, y = 1, n = 3, log = FALSE)
poch(5, y = 1:4, n = 3, log = FALSE)

# increment being NULL is equivalent to ratio of gamma functions
a <- 7
b <- 3
out1 <- poch(a, y = NULL, n = b, log = FALSE)
out2 <- gamma(a + b) / gamma(a)
```
Compute the Posterior Pairwise Similarity for All Pairs of Items

Description
Compute the Posterior Pairwise Similarity for All Pairs of Items

Usage
psm(partitions)

Arguments
partitions A matrix, with each row a numeric vector cluster labels

Value
A symmetric matrix of pairwise similarities based on the partitions given.

Examples

# Neal (2000) model and data
nealData <- c(-1.48, -1.40, -1.16, -1.08, -1.02, 0.14, 0.51, 0.53, 0.78)

mkLogPosteriorPredictiveDensity <- function(data = nealData,
                                         sigma2 = 0.1^2,
                                         mu0 = 0,
                                         sigma02 = 1) {
  function(i, subset) {
    posteriorVariance <- 1 / ( 1/sigma02 + length(subset)/sigma2 )
    posteriorMean <- posteriorVariance * ( mu0/sigma02 + sum(data[subset])/sigma2 )
    posteriorPredictiveSD <- sqrt(posteriorVariance + sigma2)
    dnorm(data[i], posteriorMean, posteriorPredictiveSD, log=TRUE)
  }
}

logPostPredict <- mkLogPosteriorPredictiveDensity()

nSamples <- 500L
partitions <- matrix(0, nrow=nSamples, ncol=length(nealData))
for ( i in 2:nSamples ) {
  partitions[i,] <- nealAlgorithm3(partitions[i-1,], logPostPredict, mass = 1.0, nUpdates = 2)
}

psm(partitions)
Merge-Split proposals for conjugate "Chinese Restaurant Process" (CRP) mixture models using sequentially-allocated elements. Allocation is performed with weights derived from a previously-calculated pairwise similarity matrix, and optionally complemented with "restricted Gibbs" scans as discussed in Jain & Neal (2004).

Usage

```r
psmMergeSplit(
  partition,
  psm,
  logPosteriorPredictiveDensity = function(i, subset) 0,
  t = 1,
  mass = 1,
  discount = 0,
  nUpdates = 1L,
  selectionWeights = NULL
)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>partition</td>
<td>A numeric vector of cluster labels representing the current partition.</td>
</tr>
<tr>
<td>psm</td>
<td>A matrix of previously-calculated pairwise similarity probabilities for each pair of data indices.</td>
</tr>
<tr>
<td>logPosteriorPredictiveDensity</td>
<td>A function taking an index i (as a numeric vector of length one) and a subset of integers subset, and returning the natural logarithm of ( p(y_i</td>
</tr>
<tr>
<td>t</td>
<td>A non-negative integer indicating the number of restricted Gibbs scans to perform for each merge/split proposal.</td>
</tr>
<tr>
<td>mass</td>
<td>A specification of the mass (concentration) parameter in the CRP prior. Must be greater than the <code>discount</code> argument.</td>
</tr>
<tr>
<td>discount</td>
<td>A numeric value on the interval ([0,1)) corresponding to the discount parameter in the two-parameter CRP prior.</td>
</tr>
<tr>
<td>nUpdates</td>
<td>An integer giving the number of merge-split proposals before returning. This has the effect of thinning the Markov chain.</td>
</tr>
<tr>
<td>selectionWeights</td>
<td>A matrix or data frame whose first two columns are the unique pairs of data indices, along with a column of weights representing how likely each pair is to be selected at the beginning of each merge-split update.</td>
</tr>
</tbody>
</table>
Value

**partition** A numeric vector giving the updated partition encoded using cluster labels.

**accept** The acceptance rate of the Metropolis-Hastings proposals, i.e. the number of accepted proposals divided by `nUpdates`.

