Package 'salso'

September 16, 2024

Type Package

Title Search Algorithms and Loss Functions for Bayesian Clustering

Version 0.3.42

Description The SALSO algorithm is an efficient randomized greedy search method to find a point estimate for a random partition based on a loss function and posterior Monte Carlo samples. The algorithm is implemented for many loss functions, including the Binder loss and a generalization of the variation of information loss, both of which allow for unequal weights on the two types of clustering mistakes. Efficient implementations are also provided for Monte Carlo estimation of the posterior expected loss of a given clustering estimate. See Dahl, Johnson, Müller (2022) <doi:10.1080/10618600.2022.2069779>.

License MIT + file LICENSE | Apache License 2.0

URL https://github.com/dbdahl/salso

BugReports https://github.com/dbdahl/salso/issues

Depends R (>= 4.2.0)

SystemRequirements Cargo (Rust's package manager), rustc (>= 1.66.1)

Encoding UTF-8

LazyData TRUE

RoxygenNote 7.3.2

NeedsCompilation yes

Author David B. Dahl [aut, cre] (<https://orcid.org/0000-0002-8173-1547>),
Devin J. Johnson [aut] (<https://orcid.org/0000-0003-2619-6649>),
Peter Müller [aut],
Alex Crichton [ctb] (Rust crates: cfg-if, proc-macro2),
Brendan Zabarauskas [ctb] (Rust crate: approx),
David B. Dahl [ctb] (Rust crates: dahl-bellnumber, dahl-partition, dahl-salso, roxido, roxido_macro),
David Tolnay [ctb] (Rust crates: proc-macro2, quote, syn, unicode-ident),
Jim Turner [ctb] (Rust crate: ndarray),
Josh Stone [ctb] (Rust crate: autocfg),
R. Janis Goldschmidt [ctb] (Rust crate: matrixmultiply),

Sean McArthur [ctb] (Rust crate: num_cpus),
Stefan Lankes [ctb] (Rust crate: hermit-abi),
The Cranelift Project Developers [ctb] (Rust crate: wasi),
The CryptoCorrosion Contributors [ctb] (Rust crates: ppv-lite86, rand_chacha),
The Rand Project Developers [ctb] (Rust crates: getrandom, rand, rand_chacha, rand_core, rand_pcg),
The Rust Project Developers [ctb] (Rust crates: libc, num-bigint, num-complex, num-integer, num-traits, rand, rand_chacha, rand_core),
Ulrik Sverdrup ``bluss'' [ctb] (Rust crate: ndarray),
bluss [ctb] (Rust crates: matrixmultiply, rawpointer)

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

Repository CRAN

Date/Publication 2024-09-16 12:00:06 UTC

Contents

`bell` *Compute the Bell Number*

Description

These functions compute the Bell number (the number of partitions of a given number of items) or its natural logarithm.

Usage

```
bell(nItems)

lbell(nItems)
```

Arguments

nItems	The size of the set $1, 2, ..., n$.

Value

A numeric vector of length one giving the Bell number or its natural logarithm.

Examples

```
bell(12)
lbell(300)
all.equal( bell(5), exp(lbell(5)) )
```

---

chips *CHiPS Partition Greedy Search*

Description

This function provides a partition to a subset of items which has high marginal probability based on samples from a partition distribution using the CHiPS greedy search method (Dahl, Page, Barrientos, 2024).

Usage

```
chips(
  x,
  threshold = 0,
  nRuns = 64,
  intermediateResults = TRUE,
  allCandidates = FALSE,
  nCores = 0
)
```

Arguments

x	A B-by-n matrix, where each of the B rows represents a clustering of n items using cluster labels. For the bth clustering, items i and j are in the same cluster if `x[b, i] == x[b, j]`.
threshold	The minimum marginal probability for the partial partition. Values closer to 1.0 will yield a partition of fewer items and values closer to 0.0 will yield a partition of more items.
nRuns	The number of runs to try, where the best result is returned.
intermediateResults	Should intermediate subset partitions be returned?
allCandidates	Should all the final subset partitions from multiple runs be returned?
nCores	The number of CPU cores to use, i.e., the number of simultaneous runs at any given time. A value of zero indicates to use all cores on the system.

