Package ‘evclust’

January 8, 2020

Type Package
Title Evidential Clustering
Version 1.1.0
Date 2020-01-07
Author Thierry Denoeux
Maintainer Thierry Denoeux <tdenoeux@utc.fr>
Description Various clustering algorithms that produce a credal partition, i.e., a set of Dempster-Shafer mass functions representing the membership of objects to clusters. The mass functions quantify the cluster-membership uncertainty of the objects. The algorithms are: Evidential c-Means (ECM), Relational Evidential c-Means (RECM), Constrained Evidential c-Means (CECM), EVCLUS, EK-NNclus and bootclus.
Depends R (>= 3.6.0),
Imports FNN, R.utils, limSolve, Matrix, mclust, quadprog, plyr
License GPL-3
LazyData TRUE
RoxygenNote 7.0.2
VignetteBuilder knitr
Suggests knitr,rmarkdown
NeedsCompilation no
Repository CRAN
Date/Publication 2020-01-08 09:10:11 UTC

R topics documented:

  bootclus .......................................................... 2
  butterfly .......................................................... 3
  cecm ............................................................... 4
  createD ............................................................ 6
  createPairs ......................................................... 7
  create_MLCL ....................................................... 8
Generating a credal partition by bootstrapping Gaussian Mixture Models

Description

bootclus generates a credal partition by bootstrapping Gaussian Mixture Models.

Usage

bootclus(
  x,  
  alpha = c(0.05, 0.95),
  B = 500,
  param = list(G = NULL),
  type = "pairs",
  Omega = FALSE
)

Arguments

X    attribute matrix or data frame of size (n,p).
alpha vector of quantiles (of length 2).
B    number of bootstrap samples (default=500)
param list of arguments passed to function Mclust in addition to 'data'.
type Type of focal sets ("simple": \emptyset, singletons and \Omega; "full": all \(2^c\) subsets of \Omega; "pairs": \emptyset, singletons, \Omega, and all or selected pairs). Argument passed to makeF.
Omega Logical. If TRUE (default), \Omega is a focal set (for types 'simple' and 'pairs'). Argument passed to makeF.
Details

This function uses the mclust package to generate and bootstrap the mixture models.

Value

A list with the following components:

clus  An object of class 'Mclust' returned by Mclust.
Clus  An object of class 'credalPart' providing the output credal partition.
CI    An array of dimension (2,n,n) containing the confidence intervals on pairwise probabilities.
BelPl An array of dimension (2,n,n) containing the pairwise Bel-Pl intervals.
Time  A matrix of size (3,5) containing the computing time as returned by function proctime for 
       (1) the parameter estimation and bootstrap, (2) the computation fo the quantiles on pairwise 
       probabilities, and (3) the computation of the credal partition.

References


See Also

ecm, recm, cecm, keclus.

Examples

## Example with the Faithful geyser data
## Not run:
data(“faithful”)
X<-faithful
param=list(G=3)
res.faithful<-bootclus(X,alpha=c(0.05,0.95),B=100,param=param)
## Plot the results
plot(res.faithful$Clus,X)
## End(Not run)

butterfly

butterfly  Butterfly dataset

Description

A toy dataset used to illustrate fuzzy and evidential clustering algorithms. Also called the 'Diamond' dataset. Adapted from Windham (1985), with one outlier added.

Usage

data(butterfly)
Format

A matrix with 12 rows and 2 column.

References


Examples

data(butterfly)
plot(butterfly[,1],butterfly[,2],xlab=expression(x[1]),ylab=expression(x[2]))

---

cecm

Constrained Evidential c-means algorithm

Description

cecm computes a credal partition from a matrix of attribute data and pairwise constraints using the Constrained Evidential c-means (CECM) algorithm.

Usage

cecm(
  x,
  c,
  type = "full",
  pairs = NULL,
  ntrials = 1,
  ML,
  CL,
  g0 = NULL,
  alpha = 1,
  delta = 10,
  bal = 0.5,
  distance = 0,
  epsi = 0.001,
  disp = TRUE
)
Arguments