References


Examples

```r
# Neal (2000) model and data
ealData <- c(-1.48, -1.40, -1.16, -1.08, -1.02, 0.14, 0.51, 0.53, 0.78)
mkLogPosteriorPredictiveDensity <- function(data = nealData, 
   sigma2 = 0.1^2, 
   mu0 = 0, 
   sigma02 = 1) {
   function(i, subset) {
     posteriorVariance <- 1 / (1/sigma02 + length(subset)/sigma2)
     posteriorMean <- posteriorVariance * (mu0/sigma02 + sum(data[subset])/sigma2)
     posteriorPredictiveSD <- sqrt(posteriorVariance + sigma2)
     dnorm(data[i], posteriorMean, posteriorPredictiveSD, log=TRUE)
   }
}

logPostPredict <- mkLogPosteriorPredictiveDensity()

nSamples <- 1100L
nBurn <- 100
partitions <- matrix(0, nrow=nSamples, ncol=length(nealData))

# initial draws to inform similarity matrix
for ( i in 2:nBurn ) {
  partitions[i,] <- nealAlgorithm3(partitions[i-1,],
    logPostPredict, 
    mass = 1,
    nUpdates = 1)
}

# Generate pairwise similarity matrix from initial draws
psm.mat <- psm(partitions[1:nBurn,])

accept <- 0
for ( i in (nBurn+1):nSamples ) {
  ms <- psmMergeSplit(partitions[i-1,],
    psm.mat, 
    logPostPredict, 
    t = 1, 
    mass = 1.0, 
  accept <- accept + ms
}
```

```r
nUpdates = 1)
    partitions[[i]] <- ms$partition
    accept <- accept + ms$accept
  }

accept / (nSamples - nBurn) # post burn-in M-H acceptance rate
nSubsets <- apply(partitions, 1, function(x) length(unique(x)))
mean(nSubsets)
sum(acf(nSubsets)$acf)-1  # Autocorrelation time

entropy <- apply(partitions, 1, partitionEntropy)
plot.ts(entropy)
```

---

**psmMergeSplit_base**  
**Base Functionality for the psmMergeSplit Function**

**Description**

Merge-split proposals for conjugate "Chinese Restaurant Process" (CRP) mixture models using sequentially-allocated elements. Allocation is performed with weights derived from a previously-calculated pairwise similarity matrix.

**Usage**

```r
psmMergeSplit_base(
  partition,
  psm,
  logPosteriorPredictiveDensity = function(i, subset) 0,
  mass = 1,
  discount = 0,
  nUpdates = 1L,
  selectionWeights = NULL
)
```

**Arguments**

- **partition**  
  A numeric vector of cluster labels representing the current partition.

- **psm**  
  A matrix of previously-calculated pairwise similarity probabilities for each pair of data indices.

- **logPosteriorPredictiveDensity**  
  A function taking an index \( i \) (as a numeric vector of length one) and a subset of integers \( \text{subset} \), and returning the natural logarithm of \( p(y_i | y_{\text{subset}}) \), i.e., that item's contribution to the log integrated likelihood given a subset of the other items. The default value "turns off" the likelihood, resulting in prior simulation (rather than posterior simulation).

- **mass**  
  A specification of the mass (concentration) parameter in the CRP prior. Must be greater than the ~discount argument.
restrictedGibbsMergeSplit

Description

Merge-split proposals for conjugate "Chinese Restaurant Process" (CRP) mixture models using restricted Gibbs scans from a uniformly random launch state, as presented in Jain & Neal (2004), with additional functionality for the two parameter CRP prior.

Usage

restrictedGibbsMergeSplit(
  partition,
  logPosteriorPredictiveDensity = function(i, subset) 0,
  t = 1,
  mass = 1,
  discount = 0,
  nUpdates = 1L,
  selectionWeights = NULL
)

Arguments

partition A numeric vector of cluster labels representing the current partition.
logPosteriorPredictiveDensity
A function taking an index \(i\) (as a numeric vector of length one) and a subset of integers \(\text{subset}\), and returning the natural logarithm of \(p(y_i|y_{\text{subset}})\), i.e., that item’s contribution to the log integrated likelihood given a subset of the other items. The default value “turns off” the likelihood, resulting in prior simulation (rather than posterior simulation).

t
A non-negative integer indicating the number of restricted Gibbs scans to perform for each merge/split proposal.

mass
A specification of the mass (concentration) parameter in the CRP prior. Must be greater than the discount argument.

discount
A numeric value on the interval \([0,1)\) corresponding to the discount parameter in the two parameter CRP prior.

nUpdates
An integer giving the number of merge-split proposals before returning. This has the effect of thinning the Markov chain.

selectionWeights
A matrix or data frame whose first two columns are the unique pairs of data indices, along with a column of weights representing how likely each pair is to be selected at the beginning of each merge-split update.