Value

If `intermediateResults` is `FALSE`, an integer vector giving the estimated subset partition, encoded using cluster labels with `-1` indicating not allocated. If `TRUE`, a matrix with intermediate subset partitions in the rows.

Examples

```
# For examples, use 'nCores = 1' per CRAN rules, but in practice omit this.
data(iris.clusterings)
draws <- iris.clusterings
chips(draws, threshold = 0, nRuns = 1)
chips(draws, nCores = 1)
```

---

dlso *Latent Structure Optimization Based on Draws*

Description

This function provides a partition to summarize a partition distribution using the draws-based latent structure optimization (DLSO) method, which is also known as the least-squares clustering method (Dahl 2006). The method seeks to minimize an estimation criterion by picking the minimizer among the partitions supplied. The implementation currently supports the minimization of several partition estimation criteria. For details on these criteria, see `partition.loss`.

Usage

```
dlso(truth, loss = VI(), estimate = NULL)
```

Arguments

`truth`	An integer vector of cluster labels for n items representing the true clustering. Two items are in the same cluster if their labels are equal. Or, a matrix of n columns where each row is a clustering.
`loss`	The loss function to use, as indicated by `"binder"`, `"omARI"`, `"VI"`, `"NVI"`, `"ID"`, `"NID"`, or the result of calling a function with these names. Also supported are `"binder.psm"`, `"VI.lb"`, `"omARI.approx"`, or the result of calling a function with these names, in which case `x` above can optionally be a pairwise similarity matrix, i.e., n-by-n symmetric matrix whose (i, j) element gives the (estimated) probability that items i and j are in the same subset (i.e., cluster) of a partition (i.e., clustering).
`estimate`	An integer vector of cluster labels having the same length as `truth` representing the estimated clustering. Or, a matrix of n columns where each row is a clustering.

Value

An integer vector giving the estimated partition, encoded using cluster labels.

References

D. B. Dahl (2006), Model-Based Clustering for Expression Data via a Dirichlet Process Mixture Model, in *Bayesian Inference for Gene Expression and Proteomics*, Kim-Anh Do, Peter Müller, Marina Vannucci (Eds.), Cambridge University Press.

See Also

partition.loss, psm, summary.salso.estimate, salso

Examples

```
data(iris.clusterings)
dlso(iris.clusterings, loss=VI())
dlso(iris.clusterings, loss=binder())

# Compute expected loss using all draws, but pick the best among the first 10.
dlso(iris.clusterings, loss=VI(), estimate=iris.clusterings[1:10,])
```

---

enumerate.partitions *Enumerate Partitions of a Set*

Description

This function produces a matrix whose rows provide all possible partitions of the set $1, 2, ..., n$. These partitions are provided as cluster labels, where two items are in the same subset (i.e., cluster) if their labels are equal.

Usage

```
enumerate.partitions(nItems)
```

Arguments

nItems	The size of the set $1, 2, ..., n$, i.e., n.

Value

A matrix of integers, where each row is a partition encoded as a vector of cluster labels.

Examples

```
enumerate.partitions(5)
```

enumerate.permutations *Enumerate Permutations of Items*

Description

This function produces a matrix whose rows provide all possible permutations of the set $1, 2, ..., n$.

Usage

```
enumerate.permutations(nItems)
```

Arguments

nItems	The size of the set $1, 2, ..., n$, i.e., n.

Value

A matrix of integers, where each row is a permutation.

Examples

```
enumerate.permutations(5)
```

iris.clusterings *Clusterings of the Iris Data*

Description

Randomly generated clusterings for the `iris` dataset.

