Package: MixAll (via r-universe)

October 13, 2024

Type Package

Encoding UTF-8

Version 1.5.16

Date 2024-04-26

Title Clustering and Classification using Model-Based Mixture Models

Author Serge Iovleff [aut, cre], Parmeet Bathia [ctb]

Maintainer Serge Iovleff <Serge.Iovleff@stkpp.org>

Copyright inria, University of Lille

Depends R (>= 4.1.0), rtkore (>= 1.6.10)

Imports methods

Description Algorithms and methods for model-based clustering and classification. It supports various types of data: continuous, categorical and counting and can handle mixed data of these types. It can fit Gaussian (with diagonal covariance structure), gamma, categorical and Poisson models. The algorithms also support missing values.

License GPL (>= 2)

LinkingTo Rcpp, rtkore (>= 1.6.10)

SystemRequirements GNU make

Collate 'ClusterAlgo.R' 'ClusterAlgoPredict.R' 'ClusterInit.R' 'ClusterStrategy.R' 'IClusterModel.R' 'ClusterModelNames.R' 'global.R' 'ClusterCategorical.R' 'ClusterDiagGaussian.R' 'ClusterGamma.R' 'ClusterMixedData.R' 'ClusterPlot.R' 'ClusterPoisson.R' 'IClusterPredict.R' 'ClusterPredict.R' 'LearnAlgo.R' 'Learners.R' 'MixAll.R' 'kmmNames.R' 'kmm.R' 'kmmMixedData.R' 'missingValues.R'

RoxygenNote 7.3.1

NeedsCompilation yes

Repository CRAN

Date/Publication 2024-05-15 08:20:02 UTC

Contents

`MixAll-package` *MixAll Allows to estimate parametric mixture models with mixed data sets and missing data.*

Description

This package contains methods allowing R users to use the clustering methods of the STK++ library.

Details

As described at the STK++ project's home page, https://www.stkpp.org, STK++ is a versatile, fast, reliable and elegant collection of C++ classes for statistics, clustering, linear algebra, arrays (with an Eigen-like API), regression, dimension reduction, etc. Some functionalities provided by the library are available in the R environment as R functions in MixAll.

The available functionalities are:

1. Clustering (clusterDiagGaussian, clusterCategorical, clusterPoisson, clusterGamma, clusterMixedData)
2. Learning ((learnDiagGaussian, learnCategorical, learnPoisson, learnGamma, learnMixedData),
3. Prediction (clusterPredict).

Author(s)

Serge Iovleff

birds *Qualitative data : morphological description of birds*

Description

The data set contains details on the morphology of birds (puffins). Each individual (bird) is described by 6 qualitative variables. One variable for the gender and 5 variables giving a morphological description of the birds. There is 69 puffins divided in 2 sub-classes: lherminieri (34) and subalaris (35).

Format

A data frame with 69 observations on the following 5 variables.

`gender` a character vector defining the gender (2 modalities, male or female).

`eyebrow` a character vector describing the eyebrow stripe (4 modalities).

`collar` a character vector describing the collar (5 modalities).

`sub-caudal` a character vector describing the sub-caudal (5 modalities).

`border` a character vector describing the border (3 modalities).

Details

This data set is also part of the Rmixmod package.

Source

Bretagnolle, V., 2007. Personal communication, source: Museum.

Examples

```
data(birds)
```

bullsEye *Quantitative Data: bullsEye*

Description

Generated data set containing two clusters with untypical ring shapes (circles)

Examples

```
data(bullsEye)
```

bullsEye.cat *label Data: bullsEye.cat*

Description

Generated data set containing two categorical variables for the two clusters with untypical ring shapes (circles)

Examples

```
data(bullsEye.cat)
```

bullsEye.target *label Data: bullsEye.target*

Description

Generated data set containing labels for the two clusters with untypical ring shapes (circles)

Examples

```
data(bullsEye.target)
```

car *Qualitative data : Car Evaluation*

Description

Car Evaluation Database was derived from a simple hierarchical decision model originally developed for the demonstration of DEX, M. Bohanec, V. Rajkovic: Expert system for decision making.

Format

A data frame with 1728 observations on the following 6 variables.

`buying` the buying price (4 modalities: vhigh, high, med, low)

`maint` the price of the maintenance (4 modalities: vhigh, high, med, low)

`doors` the number of doors (4 modalities: 2, 3, 4, 5more)

`persons` the capacity in terms of persons to carry (3 modalities: 2, 4, more)

`lug_boot` the size of luggage boot (3 modalities: small, med, big)

`safety` the estimated safety of the car (3 modalities: low, med, high)

`acceptability` the car acceptability (4 modalities: unacc, acc, good, vgood)

Source

Creator: Marko Bohanec Donors: Marko Bohanec & Blaz Zupan http://archive.ics.uci.edu/ml/datasets/Car+Evaluation

Examples

```
data(car)
```

`ClusterAlgo` *[ClusterAlgo] class for Cluster algorithms.*

Description

This class encapsulates the parameters of clustering estimation algorithms methods.

Slots

`algo` A character string with the algorithm. Possible values: "SEM", "CEM", "EM", "SemiSEM". Default value: "EM".

`nbIteration` Integer defining the maximal number of iterations. Default value: 200.

`epsilon` real defining the epsilon value for the algorithm. epsilon is note used if `algo` is "SEM" or "SemiSEM". Default value: 1e-07.

Examples

```
getSlots("ClusterAlgo")
new("ClusterAlgo")
new("ClusterAlgo", algo="SEM", nbIteration=1000)
```

`clusterAlgo` *Create an instance of the [ClusterAlgo] class*

Description

There is three algorithms and two stopping rules possibles for an algorithm.

- Algorithms:
  - `EM`: The Expectation Maximisation algorithm
  - `CEM`: The Classification EM algorithm
  - `SEM`: The Stochastic EM algorithm
  - `SemiSEM`: The Semi-Stochastic EM algorithm
- Stopping rules:
  - `nbIteration`: Set the maximum number of iterations
  - `epsilon`: Set relative increase of the log-likelihood criterion
- Default values are 200 `nbIteration` of `EM` with an `epsilon` value of $1.e-8$.

The epsilon value is not used when the algorithm is "SEM" or "SemiSEM".

Usage

```
clusterAlgo(algo = "EM", nbIteration = 200, epsilon = 1e-07)
```

Arguments

algo	character string with the estimation algorithm. Possible values are "EM", "SEM", "CEM", "SemiSEM". Default value is "EM".
nbIteration	Integer defining the maximal number of iterations. Default value is 200.
epsilon	Real defining the epsilon value for the algorithm. Not used by the "SEM" and "SemiSEM" algorithms. Default value is 1.e-7.

Value

a [ClusterAlgo] object

Author(s)

Serge Iovleff

Examples

```
clusterAlgo()
clusterAlgo(algo="SEM", nbIteration=50)
clusterAlgo(algo="CEM", epsilon = 1e-06)
```

ClusterAlgoPredict *[ClusterAlgoPredict] class for predict algorithm.*

Description

This class encapsulates the parameters of prediction methods.

Slots

- `algo` A character string with the algorithm. Possible values: "EM", "SemiSEM". Default value: "SemiSEM".
- `nbIterBurn` Integer defining the number of burning iterations. Default value is 50.
- `nbIterLong` Integer defining the number of iterations. Default value is 100.
- `epsilon` real defining the epsilon value for the long algorithm. epsilon is note used if `algo` is "SemiSEM". Default value: 1e-07.

Examples

```
getSlots("ClusterAlgoPredict")
new("ClusterAlgoPredict")
new("ClusterAlgoPredict", algo="SemiSEM", nbIterBurn=10)
```

clusterAlgoPredict *Create an instance of the [ClusterAlgoPredict] class*

Description

A prediction algorithm is a two stage algorithm. In the first stage we perform a Monte Carlo algorithm for simulating both missing values and latent class variables. In the second stage, we simulate or impute missing values.

Usage

```
clusterAlgoPredict(
  algo = "EM",
  nbIterBurn = 50,
  nbIterLong = 100,
  epsilon = 1e-07
)
```

Arguments

algo	character string with the second stage estimation algorithm. Possible values are "EM", "SemiSEM". Default value is "EM".
nbIterBurn	Integer defining the maximal number of burning iterations. Default value is 50.
nbIterLong	Integer defining the maximal number of iterations. Default value is 100.
epsilon	Real defining the epsilon value for the algorithm. Not used with "semiSEM" algorithms. Default value is 1.e-7.

Details

The epsilon value is not used when the algorithm is "SemiSEM".

Value

a [ClusterAlgoPredict] object

Author(s)

Serge Iovleff

Examples

```
clusterAlgoPredict()
clusterAlgoPredict(algo="SemiSEM", nbIterBurn=0)
clusterAlgoPredict(algo="EM", epsilon = 1e-06)
```

clusterCategorical *Create an instance of the [ClusterCategorical] class*

Description

This function computes the optimal Categorical mixture model according to the `criterion` among the list of model given in `models` and the number of clusters given in `nbCluster`, using the strategy specified in `strategy`.

Usage

```
clusterCategorical(
  data,
  nbCluster = 2,
  models = clusterCategoricalNames(probabilities = "free"),
  strategy = clusterStrategy(),
  criterion = "ICL",
  nbCore = 1
)
```

Arguments

`data`	a data.frame or a matrix containing the data. Rows correspond to observations and columns correspond to variables. data will be coerced as an integer matrix. If data set contains NA values, they will be estimated during the estimation process.
`nbCluster`	[vector] listing the number of clusters to test.
`models`	[vector] of model names to run. By default the categorical models "categorical_pk_pjk" and "categorical_p_pjk" are estimated.
`strategy`	a [ClusterStrategy] object containing the strategy to run. [clusterStrategy]() method by default.
`criterion`	character defining the criterion to select the best model. The best model is the one with the lowest criterion value. Possible values: "BIC", "AIC", "ICL", "ML". Default is "ICL".
`nbCore`	integer defining the number of processors to use (default is 1, 0 for all).

