Package 'Rmixmod'

October 18, 2022

Version 2.1.7

Date 2022-10-18

Title Classification with Mixture Modelling

Description Interface of 'MIXMOD' software for supervised, unsupervised and semi-supervised classification with mixture modelling <doi:10.18637/jss.v067.i06>.

SystemRequirements GNU make

Copyright Authors, and Mixmod use Eigen3 C++ library

BugReports https://github.com/mixmod/mixmod/issues

License GPL-3

LinkingTo Rcpp, RcppEigen

Imports methods

Depends R(>= 2.12.0), Rcpp(>= 0.8.0)

Collate 'A_NAMESPACE.R' 'global.R' 'Model.R' 'MultinomialModel.R' 'GaussianModel.R' 'CompositeModel.R' 'Parameter.R' 'MultinomialParameter.R' 'GaussianParameter.R' 'CompositeParameter.R' 'MixmodResults.R' 'Mixmod.R' 'Strategy.R' 'MixmodCluster.R' 'MixmodDAResults.R' 'MixmodLearn.R' 'MixmodPredict.R' 'MixmodXml.R' 'Rmixmod.R' 'zzz.R'

RoxygenNote 7.2.1

Encoding UTF-8

NeedsCompilation yes

Author Florent Langrognet [aut],
Remi Lebret [aut],
Christian Poli [aut],
Serge Iovleff [aut],
Benjamin Auder [aut],
Parmeet Bhatia [ctb],
Anwuli Echenim [ctb],
Serge Iovleff [aut],
Christophe Biernacki [ctb],

Gilles Celeux [ctb],
Gerard Govaert [ctb],
Quentin Grimonprez [cre]

Maintainer Quentin Grimonprez <quentingrim@yahoo.fr>

Repository CRAN

Date/Publication 2022-10-18 14:17:59 UTC

R topics documented:

`Rmixmod-package` *Rmixmod a MIXture MODelling package*

Description

Rmixmod is a package based on the existing MIXMOD software. MIXMOD is a tool for fitting a mixture model of multivariate gaussian or multinomial components to a given data set with either a clustering, a density estimation or a discriminant analysis point of view.

Details

The general purpose of the package is to discover, or explain, group structures in multivariate data sets with unknown (cluster analysis or clustering) or known class discriminant analysis or classification). It is an exploratory data analysis tool for solving clustering and classification problems. But it can also be regarded as a semi-parametric tool to estimate densities with Gaussian mixture distributions and multinomial distributions.

Mathematically, mixture probability density function (pdf) f is a weighted sum of K components densities:

$$f(\mathbf{x}_i|\theta) = \sum_{k=1}^{K} p_k h(\mathbf{x}_i|\lambda_k)$$

where $h(.|\lambda_k)$ denotes a d-dimensional distribution parametrized by λ_k. The parameters are the mixing proportions p_k and the component of the distribution λ_k.

In the Gaussian case, h is the density of a Gaussian distribution with mean μ_k and variance matrix Σ_k, and thus $\lambda_k = (\mu_k, \Sigma_k)$.

In the qualitative case, h is a multinomial distribution and $\lambda_k = (a_k, \epsilon_k)$ is the parameter of the distribution.

Estimation of the mixture parameters is performed either through maximum likelihood via the EM (*Expectation Maximization*, Dempster et al. 1977), the SEM (*Stochastic EM*, Celeux and Diebolt 1985) algorithm or through classification maximum likelihood via the CEM algorithm (*Clustering EM*, Celeux and Govaert 1992). These three algorithms can be chained to obtain original fitting strategies (e.g. CEM then EM with results of CEM) to use advantages of each of them in the estimation process. As mixture problems usually have multiple relative maxima, the program will produce different results, depending on the initial estimates supplied by the user. If the user does not input his own initial estimates, some initial estimates procedures are proposed (random centers for instance).

It is possible to constrain some input parameters. For example, dispersions can be equal between classes, etc.

In the Gaussian case, fourteen models are implemented. They are based on the eigenvalue decomposition, are most generally used. They depend on constraints on the variance matrix such as same variance matrix between clusters, spherical variance matrix... and they are suitable for data sets in any dimension.

In the qualitative case, five multinomial models are available. They are based on a reparametrization of the multinomial probabilities.

In both cases, the models and the number of clusters can be chosen by different criteria: BIC (Bayesian Information Criterion), ICL (Integrated Completed Likelihood, a classification version of BIC), NEC (Entropy Criterion), or Cross-Validation (CV).