Value

partition
An integer vector giving the updated partition encoded using cluster labels.

accept
The acceptance rate of the Metropolis-Hastings proposals, i.e. the number accepted proposals divided by nUpdates.

References


Examples

```r
# Neal (2000) model and data
nealData <- c(-1.48, -1.40, -1.16, -1.08, -1.02, 0.14, 0.51, 0.53, 0.78)

mkLogPosteriorPredictiveDensity <- function(data = nealData,
                                         sigma2 = 0.1^2,
                                         mu0 = 0,
                                         sigma02 = 1) {
  posteriorVariance <- 1 / ( 1/sigma02 + length(subset)/sigma2 )
  posteriorMean <- posteriorVariance * ( mu0/sigma02 + sum(data[subset])/sigma2 )
  posteriorPredictiveSD <- sqrt(posteriorVariance + sigma2)
  dnorm(data[i], posteriorMean, posteriorPredictiveSD, log = TRUE)
}

logPostPredict <- mkLogPosteriorPredictiveDensity()
```
nSamples <- 1000L
partitions <- matrix(0, nrow = nSamples, ncol = length(nealData))
accept <- 0
for (i in 2:nSamples) {
  ms <- restrictedGibbsMergeSplit(partitions[i-1,],
                                 logPostPredict,
                                 t = 1,
                                 mass = 1.0,
                                 nUpdates = 2)
  partitions[i,] <- ms$partition
  accept <- accept + ms$accept
}

accept / nSamples # M-H acceptance rate

# convergence and mixing diagnostics
nSubsets <- apply(partitions, 1, function(x) length(unique(x)))
mean(nSubsets)
sum(acf(nSubsets)$acf)-1 # Autocorrelation time

entropy <- apply(partitions, 1, partitionEntropy)
plot.ts(entropy)

---

seqAllocatedMergeSplit

**Merge-split Sampling for a Partition Based on Sequential Allocation of Items**

**Description**

Merge-split proposals for conjugate "Chinese Restaurant Process" (CRP) mixture models using sequential allocation of items, as originally described in Dahl (2003), with additional functionality for the two parameter CRP prior, as well as complementing these allocations with restricted Gibbs scans such as those discussed in Jain & Neal (2004).

**Usage**

seqAllocatedMergeSplit(
  partition,
  logPosteriorPredictiveDensity = function(i, subset) 0,
  t = 1,
  mass = 1,
  discount = 0,
  nUpdates = 1L,
  selectionWeights = NULL
)
Arguments

partition  A numeric vector of cluster labels representing the current partition.

logPosteriorPredictiveDensity  
A function taking an index \( i \) (as a numeric vector of length one) and a subset of integers \( \text{subset} \), and returning the natural logarithm of \( p(y_i | y_{\text{subset}}) \), i.e., that item's contribution to the log integrated likelihood given a subset of the other items. The default value "turns off" the likelihood, resulting in prior simulation (rather than posterior simulation).

t  A non-negative integer indicating the number of restricted Gibbs scans to perform for each merge/split proposal.

mass  A specification of the mass (concentration) parameter in the CRP prior. Must be greater than the -discount argument.

discount  A numeric value on the interval \([0,1)\) corresponding to the discount parameter in the two parameter CRP prior.

nUpdates  An integer giving the number of merge-split proposals before returning. This has the effect of thinning the Markov chain.

selectionWeights  A matrix or data frame whose first two columns are the unique pairs of data indices, along with a column of weights representing how likely each pair is to be selected at the beginning of each merge-split update.