Value

An instance of the [ClusterCategorical] class.

Author(s)

Serge Iovleff

Examples

```
## A quantitative example with the birds data set
data(birds)
## add 10 missing values
x = as.matrix(birds); n <- nrow(x); p <- ncol(x)
indexes <- matrix(c(round(runif(5,1,n)), round(runif(5,1,p))), ncol=2)
x[indexes] <- NA
## estimate model (using fast strategy, results may be misleading)
model <- clusterCategorical( data=x, nbCluster=2:3
                           , models=c( "categorical_pk_pjk", "categorical_p_pjk")
                           , strategy = clusterFastStrategy()
                           )

## use graphics functions

plot(model)


## get summary
summary(model)

## print model (a detailed and very long output)
print(model)

## get estimated missing values
missingValues(model)
```

ClusterCategorical-class

Definition of the [ClusterCategorical] class

Description

This class defines a categorical mixture model. It inherits from the [IClusterModel] class. A categorical mixture model is a mixture model of the form

Details

$$f(x|\boldsymbol{\theta}) = \sum_{k=1}^{K} p_k \prod_{j=1}^{d} \mathcal{M}(x_j; p_{jk}, 1) \quad x \in \{1, \ldots, L\}^d.$$

The probabilities can be assumed equal between all variables in order to reduce the number of parameters.

Slots

`component` A [ClusterCategoricalComponent] with the probabilities of the categorical component

Author(s)

Serge Iovleff

Examples

```
getSlots("ClusterCategorical")
data(birds)
new("ClusterCategorical", data=birds)
```

`ClusterCategoricalComponent`
Definition of the [ClusterCategoricalComponent] class

Description

This class defines a categorical component of a mixture model. It inherits from [IClusterComponent].

Slots

`plkj` Array with the probability for the jth variable in the kth cluster to be l.

`nbModalities` Integer with the (maximal) number of modalities of the categorical data.

`levels` list with the original levels of the variables

Author(s)

Serge Iovleff

See Also

[IClusterComponent] class

Examples

```
getSlots("ClusterCategoricalComponent")
```

clusterCategoricalNames *Create a vector of Categorical mixture model names.*

Description

In a Categorical mixture model, we can build 4 models:

1. The proportions can be equal or free
2. The probabilities can be equal or free for all the variables

Usage

```
clusterCategoricalNames(prop = "all", probabilities = "all")

clusterValidCategoricalNames(names)
```

Arguments

prop	A character string equal to "equal", "free" or "all". Default is "all".
probabilities	A character string equal to "equal", "free" or "all". Default is "all".
names	a vector of character

Details

The model names are summarized in the following array:

Model Name	Proportions	Probabilities between variables
categorical_p_pjk	Equal	Free
categorical_p_pk	Equal	Equal
categorical_pk_pjk	Free	Free
categorical_pk_pk	Free	Equal

Value

A vector of character with the model names.

Examples

```
clusterCategoricalNames()
clusterCategoricalNames("all", "equal") # same as c( "categorical_pk_pk", "categorical_p_pk")
```

ClusterDiagGaussian *Definition of the [ClusterDiagGaussian] class*

Description

This class defines a diagonal Gaussian mixture Model.

Details

This class inherits from the [IClusterModel] class. A diagonal gaussian model is a mixture model of the form:

$$f(x|\boldsymbol{\theta}) = \sum_{k=1}^{K} p_k \prod_{j=1}^{d} \phi(x_j; \mu_{jk}, \sigma_{jk}^2) \quad x \in R^d.$$

Some constraints can be added to the variances in order to reduce the number of parameters.

Slots

`component` A [ClusterDiagGaussianComponent] with the mean and standard deviation of the diagonal mixture model.

Author(s)

Serge Iovleff

See Also

[IClusterModel] class

Examples

```
getSlots("ClusterDiagGaussian")
data(geyser)
new("ClusterDiagGaussian", data=geyser)
```

---

clusterDiagGaussian *Create an instance of the [ClusterDiagGaussian] class*

Description

This function computes the optimal diagonal Gaussian mixture model according to the `criterion` among the list of model given in `models` and the number of clusters given in `nbCluster`, using the strategy specified in `strategy`.

Usage

```
clusterDiagGaussian(
  data,
  nbCluster = 2,
  models = clusterDiagGaussianNames(),
  strategy = clusterStrategy(),
  criterion = "ICL",
  nbCore = 1
)
```

Arguments

data	frame or matrix containing the data. Rows correspond to observations and columns correspond to variables. If the data set contains NA values, they will be estimated during the estimation process.
nbCluster	[vector] listing the number of clusters to test.
models	[vector] of model names to run. By default all diagonal Gaussian models are estimated. All the model names are given by the method [clusterDiagGaussianNames].
strategy	a [ClusterStrategy] object containing the strategy to run. [clusterStrategy]() method by default.
criterion	character defining the criterion to select the best model. The best model is the one with the lowest criterion value. Possible values: "BIC", "AIC", "ICL", "ML". Default is "ICL".
nbCore	integer defining the number of processors to use (default is 1, 0 for all).

Value

An instance of the [ClusterDiagGaussian] class.

Author(s)

Serge Iovleff

Examples

```
## A quantitative example with the famous geyser data set
data(geyser)
## add 10 missing values as random
x = as.matrix(geyser); n <- nrow(x); p <- ncol(x);
indexes <- matrix(c(round(runif(5,1,n)), round(runif(5,1,p))), ncol=2);
x[indexes] <- NA;
## estimate model (using fast strategy, results may be misleading)
model <- clusterDiagGaussian( data=x, nbCluster=2:3
                            , models=c( "gaussian_pk_sjk")
                            , strategy = clusterFastStrategy()
                            )

## use graphics functions
```

```
plot(model)


## get summary
summary(model)

## print model (a detailed and very long output)
print(model)

## get estimated missing values
missingValues(model)
```

ClusterDiagGaussianComponent

Definition of the [ClusterDiagGaussianComponent] class

Description

This class defines a diagonal Gaussian component of a mixture model. It inherits from [IClusterComponent].

Slots

`mean` Matrix with the mean of the jth variable in the kth cluster.

`sigma` Matrix with the standard deviation of the jth variable in the kth cluster.

Author(s)

Serge Iovleff

See Also

[IClusterComponent] class

Examples

```
getSlots("ClusterDiagGaussianComponent")
```

`clusterDiagGaussianNames` *Create a vector of diagonal Gaussian mixture model names.*

Description

In a diagonal Gaussian mixture model, we assume that the variance matrices are diagonal in each cluster. Assumptions on the proportions and standard deviations give rise to 8 models:

1. The proportions can be equal or free
2. The standard deviations can be equal or free for all the variables
3. The standard deviations can be equal or free for all the clusters

Usage

```
clusterDiagGaussianNames(
  prop = "all",
  sdInCluster = "all",
  sdBetweenCluster = "all"
)

clusterValidDiagGaussianNames(names)
```

Arguments

`prop`	A character string equal to "equal", "free" or "all". Default is "all".
`sdInCluster`	A character string equal to "equal", "free" or "all". Default is "all".
`sdBetweenCluster`	A character string equal to "equal", "free" or "all". Default is "all".
`names`	a vector of character

Details

The model names are summarized in the following array:

Model Name	Proportions	s.d. in variables	s.d. in clusters
gaussian_p_sjk	Equal	Free	Free
gaussian_p_sj	Equal	Free	Equal
gaussian_p_sk	Equal	Equal	Free
gaussian_p_s	Equal	Equal	Equal
gaussian_pk_sjk	Free	Free	Free
gaussian_pk_sj	Free	Free	Equal
gaussian_pk_sk	Free	Equal	Free
gaussian_pk_s	Free	Equal	Equal

Value

A vector of character with the model names.

Examples

```
clusterDiagGaussianNames()
## same as c("gaussian_p_sk", "gaussian_pk_sk")
clusterDiagGaussianNames(prop="all", sdInCluster="equal", sdBetweenCluster= "free")
```

ClusterGamma *Definition of the [ClusterGamma] class*

Description

This class inherits from the [IClusterModel] class. A gamma mixture model is a mixture model of the form:

$$f(x|\boldsymbol{\theta}) = \sum_{k=1}^{K} p_k \prod_{j=1}^{d} \gamma(x_j; a_{jk}, b_{jk}) \quad x \in R^d.$$

Constraints can be added to the shapes and/or scales in order to reduce the number of parameters.

Slots

`component` A [ClusterGammaComponent] with the shapes and the scales of the component mixture model.

Author(s)

Serge Iovleff

See Also

[IClusterModel] class

Examples

```
getSlots("ClusterGamma")
data(geyser)
new("ClusterGamma", data=geyser)
```

clusterGamma *Create an instance of the [ClusterGamma] class*

Description

This function computes the optimal gamma mixture model according to the `criterion` among the list of model given in `models` and the number of clusters given in `nbCluster`, using the strategy specified in `strategy`.

Usage

```
clusterGamma(
  data,
  nbCluster = 2,
  models = "gamma_pk_ajk_bjk",
  strategy = clusterStrategy(),
  criterion = "ICL",
  nbCore = 1
)
```

Arguments

`data`	frame or matrix containing the data. Rows correspond to observations and columns correspond to variables. If the data set contains NA values, they will be estimated during the estimation process.
`nbCluster`	[vector] listing the number of clusters to test.
`models`	[vector] of model names to run. By default all gamma models with free shape are estimated. All the model names are given by the method [clusterGammaNames].
`strategy`	a [ClusterStrategy] object containing the strategy to run. [clusterStrategy]() method by default.
`criterion`	character defining the criterion to select the best model. The best model is the one with the lowest criterion value. Possible values: "BIC", "AIC", "ICL", "ML". Default is "ICL".
`nbCore`	integer defining the number of processor to use (default is 1, 0 for all).