- **x**: input matrix of size n x d, where n is the number of objects and d the number of attributes.
- **c**: Number of clusters.
- **type**: Type of focal sets ("simple": empty set, singletons and Omega; "full": all \(2^c\) subsets of \(\Omega\); "pairs": \(\emptyset\), singletons, \(\Omega\), and all or selected pairs).
- **pairs**: Set of pairs to be included in the focal sets; if NULL, all pairs are included. Used only if type="pairs".
- **ntrials**: Number of runs of the optimization algorithm (set to 1 if \(g0\) is supplied).
- **ML**: Matrix nbML x 2 of must-link constraints. Each row of ML contains the indices of objects that belong to the same class.
- **CL**: Matrix nbCL x 2 of cannot-link constraints. Each row of CL contains the indices of objects that belong to different classes.
- **g0**: Initial prototypes, matrix of size c x d. If not supplied, the prototypes are initialized randomly.
- **alpha**: Exponent of the cardinality in the cost function.
- **delta**: Distance to the empty set.
- **bal**: Tradeoff between the objective function Jecm and the constraints: \(J_{cecm}=(1-bal)J_{ecm} + bal J_{const}\).
- **distance**: Type of distance use: 0=Euclidean, 1=Mahalanobis.
- **epsi**: Minimum amount of improvement.
- **disp**: If TRUE (default), intermediate results are displayed.

Details

CECM is a version of ECM allowing the user to specify pairwise constraints to guide the clustering process. Pairwise constraints are of two kinds: must-link constraints are pairs of objects that are known to belong to the same class, and cannot-link constraints are pairs of objects that are known to belong to different classes. CECM can also learn a metric for each cluster, like the Gustafson-Kessel algorithm in fuzzy clustering. At each iteration, the algorithm solves a quadratic programming problem using an interior ellipsoidal trust region and barrier function algorithm with dual solution updating technique in the standard QP form (Ye, 1992).

If initial prototypes \(g0\) are provided, the number of trials is automatically set to 1.

Remark: Due to the use of the Matrix package, messages may be generated by R’s (S4) method dispatch mechanism. They are not error messages, and they can be ignored.

Value

The credal partition (an object of class "credpart").

Author(s)

Thierry Denoeux (from a MATLAB code written by Violaine Antoine).
createD

Computation of a Euclidean distance matrix

Description

createD constructs an n x k matrix of Euclidean distances from an n x p matrix of attribute data. For each object, the distances to k randomly selected objects are computed.

Usage

createD(x, k)

Arguments

x n x p data matrix.
k Number of distances. If missing, an n x n distance matrix is computed.

Value

A list with two elements:

D n x k distance matrix.
J n x k matrix of indices. D[i,j] is the Euclidean distance between x[i,] and x[J[i,j]].
createPairs

See Also

kevclus

Examples

data(fourclass)
x<-as.matrix(fourclass[,1:2])
dist<-createD(x,k=10)
dim(dist$D)
dim(dist$J)

createPairs

Finding overlapping pairs of clusters

Description

createPairs finds pairs of clusters that are mutual k nearest neighbors in a credal partition. The similarity between two clusters k and l is defined as \( \sum_{i=1}^{n} p_{ik}p_{li} \), where \( p_{ik} \) is the plausibility of object \( i \) belonging to cluster \( k \).

Usage

createPairs(clus, k = 1)

Arguments

clus An object of class credpart. It should contain at least two fields: clus$mass (the credal partition) and clus$pl.n (the normalized plausibilities). The focal sets of the credal partition must be the empty set, the singletons, and (optionally) the whole set of clusters.

k The number of neighbors.

Details

This function allows one to use evidential clustering when the number of clusters is large. A clustering algorithm is first run with a limited number of focal sets (the empty set, the singletons and, optionally, the whole frame). Then, the similarity between clusters is analysed to determine the pairs of neighboring (overlapping) clusters. The clustering algorithm is then run again, adding these pairs to the focal sets (see the example). The focal sets of the passed credal partition must be the empty set (first row), the singletons (next c rows) and, optionally, the whole frame (last row).
create_MLCL

Random generation of Must-Link and Cannot-Link constraints

Description

create_MLCL randomly generates Must-Link (ML) and Cannot-Link (CL) constraints from a vector of class labels.

Usage

create_MLCL(y, nbConst)
Argument

y Vector of class labels.
nbConst Number of constraints.

Value

A list with two components:

ML Matrix of ML constraints. Each row corresponds to a constraint.
CL Matrix of ML constraints. Each row corresponds to a constraint.

