Package ‘shallot’

July 3, 2019

Type Package
Title Random Partition Distribution Indexed by Pairwise Information
Version 0.4.6
Date 2019-07-02
Description Implementations are provided for the models described in the paper D. B. Dahl, R. Day, J. Tsai (2017) <DOI:10.1080/01621459.2016.1165103>. The Ewens, Ewens-Pitman, Ewens attraction, Ewens-Pitman attraction, and ddCRP distributions are available for prior and posterior simulation. Posterior simulation is based on a user-supplied likelihood. Supporting functions for partition estimation and plotting are also provided.

URL https://github.com/dbdahl/shallot
BugReports https://github.com/dbdahl/shallot/issues
Imports rscala (>= 3.2.14), sdols (>= 1.7.5)
License Apache License 2.0 | file LICENSE
RoxygenNote 6.1.1
Encoding UTF-8
NeedsCompilation no
Author David B. Dahl [aut, cre]
Maintainer David B. Dahl <dahl@stat.byu.edu>
Repository CRAN
Date/Publication 2019-07-03 07:50:18 UTC

R topics documented:

shallot-package .................................................. 2
adj.rand.index .................................................. 3
association.matrix .............................................. 4
attraction ....................................................... 5
decay.reciprocal ............................................... 6
default.mass .................................................... 7
describe.partitions ............................................. 8
enumerate.partitions ........................................... 8
shallot-package

Random Partition Distribution Indexed by Pairwise Information

Description


Author(s)

David B. Dahl <dahl@stat.byu.edu>

References


See Also

ewens.pitman.attraction, sample.partitions, estimate.partition
Examples

data <- iris[, -ncol(iris)]
truth <- as.integer(iris[, ncol(iris)])
distance <- as.dist(as.matrix(dist(scale(data)) + 0.001))

decay <- decay.exponential(temperature(9.0, fixed = TRUE), distance)
permutation <- permutation(n.items = nrow(data), fixed = FALSE)
attraction <- attraction(permutation, decay)
mass <- mass(1.0, fixed = TRUE)
discount <- discount(0.2, fixed = TRUE)
distribution <- ewens.pitman.attraction(mass, discount, attraction)

raw <- sample.partitions(distribution, 500, parallel = FALSE)
samples <- process.samples(raw)

library(sdols)
pp <- expectedPairwiseAllocationMatrix(samples$labels)
est <- salso(pp)
conf <- confidence(est, pp)
plot(conf)
plot(conf, data = data)

adj.rand.index

Adjusted Rand Index

Description

This function calculates the adjusted Rand index between two clusterings/partitions.

Usage

adj.rand.index(c1, c2)

Arguments

c1 A vector containing cluster labels for a clustering/partition.

c2 A vector containing cluster labels for a clustering/partition.

Details

The adj.rand.index function takes as its input two clusterings/partitions in cluster label notation and computes the adjusted Rand index. The adjusted Rand index is at most 1.0 and large numbers indicate high similarity.
Value
A numeric vector of length one representing the adjusted Rand index between the two clusterings/partitions.

Author(s)
David B. Dahl <dahl@stat.byu.edu>

References

See Also
*estimate.partition*

Examples

```r
truth <- c(1,1,2,2,1,3,3,3)
estimate <- c(1,2,2,2,1,2,3,3)
adj.rand.index(truth, estimate)
```

### Description

This function creates an association matrix for a clustering/partition. The \((i,j)\) element of the matrix is 1 if item \(i\) and \(j\) are in the same cluster/subset and 0 otherwise.

### Usage

```r
association.matrix(cl)
```

### Arguments

c1
A vector containing cluster labels for a clustering/partition.

### Value

A matrix of 0s and 1s indicating whether items \(i\) and \(j\) are in the same cluster/subset.

### Examples

```r
cl <- rep(1:3, times=c(2,4,3))
association.matrix(cl)
```
**attraction**

---

## Description

This function creates an attraction from a permutation and a decay in preparation for use in the `ewens.attraction`, `ewens.pitman.attraction`, and `ddcrp` functions. For details on each of these arguments, please see the links below.