Author(s)

Author: Florent Langrognet and Remi Lebret and Christian Poli and Serge Iovleff, with contributions from C. Biernacki and G. Celeux and G. Govaert <contact@mixmod.org>

References

Biernacki C., Celeux G., Govaert G., Langrognet F., 2006. "Model-Based Cluster and Discriminant Analysis with the MIXMOD Software". Computational Statistics and Data Analysis, vol. 51/2, pp. 587-600.

Lebret R., Iovleff S., Langrognet F., Biernacki C., Celeux G., Govaert G. (2015). "Rmixmod: The R Package of the Model-Based Unsupervised, Supervised, and Semi-Supervised Classification Mixmod Library". Journal of Statistical Software, 67(6), 1–29. https://doi.org/10.18637/jss.v067.i06

Examples

```
## Not run:
## Clustering Analysis
# load quantitative data set
data(geyser)
# Clustering in gaussian case
xem1 <- mixmodCluster(geyser, 3)
```

```
summary(xem1)
plot(xem1)
hist(xem1)

# load qualitative data set
data(birds)
# Clustering in multinomial case
xem2 <- mixmodCluster(birds, 2)
summary(xem2)
barplot(xem2)

# load heterogeneous data set
data(finance)
# Clustering in composite case
xem3 <- mixmodCluster(finance, 2:6)
summary(xem3)

## Discriminant Analysis
# start by extract 10 observations from iris data set
remaining.obs <- sample(1:nrow(iris), 10)
# then run a mixmodLearn() analysis without those 10 observations
learn <- mixmodLearn(iris[-remaining.obs, 1:4], iris$Species[-remaining.obs])
# create a MixmodPredict to predict those 10 observations
prediction <- mixmodPredict(
  data = iris[remaining.obs, 1:4],
  classificationRule = learn["bestResult"]
)
# show results
prediction
# compare prediction with real results
paste("accuracy= ", mean(as.integer(iris$Species[remaining.obs]) == prediction["partition"]) * 100,
  "%",
  sep = ""
)

## End(Not run)
```

`asQualitative` *Convert a data frame containing integers to a qualitative data set with factors.*

Description

Convert a data frame containing integers to a qualitative data set with factors.

Usage

```
asQualitative(x)
```

Arguments

x	a vector or a matrix or a data frame

Value

a qualitative data set with factors

---

barplot *Barplot of a class [Mixmod]*

Description

Barplot of qualitative data from a [Mixmod] object using parameters to plot probabilities of modalities.

Usage

```
## S4 method for signature 'Mixmod'
barplot(height, ...)
```

Arguments

height	an object of class [Mixmod] (???)
...	further arguments passed to or from other methods

Details

Each line corresponds to one variable. Barplot is drawn for each cluster with the probabilities for each modality to be in that cluster.

See Also

barplot

Examples

```
data(birds)
xem2 <- mixmodCluster(birds, 2)
barplot(xem2)
barplot(xem2, variables = c(2, 3, 4))
barplot(xem2, variables = c("eyebrow", "collar"))
```

barplotCluster *Barplot of a class [MixmodResults]*

Description

Barplot of qualitative data object using parameters from a [MixmodResults] to plot probabilities of modalities.

Usage

```
barplotCluster(
  x,
  data,
  variables = colnames(data),
  main = paste("Barplot of", variables),
  ...
)
```

Arguments

x	an object of class [MixmodResults]
data	a vector or data frame containing a qualitative data set.
variables	list of variables names (or indices) to compute a barplot. All variables from data by default.
main	a list of title for the barplot. main must have the same length than variables.
...	further arguments passed to or from other methods

Details

Each line corresponds to one variable. A barplot is drawn for each cluster with the probabilities for each modality to be in that cluster.

See Also

barplot

Examples

```
data(birds)
xem <- mixmodCluster(birds, 2)
barplotCluster(xem["bestResult"], birds)
barplotCluster(xem["bestResult"], birds, variables = c(2, 3, 4))
barplotCluster(xem["bestResult"], birds, variables = c("eyebrow", "collar"))
```

birds *Qualitative data: morphological description of birds*

Description

The dataset 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 numeric vector defining the gender (2 modalities, male or female).