Usage

```
iris.clusterings
```

Format

A 1000-by-150 matrix of 1000 randomly generated clusterings of the 150 observations in the `iris` dataset.

Source

Unknown.

See Also

`iris`

`partition.loss` *Compute Partition Loss or the Expectation of Partition Loss*

Description

Given two partitions $\pi*$ and π, these functions compute the specified loss when using $\pi*$ to estimate π. Smaller loss values indicate higher concordance between partitions. These functions also compute a Monte Carlo estimate of the expectation for the specified loss based on samples or a pairwise similarity matrix. This function also supports computing approximations to the expectation of several losses. Supported criteria are described below. Some criteria only require the pairwise similarity matrix (as computed, for example, by `psm`) whereas others require samples from a partition distribution. For those criteria that only need the pairwise similarity matrix, posterior samples can still be provided in the `truth` argument and the pairwise similarity matrix will automatically be computed as needed.

Usage

```
partition.loss(truth, estimate, loss = VI())

binder(truth, estimate, a = 1)

RI(truth, estimate)

omARI(truth, estimate)

omARI.approx(truth, estimate)

ARI(truth, estimate)

VI(truth, estimate, a = 1)

VI.lb(truth, estimate)

NVI(truth, estimate)

ID(truth, estimate)

NID(truth, estimate)
```

Arguments

`truth`	An integer vector of cluster labels for n items representing the true clustering. Two items are in the same cluster if their labels are equal. Or, a matrix of n columns where each row is a clustering.
`estimate`	An integer vector of cluster labels having the same length as `truth` representing the estimated clustering. Or, a matrix of n columns where each row is a clustering.

`loss` The loss function to use, as indicated by `"binder"`, `"omARI"`, `"VI"`, `"NVI"`, `"ID"`, `"NID"`, or the result of calling a function with these names. Also supported are `"binder.psm"`, `"VI.lb"`, `"omARI.approx"`, or the result of calling a function with these names, in which case `x` above can optionally be a pairwise similarity matrix, i.e., n-by-n symmetric matrix whose (i, j) element gives the (estimated) probability that items i and j are in the same subset (i.e., cluster) of a partition (i.e., clustering).

`a` (Only used for Binder and VI loss) The argument `a` is either: i. a nonnegative scalar in $[0, 2]$ giving (for Binder loss) the cost of placing two items in separate clusters when in truth they belong to the same cluster, ii. `NULL`, in which case `a` that maximizes the expected loss is found, and iii. a list containing the desired number of clusters (`"nClusters"`) when searching for `a` that yields this number of clusters. In all but the first case, one may want to modifying `maxSize` in the `salso` function. To increase the probability of hitting exactly the desired number of clusters, the `nRuns` in the `salso` function may need to be increased. Without loss of generality, the cost (under Binder loss) of placing two items in the same cluster when in truth they belong to separate clusters is fixed `2-a`. For VI, `a` has a similar interpretation, although is not a unit cost. See Dahl, Johnson, Müller (2021).

Details

The partition estimation criterion can be specified using the `loss` argument, which is either a string or a result of calling the associated functions. These losses are described below:

`"binder"` Binder. Whereas high values of the Rand index R between $\pi*$ and π correspond to high concordance between the partitions, the N-invariant Binder loss L for a partition $\pi*$ in estimating π is $L = (1 - R) * (n - 1)/n$, meaning that low values correspond to high concordance between the partitions. This package reports the N-invariant Binder loss. The original Binder loss equals the N-invariant Binder loss multiplied by $n^2/2$. Only the pairwise similarity matrix is required for this loss, but samples can be provided. As originally discussed by Binder (1978), two mistakes are possible: 1. Placing two items in separate clusters when in truth they belong to the same cluster, and 2. Placing two items in the same cluster when in truth they belong to separate clusters. The default cost of the first mistake is one, but can be specified with the argument `a` in $[0, 2]$. Without loss of generality, the cost of the second mistake is set as `2-a`. For a discussion of general weights, see Dahl, Johnson, and Müller (2021). For a discussion of the equal weights case, see also Dahl (2006), Lau and Green (2007), Dahl and Newton (2007), Fritsch and Ickstadt (2009), and Wade and Ghahramani (2018).