## Usage

```r
attraction(permutation, decay)
```

```r
## S3 method for class 'shallot.attraction'
print(x, ...)
```

```r
## S3 method for class 'shallot.attraction'
as.matrix(x, ...)
```

## Arguments

- **permutation**: An object of class `shallot.permutation` encoding the permutation of the items.
- **decay**: An object of class `shallot.decay` detailing the transformation from distances to attractions.
- **x**: An object of class `shallot.attraction`.
- **...**: Currently ignored.

## Value

An object of class `shallot.attraction`.

## Author(s)

David B. Dahl &lt;dahl@stat.byu.edu&gt;

## References


## See Also

`ddcrp`, `decay`, `ewens.attraction`, `ewens.pitman.attraction`, `permutation`
decay.reciprocal

Examples

```r
c permutation <- permutation(n.items=50, fixed=FALSE)
decay <- decay.exponential(temperature(1.0), dist(scale(USArrests)))
attraction(permutation, decay)
```

---

decay.reciprocal  Decay Functions

description

These functions specify the decay to map distances to attractions.

Usage

- decay.reciprocal(temperature, distance)
- decay.exponential(temperature, distance)
- decay.subtraction(temperature, distance, multiplier = 1.01)

```r
## S3 method for class 'shallot.decay'
print(x, ...)
```

Arguments

- `temperature`: An object of class `shallot.temperature`.
- `distance`: An object of class `dist`.
- `multiplier`: An scalar greater than 1.0 to ensure that attractions from `decay.subtraction` are finite.
- `x`: An object of class `shallot.decay`.
- `...`: Currently ignored.

Details

There are currently three choices for decay functions: reciprocal, exponential, and subtraction.

The reciprocal decay maps a distance $d$ to an attraction $a$ as follows: $a = 1/d^t$, where $t$ is the temperature.

The exponential decay maps a distance $d$ to an attraction $a$ as follows: $a = \exp(-t*d)$, where $t$ is the temperature.

The subtract decay maps a distance $d$ to an attraction $a$ as follows: $a = (m-d)^t$, where $t$ is the temperature and $m$ is the maximum distance in distance multiplied by the supplied `multiplier`. 
Author(s)

David B. Dahl <dahl@stat.byu.edu>

References


See Also

dist, temperature, attraction

Examples

```r
temp <- temperature(1:0)
distance <- dist(scale(USArrests))
decay1 <- decay.reciprocal(temp, distance)
decay2 <- decay.exponential(temp, distance)
decay3 <- decay.subtraction(temp, distance)
```

---

**default.mass**

*Default Mass Selection*

**Description**

This function selects an optimal mass value for Cluster Analysis via Random Partition Distributions, using the Ewens-Pitman Attraction distribution.

**Usage**

```r
default.mass(mass, list.epam, dis, new.draws = TRUE, w = c(1, 1, 1),
  discount = 0, temp = 10, loss = "binder", n.draws = 100L,
  two.stage = TRUE, parallel = TRUE)
```

```r
## S3 method for class 'shallot.default.mass'
print(x, ...)
```

**Arguments**

- **mass**: optional, a vector of mass values.
- **list.epam**: optional, a list of expected pairwise allocation matrices. Each matrix in the list needs the attributes "mass" and "n.draws".
- **dis**: a dissimilarity structure of class dist.
- **new.draws**: logical; if TRUE then new draws are obtained at each mass value.
- **w**: a vector of length 3 of the weights to be used in the *mass.algorithm*. 
enumerate.partitions

\text{discount} \quad \text{parameter of the Ewens-Pitman Attraction distribution.}
\text{temp} \quad \text{temperature parameter of the Ewens-Pitman Attraction distribution.}
\text{loss} \quad \text{One of "\text{squareError}, "\text{absoluteError}, "\text{binder}, or "\text{lowerBoundVariationOfInformation}" to indicate the optimization should seek to minimize squared error loss, absolute error loss, Binder loss (Binder 1978), or the lower bound of the variation of information loss (Wade & Ghahramani 2017), respectively.}
\text{n.draws} \quad \text{number of draws of partitions to be obtained at each mass value.}
\text{two.stage} \quad \text{logical; if \text{TRUE}, the two stage algorithm is implemented in \text{mass.algorithm}.}
\text{parallel} \quad \text{logical; if \text{TRUE} computations will take advantage multiple CPU cores.}
\text{x} \quad \text{An object from the \text{default.mass} function.}
\ldots \quad \text{currently ignored}

Details

The function draws \text{n.draws} partitions at each specified mass value. If a vector of mass values is not given, then the default of seq(0.1, 10, 0.2) is used for loss \text{lowerBoundVariationOfInformation} and seq(0.1, 5, 0.05) used for the other loss functions.