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

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

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

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

Source

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

Examples

```
data(birds)
```

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)
```

`clusteringMain` *clusteringMain*

Description

TODO: describe

`composeModelName` *Get the heterogeneous model name using Gaussian and Multinomial model name*

Description

Get the heterogeneous model name using Gaussian and Multinomial model name

Usage

```
composeModelName(g_modelname, m_modelname)
```

Arguments

`g_modelname`	Name of Gaussian model
`m_modelname`	Name of Multinomial model

Value

name of heterogeneous model

CompositeModel-class *Constructor of [CompositeModel] class*

Description

This class defines a Composite Model. Inherits the [Model] class.

Details

variable.independency logical

component.independency logical

Examples

```
new("CompositeModel")
new("CompositeModel", listModels = c("Heterogeneous_pk_E_L_B", "Heterogeneous_pk_Ekj_L_B"))
new("CompositeModel", free.proportions = FALSE, variable.independency = TRUE)

getSlots("CompositeModel")
```

CompositeParameter-class *Constructor of [CompositeParameter] class*

Description

This class defines parameters of a Heterogeneous Mixture Model. Inherits the [Parameter] class.

Details

g_parameter an object of class CompositeParameter

m_parameter an object of class MultinomialParameter

factor a numeric vector

Examples

```
new("CompositeParameter")

getSlots("CompositeParameter")
```

finance *Composite data: Financial health of companies*

Description

This data set is made up of 216 healthy firms and 212 bankruptcy firms (year 2002) and also 241 healthy firms and 220 bankruptcy firms (year 2003). Companies are described by four financial ratios expected to provide some meaningful information about their health: EBITDA/Total Assets, Value Added/Total Sales, Quick Ratio, Accounts Payable/Total Sales. This data set offers the possibility to predict the company's ability to cover its financial obligations and also to study its stability over the years.

Format

A data frame with 889 companies (rows) and 6 variables (columns).

`Year` categorical variable with two modalities (2002 & 2003).

`Health` categorical variable with two modalities (bankruptcy & healthy).

`EBITDA.Total.Assets` numeric variable.

`Value.Added.Total.Sales` numeric variable.

`Quick.Ratio` numeric variable.

`Accounts.Payable.Total.Sales` numeric variable.

Source

Lourme A, Biernacki C (2011). *Simultaneous t-Model-Based Clustering for Data Differing over Time Period: Application for Understanding Companies Financial Health.* Case Studies in Business, Industry and Government Statistics, 4(2), 73-82.

Du Jardin P, S\'everin E (2010). *Dynamic analysis of the business failure process: a study of bankruptcy trajectories.* In Portuguese Finance Network. Ponte Delgada, Portugal.

Examples

```
data(finance)
summary(finance)
```

GaussianModel-class *Constructor of [GaussianModel] class*

Description

This class defines a gaussian Model. Inherits the [Model] class.

Details

family character defining a family of models.

Examples

```
new("GaussianModel")
new("GaussianModel", family = "general")

getSlots("GaussianModel")
```

GaussianParameter-class *Constructor of [GaussianParameter] class*

Description

This class defines parameters of a Gaussian Mixture Model. Inherits the [Parameter] class.

Details

mean a numeric vector containing mean of each cluster.

variance a vector of matrix containing variance matrix of each cluster.

Examples

```
new("GaussianParameter")

getSlots("GaussianParameter")
```

`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 litres) 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)
```

heterodata *Composite data with training and testing set*

Description

The data set is made up of 5 variables: 3 categorical variables and 2 quantitative variables. The original data set contains 200 individuals. The training data set has 300 individuals while the testing data set has 100 individuals.

Format

A data frame with 200 individuals (rows) and 5 variables (columns).

`V1` categorical variable with two modalities (1 & 2).

`V2` categorical variable with two modalities (1 & 2).

`V3` categorical variable with two modalities (1 & 2).

`V4` numeric variable.

`V5` numeric variable.

See Also

`heterodatatrain` and `heterodatatest`

Examples

```
data(heterodata)
summary(heterodata)
```

heterodatatest *Composite data: A testing set*

Description

The data set is made up of 5 variables: 3 categorical variables and 2 quantitative variables. The testing data set has 100 individuals.

Format

A data frame with 100 individuals (rows) and 5 variables (columns).

`V1` categorical variable with two modalities (1 & 2).

`V2` categorical variable with two modalities (1 & 2).

`V3` categorical variable with two modalities (1 & 2).

`V4` numeric variable.

`V5` numeric variable.

See Also

heterodatatrain

Examples

```
data(heterodatatest)
summary(heterodatatest)
```

---

heterodatatrain *Composite data: A training set*

Description

The data set is made up of 5 variables: 3 categorical variables and 2 quantitative variables. The training data set has 300 individuals.

Format

A data frame with 300 individuals (rows) and 5 variables (columns).

`V1` categorical variable with two modalities (1 & 2).