See Also

ccem

Examples

y<-sample(3,100,replace=TRUE)
const<-create_MLCL(y,nbConst=10)
const$ML
const$CL
Arguments

- **x**: input matrix of size \( n \times d \), where \( n \) is the number of objects and \( d \) the number of attributes.
- **c**: Number of clusters.
- **g0**: Initial prototypes, matrix of size \( c \times d \). If not supplied, the prototypes are initialized randomly.
- **type**: Type of focal sets ("simple": empty set, singletons and Omega; "full": all \( 2^c \) subsets of \( \Omega \); "pairs": \( \emptyset \), singletons, \( \Omega \), and all or selected pairs).
- **pairs**: Set of pairs to be included in the focal sets; if NULL, all pairs are included. Used only if type="pairs".
- **Omega**: Logical. If TRUE (default), the whole frame is included (for types 'simple' and 'pairs').
- **ntrials**: Number of runs of the optimization algorithm (set to 1 if m0 is supplied).
- **alpha**: Exponent of the cardinality in the cost function.
- **beta**: Exponent of masses in the cost function.
- **delta**: Distance to the empty set.
- **epsi**: Minimum amount of improvement.
- **disp**: If TRUE (default), intermediate results are displayed.

Details

ECM is an evidential version algorithm of the Hard c-Means (HCM) and Fuzzy c-Means (FCM) algorithms. As in HCM and FCM, each cluster is represented by a prototype. However, in ECM, some sets of clusters are also represented by a prototype, which is defined as the center of mass of the prototypes in each individual cluster. The algorithm iteratively optimizes a cost function, with respect to the prototypes and to the credal partition. By default, each mass function in the credal partition has \( 2^c \) focal sets, where \( c \) is the supplied number of clusters. We can also limit the number of focal sets to subsets of clusters with cardinalities 0, 1 and \( c \) (recommended if \( c \geq 10 \)), or to all or some selected pairs of clusters. If initial prototypes \( g0 \) are provided, the number of trials is automatically set to 1.

Value

The credal partition (an object of class "credpart").

Author(s)

Thierry Denoeux (from a MATLAB code written by Marie-Helene Masson).

References

EkNNclus

Description
EkNNclus computes hard and credal partitions from dissimilarity or attribute data using the EkNNclus algorithm.

Usage
EkNNclus(x, D, K, y0, ntrials = 1, q = 0.5, p = 1, disp = TRUE, tr = FALSE)

Arguments
- x: n x p data matrix (n instances, p attributes).
- D: n x n dissimilarity matrix (used only if x is not supplied).
- K: Number of neighbors.
- y0: Initial partition (vector of length n, with values in 1,2,...).
- ntrials: Number of runs of the algorithm (the best solution is kept).
- q: Parameter in (0,1). Gamma is set to the inverse of the q-quantile of distances from the K nearest neighbors (same notation as in the paper).
- p: Exponent of distances, \( \alpha_{ij} = \phi(d_{ij}^p) \).
- disp: If TRUE, intermediate results are displayed.
- tr: If TRUE, a trace of the cost function is returned.

Details
The number of clusters is not specified. It is influenced by parameters K and q. (It is advised to start with the default values.) For n not too large (say, until one thousand), y0 can be defined as the vector (1,2,...,n). For larger values of n, it is advised to start with a random partition of c clusters, c<n.

See Also
makeF, extractMass, recm, cecm, plot.credpart

Examples
## Clustering of the Four-class dataset
data(fourclass)
x<-fourclass[,1:2]
y<-fourclass[,3]
clus<-ecm(x,c=4,type='full',alpha=1,beta=2,delta=sqrt(20),epsi=1e-3,disp=TRUE)
plot(clus,X=x,mfrow=c(2,2),ytrue=y,Outliers=TRUE,Approx=2)
Value

The credal partition (an object of class "credpart"). In addition to the usual attributes, the output credal partition has the following attributes:

- **trace** Trace of the algorithm (sequence of values of the cost function).
- **W** The weight matrix.

Author(s)

Thierry Denoeux.

References


Examples

```r
## Clustering of the fourclass dataset
data(fourclass)
n<-nrow(fourclass)
N=2
clus<- EkNNclus(fourclass[,1:2],K=60,y0=(1:n),ntrials=N,q=0.9,p=2,disp=TRUE,tr=TRUE)
## Plot of the partition
plot(clus,X=fourclass[,1:2],y=fourclass$y,Outliers=FALSE)
## Plot of the cost function vs number of iteration
L<-vector(length=N)
for(i in 1:N) L[i]<-dim(clus$trace[clus$trace[,1]==i,])[1]
imax<-which.max(L)
plot(0:(L[imax]-1),-clus$trace[clus$trace[,1]==imax,3],type="l",lty=imax,
xlab="time steps",ylab="energy")
for(i in (1:N)) if(i != imax) lines(0:(L[i]-1),-clus$trace[clus$trace[,1]==i,3],type="l",lty=i)
```