If a list of expected pairwise allocation matrices (EPAM) is provided, additional draws at matching mass values are added to the corresponding matrix. Additionally, no new draws are needed for estimation, if a list of EPAMs is provided.

A partition/clustering estimate from each EPAM is obtained using the SALSO method in \text{salso}. The estimate given minimizes the specified loss function with respect to the EPAM.

The function then uses the \text{mass.algorithm} to select the optimal mass value for clustering estimation.

Value

An object of class \text{shallot.default.mass}. This object is a list containing a matrix of ‘best’ possible mass values to maximize partition confidence and minimize the variance ratio, the clustering estimate, the expected pairwise allocation matrix, parameters used for optimization and the EPA distribution, and the list of expected pairwise allocation matrices for each mass value.

See Also

Other Default Mass Selection: \text{mass.algorithm, partition.confidence, variance.ratio}

\begin{verbatim}
enumerate.partitions  Enumerate Partitions
\end{verbatim}

Description

This function enumerates all possible partitions for a given number of items.
Usage

enumerate.partitions(n.items)

Arguments

n.items An integer given then number of items to partition.

Details

This function returns an enumeration of the partition of n.items items.

Value

A matrix of cluster labels in which each row represents a clusterings.

Author(s)

David B. Dahl <dahl@stat.byu.edu>

See Also

process.samples

Examples

## Not run:
example(shallot)

## End(Not run)

estimate.partition Estimate Partition

Description

This function returns a partition that summarizes the partition distribution using the least-square clustering method (Dahl 2006), with extensions to perform greedy optimization and limit the number of subsets.

Usage

estimate.partition(x, pairwise.probabilities = NULL, max.subsets = 0,
max.scans = 0, parallel = TRUE)
Arguments

- **x**  
  An object from the `sample.partitions` function.

- **pairwise.probabilities**  
  An object of class `shallot.pairwiseProbability` obtained from `pairwise.probabilities`. If not supplied, it will be computed from `x`.

- **max.subsets**  
  An integer limiting the number of subsets. Defaults to 0, which does not impose a constraint on the number of subsets.

- **max.scans**  
  An integer controlling the greedy search. Defaults to 0, which disables the greedy search.

- **parallel**  
  Should all of the CPU cores should be used? Defaults to `TRUE`.

Value

A partition as a vector of cluster labels.

Author(s)

David B. Dahl <dahl@stat.byu.edu>

References


See Also

- `sample.partitions`, `process.samples`, `plot.partition`, `adj.rand.index`

Examples

```r
## Not run:
exmaple(shallot)

## End(Not run)
```
Description

These functions specify the Ewens, Ewens-Pitman, Ewens attraction, Ewens-Pitman attraction, and ddCRP distributions which would then be used in the `sample.partitions` function.

Usage

ewens(mass, n.items, names = paste0("c", 1:n.items))

## S3 method for class 'shallot.distribution.ewens'
print(x, ...)

ewens.pitman(mass, discount, n.items, names = paste0("c", 1:n.items))

## S3 method for class 'shallot.distribution.ewensPitman'
print(x, ...)

ewens.attraction(mass, attraction)

## S3 method for class 'shallot.distribution.ewensAttraction'
print(x, ...)

ewens.pitman.attraction(mass, discount, attraction)

## S3 method for class 'shallot.distribution.ewensPitmanAttraction'
print(x, ...)

ddcrp(mass, attraction)

## S3 method for class 'shallot.distribution.ddcrp'
print(x, ...)

Arguments

- **mass**: An object of class `shallot.mass`.
- **n.items**: An integer containing the number of items to partition.
- **names**: A character vector containing the names of the items. The default names are of the form “c1”, “c2”, etc.
- **x**: An object of class `shallot.distribution`.
- **...**: Currently ignored.
- **discount**: An object of class `shallot.discount`.
- **attraction**: An object of class `shallot.attraction`. 
Value

An object of class shallot.distribution.