`V2` categorical variable with two modalities (1 & 2).

`V3` categorical variable with two modalities (1 & 2).

`V4` numeric variable.

`V5` numeric variable.

See Also

heterodatatest

Examples

```
data(heterodatatrain)
summary(heterodatatrain)
```

hist *Histograms of a class [Mixmod]*

Description

Histograms of quantitative data from a [Mixmod] object using parameters to plot densities.

Usage

```
## S4 method for signature 'Mixmod'
hist(x, hist_x_dim = 10000, ...)
```

Arguments

x	an object of class [Mixmod]
hist_x_dim	Dimension of the histogram (???)
...	further arguments passed to or from other methods

Details

Data with the density of each cluster and the mixture density are drawn for each variable.

See Also

hist

Examples

```
data(iris)
xem <- mixmodCluster(iris[1:4], 3)
hist(xem)
hist(xem, variables = c(1, 3))
hist(xem, variables = c("Sepal.Length", "Sepal.Width"))
```

histCluster *Histogram of a class [MixmodResults]*

Description

Histograms of data object using parameters from a [MixmodResults] to plot densities.

Usage

```
histCluster(
  x,
  data,
  variables = colnames(data),
  xlab = rep("", length(variables)),
  main = paste("Histogram of", variables),
  hist_x_dim = 10000,
  ...
)
```

Arguments

x	an object of class [MixmodResults]
data	a vector or data frame containing a quantitative data set.
variables	list of variables names (or indices) to compute a histogram. All variables from data by default.
xlab	a list of title for the x axis. xlab must have the same length than variables.
main	a list of title for the histogram. main must have the same length than variables.
hist_x_dim	Dimension of the histogram (???)
...	further arguments passed to or from other methods

Details

Data with the density of each cluster and the mixture density are drawn for each variable.

See Also

hist

Examples

```
data(geyser)
xem1 <- mixmodCluster(geyser, 3)
## Not run:
histCluster(xem1["bestResult"], geyser)

## End(Not run)
histCluster(xem1["bestResult"], geyser, variables = 1)
```

is.dataType *Say if a data frame is quantitative, qualitative or composite*

Description

Say if a data frame is quantitative, qualitative or composite

Usage

```
is.dataType(x)
```

Arguments

x	a vector, a factor or a data frame

Value

a string with the data type

isQualitative *Say if a data frame contains only qualitative variables.*

Description

Say if a data frame contains only qualitative variables.

Usage

```
isQualitative(x)
```

Arguments

x	a vector or a matrix or a data frame

Value

a boolean

learnMain *learnMain*

Description

TODO: describe

Mixmod-class *Constructor of [Mixmod] class*

Description

This is a class to run mixmod library.

Details

data numeric vector or a data frame of observations. Can be qualitative,quantitative or both(heterogeneous)

dataType character. Type of data. It defines whether data is quantitative, qualitative or composite

nbCluster integer. It indicates the number of classes.

knownLabels numeric. It contains the known labels.

weight numeric vector with n (number of individuals) rows. Weight is optional. This option is to be used when weight is associated to the data.

nbVariable integer. The number of variables.

nbSample integer. The number of observations.

criterion list of character. This option permits to select the criterion giving the best configuration of an execution.

models a S4 [Model] object. Defining the list of models to be tested.

error logical. Say if at least one model finished with no error in MIXMOD.

results a list of S4 [MixmodResults] object containing all results. Results are sorted into a ascending order according to the first criterion (descending order for the CV criterion). This order can be changed by using the sortByCriterion() method.

Examples

```
getSlots("Mixmod")
```

mixmodCluster *Create an instance of the [MixmodCluster] class*

Description

This function computes an optimal mixture model according to the criteria furnished, and the list of model defined in [Model], using the algorithm specified in [Strategy].

Usage

```
mixmodCluster(...)
```

Arguments

...	all arguments are transfered to the MixmodCluster constructor. Valid arguments are:

data: frame containing quantitative,qualitative or heterogeneous data. Rows correspond to observations and columns correspond to variables.

nbCluster: numeric listing the number of clusters.

dataType: character. Type of data is "quantitative", "qualitative" or "composite". Set as NULL by default, type will be guessed depending on variables type.

models: a [Model] object defining the list of models to run. For quantitative data, the model "Gaussian_pk_Lk_C" is called (see mixmodGaussianModel() to specify other models). For qualitative data, the model "Binary_pk_Ekjh" is called (see mixmodMultinomialModel() to specify other models).

strategy: a [Strategy] object containing the strategy to run. Call mixmodStrategy() method by default.

criterion: list of character defining the criterion to select the best model. The best model is the one with the lowest criterion value. Possible values: "BIC", "ICL", "NEC", c("BIC", "ICL", "NEC"). Default is "BIC".

weight: numeric vector with n (number of individuals) rows. Weight is optional. This option is to be used when weight is associated to the data.

knownLabels: vector of size nbSample. it will be used for semi-supervised classification when labels are known. Each cell corresponds to a cluster affectation.