---

**evclust**

*evclust: A package for evidential clustering*

Description

Various clustering algorithms that generate a credal partition, i.e., a set of mass functions. Mass functions quantify the cluster-membership uncertainty of the objects. The package consists in five main functions, implementing five different evidential clustering algorithms:

- **ecm** Evidential c-means algorithm (Masson and Denoeux, 2008)
- **recm** Relational Evidential c-means algorithm (Masson and Denoeux, 2009)
- **kevclus** $k$-EVCLUS algorithm (Denoeux and Masson, 2004; Denoeux et al., 2016)
- **EkNNclus** E$k$-NNclus algorithm (Denoeux et al., 2015)
- **cecm** Constrained Evidential c-means algorithm (Antoine et al, 2012)
extractMass

References


See Also

ecm, recm, cecm, kevclus, EkNNclus.

extractMass

 Creates an object of class "credalPart"

Description

extractMass computes different outputs (hard, fuzzy, rough partitions, etc.) from a credal partition and creates an object of class "credalPart".

Usage

extractMass(
  mass,
  F,
  g = NULL,
  S = NULL,
  method,
  crit,
  Kmat = NULL,
  trace = NULL,
  D = NULL,
  W = NULL
)
extractMass

**Arguments**

- **mass**: A credal partition (a matrix of n rows and f columns, where n is the number of objects and f is the number of focal sets).
- **F**: Matrix (f,c) of focal sets.
- **g**: A c x d matrix of prototypes.
- **S**: A list of length f containing the matrices \( S_j \) defining the metrics for each cluster and each group of cluster.
- **method**: The method used to construct the credal partition (a character string).
- **crit**: The value of the optimized criterion (depends on the method used).
- **Kmat**: The matrix of degrees of conflict. Same size as \( D \) (for method `kevclus`).
- **trace**: The trace of criterion values (for methods `kevclus` and `EkNNclus`).
- **D**: The normalized dissimilarity matrix (for method `kevclus`).
- **W**: The weight matrix (for method `EkNNclus`).

**Details**

This function collects varied information on a credal partition and stores it in an object of class "credalPart". The lower and upper approximations of clusters define rough partitions. They can be computed in two ways: either from the set of clusters with maximum mass, or from the set of non dominated clusters. A cluster \( \omega_k \) is non dominated if \( \text{pl}(\omega_k) \geq \text{bel}(\omega_l) \) for all \( l \) different from \( k \). Once a set of cluster \( Y_i \) has been computed for each object, object \( i \) belongs to the lower approximation of cluster \( k \) if \( Y_i = \omega_k \). It belongs to the upper approximation of cluster \( k \) if \( \omega_k \in Y_i \). See Masson and Denoeux (2008) for more details, and Denoeux and Kanjanatarakul (2016) for the interval dominance rule. The function creates an object of class "credalPart". There are two methods for this class: `plot.credpart` and `summary.credpart`.

**Value**

An object of class "credalpart" with the following components:

- **method**: The method used to construct the credal partition (a character string).
- **F**: Matrix of focal sets.
- **conf**: Masses assigned to the empty set, vector of length n.
- **mass**: Mass functions, matrix of size (n,f).
- **mass.n**: Normalized mass functions, matrix of size (n,f-1).
- **g**: The prototypes (if defined).
- **S**: The matrices \( S_j \) defining the metrics for each cluster and each group of cluster (if defined).
- **pl**: Unnormalized plausibilities of the singletons, matrix of size (n,c).
- **pl.n**: Normalized plausibilities of the singletons, matrix of size (n,c).
- **bel**: Unnormalized beliefs of the singletons, matrix of size (n,c).
- **bel.n**: Normalized beliefs of the singletons, matrix of size (n,c).
- **y.pl**: Maximum plausibility clusters, vector of length n.
y.bel  Maximum belief clusters, vector of length n.
betp  Unnormalized pignistic probabilities of the singletons.
betpn  Normalized pignistic probabilities of the singletons.
Y  Sets of clusters with maximum mass, matrix of size (n,c).
outlier  n-vector of 0’s and 1’s, indicating which objects are outliers. An outlier is an object such
that the largest mass is assigned to the empty set.
lower.approx  Lower approximations of clusters, a list of length c. Each element lower.approx[[i]]
is a vector of object indices.
upper.approx  Upper approximations of clusters, a list of length c. Each element upper.approx[[i]]
is a vector of object indices.
Ynd  Sets of clusters selected by the interval dominance rule, matrix of size (n,c).
lower.approx.nd  Lower approximations of clusters using the interval dominance rule, a list of
length c. Each element lower.approx.nd[[i]] is a vector of objects.
upper.approx.nd  Upper approximations of clusters using the interval dominance rule, a list of
length c. Each element upper.approx.nd[[i]] is a vector of objects.
N  Average nonspecificity.
crit  The value of the optimized criterion (depends on the method used).
Kmat  The matrix of degrees of conflict. Same size as D (for method kevclus).
D  The normalized dissimilarity matrix (for method kevclus).
trace  The trace of criterion values (for methods kevclus and EkNNclus).
W  The weight matrix (for method EkNNclus).