Value

An instance of the [ClusterGamma] class.

Author(s)

Serge Iovleff

Examples

```
## A quantitative example with the famous geyser data set
data(geyser)
## add 10 missing values
x = geyser;
x[round(runif(5,1,nrow(geyser))), 1] <- NA
x[round(runif(5,1,nrow(geyser))), 2] <- NA

## use graphics functions
set.seed(2)
model <- clusterGamma( data=x, nbCluster=2:3
                     , models="gamma_pk_ajk_bjk"
                     , strategy = clusterFastStrategy())

## use plot

plot(model)


## get summary
summary(model)

## print model (a detailed and very long output)
print(model)

## get estimated missing values
missingValues(model)
```

ClusterGammaComponent *Definition of the [ClusterGammaComponent] class*

Description

This class defines a gamma component of a mixture Model. It inherits from [IClusterComponent].

Slots

`shape` Matrix with the shapes of the jth variable in the kth cluster.

`scale` Matrix with the scales of the jth variable in the kth cluster.

Author(s)

Serge Iovleff

See Also

[IClusterComponent] class

Examples

```
getSlots("ClusterGammaComponent")
```

`clusterGammaNames` *Create a vector of gamma mixture model names.*

Description

In a gamma mixture model, we can assume that the shapes are equal in each/all cluster(s) or not. We can also assume that the scales are equal in each/all cluster(s) or not.

Usage

```
clusterGammaNames(
  prop = "all",
  shapeInCluster = "all",
  shapeBetweenCluster = "all",
  scaleInCluster = "all",
  scaleBetweenCluster = "all"
)

clusterValidGammaNames(names)
```

Arguments

`prop`	A character string equal to "equal", "free" or "all". Default is "all".
`shapeInCluster`	A character string equal to "equal", "free" or "all". Default is "all".
`shapeBetweenCluster`	A character string equal to "equal", "free" or "all". Default is "all".
`scaleInCluster`	A character string equal to "equal", "free" or "all". Default is "all".
`scaleBetweenCluster`	A character string equal to "equal", "free" or "all". Default is "all".
`names`	a vector of character

Details

Some configuration are impossibles. If the shapes are equal between all the clusters, then the scales cannot be equal between all the clusters. Conversely if the scales are equal between all the cluster, then the shapes cannot be equal between all the clusters.

This gives rise to 24 models:

1. The proportions can be equal or free
2. The shapes can be equal or free in each clusters
3. The shapes can be equal or free between all clusters

4. The scales can be equal or free for each clusters
5. The scales can be equal or free between all clusters

The model names are summarized in the following array:

& ajk & ak & aj & a
bjk & gamma_*_ajk_bjk & gamma_*_ak_bjk & gamma_*_aj_bjk & gamma_*_a_bjk
bk & gamma_*_ajk_bk & gamma_*_ak_bk & gamma_*_aj_bk & gamma_*_a_bk
bj & gamma_*_ajk_bj & gamma_*_ak_bj & NA & NA
b & gamma_*_ajk_b & gamma_*_ak_b & NA & NA

Value

A vector of character with the model names.

Examples

```
clusterGammaNames()
## same as c("gamma_p_ak_bj", "gamma_pk_ak_bj")
clusterGammaNames("all", "equal", "free", "free", "equal")
```

`ClusterInit` *Constructor of the [`ClusterInit`] class*

Description

This class encapsulates the parameters of clustering initialization methods.

Slots

`method` Character string with the initialization method to use. Default value: "class"

`nbInit` Integer defining the number of initialization to perform. Default value: 5.

`algo` An instance of `ClusterAlgo` class. Default value: `clusterAlgo("EM", 20, 0.01)`.

Author(s)

Serge Iovleff

Examples

```
getSlots("ClusterInit")
new("ClusterInit")
new("ClusterInit", nbInit=1)
```

clusterInit *Create an instance of [ClusterInit] class*

Description

The initialization step is a two stages process: the proper initialization step and some (optionnals) iterations of an algorithm [clusterAlgo].

Usage

```
clusterInit(
  method = "class",
  nbInit = 5,
  algo = "EM",
  nbIteration = 20,
  epsilon = 0.01
)
```

Arguments

method	Character string with the initialisation method. Possible values: "random", "class", "fuzzy". Default value is "class".
nbInit	integer defining the number of initialization point to test. Default value is 5.
algo	String with the initialisation algorithm. Possible values: "EM", "CEM", "SEM", "SemiSEM". Default value is "EM".
nbIteration	Integer defining the number of iteration in algo. nbIteration must be a positive integer. Default values is 20. if .
epsilon	threshold to use in order to stop the iterations. Default value is 0.01.

Details

There is three ways to initialize the parameters:

- random: The initial parameters of the mixture are chosen randomly
- class: The initial membership of individuals are sampled randomly
- fuzzy: The initial probabilities of membership of individuals are sampled randomly

A few iterations of an algorithm [clusterAlgo] are then performed. It is strongly recommended to use a few number of iterations of the EM or SEM algorithms after initialization. This allows to detect "bad" initialization starting point.

These two stages are repeated until nbInit is reached. The initial point with the best log-likelihood is conserved as the initial starting point.

Value

a [ClusterInit] object

Author(s)

Serge Iovleff

Examples

```
clusterInit(method = "class", nbInit=1, algo="CEM",nbIteration=50, epsilon=0.00001)
clusterInit(nbIteration=0) # no algorithm
```

---

clusterMixedData *Create an instance of the [ClusterMixedDataModel] class*

Description

This function computes the optimal mixture model for mixed data according to the `criterion` among the number of clusters given in `nbCluster` using the strategy specified in [`strategy`].

Usage

```
clusterMixedData(
  data,
  models,
  nbCluster = 2,
  strategy = clusterStrategy(),
  criterion = "ICL",
  nbCore = 1
)
```

Arguments

`data`	[`list`] containing the data sets (matrices and/or data.frames). If data sets contain NA values, these missing values will be estimated during the estimation process.
`models`	a [`vector`] of character or a [`list`] of same length than data. It contains the model names to use in order to fit each data set.
`nbCluster`	[`vector`] with the number of clusters to test.
`strategy`	a [`ClusterStrategy`] object containing the strategy to run. Default is clusterStrategy().
`criterion`	character defining the criterion to select the best model. The best model is the one with the lowest criterion value. Possible values: "BIC", "AIC", "ICL", "ML". Default is "ICL".
`nbCore`	integer defining the number of processors to use (default is 1, 0 for all).

Value

An instance of the [`ClusterMixedDataModel`] class.

Author(s)

Serge Iovleff

Examples

```
## A quantitative example with the heart disease data set
data(HeartDisease.cat)
data(HeartDisease.cont)
## with default values
ldata = list(HeartDisease.cat, HeartDisease.cont);
models = c("categorical_pk_pjk","gaussian_pk_sjk")
model <- clusterMixedData(ldata, models, nbCluster=2:5, strategy = clusterFastStrategy())

## get summary
summary(model)

## get estimated missing values
missingValues(model)


## print model (a very detailed output)
print(model)
## use graphics functions
plot(model)
```

ClusterMixedDataModel *Definition of the [ClusterMixedDataModel] class*

Description

This class defines a mixed data mixture Model.

Details

This class inherits from the [IClusterModel] class. A model for mixed data is a mixture model of the form:

$$f(x_i = (x_{1i}, x_{2i}, \dots x_{Li})|\theta) = \sum_{k=1}^{K} p_k \prod_{l=1}^{L} h(x_{li}|\lambda_{lk}, \alpha_l).$$

The density functions (or probability distribution functions)

$$h(.|\lambda_{lk}, \alpha_l)$$

can be any implemented model (Gaussian, Poisson,...).

Slots

`lcomponent` a list of [IClusterComponent]

Author(s)

Serge Iovleff

See Also

[IClusterModel] class

Examples

```
getSlots("ClusterMixedDataModel")
```

ClusterPoisson *Definition of the [ClusterPoisson] class*

Description

This class inherits from the [IClusterModel] class. A poisson mixture model is a mixture model of the form:

$$f(x|\boldsymbol{\theta}) = \sum_{k=1}^{K} p_k \prod_{j=1}^{d} Pois(x_j; \lambda_{jk}) \quad x \in N^d.$$

Slots

component A [ClusterPoissonComponent] with the lambda of the component mixture model.

Author(s)

Serge Iovleff

See Also

[IClusterModel] class

Examples

```
getSlots("ClusterPoisson")
data(DebTrivedi)
dt <- DebTrivedi[, c(1, 6,8, 15)]
new("ClusterPoisson", data=dt)
```

clusterPoisson *Create an instance of the [ClusterPoisson] class*

Description

This function computes the optimal poisson mixture model according to the [criterion] among the list of model given in [models] and the number of clusters given in [nbCluster], using the strategy specified in [strategy].

Usage

```
clusterPoisson(
  data,
  nbCluster = 2,
  models = clusterPoissonNames(),
  strategy = clusterStrategy(),
  criterion = "ICL",
  nbCore = 1
)
```

Arguments

data	a data.frame or matrix containing the data. Rows correspond to observations and columns correspond to variables. data will be coerced as an integer matrix. If data set contains NA values, they will be estimated during the estimation process.
nbCluster	[vector] listing the number of clusters to test.
models	[vector] of model names to run. By default all poisson models are estimated. All the model names are given by the method [clusterPoissonNames].
strategy	a [ClusterStrategy] object containing the strategy to run. [clusterStrategy]() method by default.
criterion	character defining the criterion to select the best model. The best model is the one with the lowest criterion value. Possible values: "BIC", "AIC", "ICL", "ML". Default is "ICL".
nbCore	integer defining the number of processor to use (default is 1, 0 for all).