Value

Returns an instance of the [MixmodCluster] class. Those two attributes will contain all outputs:

results a list of [MixmodResults] object containing all the results sorted in ascending order according to the given criterion.

bestResult a S4 [MixmodResults] object containing the best model results.

Author(s)

Florent Langrognet and Remi Lebret and Christian Poli ans Serge Iovleff, with contributions from C. Biernacki and G. Celeux and G. Govaert <contact@mixmod.org>

Examples

```
## A quantitative example with the famous geyser data set
data(geyser)
## with default values
mixmodCluster(geyser, nbCluster = 2:6)

## A qualitative example with the birds data set
data(birds)
mixmodCluster(
```

```
  data = birds, nbCluster = 2:5, criterion = c("BIC", "ICL", "NEC"),
  model = mixmodMultinomialModel()
)

## use graphics functions
xem <- mixmodCluster(data = geyser, nbCluster = 3)
## Not run:
plot(xem)
hist(xem)

## End(Not run)

## get summary
summary(xem)

## A composite example with a heterogeneous data set
data(heterodata)
mixmodCluster(heterodata, 2)
```

MixmodCluster-class *Constructor of [MixmodCluster] class*

Description

This is a class to run clustering with mixmod. Inherits the [Mixmod] class.

Details

strategy a S4 [Strategy] object. Defining the strategy used to run MIXMOD.

bestResult a S4 [MixmodResults] object containing the best model results.

Examples

```
## A quantitative example with the famous iris data set
data(iris)
## with default values
new("MixmodCluster", data = iris[1:4], nbCluster = 3)

getSlots("MixmodCluster")
```

`mixmodCompositeModel` *Create an instance of the [CompositeModel] class*

Description

Define a list of heterogeneous model to test in MIXMOD.

Usage

```
mixmodCompositeModel(
  listModels = NULL,
  free.proportions = TRUE,
  equal.proportions = TRUE,
  variable.independency = NULL,
  component.independency = NULL
)
```

Arguments

`listModels`	a list of characters containing a list of models. It is optional.
`free.proportions`	logical to include models with free proportions. Default is TRUE.
`equal.proportions`	logical to include models with equal proportions. Default is TRUE.
`variable.independency`	logical to include models where $[\varepsilon_k^j]$ is independent of the variable j. Optional.
`component.independency`	logical to include models where $[\varepsilon_k^j]$ is independent of the component k. Optional.

Details

In heterogeneous case, Gaussian model can only belong to the diagonal family. We assume that the variance matrices Σ_k are diagonal. In the parameterization, it means that the orientation matrices D_k are permutation matrices. We write $\Sigma_k = \lambda_k B_k$ where B_k is a diagonal matrix with $|B_k| = 1$. This particular parameterization gives rise to 4 models: $[\lambda B]$, $[\lambda_k B]$, $[\lambda B_k]$ and $[\lambda_k B_k]$. The multinomial distribution is associated to the jth variable of the kth component is reparameterized by a center a_k^j and the dispersion ε_k^j around this center. Thus, it allows us to give an interpretation similar to the center and the variance matrix used for continuous data in the Gaussian mixture context. In the following, this model will be denoted by $[\varepsilon_k^j]$. In this context, three other models can be easily deduced. We note $[\varepsilon_k]$ the model where ε_k^j is independent of the variable j, $[\varepsilon^j]$ the model where ε_k^j is independent of the component k and, finally, $[\varepsilon]$ the model where ε_k^j is independent of both the variable \$j\$ and the component k. In order to maintain some unity in the notation, we will denote also $[\varepsilon_k^{jh}]$ the most general model introduced at the previous section.