References

T. Denoeux and O. Kanjanatarakul. Beyond Fuzzy, Possibilistic and Rough: An Investigation of
Belief Functions in Clustering. 8th International conference on soft methods in probability and
M.-H. Masson and T. Denoeux. ECM: An evidential version of the fuzzy c-means algorithm. Pattern
Available from https://www.hds.utc.fr/~tdenoeux.

See Also

plot.credpart, summary.credpart

Examples

## Four-class data
data(fourclass)
x<-fourclass[,1:2]
y<-fourclass[,3]
D<-as.matrix(dist(x))^2
clus<-recm(D,c=4,delta=10,ntrials=1)
summary(clus)
plot(clus,X=x,mfrow=c(1,1),ytrue=y,Outliers=TRUE)
fourclass  

*Synthetic four-class dataset*

**Description**

A synthetic dataset with two attributes and four classes of 100 points each, generated from a multivariate $t$ distribution with five degrees of freedom and centered, respectively, on $[0;0]$, $[0;4]$, $[4;0]$ and $[4;4]$.

**Usage**

```r
data(fourclass)
```

**Format**

A data frame with three variables: $x_1$, $x_2$ and $y$ (the true class).

**References**


**Examples**

```r
data(fourclass)
plot(fourclass$x1,fourclass$x2,xlab=expression(x[1]),ylab=expression(x[2]),
col=fourclass$y,pch=fourclass$y)
```

kevclus  

*k-EVCLUS algorithm*

**Description**

kevclus computes a credal partition from a dissimilarity matrix using the k-EVCLUS algorithm.

**Usage**

```r
text = "\"simple\"",
pairs = NULL,
m0 = NULL,"
ntrials = 1,
disp = TRUE,
maxit = 1000,
epsi = 1e-05,
d0 = quantile(D, 0.9),
tr = FALSE,
change.order = FALSE
)

Arguments

- **x**: nxp matrix of p attributes observed for n objects (optional).
- **k**: Number of distances to compute for each object (default: n).
- **D**: nxn or nxk dissimilarity matrix (used only if x is not supplied).
- **J**: nxk matrix of indices. D[i,j] is the distance between objects i and J[i,j]. (Used only if D is supplied and ncol(D)=n; then k is set to ncol(D).)
- **c**: Number of clusters
- **type**: Type of focal sets ("simple": empty set, singletons and Omega; "full": all \(2^c\) subsets of \(\Omega\); "pairs": \(\emptyset\), singletons, \(\Omega\), and all or selected pairs).
- **pairs**: Set of pairs to be included in the focal sets; if NULL, all pairs are included. Used only if type="pairs".
- **m0**: Initial credal partition. Should be a matrix with n rows and a number of columns equal to the number f of focal sets specified by 'type' and 'pairs'.
- **ntrials**: Number of runs of the optimization algorithm (set to 1 if m0 is supplied and change.order=FALSE).
- **disp**: If TRUE (default), intermediate results are displayed.
- **maxit**: Maximum number of iterations.
- **epsi**: Minimum amount of improvement.
- **d0**: Parameter used for matrix normalization. The normalized distance corresponding to d0 is 0.95.
- **tr**: If TRUE, a trace of the stress function is returned.
- **change.order**: If TRUE, the order of objects is changed at each iteration of the Iterative Row-wise Quadratic Programming (IRQP) algorithm.

Details

This version of the EVCLUS algorithm uses the Iterative Row-wise Quadratic Programming (IRQP) algorithm (see ter Braak et al., 2009). It also makes it possible to use only a random sample of the dissimilarities, reducing the time and space complexity from quadratic to roughly linear (Denoeux et al., 2016). The user must supply: 1) a matrix x or size (n,p) containing the values of p attributes for n objects, or 2) a matrix D of size (n,n) of dissimilarities between n objects, or 3) a matrix D of size (n,k) of dissimilarities between the n objects and k randomly selected objects, AND a matrix J of size (n,k) of indices, such that D[i,j] is the distance between objects i and J[i,j]. In cases 1 and 2, the user may supply the number \(k\) of distances to be picked randomly for each object. In case 3, \(k\) is set to the number of columns of D.
Value

The credal partition (an object of class "credpart"). In addition to the usual attributes, the output credal partition has the following attributes:

**Kmat**  The matrix of degrees of conflict. Same size as D.
**D**  The normalized dissimilarity matrix.
**trace**  Trace of the algorithm (Stress function vs iterations).

Author(s)

Thierry Denoeux.

References


Available from [https://www.hds.utc.fr/~tdenoeux](https://www.hds.utc.fr/~tdenoeux).

See Also

`createD`, `makeF`, `extractMass`

Examples