Value

partition  An integer vector giving the updated partition encoded using cluster labels.

accept  The acceptance rate of the Metropolis-Hastings proposals, i.e. the number accepted proposals divided by nUpdates.

References


Examples

```r
# Neal (2000) model and data
nealData <- c(-1.48, -1.40, -1.16, -1.08, -1.02, 0.14, 0.51, 0.53, 0.78)
mkLogPosteriorPredictiveDensity <- function(data = nealData, 
                                          sigma2 = 0.1^2, 
                                          mu0 = 0, 
                                          sigma02 = 1) {
  posteriorVariance <- 1 / (1/sigma02 + length(subset)/sigma2)
  posteriorMean <- posteriorVariance * (mu0/sigma02 + sum(data[subindex])/sigma2)
  posteriorPredictiveSD <- sqrt(posteriorVariance + sigma2)
  dnorm(data[i], posteriorMean, posteriorPredictiveSD, log=TRUE)
}
logPostPredict <- mkLogPosteriorPredictiveDensity()

nSamples <- 1000L
partitions <- matrix(0, nrow = nSamples, ncol = length(nealData))
accept <- 0
for (i in 2:nSamples) {
  ms <- seqAllocatedMergeSplit(partitions[[i-1]],
                              logPostPredict,
                              t = 1,
                              mass = 1.0,
                              nUpdates = 2)
  partitions[[i]] <- ms$partition
  accept <- accept + ms$accept
}

accept / nSamples # M-H acceptance rate

# convergence and mixing diagnostics
nSubsets <- apply(partitions, 1, function(x) length(unique(x)))
mean(nSubsets)
sum(acf(nSubsets)$acf)-1 # Autocorrelation time

entropy <- apply(partitions, 1, partitionEntropy)
plot.ts(entropy)

seqAllocatedMergeSplit_base

---

**Description**

Merge-split proposals for conjugate "Chinese Restaurant Process" (CRP) mixture models using
sequential allocation of items, as originally described in Dahl (2003), with additional functionality
for the two parameter CRP prior.

**Usage**

```r
seqAllocatedMergeSplit_base(
  partition,
  logPosteriorPredictiveDensity = function(i, subset) 0,
  mass = 1,
  discount = 0,
  nUpdates = 1L,
  selectionWeights = NULL
)
```
simpleMergeSplit

Arguments

- **partition** A numeric vector of cluster labels representing the current partition.
- **logPosteriorPredictiveDensity** A function taking an index $i$ (as a numeric vector of length one) and a subset of integers $\text{subset}$, and returning the natural logarithm of $p(y_i|y_{\text{subset}})$, i.e., that item’s contribution to the log integrated likelihood given a subset of the other items. The default value “turns off” the likelihood, resulting in prior simulation (rather than posterior simulation).
- **mass** A specification of the mass (concentration) parameter in the CRP prior. Must be greater than the -discount argument.
- **discount** A numeric value on the interval [0,1) corresponding to the discount parameter in the two-parameter CRP prior.
- **nUpdates** An integer giving the number of merge-split proposals before returning. This has the effect of thinning the Markov chain.
- **selectionWeights** A matrix or data frame whose first two columns are the unique pairs of data indices, along with a column of weights representing how likely each pair is to be selected at the beginning of each merge-split update.

Value

- **partition** An integer vector giving the updated partition encoded using cluster labels.
- **accept** The acceptance rate of the Metropolis-Hastings proposals, i.e. the number accepted proposals divided by nUpdates.

References


See Also

seqAllocatedMergeSplit

---

**Description**

Merge-split proposals for conjugate "Chinese Restaurant Process" (CRP) mixture models using uniformly random allocation of items, as presented in Jain & Neal (2004), with additional functionality for the two parameter CRP prior.
simpleMergeSplit

Usage