Value

an object of [CompositeModel] which contains some of the 40 heterogeneous Models:

Model	Prop.	Var.	Comp.	Volume	Shape
Heterogeneous_p_E_L_B	Equal	TRUE	TRUE	Equal	Equal
Heterogeneous_p_E_Lk_B		TRUE	TRUE	Free	Equal
Heterogeneous_p_E_L_Bk		TRUE	TRUE	Equal	Free
Heterogeneous_p_E_Lk_Bk		TRUE	TRUE	Free	Free
Heterogeneous_p_Ek_L_B		TRUE	FALSE	Equal	Equal
Heterogeneous_p_Ek_Lk_B		TRUE	FALSE	Free	Equal
Heterogeneous_p_Ek_L_Bk		TRUE	FALSE	Equal	Free
Heterogeneous_p_Ek_Lk_Bk		TRUE	FALSE	Free	Free
Heterogeneous_p_Ej_L_B		FALSE	TRUE	Equal	Equal
Heterogeneous_p_Ej_Lk_B		FALSE	TRUE	Free	Equal
Heterogeneous_p_Ej_L_Bk		FALSE	TRUE	Equal	Free
Heterogeneous_p_Ej_Lk_Bk		FALSE	TRUE	Free	Free
Heterogeneous_p_Ekj_L_B		FALSE	FALSE	Equal	Equal
Heterogeneous_p_Ekj_Lk_B		FALSE	FALSE	Free	Equal
Heterogeneous_p_Ekj_L_Bk		FALSE	FALSE	Equal	Free
Heterogeneous_p_Ekj_Lk_Bk		FALSE	FALSE	Free	Free
Heterogeneous_p_Ekjh_L_B		FALSE	FALSE	Equal	Equal
Heterogeneous_p_Ekjh_Lk_B		FALSE	FALSE	Free	Equal
Heterogeneous_p_Ekjh_L_Bk		FALSE	FALSE	Equal	Free
Heterogeneous_p_Ekjh_Lk_Bk		FALSE	FALSE	Free	Free
Heterogeneous_pk_E_L_B	Free	TRUE	TRUE	Equal	Equal
Heterogeneous_pk_E_Lk_B		TRUE	TRUE	Free	Equal
Heterogeneous_pk_E_L_Bk		TRUE	TRUE	Equal	Free
Heterogeneous_pk_E_Lk_Bk		TRUE	TRUE	Free	Free
Heterogeneous_pk_Ek_L_B		TRUE	FALSE	Equal	Equal
Heterogeneous_pk_Ek_Lk_B		TRUE	FALSE	Free	Equal
Heterogeneous_pk_Ek_L_Bk		TRUE	FALSE	Equal	Free
Heterogeneous_pk_Ek_Lk_Bk		TRUE	FALSE	Free	Free
Heterogeneous_pk_Ej_L_B		FALSE	TRUE	Equal	Equal
Heterogeneous_pk_Ej_Lk_B		FALSE	TRUE	Free	Equal
Heterogeneous_pk_Ej_L_Bk		FALSE	TRUE	Equal	Free
Heterogeneous_pk_Ej_Lk_Bk		FALSE	TRUE	Free	Free
Heterogeneous_pk_Ekj_L_B		FALSE	FALSE	Equal	Equal
Heterogeneous_pk_Ekj_Lk_B		FALSE	FALSE	Free	Equal
Heterogeneous_pk_Ekj_L_Bk		FALSE	FALSE	Equal	Free
Heterogeneous_pk_Ekj_Lk_Bk		FALSE	FALSE	Free	Free
Heterogeneous_pk_Ekjh_L_B		FALSE	FALSE	Equal	Equal
Heterogeneous_pk_Ekjh_Lk_B		FALSE	FALSE	Free	Equal
Heterogeneous_pk_Ekjh_L_Bk		FALSE	FALSE	Equal	Free
Heterogeneous_pk_Ekjh_Lk_Bk		FALSE	FALSE	Free	Free

Author(s)

Florent Langrognet and Remi Lebret and Christian Poli ans Serge Iovleff, with contributions from C. Biernacki and G. Celeux and G. Govaert <contact@mixmod.org>

References

C. Biernacki, G. Celeux, G. Govaert, F. Langrognet. "Model-Based Cluster and Discriminant Analysis with the MIXMOD Software". Computational Statistics and Data Analysis, vol. 51/2, pp. 587-600. (2006)

Examples

```
mixmodCompositeModel()
# composite models with equal proportions
mixmodCompositeModel(free.proportions = FALSE)
# composite models with equal proportions and independent of the variable
mixmodCompositeModel(free.proportions = FALSE, variable.independency = TRUE)
# composite models with a pre-defined list
mixmodCompositeModel(listModels = c("Heterogeneous_pk_Ekjh_L_Bk", "Heterogeneous_pk_Ekjh_Lk_B"))
```

MixmodDAResults-class *Constructor of [MixmodDAResults] class*

Description

This is a class to contain results after a discriminant analysis with MIXMOD. Inherits the [MixmodResults] class.