Value

An instance of the [ClusterPoisson] class.

Author(s)

Serge Iovleff

Examples

```
## A quantitative example with the DebTrivedi data set.
data(DebTrivedi)
dt <- DebTrivedi[1:500, c(1, 6,8, 15)]

model <- clusterPoisson( data=dt, nbCluster=2
                       , models=clusterPoissonNames(prop = "equal")
                       , strategy = clusterFastStrategy())

## use graphics functions

plot(model)


## get summary
summary(model)

## print model (a very detailed output)
print(model)

## get estimated missing values
missingValues(model)
```

`clusterPoissonNames` *Create a vector of Poisson mixture model names.*

Description

In a Poisson mixture model, we can build 4 models:

1. The proportions can be equal or free
2. The means can be equal, free or proportional for all the variables

Usage

```
clusterPoissonNames(prop = "all", mean = "all")

clusterValidPoissonNames(names)
```

Arguments

`prop`	A character string equal to "equal", "free" or "all". Default is "all".
`mean`	A character string equal to "equal", "free", "proportional or "all". Default is "all".
`names`	a vector of character

Details

The model names are summarized in the following array:

Model Name	Proportions	Mean between variables
poisson_p_ljk	Equal	Free
poisson_p_lk	Equal	Equal
poisson_p_ljlk	Equal	Proportional
poisson_pk_ljk	Free	Free
poisson_pk_lk	Free	Equal
poisson_pk_ljlk	Free	Proportional

Value

A vector of character with the model names.

Examples

```
clusterPoissonNames()
clusterPoissonNames("all", "proportional") # same as c( "poisson_pk_ljlk", "poisson_p_ljlk")
```

ClusterPredict *Class [ClusterPredict] for predicting*

Description

This class encapsulate the parameters for predicted data.

Slots

`data` Matrix with the data set

`missing` Matrix with the indexes of the missing values

Author(s)

Serge Iovleff

See Also

[IClusterPredict] class

Examples

```
getSlots("ClusterPredict")
```

clusterPredict *Create an instance of [ClusterPredict] class*

Description

This function predicts the best cluster each sample in data belongs to.

Usage

```
clusterPredict(data, model, algo = clusterAlgoPredict(), nbCore = 1)
```

Arguments

data	dataframe or matrix containing the data. Rows correspond to observations and columns correspond to variables. If the data set contains NA values, they will be estimated during the predicting process.
model	(estimated) clustering model to use, i.e. an instance of ClusterCategorical, ClusterDiagGaussian,.. produced by clusterCategorical, clusterDiagGaussian,... learnCategorical, learnDiagGaussian, etc. functions.
algo	an instance of ClusterAlgoPredict S4 class. Will not be used if there is no missing values.
nbCore	integer defining the number of processors to use (default is 1, 0 for all).

Value

An instance of [ClusterPredict] with predicted values

Author(s)

Serge Iovleff

Examples

```
## A quantitative example with the famous iris data set
data(iris)
## get quantitatives
x = as.matrix(iris[1:4])
## sample train and test data sets
indexes <- sample(1:nrow(x), nrow(x)/2)
train <- x[ indexes,]
test  <- x[-indexes,]
## estimate model (using fast strategy, results may be misleading)
model1 <- clusterDiagGaussian( data =train, nbCluster=2:3
                             , models=c( "gaussian_p_sjk")
                             )
## get summary
summary(model1)
## compute prediction and compare
```

```
model2 <- clusterPredict(test, model1)
show(model2)
as.integer(iris$Species[-indexes])
```

`ClusterPredictMixedData` *Class [ClusterPredictMixedData] for predicting*

Description

This class encapsulate the parameters for predicted data.

Slots

`ldata` list of matrix with the data sets

`lmissing` list of matrix with the indexes of the missing values

Author(s)

Serge Iovleff

See Also

[IClusterPredict] class

Examples

```
getSlots("ClusterPredictMixedData")
```

`ClusterStrategy` *Constructor of [ClusterStrategy] class*

Description

This class encapsulate the parameters of the clustering estimation strategies.

Details

@slot nbTry Integer defining the number of tries. Default value: 1. @slot nbShortRun Integer defining the number of short run. Recall that the strategy launch an initialization before each short run. Default value is 5. @slot initMethod A [ClusterInit] object defining the way to initialize the estimation method. Default value is [ClusterInit]. @slot shortAlgo A [ClusterAlgo] object defining the algorithm to use during the short runs of the estimation method. Default value is `clusterAlgo("EM",100,1e-04)`. @slot longAlgo A [ClusterAlgo] object defining the algorithm to use during the long run of the estimation method. Default value is `clusterAlgo("EM",1000,1e-07)`.

Author(s)

Serge Iovleff

Examples

```
new("ClusterStrategy")
shortAlgo=clusterAlgo("SEM",1000)
longAlgo =clusterAlgo("SemiSEM",200,1e-07)
new("ClusterStrategy", shortAlgo=shortAlgo, longAlgo=longAlgo)
getSlots("ClusterStrategy")
```

`clusterStrategy` *A strategy is a multistage empirical process for finding a good estimate in the clustering estimation process.*

Description

A strategy is a way to find a good estimate of the parameters of a mixture model when using an EM algorithm or its variants. A "try" is composed of three stages

- `nbShortRun` short iterations of the initialization step and of the `EM`, `CEM`, `SEM` or `SemiSEM` algorithm.
- `nbInit` initializations using the [`clusterInit`] method.
- A long run of the `EM`, `CEM`, `SEM` or `SemiSEM` algorithm.

For example if `nbInit` is 5 and `nbShortRun` is also 5, there will be 5 packets of 5 models initialized. In each packet, the best model will be ameliorated using a short run. Among the 5 models ameliorated the best one will be estimated until convergence using a long run. In total there will be 25 initializations, 5 short runs and one long-run.

`clusterSemiSEMStrategy()` create an instance of [`ClusterStrategy`] for users with many missing values uning a semiSem algorithm.

`clusterSEMStrategy()` create an instance of [`ClusterStrategy`] for users with many missing values using a SEM algorithm.

`clusterFastStrategy()` create an instance of [`ClusterStrategy`] for impatient user.

Usage

```
clusterStrategy(
  nbTry = 1,
  nbInit = 5,
  initMethod = "class",
  initAlgo = "EM",
  nbInitIteration = 20,
  initEpsilon = 0.01,
  nbShortRun = 5,
```

```
  shortRunAlgo = "EM",
  nbShortIteration = 100,
  shortEpsilon = 1e-04,
  longRunAlgo = "EM",
  nbLongIteration = 1000,
  longEpsilon = 1e-07
)

clusterSemiSEMStrategy()

clusterSEMStrategy()

clusterFastStrategy()
```

Arguments

nbTry	number of estimation to attempt.
nbInit	Integer defining the number of initialization to try. Default value: 5.
initMethod	Character string with the initialization method, see [clusterInit]$ for possible values. Default value: "class".
initAlgo	Character string with the algorithm to use in the initialization stage, [clusterAlgo] for possible values. Default value: "EM".
nbInitIteration	Integer defining the maximal number of iterations in initialization algorithm. If initAlgo = "EM", "CEM" or "SemiSEM", this is the number of iterations if initAlgo = "SEM". Default value: 20.
initEpsilon	Real defining the epsilon value for the algorithm. initEpsilon is not used by the SEM algorithm. Default value: 0.01.
nbShortRun	Integer defining the number of short run to try (the strategy launch an initialization before each short run). Default value: 5.
shortRunAlgo	A character string with the algorithm to use in the short run stage. Default value: "EM".
nbShortIteration	Integer defining the maximal number of iterations in a short run if shortRunAlgo = "EM", "CEM" or "semiSEM", or the number of iterations if shortRunAlgo = "SEM". Default value: 100.
shortEpsilon	Real defining the epsilon value for the algorithm. shortEpsilon is not used by the SEM algorithm. Default value: 1e-04.
longRunAlgo	A character string with the algorithm to use in the long run stage Default value: "EM".
nbLongIteration	Integer defining the maximal number of iterations in the short runs if shortRunAlgo = "EM", "CEM" or "SemiSEM", or the number of iterations if shortRunAlgo = "SEM". Default value: 1000.
longEpsilon	Real defining the epsilon value for the algorithm. longEpsilon is not used by the SEM algorithm. Default value: 1e-07.

Details

The whole process can be repeated at least nbTry times. If a try success, the estimated model is returned, otherwise an empty model is returned (with an error message).

Value

a [ClusterStrategy] object

Author(s)

Serge Iovleff

Examples

```
clusterStrategy()
clusterStrategy(longRunAlgo= "CEM", nbLongIteration=100)
clusterStrategy(nbTry = 1, nbInit= 1, shortRunAlgo= "SEM", nbShortIteration=100)

clusterSemiSEMStrategy()

clusterSEMStrategy()

clusterFastStrategy()
```

DebTrivedi *Counting Data: DebTrivedi*

Description

Deb and Trivedi (1997) analyze data on 4406 individuals, aged 66 and over, who are covered by Medicare, a public insurance program. Originally obtained from the US National Medical Expenditure Survey (NMES) for 1987/88, the data are available from the data archive of the *Journal of Applied Econometrics*. It was prepared for an R package accompanying Kleiber and Zeileis (2008) and is also available asDebTrivedi.rda in the Journal of Statistical Software together with Zeileis (2006). The objective is to model the demand for medical care -as captured by the number of physician/non-physician office and hospital outpatient visits- by the covariates available for the patients.