```r
## Example with a non metric dissimilarity matrix: the Protein dataset
## Not run:
data(protein)
clus <- kevclus(D=protein$D,c=4,type='simple',d0=max(protein$D))
z<- cmdscale(protein$D,k=2)  # Computation of 2 attributes by Multidimensional Scaling
plot(clus,X=z,mfrow=c(2,2),ytrue=protein$y,Outliers=FALSE,Approx=1)

## Example with k=30
clus <- kevclus(D=protein$D,k=30,c=4,type='simple',d0=max(protein$D))
z<- cmdscale(protein$D,k=2)  # Computation of 2 attributes by Multidimensional Scaling
plot(clus,X=z,mfrow=c(2,2),ytrue=protein$y,Outliers=FALSE,Approx=1)

## End(Not run)
```
**knn_dist**

*K nearest neighbors in a dissimilarity matrix*

**Description**

knn_dist searches for nearest neighbors in a dissimilarity matrix matrix.

**Usage**

`knn_dist(D, K)`

**Arguments**

- `D`  
  Dissimilarity matrix of size (n,n), where n is the number of objects.
- `K`  
  Number of neighbors

**Details**

This function is called by `EkNNclus` if argument x is not supplied. It is not optimized and cannot be used for very large D. If an attribute matrix x is supplied and D is the matrix of Euclidean distances, it is preferable to use function `get.knn` from package FNN.

**Value**

A list with two components:

- `nn.dist`  
  An (n,K) matrix for the nearest neighbor dissimilarities.
- `nn.index`  
  An (n,K) matrix for the nearest neighbor indices.

**Author(s)**

Thierry Denoeux.

**See Also**

`get.knn`, `EkNNclus`

**Examples**

```r
data(butterfly)
n <- nrow(butterfly)
D <- as.matrix(dist(butterfly))
knn <- knn_dist(D, K=2)
knn$nn.dist
knn$nn.index
```
makeF  

*Creation of a matrix of focal sets*

**Description**

`makeF` creates a matrix of focal sets.

**Usage**

```r
makeF(c, type = c("simple", "full", "pairs"), pairs = NULL, Omega = TRUE)
```

**Arguments**

- `c`: Number of clusters.
- `type`: Type of focal sets ("simple": ∅, singletons and Ω; "full": all $2^c$ subsets of Ω; "pairs": ∅, singletons, Ω, and all or selected pairs).
- `pairs`: Set of pairs to be included in the focal sets; if NULL, all pairs are included. Used only if type="pairs".
- `Omega`: Logical. If TRUE (default), Ω is a focal set (for types 'simple' and 'pairs').

**Value**

A matrix (f,c) of focal sets.

**Examples**

```r
c<-4
## Generation of all 16 focal sets
F<-makeF(c,type='full')
## Generation of focal sets of cardinality 0, 1 and c
F<-makeF(c,type='simple')
## Generation of focal sets of cardinality 0, 1, and 2
F<-makeF(c,type='pairs',Omega=FALSE)
## Generation of focal sets of cardinality 0, 1, and c, plus the pairs (1,2) and (1,3)
F<-makeF(c,type='pairs',pairs=matrix(c(1,2,1,3),nrow=2,byrow=TRUE))
```

---

plot.credpart  

*Plotting a credal partition*

**Description**

Generates plots of a credal partition.
Usage