Details

CVLabel vector of integers containing labels defined by cross validation.

CVClassification classification table after cross validation.

MAPErrorRate error rate done by MAP algorithm.

MAPClassification classification table after MAP algorithm.

Examples

```
getSlots("MixmodDAResults")
```

---

mixmodGaussianModel *Create an instance of the [GaussianModel] class*

Description

Define a list of Gaussian model to test in MIXMOD.

Usage

```
mixmodGaussianModel(
  family = "all",
  listModels = NULL,
  free.proportions = TRUE,
  equal.proportions = TRUE
)
```

Arguments

`family`	character defining a family of models. "general" for the general family, "diagonal" for the diagonal family, "spherical" for the spherical family and "all" for all families. Default is "general".
`listModels`	a list of characters containing a list of models. It is optional.
`free.proportions`	logical to include models with free proportions. Default is TRUE.
`equal.proportions`	logical to include models with equal proportions. Default is TRUE.

Details

In the Gaussian mixture model, following Banfield and Raftery (1993) and Celeux and Govaert (1995), we consider a parameterization of the variance matrices of the mixture components consisting of expressing the variance matrix Σ_k in terms of its eigenvalue decomposition

$$\Sigma_k = \lambda_k D_k A_k D_k'$$

where $\lambda_k = |\Sigma_k|^{1/d}$, D_k is the matrix of eigenvectors of Σ_k and A_k is a diagonal matrix, such that $|A_k| = 1$, with the normalized eigenvalues of Σ_k on the diagonal in a decreasing order. The parameter λ_k determines the *volume* of the kth cluster, D_k its *orientation* and A_k its *shape*. By allowing some but not all of these quantities to vary between clusters, we obtain parsimonious and easily interpreted models which are appropriate to describe various clustering situations.

In general family, we can allow the volumes, the shapes and the orientations of clusters to vary or to be equal between clusters. Variations on assumptions on the parameters λ_k, D_k and A_k $(1 \leq k \leq K)$ lead to 8 general models of interest. For instance, we can assume different volumes and keep the shapes and orientations equal by requiring that $A_k = A$ (A unknown) and $D_k = D$ (D unknown) for $k = 1, \ldots, K$. We denote this model $[\lambda_k DAD']$. With this convention, writing $[\lambda D_k A D_k']$ means that we consider the mixture model with equal volumes, equal shapes and different orientations. In diagonal family, we assume that the variance matrices Σ_k are diagonal. In the parameterization, it means that the orientation matrices D_k are permutation matrices. We write $\Sigma_k = \lambda_k B_k$ where B_k is a diagonal matrix with $|B_k| = 1$. This particular parameterization gives rise to 4 models: $[\lambda B]$, $[\lambda_k B]$, $[\lambda B_k]$ and $[\lambda_k B_k]$.

In spherical family, we assume spherical shapes, namely $A_k = I$, I denoting the identity matrix. In such a case, two parsimonious models are in competition: $[\lambda I]$ and $[\lambda_k I]$.

Value

an object of [GaussianModel] which contains some of the 28 Gaussian Models:

Model	Family	Prop.	Volume	Shape	Orient.
Gaussian_p_L_C	General	Equal	Equal	Equal	Equal
Gaussian_p_Lk_C			Free	Equal	Equal
Gaussian_p_L_D_Ak_D			Equal	Free	Equal
Gaussian_p_Lk_D_Ak_D			Free	Free	Equal
Gaussian_p_L_Dk_A_Dk			Equal	Equal	Free
Gaussian_p_Lk_Dk_A_Dk			Free	Equal	Free
Gaussian_p_L_Ck			Equal	Free	Free
Gaussian_p_Lk_Ck			Free	Free	Free
Gaussian_p_L_B	Diagonal	Equal	Equal	Equal	Axes
Gaussian_p_Lk_B			Free	Equal	Axes
Gaussian_p_L_Bk			Equal	Free	Axes
Gaussian_p_Lk_Bk			Free	Free	Axes
Gaussian_p_L_I	Spherical	Equal	Equal	Equal	NA
Gaussian_p_Lk_I			Free	Equal	NA
Gaussian_pk_L_C	General	Free	Equal	Equal	Equal
Gaussian_pk_Lk_C			Free	Equal	Equal
Gaussian_pk_L_D_Ak_D			Equal	Free	Equal
Gaussian_pk_Lk_D_Ak_D			Free	Free	Equal
Gaussian_pk_L_Dk_A_Dk			Equal	Equal	Free
Gaussian_pk_Lk_Dk_A_Dk			Free	Equal	Free
Gaussian_pk_L_Ck			Equal	Free	Free
Gaussian_pk_Lk_Ck			Free	Free	Free
Gaussian_pk_L_B	Diagonal	Free	Equal	Equal	Axes
Gaussian_pk_Lk_B			Free	Equal	Axes
Gaussian_pk_L_Bk			Equal	Free	Axes
Gaussian_pk_Lk_Bk			Free	Free	Axes
Gaussian_pk_L_I	Spherical	Free	Equal	Equal	NA
Gaussian_pk_Lk_I			Free	Equal	NA