`"omARI"` One Minus Adjusted Rand Index. Computes the expectation of one minus the adjusted Rand index (Hubert and Arabie, 1985). Whereas high values of the adjusted Rand index between $\pi*$ and π correspond to high concordance between the partitions, the loss associated with the adjusted Rand index for a partition $\pi*$ in estimating π is one minus the adjusted Rand index between the partitions, meaning that low values correspond to high concordance between the partitions. Samples from a partition distribution are required for this loss. See Fritsch and Ickstadt (2009).

`"omARI.approx"` Approximation of One Minus Adjusted Rand Index. Computes the first-order approximation of the expectation of one minus the adjusted Rand index. The adjusted Rand index involves a ratio and the first-order approximation of the expectation is based on $E(X/Y) \approx$

$E(X)/E(Y)$. Only the pairwise similarity matrix is required for this criterion, but samples can be provided. See Fritsch and Ickstadt (2009).

`"VI"` Variation of Information. Computes the expectation of the (generalized) variation of information loss. Samples from a partition distribution are required for this loss. See Meilă (2007), Wade and Ghahramani (2018), and Rastelli and Friel (2018). The original variation of information of Meilă (2007) has been extended to the generalized variation of information of Dahl, Johnson, and Müller (2021) to allow for unequal weighting of two possible mistakes: 1. Placing two items in separate clusters when in truth they belong to the same cluster, and 2. Placing two items in the same cluster when in truth they belong to separate clusters. The value `a` controls the cost of the first mistake and defaults to one, but can be specified with the argument `a` in $[0, 2]$. Without loss of generality, the cost of the second mistake is controlled by `2-a`. See Dahl, Johnson, Müller (2021).

`"VI.lb"` Lower Bound of the Variation of Information. Computes the lower bound of the expectation of the variation of information loss, where the lower bound is obtained by Jensen's inequality. Only the pairwise similarity matrix is required for this criterion, but samples can be provided. See Wade and Ghahramani (2018).

`"NVI"` Normalized Variation of Information. Computes the expectation of the normalized variation of information loss. Samples from a partition distribution are required for this loss. See Vinh, Epps, and Bailey (2010) and Rastelli and Friel (2018).

`"ID"` Information Distance. Computes the expectation of the information distance (D_{max}) loss. Samples from a partition distribution are required for this loss. See Vinh, Epps, and Bailey (2010).

`"NID"` Normalized Information Distance. Computes the expectation of the normalized information distance loss. Samples from a partition distribution are required for this loss. See Vinh, Epps, and Bailey (2010) and Rastelli and Friel (2018).

The functions `RI` and `ARI` are convenience functions. Note that:

- `binder(p1, p2, a=1) = ( 1 - RI(p1, p2) )*(n-1)/n`
- `omARI(p1, p2) = 1 - ARI(p1, p2)`

Value

A numeric vector.

References

W. M. Rand (1971), Objective Criteria for the Evaluation of Clustering Methods. *Journal of the American Statistical Association*, **66**: 846–850.

D. A. Binder (1978), Bayesian cluster analysis, *Biometrika* **65**, 31-38.

L. Hubert and P. Arabie (1985), Comparing Partitions. *Journal of Classification*, **2**, 193–218.

D. B. Dahl (2006), Model-Based Clustering for Expression Data via a Dirichlet Process Mixture Model, in *Bayesian Inference for Gene Expression and Proteomics*, Kim-Anh Do, Peter Müller, Marina Vannucci (Eds.), Cambridge University Press.