```r
## S3 method for class 'credpart'
plot(
  x,
  X = NULL,
  ..., 
  mfrow = c(1, 1),
  ytrue = NULL,
  Outliers = TRUE,
  Approx = 1,
  cex = 0.7,
  cex_outliers = 1.3,
  lwd = 2,
  ask = FALSE
)
```

Arguments

- `x`: An object of class "credpart", encoding a credal partition.
- `X`: A data matrix. If it has more than two columns (attributes), only the first two columns are used.
- `...`: Other arguments to be passed to the plot function.
- `mfrow`: A 2-vector defining the number of rows and columns of the plot. If `mfrow=c(1,1)`, only one figure is drawn. Otherwise, `mfrow[1] x mfrow[2]` should not be less than `x`, the number of clusters.
- `ytrue`: The vector of true class labels. If not supplied, the hard partition corresponding to the maximum plausibility is used instead.
- `Outliers`: If `TRUE`, the outliers are plotted, and they are not included in the lower and upper approximations of the clusters.
- `Approx`: If `Approx==1` (default), the lower and upper cluster approximations are computed using the interval dominance rule. Otherwise, the maximum mass rule is used.
- `cex`: Size of data points.
- `cex_outliers`: Size of data points for outliers.
- `lwd`: Line width for drawing the lower and upper approximations.
- `ask`: Logical; if `TRUE`, the user is asked before each plot.

Details

This function plots the hard and rough partitions (lower and upper approximations) extracted from a credal partition, together with two dimensional attribute data.
Value

The maximum plausibility hard partition, as well as the lower and upper approximations of each cluster are drawn in the two-dimensional space specified by matrix $X$. If prototypes are defined (for methods "ecm" and "cecm"), they are also represented on the plot. For method "kevclus", a second plot with Shepard’s diagram (degrees of conflict vs. transformed dissimilarities) is drawn. If input X is not supplied, and method="kevclus", then only the Shepard diagram is drawn.

References


Available from https://www.hds.utc.fr/~tdenoeux.

See Also

extractMass, summary.credpart, ecm, recm, cecm, kevclus.

Examples

```r
## Example with Four-class data
data("fourclass")
x<-fourclass[,1:2]
y<-fourclass[,3]
c=4
## Running k-EVCLUS with singletons
clus<-kevclus(x=x,k=100,c=c,type='simple')
## Plot the results
plot(clus,X=x,mfrow=c(2,2),ytrue=y)
```

protein

<table>
<thead>
<tr>
<th>Protein dataset</th>
</tr>
</thead>
</table>

Description

This real data set consists of a dissimilarity matrix derived from the structural comparison of 213 protein sequences. Each of these proteins is known to belong to one of four classes of globins: hemoglobin-alpha (HA), hemoglobin-beta (HB), myoglobin (M) and heterogeneous globins (G).

Usage

data(protein)
recm

Format
A list with three elements:

- **D**: The 213x213 dissimilarity matrix.
- **class**: A 213-vector containing the class encoded as a factor with four levels: "G", "HA", "HB", "M".
- **y**: A 213-vector containing the class encoded by an integer between 1 and 4.

References


Examples

```r
data(protein)
z <- cmdscale(protein$D, k=2)  # Multidimensional scaling
plot(z[,1],z[,2],xlab=expression(z[1]),ylab=expression(z[2]),pch=protein$y,col=protein$y)
```

rebm  
Relational Evidential c-means algorithm

Description

rebm computes a credal partition from a dissimilarity matrix using the Relational Evidential c-means (RECM) algorithm.

Usage