Source

https://www.jstatsoft.org/htaccess.php?volume=27&type=i&issue=08&filename=paper

References

Zeileis, A. and Kleiber, C. and Jackma, S. (2008). "Regression Models for Count Data in R". JSS 27, 8, 1–25.

Examples

```
data(DebTrivedi)
```

`geyser` *Quantitative data: Old Faithful Geyser*

Description

The file geyser.rda contains 272 observations from the Old Faithful Geyser in the Yellowstone National Park. Each observation consists of two measurements: the duration (in minutes) of the eruption and the waiting time (in minutes) to the next eruption.

Format

A data frame with 272 observations on the following 2 variables.

`Duration` a numeric vector containing the duration (in minutes) of the eruption

`Waiting.Time` a numeric vector containing the waiting time (in minutes) to the next eruption

Details

Old Faithful erupts more frequently than any other big geyser, although it is not the largest nor the most regular geyser in the park. Its average interval between two eruptions is about 76 minutes, varying from 45 - 110 minutes. An eruption lasts from 1.1/2 to 5 minutes, expels 3,700 - 8,400 gallons (14,000 - 32,000 liters) of boiling water, and reaches heights of 106 - 184 feet (30 - 55m). It was named for its consistent performance by members of the Washburn Expedition in 1870. Old Faithful is still as spectacular and predictable as it was a century ago.

Source

https://web.archive.org/web/20191110083004/http://www.geyserstudy.org/geyser.aspx?pGeyserNo=OLDFAITHFUL

References

Hardle, W. (1991). "Smoothing Techniques with Implementation in S". Springer-Verlag, New York.
Azzalini, A. and Bowman, A. W. (1990). "A look at some data on the Old Faithful geyser". Applied Statistics 39, 357-365.

Examples

```
data(geyser)
```

HeartDisease.cat *Mixed data : Cleveland Heart Disease Data*

Description

The Cleveland Heart Disease Data found in the UCI machine learning repository consists of 14 variables measured on 303 individuals who have heart disease. The individuals had been grouped into five levels of heart disease. The information about the disease status is in the HeartDisease.target data set.

Format

Three data frames with 303 observations on the following 14 variables.

age age in years

sex sex (1 = male; 0 = female)

cp chest pain type. 1: typical angina, 2: atypical angina, 3: non-anginal pain, 4: asymptomatic

trestbps resting blood pressure (in mm Hg on admission to the hospital)

chol serum cholestoral in mg/dl

fbs (fasting blood sugar > 120 mg/dl) (1 = true; 0 = false)

restecg resting electrocardiographic results. 0: normal, 1: having ST-T wave abnormality (T wave inversions and/or ST, elevation or depression of > 0.05 mV) 2: showing probable or definite left ventricular hypertrophy by Estes\' criteria

thalach maximum heart rate achieved

exang exercise induced angina (1 = yes; 0 = no)

oldpeak ST depression induced by exercise relative to rest

slope the slope of the peak exercise ST segment 1: upsloping, 2: flat, 3: downsloping

ca number of major vessels (0-3) colored by flourosopy (4 missing values)

thal 3 = normal; 6 = fixed defect; 7 = reversable defect (2 missing values)

num diagnosis of heart disease (angiographic disease status). 0: < 50 1: > 50 (in any major vessel: attributes 59 through 68 are vessels)

Details

The variables consist of five continuous and eight discrete attributes, the former in the HeartDisease.cont data set and the later in the HeartDisease.cat data set. Three of the discrete attributes have two levels, three have three levels and two have four levels. There are six missing values in the data set.

Source

Author: David W. Aha (aha 'AT' ics.uci.edu) (714) 856-8779

Donors: The data was collected from the Cleveland Clinic Foundation (cleveland.data)

https://archive.ics.uci.edu/ml/datasets/Heart+Disease

Detrano, R., Janosi, A., Steinbrunn, W., Pfisterer, M., Schmid, J., Sandhu, S., Guppy, K., Lee, S., & Froelicher, V. (1989). International application of a new probability algorithm for the diagnosis of coronary artery disease. American Journal of Cardiology, 64,304–310.

David W. Aha & Dennis Kibler. "Instance-based prediction of heart-disease presence with the Cleveland database."

Gennari, J.H., Langley, P, & Fisher, D. (1989). Models of incremental concept formation. Artificial Intelligence, 40, 11–61.

Examples

```
summary(data(HeartDisease.cat))
summary(data(HeartDisease.cont))
summary(data(HeartDisease.target))
```

IClusterComponent *Definition of the [IClusterComponent] class*

Description

Interface base class defining a component of a mixture Model

This class defines a poisson component of a mixture Model. It inherits from [IClusterComponent].

Slots

`data` Matrix with the data set

`missing` Matrix with the indexes of the missing values

`modelName` model name associated with the data set

`lambda` Matrix with the mean of the jth variable in the kth cluster.

Author(s)

Serge Iovleff

See Also

[IClusterComponent] class

Examples

```
getSlots("IClusterComponent")

getSlots("ClusterPoissonComponent")
```

IClusterModel *Interface base Class [IClusterModel] for Cluster models.*

Description

This class encapsulate the common parameters of all the Cluster models.

Details

A Cluster model is a model of the form

$$f(x|\boldsymbol{\theta}) \sum_{k=1}^{K} p_k h(x; \boldsymbol{\lambda}_k, \boldsymbol{\alpha}) \quad x \in J.$$

where h can be either a pdf, a discrete probability, (homogeneous case) or a product of arbitrary pdf and discrete probabilities (mixed data case).

Slots

`nbSample` Integer with the number of samples of the model.

`nbCluster` Integer with the number of cluster of the model.

`pk` Vector of size K with the proportions of each mixture.

`tik` Matrix of size $n \times K$ with the posterior probability of the ith individual to belong to kth cluster.

`lnFi` Vector of size n with the log-likelihood of the ith individuals.

`zi` Vector of integer of size n with the attributed class label of the individuals.

`ziFit` Vector of integer of size n with the fitted class label of the individuals (only used in supervised learning).

`lnLikelihood` Real given the ln-liklihood of the Cluster model.

`criterion` Real given the value of the AIC, BIC, ICL or ML criterion.

`criterionName` string with the name of the criterion. Possible values are "BIC", "AIC", "ICL" or "ML". Default is "ICL".

`nbFreeParameter` Integer given the number of free parameters of the model.

`strategy` the instance of the [ClusterStrategy] used in the estimation process of the mixture. Default is clusterStrategy().

Author(s)

Serge Iovleff

Examples

```
getSlots("IClusterModel")
```

IClusterPredict *Interface class [IClusterPredict] for predicting*

Description

Interface base class for predicting clusters

Slots

`nbSample` Integer with the number of samples

`nbCluster` Integer with the number of cluster

`pk` Vector of size K with the proportions of each mixture.

`tik` Matrix of size $n \times K$ with the posterior probability of the ith individual to belong to kth cluster.

`lnFi` Vector of size n with the log-likelihood of the ith individuals.

`zi` Vector of integer of size n with the attributed class label of the individuals

`algo` an instance of [ClusterAlgoPredict]

`model` an instance of a (derived) [IClusterModel]

Author(s)

Serge Iovleff

Examples

```
getSlots("IClusterPredict")
```

kmm *Create an instance of the [KmmModel] class*

Description

This function computes the optimal kernel mixture model (KMM) according to the [criterion] among the number of clusters given in [nbCluster], using the strategy specified in [strategy].

Usage

```
kmm(
  data,
  nbCluster = 2,
  dim = 10,
  models = "kmm_pk_s",
  kernelName = "Gaussian",
  kernelParameters = c(1),
  kernelComputation = TRUE,
  strategy = kmmStrategy(),
  criterion = "ICL",
  nbCore = 1
)
```

Arguments

data	frame or matrix containing the data. Rows correspond to observations and columns correspond to variables.
nbCluster	[vector] listing the number of clusters to test.
dim	integer giving the dimension of the Gaussian density. Default is 10.
models	[vector] of model names to run. By default only "kmm_pk_s" is estimated. All the model names are given by the method [kmmNames].
kernelName	string with a kernel name. Possible values: "Gaussian", "polynomial", "Laplace", "linear", "rationalQuadratic_", "Hamming". Default is "Gaussian".
kernelParameters	[vector] with the parameters of the chosen kernel. Default is c(1).
kernelComputation	[logical] parameter. Should be TRUE if the Gram matrix is to be computed (faster but can be memory consuming), FALSE otherwise (times consuming). Default is TRUE. Recall that Gram matrix is a square matrix of size nbSample.
strategy	a [ClusterStrategy] object containing the strategy to run. [kmmStrategy]() method by default.
criterion	character defining the criterion to select the best model. The best model is the one with the lowest criterion value. Possible values: "BIC", "AIC", "ICL", "ML". Default is "ICL".
nbCore	integer defining the number of processor to use (default is 1, 0 for all).

Value

An instance of the [KmmModel] class.

Note

in KmmModel instance returned, the gram matrix is computed if and only if kernelComputation is TRUE.

Author(s)

Serge Iovleff

Examples

```
## A quantitative example with the famous bulls eye model
data(bullsEye)
## estimate model
model <- kmm( data=bullsEye, nbCluster=2:3, models= "kmm_pk_s")


## get summary
summary(model)
## use graphics functions

plot(model)
```

KmmComponent *Definition of the [KmmComponent] class*

Description

This class defines a kernel component of a mixture Model. It inherits from [IClusterComponent].

Slots

`dim` Vector with the dimension of the kth cluster

`sigma2` Vector with the standard deviation in the kth cluster.

`gram` Matrix storing the gram matrix if its computation is needed

`kernelName` string with the name of the kernel to use. Possible values: "Gaussian", "polynomial", "Laplace", "linear","rationalQuadratic", "Hamming". Default is "Gaussian".