J. W. Lau and P. J. Green (2007), Bayesian model based clustering procedures, *Journal of Computational and Graphical Statistics* **16**, 526-558.

M. Meilă (2007), Comparing Clusterings - an Information Based Distance. *Journal of Multivariate Analysis*, **98**: 873–895.

D. B. Dahl and M. A. Newton (2007), Multiple Hypothesis Testing by Clustering Treatment Effects, *Journal of the American Statistical Association*, **102**, 517-526.

A. Fritsch and K. Ickstadt (2009), An improved criterion for clustering based on the posterior similarity matrix, *Bayesian Analysis*, **4**, 367-391.

N. X. Vinh, J. Epps, and J. Bailey (2010), Information Theoretic Measures for Clusterings Comparison: Variants, Properties, Normalization and Correction for Chance, *Journal of Machine Learning Research*, **11**, 2837-2854.

S. Wade and Z. Ghahramani (2018), Bayesian cluster analysis: Point estimation and credible balls. *Bayesian Analysis*, **13:2**, 559-626.

R. Rastelli and N. Friel (2018), Optimal Bayesian estimators for latent variable cluster models. *Statistics and Computing*, **28**, 1169-1186.

D. B. Dahl, D. J. Johnson, and P. Müller (2022), Search Algorithms and Loss Functions for Bayesian Clustering, *Journal of Computational and Graphical Statistics*, 31(4), 1189-1201, doi:10.1080/10618600.2022.2069779.

See Also

psm, salso, dlso

Examples

```
# For examples, use 'nCores=1' per CRAN rules, but in practice omit this.
data(iris.clusterings)
partitions <- iris.clusterings[1:5,]

all.equal(partition.loss(partitions, partitions, loss=binder(a=1.4)),
          binder(partitions, partitions, a=1.4))
all.equal(partition.loss(partitions, partitions, loss=omARI()),
          omARI(partitions, partitions))
all.equal(partition.loss(partitions, partitions, loss=VI(a=0.8)),
          VI(partitions, partitions, a=0.8))

truth <- iris.clusterings[1,]
estimate <- iris.clusterings[2,]

VI(truth, estimate, a=1.0)
n <- length(truth)
all.equal(binder(truth, estimate), ( 1 - RI(truth, estimate) ) * (n-1) / n)
all.equal(omARI(truth, estimate), 1 - ARI(truth, estimate))
```

`plot.salso.summary` *Heatmap, Multidimensional Scaling, Pairs, and Dendrogram Plotting for Partition Estimation*

Description

This function produces one of four plots: 1. `"heatmap"`: A heatmap showing the pairwise allocation probabilities that items are clustered. 2. `"mds"`: A scatter plot using classical multidimensional scaling (also known as principal coordinates analysis) with the exemplar (i.e., the most representative observation) of each cluster emphasized. 3. `"pairs"`: Pairs plots of all the variables with the exemplar (i.e., the most representative observation) of each cluster emphasized. 4. `"dendrogram"`: A dendrogram based on expected partition loss showing the relationships among clusters when merging pairs of clusters such that the increase in the expectation of the posterior loss is minimized.

Usage

```
## S3 method for class 'salso.summary'
plot(
  x,
  type = c("heatmap", "mds", "pairs", "dendrogram")[1],
  data = NULL,
  showLabels = TRUE,
  showIDs = length(x$estimate) <= 50,
  cexAdjustment = 0.7,
  ...
)
```

Arguments

`x`	An object returned by `summary(y)`, where `y` itself is returned by the `salso` function.
`type`	A string equal to `"heatmap"`, `"mds"`, `"pairs"`, or `"dendrogram"`.
`data`	The data from which the partition estimation was obtained. This is required when `type='pairs'` and ignored otherwise.
`showLabels`	Should the cluster labels be shown in the plot when `type="heatmap"`?
`showIDs`	Should the ID of the items be shown in the plot?
`cexAdjustment`	Scalar multiplier for adjust text size.
`...`	Arguments to be passed to methods, such as graphical parameters (see `par`).