Author(s)

David B. Dahl <dahl@stat.byu.edu>

References


See Also

mass, discount, attraction, sample.partitions

Examples

```r
pd1 <- ewens(mass(1),50)

decay <- decay.exponential(temperature(1.0),dist(scale(USArests)))
attraction <- attraction(permutation(n.items=50, fixed=FALSE), decay)

pd2 <- ewens.pitman.attraction(mass(1), discount(0.05), attraction)

pd3 <- ddcrap(mass(1), attraction)
```

### mass

*Mass, Discount, and Temperature Parameters*

Description

These functions set the mass, discount, and temperature parameters and, in the case of them being random, specify the parameters of their distribution.

Usage

```r
mass(..., fixed = TRUE)
```

## S3 method for class 'shallot.mass'

print(x, ...)

discount(..., fixed = TRUE)

## S3 method for class 'shallot.discount'

print(x, ...)
mass

\[
\text{temperature}(\ldots, \text{fixed} = \text{TRUE})
\]

## S3 method for class 'shallot.temperature'
print(x, ...)  

**Arguments**

... A number greater than 0.0 representing the value of the mass, discount, or temperature parameters. Or, in the case of them being random, a vector of two numbers representing either: i. the shape and rate parameters of the gamma distribution for the mass or temperature, or ii. the shape parameters of the beta distribution for the discount. This argument is currently ignored for the associated print functions.

fixed If TRUE, the parameter is fixed. If FALSE, the parameter value is samples from either: i. a gamma distribution for the mass or temperature, or ii. a beta distribution for the discount.

x An object from the `mass`, `discount`, or `temperature` functions.

**Details**

If no parameters are specified, the mass parameter defaults to 1.2, the discount parameter defaults to 0.05, the temperature parameter defaults to 3.0. If the mass parameter is random, the default shape and rate parameters of the gamma distribution are 2.5 and 2, respectively. If the discount parameter is random, the default shape parameters of the beta distribution are 1.0 and 1.0. If the temperature parameter is random, the default shape and rate parameters of the gamma distribution are 2 and 0.5, respectively.

**Value**

An object of class `shallot.mass`, `shallot.discount`, or `shallot.temperature`.

**Author(s)**

David B. Dahl <dahl@stat.byu.edu>

**Examples**

mass()
mass(1.0)
mass(1.4, fixed=FALSE)
mass(0.5, 1, fixed=FALSE)
discount()
discount(0.2)
discount(1, 3, fixed=FALSE)
temperature()
temperature(2)
temperature(2, 4, fixed=FALSE)
mass.algorithm  

**Mass Selection Algorithm**

**Description**

This function selects the optimal mass value for Cluster Analysis via Random Partition distributions using the Ewens-Pitman attraction distribution.

**Usage**

```r
mass.algorithm(mass, pc, vr, n, w = c(1, 1, 1), two.stage = TRUE)
```

**Arguments**

- `mass`: a vector of mass values
- `pc`: a vector of partition confidences for the partition estimates at the corresponding mass values
- `vr`: a vector of variance ratios for the partition estimates at the corresponding mass values
- `n`: a vector of the number of subsets in the partition estimates at the corresponding mass values
- `w`: a vector of length 3 specifying the weights of `pc`, `vr`, and `n`
- `two.stage`: logical; if `TRUE`, the two stage algorithm is implemented

**Details**

The `mass.algorithm` function is used internally in the `default.mass` function.

The default value for `w` is `c(1, 1, 1).`

The general algorithm is as follows:

1. Rank the partition confidence (`pc`) and variance ratio (`vr`). Select the `mass_i` value which minimizes the weighted sum of \( w_1pc_i + w_2vr_i + w_3n_i \).

The two stage algorithm proceeds as follows:

1. Rank the partition confidence (`pc`) and variance ratio (`vr`). For each number of clusters `n` select the index which minimizes the weighted sum of \( w_1pc_i + w_2vr_i \).

2. Rerank the `pc` and `vr` of the selected indices and select the `mass_i` value which minimizes the weighted sum of \( w_1pc_i + w_2vr_i + w_3n_i \) from among the selected indices.