`kernelParameters` vector with the parameters of the kernel.

`kernelComputation` boolean value set as `TRUE` if Gram matrix is to be computed `FALSE` othewise. Default is `TRUE`.

Author(s)

Serge Iovleff

See Also

[IClusterComponent] class

Examples

```
getSlots("KmmComponent")
```

kmmMixedData *Create an instance of the [KmmMixedDataModel] class*

Description

This function computes the optimal mixture model for mixed data using kernel mixture models according to the criterion among the number of clusters given in nbCluster using the strategy specified in [strategy].

Usage

```
kmmMixedData(
  ldata,
  lmodels,
  nbCluster = 2,
  strategy = clusterStrategy(),
  criterion = "ICL",
  nbCore = 1
)
```

Arguments

ldata	[list] containing the data sets (matrices and/or data.frames).
lmodels	a [list] of same length than data. It contains the model names, kernel names and kernel parameter names to use in order to fit each data set.
nbCluster	[vector] with the number of clusters to test.
strategy	a [ClusterStrategy] object containing the strategy to run. Default is clusterStrategy().
criterion	character defining the criterion to select the best model. The best model is the one with the lowest criterion value. Possible values: "BIC", "AIC", "ICL", "ML". Default is "ICL".
nbCore	integer defining the number of processors to use (default is 1, 0 for all).

Details

For each data set in data, we need to specify a list of parameters

Value

An instance of the [KmmMixedDataModel] class.

Author(s)

Serge Iovleff

Examples

```
## An example with the bullsEye data set
data(bullsEye)
data(bullsEye.cat)
## with default values
ldata     <- list(bullsEye, bullsEye.cat)
modelcont <- list(modelName="kmm_pk_s", dim = 10, kernelName="Gaussian")
modelcat  <- list(modelName="kmm_pk_s", dim = 20, kernelName="Hamming", kernelParameters = c(0.6))
lmodels   <- list( modelcont, modelcat)

model <- kmmMixedData(ldata, lmodels, nbCluster=2:5, strategy = clusterFastStrategy())

## get summary
summary(model)



## use graphics functions
plot(model)
```

KmmMixedDataModel *Definition of the [KmmMixedDataModel] class*

Description

This class defines a mixed data kernel mixture Model (KMM).

Details

This class inherits from the [IClusterModel] class. A model for mixed data is a mixture model of the form:

$$f(x_i = (x_{1i}, x_{2i}, \ldots x_{Li})|\theta) = \sum_{k=1}^{K} p_k \prod_{l=1}^{L} h(x_{li}).$$

The density functions (or probability distribution functions)

$$h(.)$$

can be any implemented kmm model on a RKHS space.

Slots

`lcomponent` a list of [KmmComponent]

Author(s)

Serge Iovleff

See Also

[IClusterModel] class

Examples

```
getSlots("KmmMixedDataModel")
```

KmmModel *Definition of the [KmmModel] class*

Description

This class defines a Kernel mixture Model (KMM).

Details

This class inherits from the [IClusterModel] virtual class. A KMM is a mixture model of the form:

$$f(x|\boldsymbol{\theta}) = \sum_{k=1}^{K} p_k \prod_{j=1}^{d} \phi(x_j; \sigma_k^2) \quad x \in R^d.$$

Some constraints can be added to the variances in order to reduce the number of parameters.

Slots

`component` A [KmmComponent] with the dimension and standard deviation of the kernel mixture model.

Author(s)

Serge Iovleff

See Also

[IClusterModel] class

Examples

```
getSlots("KmmModel")
data(bullsEye)
new("KmmModel", data=bullsEye)
```

kmmNames *Create a vector of Kernel mixture model (KMM) names.*

Description

In a Kernel mixture model, sssumptions on the proportions and standard deviations give rise to 4 models:

1. Proportions can be equal or free.
2. Standard deviations are equal or free for all clusters.

Usage

```
kmmNames(prop = "all", sdBetweenCluster = "all")

kmmValidModelNames(names)

kmmValidKernelNames(names)
```

Arguments

prop	A character string equal to "equal", "free" or "all". Default is "all".
sdBetweenCluster	A character string equal to "equal", "free" or "all". Default is "all".
names	a vector of character with the names to check

Details

The model names are summarized in the following array:

Model Name	Proportions	s. d. between clusters
kmm_p_sk	equal	Free
kmm_p_s	equal	Equal
kmm_pk_sk	equal	Free
kmm_pk_s	equal	Equal

Value

A vector of character with the model names.

TRUE if the names in the vector names are valid, FALSE otherwise.

Examples

```
kmmNames()
## same as c("kmm_p_sk")
kmmNames( prop = "equal", sdBetweenCluster= "free")
```

kmmStrategy *Create an instance of [ClusterStrategy] class*

Description

A strategy is a multistage empirical process for finding a good estimate in the clustering estimation process.

Usage

```
kmmStrategy(
  nbTry = 1,
  nbInit = 5,
  initMethod = "class",
  initAlgo = "EM",
  nbInitIteration = 20,
  initEpsilon = 0.01,
  nbShortRun = 5,
  shortRunAlgo = "EM",
  nbShortIteration = 100,
  shortEpsilon = 1e-04,
  longRunAlgo = "EM",
  nbLongIteration = 1000,
  longEpsilon = 1e-07
)
```

Arguments

nbTry	Integer defining the number of estimation to attempt.
nbInit	Integer defining the number of initialization to try. Default value: 3.
initMethod	Character string with the initialization method, see [clusterInit]$ for possible values. Default value: "class".
initAlgo	Character string with the algorithm to use in the initialization stage, [clusterAlgo] for possible values. Default value: "EM".
nbInitIteration	Integer defining the maximal number of iterations in initialization algorithm if initAlgo = "EM" or "CEM", the number of iterations if initAlgo = "SEM". Default value: 20.
initEpsilon	Real defining the epsilon value for the initialization algorithm. Not used if initAlgo = "SEM". Default value: 0.01.
nbShortRun	Integer defining the number of short run to try (the strategy launch an initialization before each short run). Default value: 5.
shortRunAlgo	A character string with the algorithm to use in the short run stage. Default value: "EM".

`nbShortIteration`
: Integer defining the maximal number of iterations during sa hort run if `shortRunAlgo` = "EM" or "CEM", the number of iterations if `shortRunAlgo` = "SEM". Default value: 100.

`shortEpsilon`
: Real defining the epsilon value for the algorithm. Not used if `shortRunAlgo` = "SEM". Default value: 1e-04.

`longRunAlgo`
: A character string with the algorithm to use in the long run stage. Default value: "EM".

`nbLongIteration`
: Integer defining the maximal number of iterations during a long run algorithm if `longRunAlgo` = "EM" or "CEM", the number of iterations if `longRunAlgo` = "SEM". Default value: 1000.

`longEpsilon`
: Real defining the epsilon value for the algorithm. Nor used if `longRunAlgo` = "SEM". Default value: 1e-07.

Details

A strategy is a way to find a good estimate of the parameters of a kernel mixture model when using an EM algorithm or its variants. A "try" of kmmStrategy is composed of three stages

- `nbShortRun` short iterations of the initialization step and of the `EM`, `CEM` or `SEM` algorithm.
- `nbInit` initializations using the [`clusterInit`] method.
- A long run of the `EM`, `CEM` or `SEM` algorithm.

For example if `nbInit` is 5 and `nbShortRun` is also 5, there will be 5 times 5 models initialized. Five time, the best model (in the likelihood sense) will be ameliorated using a short run. Among the 5 models ameliorated one will be estimated until convergence using a long run. In total there is 25 initializations.

The whole process can be repeated at least `nbTry` times. If a try success, the estimated model is returned, otherwise an empty model is returned.

Value

a [`ClusterStrategy`] object

Author(s)

Serge Iovleff

Examples

```
kmmStrategy()
kmmStrategy(longRunAlgo= "CEM", nbLongIteration=100)
kmmStrategy(nbTry = 1, nbInit= 1, shortRunAlgo= "EM", nbShortIteration=100)
```

LearnAlgo *[LearnAlgo] class for Cluster algorithms.*

Description

This class encapsulates the parameters of clustering estimation algorithms methods.

Slots

`algo` A character string with the algorithm. Possible values: "Simul", "Impute. Default value: "Simul".

`nbIteration` Integer defining the maximal number of iterations. Default value: 200.

`epsilon` real defining the epsilon value for the algorithm. epsilon is note used if `algo` is "Simul". Default value: 1e-07.

Examples

```
getSlots("LearnAlgo")
new("LearnAlgo")
new("LearnAlgo", algo="Impute", nbIteration=100)
```

learnAlgo *Create an instance of the [LearnAlgo] class*

Description

There is two algorithms and two stopping rules possibles for a learning algorithm.

- Algorithms:
  - `Impute`: Impute the missing values during the iterations
  - `Simul`: Simulate the missing values during the iterations
- Stopping rules:
  - `nbIteration`: Set the maximum number of iterations
  - `epsilon`: Set relative increase of the log-likelihood criterion
- Default values are 200 `nbIteration` of `Simul`.

The `epsilon` value is not used when the algorithm is "Simul". It is worth noting that if there is no missing values, the method should be "Impute" and nbIteration should be set to 1!

Usage

```
learnAlgo(algo = "Simul", nbIteration = 200, epsilon = 1e-07)
```

Arguments

algo	character string with the estimation algorithm. Possible values are "Simul", "Impute". Default value is "Simul".
nbIteration	Integer defining the maximal number of iterations. Default value is 200.
epsilon	Real defining the epsilon value for the algorithm. Not used by the "Simul" algorithm. Default value is 1.e-7.