Value

`NULL`, invisibly.

See Also

`salso`, `summary.salso.estimate`, `cmdscale`.

Examples

```
# For examples, use 'nCores=1' per CRAN rules, but in practice omit this.
data(iris.clusterings)
draws <- iris.clusterings
est <- salso(draws, nCores=1)
summ <- summary(est)
plot(summ, type="heatmap")
plot(summ, type="mds")
plot(summ, type="pairs", data=iris)
plot(summ, type="dendrogram")
```

---

psm *Compute an Adjacency or Pairwise Similarity Matrix*

Description

If only one sample is provided, this function computes an adjacency matrix, i.e., a binary matrix whose (i, j) element is one if and only if elements i and j in the partition have the same cluster label. If multiple samples are provided (as rows of the `x` matrix), this function computes the n-by-n matrix whose (i, j) element gives the relative frequency (i.e., estimated probability) that items i and j are in the same subset (i.e., cluster). This is the mean of the adjacency matrices of the provided samples.

Usage

```
psm(x, nCores = 0)
```

Arguments

`x`	A B-by-n matrix, where each of the B rows represents a clustering of n items using cluster labels. For the bth clustering, items i and j are in the same cluster if `x[b,i] == x[b,j]`.
`nCores`	The number of CPU cores to use, i.e., the number of simultaneous runs at any given time. A value of zero indicates to use all cores on the system.

Value

A n-by-n symmetric matrix whose (i, j) element gives the relative frequency that that items i and j are in the same subset (i.e., cluster).

Examples

```
# For examples, use 'nCores=1' per CRAN rules, but in practice omit this.
data(iris.clusterings)
partition <- iris.clusterings[1,]
psmatrix <- psm(partition, nCores=1)
psmatrix[1:6, 1:6]
```

```
dim(iris.clusterings)
probs <- psm(iris.clusterings, nCores=1)
dim(probs)
probs[1:6, 1:6]
```

`salso` *SALSO Greedy Search*

Description

This function provides a partition to summarize a partition distribution using the SALSO greedy search method (Dahl, Johnson, and Müller, 2022). The implementation currently supports the minimization of several partition estimation criteria. For details on these criteria, see `partition.loss`.

Usage

```
salso(
  x,
  loss = VI(),
  maxNClusters = 0,
  nRuns = 16,
  maxZealousAttempts = 10,
  probSequentialAllocation = 0.5,
  nCores = 0,
  ...
)
```

Arguments

`x`	A B-by-n matrix, where each of the B rows represents a clustering of n items using cluster labels. For the bth clustering, items i and j are in the same cluster if `x[b,i] == x[b,j]`.
`loss`	The loss function to use, as indicated by `"binder"`, `"omARI"`, `"VI"`, `"NVI"`, `"ID"`, `"NID"`, or the result of calling a function with these names. Also supported are `"binder.psm"`, `"VI.lb"`, `"omARI.approx"`, or the result of calling a function with these names, in which case `x` above can optionally be a pairwise similarity matrix, i.e., n-by-n symmetric matrix whose (i, j) element gives the (estimated) probability that items i and j are in the same subset (i.e., cluster) of a partition (i.e., clustering). The loss functions `"binder.psm"`, `"VI.lb"`, and `"omARI.approx"` are generally not recommended and the current implementation requires that `maxZealousAttempts = 0` and `probSequentialAllocation = 1.0`.
`maxNClusters`	The maximum number of clusters that can be considered by the optimization algorithm, which has important implications for the interpretability of the resulting clustering and can greatly influence the RAM needed for the optimization algorithm. If the supplied value is zero and `x` is a matrix of clusterings, the

optimization is constrained by the maximum number of clusters among the clusterings in `x`. If the supplied value is zero and `x` is a pairwise similarity matrix, there is no constraint.