**Value**

A matrix containing the ‘best’ mass value and corresponding values for `pc`, `vr`, and `n`. The matrix also contains the mass values for the partitions estimate with more one more and one less subset that the selected mass value.
See Also

Other Default Mass Selection: `default.mass.partition.confidence`, `variance.ratio`

### nsubsets.random

**Number of Subsets**

**Description**

These functions either sample the number of subsets for supported partition distributions or computes probabilities, means, and variances of these distributions.

**Usage**

```r
nsubsets.random(x, n.samples)
nsubsets.probability(x, n.subsets)
nsubsets.average(x)
nsubsets.variance(x)
```

**Arguments**

- `x`: An object of class `shallot.distribution`.
- `n.samples`: An integer containing the number of samples.
- `n.subsets`: An integer containing the number of subsets.
- `...`: Currently ignored.

**Value**

The `nsubsets.random` function returns a vector of random samples of the number of subsets in the distribution `x`.

The `nsubsets.probability` function returns the probability that the number of subsets is `n.subsets` in the distribution `x`. Depending on the number of items and the value of `n.subsets`, this function can be computationally intensive.

The `nsubsets.average` and `nsubsets.variance` functions return the mean and variances, respectively, of the number of subsets in the distribution `x`.

**Author(s)**

David B. Dahl <dahl@stat.byu.edu>

**References**

See Also

\text{partition.distribution}

Examples

```r
pd <- ewens.pitman.attraction(
  mass(1),
  discount(0.05),
  attraction(permutation(n.items=50, fixed=FALSE),
             decay.exponential(temperature(1.0), dist(scale(USArrests)))))
mean(nsubsets.random(pd,1000))
nsubsets.average(pd)

pde <- ewens(mass(1),50)
nsubsets.variance(pde)
nsubsets.probability(pde,4)
```
Value

The `pairwise.probabilities` function returns an object of class `shallot.pairwiseProbability`. The `as.matrix` function returns a square matrix.

Author(s)

David B. Dahl <dahl@stat.byu.edu>

See Also

`sample.partitions`, `process.samples`, `estimate.partition`

Examples

```r
## Not run:
exampleshallot)
## End(Not run)
```

---

**partition.confidence**

**Partition Confidence**

Description

This function calculates the partition confidence of a partition estimate from the corresponding expected pairwise allocation matrix (EPAM).

Usage

```r
partition.confidence(x, y)
```

Arguments

- `x`: If `y` is not specified then `x` must be an object of class `sdols.confidence`. Otherwise, `x` is a vector of cluster labels and `y` is an expected pairwise allocation matrix.
- `y`: If `y` is not specified then `x` must be an object of class `sdols.confidence`. Otherwise, `x` is a vector of cluster labels and `y` is an expected pairwise allocation matrix.

Details

The `partition.confidence` takes as input an object of class `sdols.confidence` and then calculates the partition confidence from the expected pairwise allocation matrix.

The partition confidence is the average values of the EPAM for items that are clustered together. Items which are in their own subset do not contribute to partition confidence.
partition.pmf

See Also

Other Default Mass Selection: default.mass, mass.algorithm, variance.ratio

Examples

```r
x <- rep(c(1,2,3), times=c(2,3,5))
y <- diag(10)
y[upper.tri(y)] <- runif(45)
partition.confidence(x,y)
```

partition.pmf Obtain the Probability Mass Function of a Partition Distribution

Description

This function returns the probability mass function (pmf) of a partition distribution.

Usage

```r
partition.pmf(x)
```

Arguments

x An object of class shallot.distribution obtained, for example, from the
  `ewens.pitman.attraction` function.

Value

A function that takes a partition (as a vector in cluster label notation) and returns the probability — or, if log=TRUE, the log of the probability — of the supplied partition.

Author(s)

David B. Dahl <dahl@stat.byu.edu>

Examples

```r
## Not run:
example(shallot)
## End(Not run)
```
permutation

permutation

Description

These function define a permutation for subsequent use.

Usage

permutation(..., n.items = NULL, fixed = TRUE)

## S3 method for class 'shallot.permutation'
print(x, ...)

Arguments

... For the function permutation, a permutation of the integers 1, 2,... n, where n
is the length of the vector. For the function print.shallot.permutation, this
is ignored.

n.items An optional argument provided instead of ... to request a random partition.
The argument fixed must be FALSE.

fixed Should the permutation be fixed?

x An object of class shallot.permutation.