```r
simpleMergeSplit(
  partition,
  logPosteriorPredictiveDensity = function(i, subset) 0,
  mass = 1,
  discount = 0,
  nUpdates = 1L,
  selectionWeights = NULL
)
```

Arguments

- **partition**: A numeric vector of cluster labels representing the current partition.
- **logPosteriorPredictiveDensity**: A function taking an index `i` (as a numeric vector of length one) and a subset of integers `subset`, and returning the natural logarithm of $p(y_i|y_{\text{subset}})$, i.e., that item’s contribution to the log integrated likelihood given a subset of the other items. The default value "turns off" the likelihood, resulting in prior simulation (rather than posterior simulation).
- **mass**: A specification of the mass (concentration) parameter in the CRP prior. Must be greater than the `discount` argument.
- **discount**: A numeric value on the interval [0,1) corresponding to the discount parameter in the two parameter CRP prior.
- **nUpdates**: An integer giving the number of merge-split proposals before returning. This has the effect of thinning the Markov chain.
- **selectionWeights**: A matrix or data frame whose first two columns are the unique pairs of data indices, along with a column of weights representing how likely each pair is to be selected at the beginning of each merge-split update.

Value

- **partition**: An integer vector giving the updated partition encoded using cluster labels.
- **accept**: The acceptance rate of the Metropolis-Hastings proposals, i.e. the number accepted proposals divided by `nUpdates`.

References

sizeOfLargestCluster  
**Calculate the Number of Items in the Largest Cluster of a Set Partition**

**Description**

Calculate the Number of Items in the Largest Cluster of a Set Partition

**Usage**

```r
sizeOfLargestCluster(partition)
```

**Arguments**

- `partition`  
  A numeric vector representing a partition of the integers 1, ..., n using cluster labels

**Value**

The number of items in the largest cluster of the given partition as a numeric vector of length one.

**Examples**

```r
p <- c(0,1,1,1,1,1,2)
sizeOfLargestCluster(p)
```

---

transformedWeights  
**Enumerate Transformed Weights for Choosing i and j Non-Uniformly**

**Description**

Enumerate Transformed Weights for Choosing i and j Non-Uniformly

**Usage**

```r
transformedWeights(m, fn = function(x) x, eps = 1e-12)
```

**Arguments**

- `m`  
  A square matrix of pairwise similarities between items

- `fn`  
  A function that maps pairwise similarities. Default is the identity function.

- `eps`  
  A numeric value close to 0 to give some nonzero weight to pairs of items with 0 or 1 pairwise similarity
Index

* datasets
  p18_bern, 11
  p18_corr_mvn, 12
  p18_mvn, 12
  p6_bern, 13
  p6_big_bern, 13
  p6_mvn, 14

asCanonical, 2
asClusterLabels, 3
asSetPartition, 3

clusterProportions, 4
cclusterTrace, 5
cclusterWithItem, 6
ccreateNewCluster, 6
dCRP, 7
ggetThetas, 8
isCanonical, 8
joinExistingCluster, 9

nClusters, 9
nealAlgorithm3, 10

p18_bern, 11
p18_bern_1 (p18_bern), 11
p18_bern_2 (p18_bern), 11
p18_bern_3 (p18_bern), 11
p18_corr_mvn, 12
p18_corr_mvn_1 (p18_corr_mvn), 12
p18_corr_mvn_2 (p18_corr_mvn), 12
p18_corr_mvn_3 (p18_corr_mvn), 12
p18_mvn, 12
p18_mvn_1 (p18_mvn), 12
p18_mvn_2 (p18_mvn), 12
p18_mvn_3 (p18_mvn), 12
p6_bern, 13

p6_bern_1 (p6_bern), 13
p6_bern_2 (p6_bern), 13
p6_bern_3 (p6_bern), 13
p6_big_bern, 13
p6_big_bern_1 (p6_big_bern), 13
p6_big_bern_2 (p6_big_bern), 13
p6_big_bern_3 (p6_big_bern), 13
p6_mvn, 14
p6_mvn_1 (p6_mvn), 14
p6_mvn_2 (p6_mvn), 14
p6_mvn_3 (p6_mvn), 14
partitionEntropy, 14
poch, 15
psm, 16
psmMergeSplit, 17, 20
psmMergeSplit_base, 19

restrictedGibbsMergeSplit, 20
seqAllocatedMergeSplit, 22, 25
seqAllocatedMergeSplit_base, 24
simpleMergeSplit, 25
sizeOfLargestCluster, 27

transformedWeights, 27