`nRuns` The number of runs to try, although the actual number may differ for the following reasons: 1. The actual number is a multiple of the number of cores specified by the `nCores` argument, and 2. The search is curtailed when the `seconds` threshold is exceeded.

`maxZealousAttempts`
The maximum number of attempts for zealous updates, in which entire clusters are destroyed and items are sequentially reallocated. While zealous updates may be helpful in optimization, they also take more CPU time which might be better used trying additional runs.

`probSequentialAllocation`
For the initial allocation, the probability of sequential allocation instead of using `sample(1:K, ncol(x), TRUE)`, where `K` is set according to the `maxNClusters` argument.

`nCores` The number of CPU cores to use, i.e., the number of simultaneous runs at any given time. A value of zero indicates to use all cores on the system.

`...` Extra arguments not intended for the end user, including: 1. `seconds`: Instead of performing all the requested number of runs, curtail the search after the specified expected number of seconds. Note that the function will finish earlier if all the requested runs are completed. The specified seconds does not account for the overhead involved in starting the search and returning results. 2. `maxScans` The maximum number of full reallocation scans. The actual number of scans may be less than `maxScans` since the method stops if the result does not change between scans, and 3. `probSingletonsInitialization`: When doing a sequential allocation to obtain the initial allocation, the probability of placing the first `maxNClusters` randomly-selected items in singletons subsets.

Value

An integer vector giving the estimated partition, encoded using cluster labels.

References

D. B. Dahl, D. J. Johnson, and P. Müller (2022), Search Algorithms and Loss Functions for Bayesian Clustering, *Journal of Computational and Graphical Statistics*, 31(4), 1189-1201, doi:10.1080/10618600.2022.2069779.

See Also

`partition.loss`, `psm`, `summary.salso.estimate`, `dlso`

Examples

```
# For examples, use 'nCores=1' per CRAN rules, but in practice omit this.
data(iris.clusterings)
draws <- iris.clusterings
salso(draws, loss=VI(), nRuns=1, nCores=1)
```

```
salso(draws, loss=VI(a=0.7), nRuns=1, nCores=1)
salso(draws, loss=binder(), nRuns=1, nCores=1)
salso(iris.clusterings, binder(a=NULL), nRuns=4, nCores=1)
salso(iris.clusterings, binder(a=list(nClusters=3)), nRuns=4, nCores=1)
```

`summary.salso.estimate` *Summary of Partitions Estimated Using Posterior Expected Loss*

Description

Assessing the quality of clusters in a partition estimate is added by this function. The result can then be plotted with `plot.salso.summary`. The current implementation of the calculation of these summaries is not terribly efficient and may be improved in the future.

Usage

```
## S3 method for class 'salso.estimate'
summary(object, alternative, orderingMethod = 1, ...)
```

Arguments

`object`	An object returned by the `salso` function.
`alternative`	An optional argument specifying an alternative clustering to use instead of that provided by `object`. Use this feature to obtain numerical and graphical summaries of a clustering estimate from other procedures. This clustering must be provided in canonical form: cluster labels as integers starting at 1 for the first observation and incrementing by one for each new label.
`orderingMethod`	An integer giving method to use to order the observations for a heatmap plot. Currently values 1 or 2 are supported.
`...`	Currently ignored.

Value

A list containing the estimate, the pairwise similarity matrix, the mean pairwise similarity matrix, the score and mean pairwise similarity for each observation, exemplar observation for each cluster, a dendrogram object, a vector for ordering observations in the heatmap plot, the size of each cluster, and the number of clusters.

Examples

```
# For examples, use 'nCores=1' per CRAN rules, but in practice omit this.
data(iris.clusterings)
draws <- iris.clusterings
est <- salso(draws, nCores=1)
summ <- summary(est)
```

```
plot(summ, type="heatmap")
plot(summ, type="mds")
plot(summ, type="pairs", data=iris)
plot(summ, type="dendrogram")
```

Index