Details

A valid permutation of length n is an integer vector of length n containing each integer 1, 2,... n
only once.

Value

An object of class shallot.permutation.

Author(s)

David B. Dahl <dahl@stat.byu.edu>

References

Dahl, D. B., Day, R., and Tsai, J. (2017), Random Partition Distribution Indexed by Pairwise Infor-

See Also

attraction
Examples

```r
## Demonstrate permutation.
permutation(c(3, 1, 2, 5, 4))
permutation(c(3, 1, 2, 5, 4), fixed=FALSE)
permutation(n.items=5, fixed=FALSE)
```

Description

This function extracts the partitions from the results of the `sample.partitions` function.

Usage

```r
process.samples(x)
```

Arguments

- `x`: An object from the `sample.partitions` function.

Details

This function extracts the sampled partitions from the results of the `sample.partitions` function.

Value

A list containing a matrix of cluster labels in which each row represents a clusterings. The list also contains sampled model parameters if `sample.parameter` is not NULL.

Author(s)

David B. Dahl <dahl@stat.byu.edu>

See Also

- `sample.partitions`

Examples

```r
## Not run:
example(shallot)

## End(Not run)
```
Sample Partitions from Partition Distributions

Description

This function samples partitions from the Ewens, Ewens-Pitman, Ewens attraction, Ewens-Pitman attraction, and ddCRP distributions.

Usage

sample.partitions(x, n(draws), parallel = TRUE)

Arguments

x
  An object of class shallot.distribution obtained, for example, from the
  ewens.pitman.attraction function.
n(draws)
  An integer representing the desired number of samples. Due to parallelization,
  slightly more samples may be returned.
parallel
  Should sampling be done in parallel by simultaneously using all CPU cores?
... 
  Currently ignored.

Value

An object of class shallot.samples.raw which can be subsequently be used in process.samples,
  pairwise.probabilities, estimate.partition,

Note

If this function is interrupted by the user, the computation engine will be broken and subsequent
calls to package functions may fail until a new session is started.

Author(s)

David B. Dahl <dahl@stat.byu.edu>

See Also

partition.distribution, process.samples, pairwise.probabilities, estimate.partition
  sample.partitions.posterior

Examples

## Not run:
exaample(shallot)

## End(Not run)
sample.partitions.posterior

**Sample Partitions from Posterior Distribution of Partition**

**Description**

This function samples partitions from the posterior distribution of a partition based on a user-supplied likelihood and the following prior partition distributions: Ewens, Ewens-Pitman, Ewens attraction, Ewens-Pitman attraction, and ddCRP distributions.

**Usage**

```r
sample.partitions.posterior(partition, sampling.model, partition.model,
                            n.draws, massRWSD = 0.5, discountRWSD = 0.1,
                            k = min(length(partition), 25), temperatureRWSD = 0.5,
                            progress.bar = interactive())
```

**Arguments**

- `partition`: An object of class `shallot.distribution.data`
- `sampling.model`: An object of class `shallot.distribution.data` obtained from the `sampling.model` function.
- `partition.model`: An object of class `shallot.distribution` obtained, for example, from the `ewens.pitman.attraction` function.
- `n.draws`: An integer representing the desired number of samples.
- `massRWSD`: The standard deviation of the random walk proposal for updating the mass parameter.
- `discountRWSD`: The standard deviation of the random walk proposal for updating the discount parameter.
- `k`: The number of items to shuffle when proposing an update for the permutation.
- `temperatureRWSD`: The standard deviation of the random walk proposal for updating the temperature parameter.
- `progress.bar`: Should a progress bar be shown while sampling?

