Package 'shallot'

October 14, 2022

Type Package

Title Random Partition Distribution Indexed by Pairwise Information

Version 0.4.10

Date 2020-11-09

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 (>= 3.2.18), commonsMath (>= 1.2.5)

License Apache License 2.0 | file LICENSE

RoxygenNote 7.1.1

Encoding UTF-8

NeedsCompilation no

Author David B. Dahl [aut, cre]

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

Repository CRAN

Date/Publication 2020-11-09 16:10:02 UTC

R topics documented:

`shallot-package` *Random Partition Distribution Indexed by Pairwise Information*

Description

This package implements models described in the paper Dahl, D. B., Day, R., and Tsai, J. (2017), Random Partition Distribution Indexed by Pairwise Information, *Journal of the American Statistical Association*, 112, 721-732. The Ewens, Ewens-Pitman, Ewens attraction, Ewens-Pitman attraction, and ddCRP distributions are available for prior simulation. We hope in the future to add posterior simulation with a user-supplied likelihood. Supporting functions for partition estimation and plotting are also planned.

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 Information, *Journal of the American Statistical Association*, 112, 721-732. <DOI:10.1080/01621459.2016.1165103>

See Also

`ewens.pitman.attraction`, `sample.partitions`

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)
```

attraction *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

```
attraction(permutation, decay)

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

## 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 <dahl@stat.byu.edu>

References

Dahl, D. B., Day, R., and Tsai, J. (2017), Random Partition Distribution Indexed by Pairwise Information, *Journal of the American Statistical Association*, 112, 721-732. <DOI:10.1080/01621459.2016.1165103>

See Also

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

Examples

```
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)

## 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

Dahl, D. B., Day, R., and Tsai, J. (2017), Random Partition Distribution Indexed by Pairwise Information, *Journal of the American Statistical Association*, 112, 721-732. <DOI:10.1080/01621459.2016.1165103>

See Also

dist, temperature, attraction

Examples

```
temp <- temperature(1.0)
distance <- dist(scale(USArrests))
decay1 <- decay.reciprocal(temp,distance)
decay2 <- decay.exponential(temp,distance)
decay3 <- decay.subtraction(temp,distance)
```

---

ewens *Partition Distributions*

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

Dahl, D. B., Day, R., and Tsai, J. (2017), Random Partition Distribution Indexed by Pairwise Information, *Journal of the American Statistical Association*, 112, 721-732. <DOI:10.1080/01621459.2016.1165103>

See Also

mass, discount, attraction, sample.partitions

Examples

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

decay <- decay.exponential(temperature(1.0),dist(scale(USArrests)))
attraction <- attraction(permutation(n.items=50,fixed=FALSE), decay)
pd2 <- ewens.pitman.attraction(mass(1), discount(0.05), attraction)

pd3 <- ddcrp(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

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

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

discount(..., fixed = TRUE)

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

temperature(..., fixed = TRUE)

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

Arguments

`...`	A number greater than `0.0` representing the value of the mass, discount, or temperaure 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)
```

---

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

```
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.

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

Dahl, D. B., Day, R., and Tsai, J. (2017), Random Partition Distribution Indexed by Pairwise Information, *Journal of the American Statistical Association*, 112, 721-732. <DOI:10.1080/01621459.2016.1165103>

See Also

partition.distribution

Examples

```
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)
```

---

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

```
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

```
## 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 Information, *Journal of the American Statistical Association*, 112, 721-732. <DOI:10.1080/01621459.2016.1165103>

See Also

attraction

Examples

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

---

process.samples *Process Sampled Partitions*

Description

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

Usage

```
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

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

## End(Not run)
```

`sample.partitions` *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?

Value

An object of class `shallot.samples.raw` which can be subsequently be used in `process.samples`.

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

Examples

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

## End(Not run)
```

Index