Value

a [LearnAlgo] object

Author(s)

Serge Iovleff

Examples

```
learnAlgo()
learnAlgo(algo="simul", nbIteration=50)
learnAlgo(algo="impute", epsilon = 1e-06)
```

---

learnDiagGaussian *Create an instance of a learn mixture model*

Description

This function learn the optimal mixture model when the class labels are known according to the criterion among the list of model given in models.

Usage

```
learnDiagGaussian(
  data,
  labels,
  prop = NULL,
  models = clusterDiagGaussianNames(prop = "equal"),
  algo = "simul",
  nbIter = 100,
  epsilon = 1e-08,
  criterion = "ICL",
  nbCore = 1
)

learnPoisson(
  data,
  labels,
```

```
  prop = NULL,
  models = clusterPoissonNames(prop = "equal"),
  algo = "simul",
  nbIter = 100,
  epsilon = 1e-08,
  criterion = "ICL",
  nbCore = 1
)

learnGamma(
  data,
  labels,
  prop = NULL,
  models = clusterGammaNames(prop = "equal"),
  algo = "simul",
  nbIter = 100,
  epsilon = 1e-08,
  criterion = "ICL",
  nbCore = 1
)

learnCategorical(
  data,
  labels,
  prop = NULL,
  models = clusterCategoricalNames(prop = "equal"),
  algo = "simul",
  nbIter = 100,
  epsilon = 1e-08,
  criterion = "ICL",
  nbCore = 1
)
```

Arguments

data	frame or matrix containing the data. Rows correspond to observations and columns correspond to variables. If the data set contains NA values, they will be estimated during the estimation process.
labels	vector or factors giving the label class.
prop	[vector] with the proportions of each class. If NULL the proportions will be estimated using the labels.
models	[vector] of model names to run. By default all models are estimated.
algo	character defining the algo to used in order to learn the model. Possible values: "simul" (default), "impute" (faster but can produce biased results).
nbIter	integer giving the number of iterations to do. algo is "impute" this is the maximal authorized number of iterations. Default is 100.
epsilon	real giving the variation of the log-likelihood for stopping the iterations. Not used if algo is "simul". Default value is 1e-08.

criterion	character defining the criterion to select the best model. The best model is the one with the lowest criterion value. Possible values: "BIC", "AIC", "ML". Default is "ICL".
nbCore	integer defining the number of processors to use (default is 1, 0 for all).

Value

An instance of a learned mixture model class.

Author(s)

Serge Iovleff

Examples

```
## A quantitative example with the famous iris data set
data(iris)

## get data and target
x <- as.matrix(iris[,1:4]);
z <- as.vector(iris[,5]);
n <- nrow(x); p <- ncol(x);

## add missing values at random
indexes <- matrix(c(round(runif(5,1,n)), round(runif(5,1,p))), ncol=2);
x[indexes] <- NA;

## learn model
model <- learnDiagGaussian( data=x, labels= z, prop = c(1/3,1/3,1/3)
                          , models = clusterDiagGaussianNames(prop = "equal")
                          )

## get summary
summary(model)

## use graphics functions

plot(model)



## print model (a detailed and very long output)
print(model)


## get estimated missing values
missingValues(model)
```

learnMixedData — *This function learn the optimal mixture model when the class labels are known according to the* criterion *among the list of model given in* models.

Description

This function learn the optimal mixture model when the class labels are known according to the criterion among the list of model given in models.

Usage

```
learnMixedData(
  data,
  models,
  labels,
  prop = NULL,
  algo = "impute",
  nbIter = 100,
  epsilon = 1e-08,
  criterion = "ICL",
  nbCore = 1
)
```

Arguments

data	[list] containing the data sets (matrices and/or data.frames). If data sets contain NA values, these missing values will be estimated during the estimation process.
models	either a [vector] of character or a [list] of same length than data. If models is a vector, it contains the model names to use in order to fit each data set. If models is a list, it must be of the form models = list(modelName, dim, kernelName, modelParameters) Only modelName is required.
labels	vector or factors giving the label class.
prop	[vector] with the proportions of each class. If NULL the proportions will be estimated using the labels.
algo	character defining the algo to used in order to learn the model. Possible values: "simul" (default), "impute" (faster but can produce biased results).
nbIter	integer giving the number of iterations to do. algo is "impute" this is the maximal authorized number of iterations. Default is 100.
epsilon	real giving the variation of the log-likelihood for stopping the iterations. Not used if algo is "simul". Default value is 1e-08.
criterion	character defining the criterion to select the best model. The best model is the one with the lowest criterion value. Possible values: "BIC", "AIC", "ICL", "ML". Default is "ICL".
nbCore	integer defining the number of processors to use (default is 1, 0 for all).

Value

An instance of the [ClusterMixedDataModel] class.

Author(s)

Serge Iovleff

Examples

```
## A quantitative example with the heart disease data set
data(HeartDisease.cat)
data(HeartDisease.cont)
## with default values
ldata = list(HeartDisease.cat, HeartDisease.cont);
models = c("categorical_pk_pjk","gaussian_pk_sjk")
model <- clusterMixedData(ldata, models, nbCluster=2:5, strategy = clusterFastStrategy())

## get summary
summary(model)

## get estimated missing values
missingValues(model)


## print model (a detailed and very long output)
print(model)
## use graphics functions
plot(model)
```

---

missingValues *Return the missing values of a component or a cluster class.*

Description

The missing methods allow the user to get the imputed mssing values from a mixture model.

Usage

```
missingValues(x)

## S4 method for signature 'ClusterMixedDataModel'
missingValues(x)

## S4 method for signature 'ClusterDiagGaussianComponent'
missingValues(x)

## S4 method for signature 'ClusterDiagGaussian'
```

```
missingValues(x)

## S4 method for signature 'ClusterGammaComponent'
missingValues(x)

## S4 method for signature 'ClusterGamma'
missingValues(x)

## S4 method for signature 'ClusterCategoricalComponent'
missingValues(x)

## S4 method for signature 'ClusterCategorical'
missingValues(x)

## S4 method for signature 'ClusterPoissonComponent'
missingValues(x)

## S4 method for signature 'ClusterPoisson'
missingValues(x)

## S4 method for signature 'ClusterPredict'
missingValues(x)

## S4 method for signature 'ClusterPredictMixedData'
missingValues(x)

## S4 method for signature 'KmmComponent'
missingValues(x)

## S4 method for signature 'KmmModel'
missingValues(x)
```

Arguments

x	an object that can return the imputed missing values

Value

A matrix with three columns (row index, column index, value)

Examples

```
## add 10 missing values as random
data(geyser)
x = as.matrix(geyser); n <- nrow(x); p <- ncol(x);
indexes <- matrix(c(round(runif(5,1,n)), round(runif(5,1,p))), ncol=2);
x[indexes] <- NA;
## estimate model (using fast strategy, results may be misleading)
model <- clusterDiagGaussian(data=x, nbCluster=2:3, strategy = clusterFastStrategy())
missingValues(model)
```

---

plot,ClusterCategorical-method

Plotting of a class [ClusterCategorical]

Description

Plotting data from a [ClusterCategorical] object using the estimated parameters and partition.

Usage

```
## S4 method for signature 'ClusterCategorical'
plot(x, y, ...)
```

Arguments

x	an object of class [ClusterCategorical]
y	a number between 1 and K-1.
...	further arguments passed to or from other methods

See Also

plot

Examples

```
## the car data set (verify car data is in your environment)
data(car)
model <- clusterCategorical(car, 3, strategy = clusterFastStrategy())
plot(model)
```

---

plot,ClusterDiagGaussian-method

Plotting of a class [ClusterDiagGaussian]

Description

Plotting data from a [ClusterDiagGaussian] object using the estimated parameters and partition.

Usage

```
## S4 method for signature 'ClusterDiagGaussian'
plot(x, y, ...)
```

Arguments

x	an object of class [ClusterDiagGaussian]
y	a list of variables to plot (subset). Variables names or indices. If missing all the variables are represented.
...	further arguments passed to or from other methods

See Also

plot

Examples

```
## the famous iris data set
data(iris)
model <- clusterDiagGaussian(iris[1:4], 3, strategy = clusterFastStrategy())
plot(model)
plot(model, c(1,3))
plot(model, c("Sepal.Length","Sepal.Width"))
```

---

plot,ClusterGamma-method

Plotting of a class [ClusterGamma]

Description

Plotting data from a [ClusterGamma] object using the estimated parameters and partition.

Usage

```
## S4 method for signature 'ClusterGamma'
plot(x, y, ...)
```

Arguments

x	an object of class [ClusterGamma]
y	a list of variables to plot (subset). Variables names or indices. If missingValues all the variables are represented.
...	further arguments passed to or from other methods

See Also

plot

Examples

```
## Example with quantitative vairables
data(iris)
model <- clusterGamma( data=iris[1:4], nbCluster=3
                     , models=clusterGammaNames(prop = "equal")
                     , strategy = clusterFastStrategy())
plot(model)
plot(model, c(1,3))
plot(model, c("Sepal.Length","Sepal.Width"))
```

---

plot,ClusterMixedDataModel-method

Plotting of a class [ClusterMixedDataModel]

Description

Plotting data from a [ClusterMixedDataModel] object using the estimated parameters and partition.

Usage

```
## S4 method for signature 'ClusterMixedDataModel'
plot(x, y, ...)
```

Arguments

x	an object of class [ClusterMixedDataModel]
y	a number between 1 and K-1.
...	further arguments passed to or from other methods

See Also

plot

plot,ClusterPoisson-method

Plotting of a class [ClusterPoisson]

Description

Plotting data from a [ClusterPoisson] object using the estimated parameters and partition.