**Value**

An object of class `shallot.samples.raw` which can be subsequently be used in `process.samples`, `pairwise.probabilities`, `estimate.partition`,

**Note**

If this function is interrupted by the user, the computation engine will be broken and subsequent calls to package functions may fail until a new session is started.
sample.partitions.posterior

Author(s)

David B. Dahl <dahl@stat.byu.edu>

See Also

partition.distribution, process.samples, pairwise.probabilities, estimate.partition sample.partitions

Examples

```r
1+2

# don't test
#mass <- mass(1.0, fixed=TRUE)
discount <- discount(0.05, fixed=TRUE)
distance <- dist(scale(USArrests[1:9,]))
if ( min(distance[upper.tri(distance)], na.rm=TRUE) == 0 )
  stop("Oops, distances must be strictly positive."
#
#n.items <- attr(distance,"Size")
#permutation <- permutation(n.items=n.items, fixed=FALSE)
#temperature <- temperature(2, fixed=TRUE)
#attraction <- attraction(permutation, decay.exponential(temperature, distance))
#partition.distribution <- ewens.pitman.attraction(mass, discount, attraction)
#
### Model inputs.
data <- c(-1.48, -1.40, -1.16, -1.08, -1.02, 0.14, 0.51, 0.53, 0.78)
sigma <- 0.1
mu0 <- 0.0
sigma0 <- 1.0
#
### Derived values.
s2 <- sigma * sigma
s02 <- sigma0 * sigma0
s02Inv <- 1.0 / s02
c <- -1.0 / (2.0 * s2)
#
### Sampling model of Neal (JCGS, 2009)
### Function to perform an MCMC update of the parameter.
sample.parameter <- function(indices=scalaType("D1"), parameter=scalaType("D0")) {
  sum <- sum(data[indices])
  variance <- 1 / (s02Inv + length(indices) / s2)
  mean <- variance * (mu0 / s02 + sum / s2)
  rnorm(1, mean, sqrt(variance))
}
#
library(rscala)
s <- shallot:::
sample.parameter.compiled <- s(data=data, mu0=mu0, s2=s2, s02=s02, s02Inv=s02Inv) ^ sample.parameter
#
### Function to evaluate the likelihood contribution for an observation.
```
#log.density <- function(i=scalaType("D0"), indices=scalaType("D1"), parameter=scalaType("D0")) {
  # resid <- data[i] - parameter
  # c * resid * resid
  #
  #log.density.compiled <- s(data=resid, c=c) ^ log.density
  #
  #sampling.model <- sampling.model(sample.parameter, log.density)
  #
  ### Perform posterior sampling.
  #initial.partition <- rep(1, length(data))
  #n.draws <- 10
  #raw <- sample.partitions.posterior(initial.partition, sampling.model, partition.distribution,  
  #massRWSD=1, temperatureRWSD=1, n.draws)
  #samples.format1 <- process.samples(raw)
  #parameterMatrix <- t(sapply(seq_len(n.draws), function(i) {  
  # unlist(samples.format1$parameters[i][1])
  #}))
  
  # tail(samples.format1$hyperparameters)
  #
  ### Shrinkage to group means?
  #plot(data, apply(parameterMatrix, 2, mean))
  #abline(a=0, b=1)
  #
  #samples.format1$hyperparameters
  #
  ### Post processing to find the partition estimate.
  #library(sdols)
  #pp <- expectedPairwiseAllocationMatrix(samples.format1$labels)
  #est <- salso(pp)
  #plot(confidence(est, pp))
  #
  #rscala::scalaDisconnect(shallot::s)
  #}

---

### sampling.model

#### Specifying a Sampling Model

**Description**

These functions set the mass, discount, and temperature parameters and, in the case of them being random, specify the parameters of their distribution.

**Usage**

`sampling.model(sample.parameter, log.density)`
Arguments

- **sampleNparameter**
  A function taking two arguments with names indices and parameter having default values \(c()\) and NULL, respectively. With those default values, the function should return a sample from the centering distribution as an object of any type. Otherwise, indices is a vector of integers indicating the cluster elements and parameter is the current value of the parameter for the cluster, and the function should return an updated value for the cluster parameter based on a valid MCMC update.

- **logNdensity**
  A function taking three arguments with names \(i\), indices, and parameter and returning a double giving the log of the likelihood contribution of item \(i\) to a clustering with members indices and parameter parameter.

Value

An object of class `shallot.distribution.data`.