```r
rebm(
D,
c,
type = "full",
pairs = NULL,
Omega = TRUE,
m0 = NULL,
ntrials = 1,
alpha = 1,
beta = 1.5,
delta2 = quantile(D[upper.tri(D) | lower.tri(D)], 0.95),
epsi = 1e-04,
maxit = 5000,
disp = TRUE
)
```
Arguments

D  Dissimilarity matrix of size (n,n), where n is the number of objects. Dissimilarities must be squared Euclidean distances to ensure convergence.

c  Number of clusters.

type  Type of focal sets ("simple": empty set, singletons and Omega; "full": all $2^c$ subsets of $\Omega$; "pairs": $\emptyset$, singletons, $\Omega$, and all or selected pairs).

pairs  Set of pairs to be included in the focal sets; if NULL, all pairs are included. Used only if type="pairs".

Omega  Logical. If TRUE (default), the whole frame is included (for types 'simple' and 'pairs').

m0  Initial credal partition. Should be a matrix with n rows and a number of columns equal to the number f of focal sets specified by 'type' and 'pairs'.

ntrials  Number of runs of the optimization algorithm (set to 1 if m0 is supplied).

alpha  Exponent of the cardinality in the cost function.

beta  Exponent of masses in the cost function.

delta2  Squared distance to the empty set.

epsi  Minimum amount of improvement.

maxit  Maximum number of iterations.

disp  If TRUE (default), intermediate results are displayed.

Details

RECM is a relational version of the Evidential c-Means (ECM) algorithm. Convergence is guaranteed only if elements of matrix D are squared Euclidean distances. However, the algorithm is quite robust and generally provides sensible results even if the dissimilarities are not metric. By default, each mass function in the credal partition has $2^c$ focal sets, where c is the supplied number of clusters. We can also limit the number of focal sets to subsets of clusters with cardinalities 0, 1 and c (recommended if c>=10), or to all or some selected pairs of clusters. If an initial credal partition m0 is provided, the number of trials is automatically set to 1.

Value

The credal partition (an object of class "credpart").

Author(s)

Thierry Denoeux (from a MATLAB code written by Marie-Helene Masson).

References

See Also

makeF, extractMass, ecm

Examples

## Clustering of the Butterfly dataset

```r
n <- nrow(butterfly)
D <- as.matrix(dist(butterfly))^2
c <- recm(D, c=2, delta2=50)
m <- cclus$mass
plot(1:n, m[,1], type="l", ylim=c(0,1), xlab="objects", ylab="masses")
lines(1:n, m[,2], lty=2)
lines(1:n, m[,3], lty=3)
lines(1:n, m[,4], lty=4)
```

## Clustering the protein data

```r
data(protein)
c <- recm(D=protein$D, c=4, type="full", alpha=0.2, beta=1.1, delta2=20)
```

```r
z <- cmdscale(protein$D, k=2)
plot(clus, X=z, mfrow=c(2,2), ytrue=protein$y, Outliers=FALSE, Approx=1)
```

---

**s2**

**S2 dataset**

Description

This dataset contains 5000 two-dimensional vectors grouped in 15 Gaussian clusters.

Usage

```r
data(s2)
```

Format

A matrix with 5000 rows and two columns.

References


**Examples**

```r
data(s2)
plot(s2[,1],s2[,2],xlab=expression(x[1]),ylab=expression(x[2]))
```

---

**Summary of a credal partition**

**Description**

`summary.credpart` is the summary method for "credpart" objects.

**Usage**

```r
## S3 method for class 'credpart'
summary(object, ...)
```

**Arguments**

- `object` An object of class "credpart", encoding a credal partition.
- `...` Additional arguments (not used).

**Details**

This function extracts basic information from "credpart" objects, such as created by `ecm`, `recm`, `cecm`, `EkNNclus` or `kevclus`.

**Value**

Prints basic information on the credal partition.

**References**


Available from https://www.hds.utc.fr/~tdenoeux.

**See Also**

`extractMass`, `plot.credpart`, `ecm`, `recm`, `cecm`, `EkNNclus`, `kevclus`. 
Examples

## Example with Four-class data
```r
data("fourclass")
x<-fourclass[,1:2]
y<-fourclass[,3]
c=4
## Running k-EVCLUS with singletons
clus<-kevclus(x=x,k=100,c=c,type='simple')
summary(clus)
```
Index

*Topic datasets
   butterfly, 3
   fourclass, 16
   protein, 22
   s2, 25

bootclus, 2
butterfly, 3

cecm, 3, 4, 8, 9, 11, 13, 22, 26
create_MLCL, 6, 8
createD, 6, 18
createPairs, 7

ecm, 3, 6, 8, 9, 13, 22, 25, 26
EKNNclus, 11, 13–15, 19, 26
evclust, 12
extractMass, 6, 8, 11, 13, 18, 22, 25, 26

fourclass, 16

get.knn, 19

kevclus, 3, 7, 8, 13–15, 16, 22, 26
knn_dist, 19

makeF, 6, 11, 18, 20, 25

plot.credpart, 11, 14, 15, 20, 26
protein, 22

recm, 3, 6, 8, 11, 13, 22, 23, 26

s2, 25
summary.credpart, 14, 15, 22, 26