Usage

```
## S4 method for signature 'ClusterPoisson'
plot(x, y, ...)
```

Arguments

x	an object of class [ClusterPoisson]
y	a list of variables to plot (subset). Variables names or indices. If missingValues all the variables are represented.
...	further arguments passed to or from other methods

See Also

plot

Examples

```
## Example with counting data
data(DebTrivedi)
dt <- DebTrivedi[, c(1, 6,8, 15)]
model <- clusterPoisson(iris[1:4], 3, strategy = clusterFastStrategy())
plot(model)
plot(model, c(1,2))
```

plot,KmmComponent-method

Plotting of a class [KmmComponent]

Description

Plotting data from a [KmmComponent] object using the estimated partition.

Usage

```
## S4 method for signature 'KmmComponent'
plot(x, y, ...)
```

Arguments

x	an object of class [KmmComponent]
y	a vector with partitions
...	further arguments passed to or from other methods

See Also

plot

Examples

```
## the bull eyes data set
 data(bullsEye)
 model <- kmm( bullsEye, 2, models= "kmm_pk_s")
 plot(model)
```

---

plot,KmmMixedDataModel-method

Plotting of a class [KmmMixedDataModel]

Description

Plotting data from a [KmmMixedDataModel] object using the estimated parameters and partition.

Usage

```
## S4 method for signature 'KmmMixedDataModel'
plot(x, y, ...)
```

Arguments

x	an object of class [KmmMixedDataModel]
y	a vector listing the data sets you want to disply
...	further arguments passed to or from other methods

See Also

plot

Examples

```
## The bullsEye data set
data(bullsEye)
data(bullsEye.cat)
## with default values
ldata  = list(bullsEye, bullsEye.cat)
modelcont <- list(modelName="kmm_pk_s", dim = 10, kernelName="Gaussian")
modelcat  <- list(modelName="kmm_pk_s", dim = 20, kernelName="Hamming", kernelParameters = c(0.6))
lmodels = list( modelcont, modelcat)

model <- kmmMixedData(ldata, lmodels, nbCluster=2:5, strategy = clusterFastStrategy())
# plot only the first continuous data set
plot(model, y=c(1))
```

---

plot,KmmModel-method *Plotting of a class [KmmModel]*

Description

Plotting data from a [KmmModel] object using the estimated parameters and partition.

Usage

```
## S4 method for signature 'KmmModel'
plot(x, y, ...)
```

Arguments

x	an object of class [KmmModel]
y	a list of variables to plot (subset). Variables names or indices. If missing all the variables are represented.
...	further arguments passed to or from other methods

See Also

plot

Examples

```
## the bull eyes data set
 data(bullsEye)
 model <- kmm( bullsEye, 2, models= "kmm_pk_s")
 plot(model)
```

print,ClusterAlgo-method

Print a MixAll S4 class to standard output.

Description

Print a MixAll S4 class to standard output.

Usage

```
## S4 method for signature 'ClusterAlgo'
print(x, ...)

## S4 method for signature 'ClusterAlgoPredict'
print(x, ...)

## S4 method for signature 'ClusterInit'
print(x, ...)

## S4 method for signature 'ClusterStrategy'
print(x, ...)

## S4 method for signature 'IClusterComponent'
print(x, ...)

## S4 method for signature 'IClusterModel'
print(x, ...)

## S4 method for signature 'ClusterCategoricalComponent'
print(x, k, ...)

## S4 method for signature 'ClusterCategorical'
print(x, ...)

## S4 method for signature 'ClusterDiagGaussianComponent'
print(x, k, ...)

## S4 method for signature 'ClusterDiagGaussian'
print(x, ...)

## S4 method for signature 'ClusterGammaComponent'
print(x, k, ...)

## S4 method for signature 'ClusterGamma'
print(x, ...)

## S4 method for signature 'ClusterMixedDataModel'
```

```
print(x, ...)

## S4 method for signature 'ClusterPoissonComponent'
print(x, k, ...)

## S4 method for signature 'ClusterPoisson'
print(x, ...)

## S4 method for signature 'IClusterPredict'
print(x, ...)

## S4 method for signature 'ClusterPredict'
print(x, ...)

## S4 method for signature 'ClusterPredictMixedData'
print(x, ...)

## S4 method for signature 'LearnAlgo'
print(x, ...)

## S4 method for signature 'KmmComponent'
print(x, k, ...)

## S4 method for signature 'KmmModel'
print(x, ...)

## S4 method for signature 'KmmMixedDataModel'
print(x, ...)
```

Arguments

x	a MixAll object: a ClusterStrategy, a ClusterInit or a ClusterAlgo.
...	further arguments passed to or from other methods
k	the number of the cluster to print

Value

NULL. Prints to standard out.

See Also

print

Examples

```
## for cluster strategy
strategy <- clusterStrategy()
print(strategy)
## for cluster init
```

```
init <- clusterInit()
print(init)
## for cluster algo
algo <- clusterAlgo()
print(algo)
```

---

show,ClusterAlgo-method *Show description of a MixAll S4 class to standard output.*

Description

Show description of a MixAll S4 class to standard output.

Usage

```
## S4 method for signature 'ClusterAlgo'
show(object)

## S4 method for signature 'ClusterAlgoPredict'
show(object)

## S4 method for signature 'ClusterInit'
show(object)

## S4 method for signature 'ClusterStrategy'
show(object)

## S4 method for signature 'IClusterComponent'
show(object)

## S4 method for signature 'IClusterModel'
show(object)

## S4 method for signature 'ClusterCategoricalComponent'
show(object)

## S4 method for signature 'ClusterCategorical'
show(object)

## S4 method for signature 'ClusterDiagGaussianComponent'
show(object)

## S4 method for signature 'ClusterDiagGaussian'
show(object)
```

```
## S4 method for signature 'ClusterGammaComponent'
show(object)

## S4 method for signature 'ClusterGamma'
show(object)

## S4 method for signature 'ClusterMixedDataModel'
show(object)

## S4 method for signature 'ClusterPoissonComponent'
show(object)

## S4 method for signature 'ClusterPoisson'
show(object)

## S4 method for signature 'IClusterPredict'
show(object)

## S4 method for signature 'ClusterPredict'
show(object)

## S4 method for signature 'ClusterPredictMixedData'
show(object)

## S4 method for signature 'LearnAlgo'
show(object)

## S4 method for signature 'KmmComponent'
show(object)

## S4 method for signature 'KmmModel'
show(object)

## S4 method for signature 'KmmMixedDataModel'
show(object)
```

Arguments

object	a MixAll object: a ClusterStrategy, a ClusterInit or a ClusterAlgo.

Value

NULL. Prints to standard out.

See Also

show

Examples

```
## for strategy
strategy <- clusterStrategy()
show(strategy)
## for cluster init
init <- clusterInit()
show(init)
## for cluster algo
algo <- clusterAlgo()
show(algo)
```

---

summary,IClusterComponent-method

Produce summary of a MixAll S4 class.

Description

Produce summary of a MixAll S4 class.

Usage

```
## S4 method for signature 'IClusterComponent'
summary(object, ...)

## S4 method for signature 'IClusterModel'
summary(object, ...)

## S4 method for signature 'ClusterCategoricalComponent'
summary(object)

## S4 method for signature 'ClusterCategorical'
summary(object, ...)

## S4 method for signature 'ClusterDiagGaussian'
summary(object, ...)

## S4 method for signature 'ClusterGamma'
summary(object, ...)

## S4 method for signature 'ClusterMixedDataModel'
summary(object, ...)

## S4 method for signature 'ClusterPoisson'
summary(object, ...)

## S4 method for signature 'IClusterPredict'
```

```
summary(object, ...)

## S4 method for signature 'ClusterPredict'
summary(object, ...)

## S4 method for signature 'ClusterPredictMixedData'
summary(object, ...)

## S4 method for signature 'KmmModel'
summary(object, ...)

## S4 method for signature 'KmmMixedDataModel'
summary(object, ...)
```

Arguments

object	any cluster model deriving from a IClusterModel or IClusterComponent object.
...	further arguments passed to or from other methods

Value

NULL. Summaries to standard out.

[,ClusterAlgo-method *Extract parts of a MixAll S4 class*

Description

Extract parts of a MixAll S4 class

Usage

```
## S4 method for signature 'ClusterAlgo'
x[i, j, drop]

## S4 replacement method for signature 'ClusterAlgo'
x[i, j] <- value

## S4 method for signature 'ClusterAlgoPredict'
x[i, j, drop]

## S4 replacement method for signature 'ClusterAlgoPredict'
x[i, j] <- value

## S4 method for signature 'ClusterInit'
x[i, j, drop]
```

```
## S4 replacement method for signature 'ClusterInit'
x[i, j] <- value

## S4 method for signature 'ClusterStrategy'
x[i, j, drop]

## S4 replacement method for signature 'ClusterStrategy'
x[i, j] <- value

## S4 method for signature 'ClusterCategoricalComponent'
x[i, j, drop]

## S4 method for signature 'ClusterDiagGaussianComponent'
x[i, j, drop]

## S4 method for signature 'ClusterGammaComponent'
x[i, j, drop]

## S4 method for signature 'ClusterPoissonComponent'
x[i, j, drop]

## S4 method for signature 'LearnAlgo'
x[i, j, drop]

## S4 replacement method for signature 'LearnAlgo'
x[i, j] <- value

## S4 method for signature 'KmmComponent'
x[i, j, drop]
```

Arguments

x	object from which to extract element(s) or in which to replace element(s).
i	the name of the element we want to extract or replace.
j	if the element designing by i is complex, j specifying elements to extract or replace.
drop	For matrices and arrays. If TRUE the result is coerced to the lowest possible dimension (see the examples). This only works for extracting elements, not for the replacement. See drop for further details.
value	typically an array-like R object of a similar class as the element of x we want to replace.

Index