Author(s)

David B. Dahl <dahl@stat.byu.edu>

Examples

```r
## Model inputs.
data <- c(-1.48, -1.40, -1.16, -1.08, -1.02, 0.14, 0.51, 0.53, 0.78)
sigma <- 0.1
mu0 <- 0.0
sigma0 <- 1.0

## Derived values.
s2 <- sigma * sigma
s02 <- sigma0 * sigma0
s02Inv <- 1.0 / s02
c <- -1.0 / (2.0 * s2)

## Sampling model of Neal (JCGS, 2009)
## Function to perform an MCMC update of the parameter.
sampleNparameter <- function(indices=c(), parameter=NULL) {
  sum <- sum(data[indices])
  variance <- 1 / (s02Inv + length(indices) / s2)
  mean <- variance * (mu0 / s02 + sum / s2)
  rnorm(1, mean=mean, sd=sqrt(variance))
}

## Function to evaluate the likelihood contribution for an observation.
logNdensity <- function(i, indices, parameter) {
  resid <- data[i] - parameter
  c * resid * resid
}
```
variance.ratio

Describing the variance of the expected pairwise allocation matrix (EPAM) within clusters/subsets over the total variance of the expected pairwise allocation matrix.

Usage

variance.ratio(x, y)

Arguments

x, y If y is not specified then x must be an object of class sdols.confidence. Otherwise, x is a vector of cluster labels and y is an expected pairwise allocation matrix.

Details

The variance.ratio function takes as input an object of class sdols.confidence and calculates the variance ratio for the estimated partition from the corresponding expected pairwise allocation matrix (EPAM).

The variance ratio is the weighted average of the within cluster variances of the EPAM, weighted by the number of pairwise EPAM values per cluster, over the total variance of the EPAM.

See Also

Other Default Mass Selection: default.mass, mass.algorithm, partition.confidence

Examples

x <- rep(c(1,2,3), times=c(2,3,5))
y <- diag(10)
y[upper.tri(y)] <- runif(45)
variance.ratio(x,y)
Index

*Topic **package**
  shallot-package, 2

adj.rand.index, 3, 3, 10
as.matrix, 16, 17
as.matrix.shallot.attraction (attraction), 5
as.matrix.shallot.pairwiseProbability (pairwise.probabilities), 16
association.matrix, 4
attraction, 5, 7, 12, 19
ddcrp, 5
ddcrp (ewens), 11
decay, 5
decay (decay.reciprocal), 6
decay.reciprocal, 6
default.mass, 7, 8, 14, 15, 18, 26
disc, 12, 13
disc (mass), 12
dist, 7

enumerate.partitions, 8
estimate.partition, 2, 4, 9, 17, 21–23
ewens, 11
ewens.attraction, 5
ewens.pitman.attraction, 2, 5, 18, 21, 22

mass, 12, 12, 13
mass.algorithm, 7, 8, 14, 18, 26

ns subsets.average, 15
ns subsets.average (ns subsets.random), 15
ns subsets.probability, 15
ns subsets.probability (ns subsets.random), 15
ns subsets.random, 15, 15
ns subsets.variance, 15
ns subsets.variance (ns subsets.random), 15

pairwise.probabilities, 10, 16, 16, 17, 21–23
partition.confidence, 8, 15, 17, 17, 26
partition.distribution, 16, 21, 23
partition.distribution (ewens), 11
partition.pmf, 18
permutation, 5, 19, 19
plot.partition, 10
print, 16
print.shallot.attraction (attraction), 5
print.shallot.decay (decay.reciprocal), 6
print.shallot.default.mass (default.mass), 7
print.shallot.discount (mass), 12
print.shallot.distribution.ddcrp (ewens), 11
print.shallot.distribution.ewens (ewens), 11
print.shallot.distribution.ewensAttraction (ewens), 11
print.shallot.distribution.ewensPitman (ewens), 11
print.shallot.distribution.ewensPitmanAttraction (ewens), 11
print.shallot.mass (mass), 12
print.shallot.pairwiseProbability (pairwise.probabilities), 16
print.shallot.permutation, 19
print.shallot.permutation (permutation), 19
print.shallot.samples.raw (sample.partitions), 21
print.shallot.temperature (mass), 12
process.samples, 9, 10, 17, 20, 21–23

salso, 8
sample.partitions, 2, 10–12, 17, 20, 21, 23
sample.partitions.posterior, 21, 22
sampling.model, 22, 24
shallot (shallot-package), 2
shallot-package, 2

temperature, 7, 13
temperature (mass), 12

variance.ratio, 8, 15, 18, 26, 26