Author(s)

Florent Langrognet and Remi Lebret and Christian Poli ans Serge Iovleff, with contributions from C. Biernacki and G. Celeux and G. Govaert <contact@mixmod.org>

References

C. Biernacki, G. Celeux, G. Govaert, F. Langrognet. "Model-Based Cluster and Discriminant Analysis with the MIXMOD Software". Computational Statistics and Data Analysis, vol. 51/2, pp. 587-600. (2006)

Examples

```
mixmodGaussianModel()
# all Gaussian models with equal proportions
mixmodGaussianModel(family = "all", free.proportions = FALSE)
# Diagonal and Spherical Gaussian models
mixmodGaussianModel(family = c("diagonal", "spherical"))
```

```
# Gaussian models with a pre-defined list
mixmodGaussianModel(listModels = c("Gaussian_p_L_C", "Gaussian_p_L_Ck", "Gaussian_pk_L_I"))
```

`mixmodLearn` *mixmodLearn*

Description

TODO: describe...

Usage

```
mixmodLearn(...)
```

Arguments

`...`	...

`MixmodLearn-class` *Constructor of [MixmodLearn] class*

Description

This is a class to run discriminant analysis with mixmod. Inherits the [Mixmod] class.

Details

bestResult a S4 [MixmodDAResults] object containing the best model results.

nbCVBlocks integer which defines the number of block to perform the Cross Validation.

Examples

```
## A quantitative example with the famous iris data set
new("MixmodLearn", data = iris[1:4], knownLabels = iris$Species)

getSlots("MixmodLearn")
```

`mixmodLearn.default` *Create an instance of the [MixmodLearn] class*

Description

This function computes the first step of a discriminant analysis. It will find the best classification rule by running an M step from the training observations.

Usage

```
mixmodLearn.default(
  data,
  knownLabels,
  dataType = NULL,
  models = NULL,
  criterion = "CV",
  nbCVBlocks = 10,
  weight = NULL
)
```

Arguments

`data`	frame containing quantitative,qualitative or heterogeneous data. Rows correspond to observations and columns correspond to variables.
`knownLabels`	an integer vector or a factor of size number of observations. Each cell corresponds to a cluster affectation. So the maximum value is the number of clusters.
`dataType`	character. Type of data is "quantitative", "qualitative" or "composite". Set as NULL by default, type will be guessed depending on variables type (in case of homogeneous data). 'composite' type must be specified explicitly.
`models`	a [Model] object defining the list of models to run. For quantitative data, the model "Gaussian_pk_Lk_C" is called (see mixmodGaussianModel() to specify other models). For qualitative data, the model "Binary_pk_Ekjh" is called (see mixmodMultinomialModel() to specify other models).
`criterion`	list of character defining the criterion to select the best model. Possible values: "BIC", "CV" or c("CV","BIC"). Default is "CV".
`nbCVBlocks`	integer which defines the number of block to perform the Cross Validation. This value will be ignored if the CV criterion is not chosen. Default value is 10.
`weight`	numeric vector with n (number of individuals) rows. Weight is optional. This option is to be used when weight is associated to the data.

Value

Returns an instance of the [MixmodLearn] class. Those two attributes will contain all outputs:

results a list of [MixmodResults] object containing all the results sorted in ascending order according to the given criterion.

bestResult a S4 [MixmodResults] object containing the best model results.

Author(s)

Florent Langrognet and Remi Lebret and Christian Poli ans Serge Iovleff, with contributions from C. Biernacki and G. Celeux and G. Govaert <contact@mixmod.org>

Examples

```
## A quantitative example with the famous iris data set
learn.iris <- mixmodLearn(iris[1:4], iris$Species)
## get summary
summary(learn.iris)

## A qualitative example with the famous birds data set
data(birds)
birds.partition <- as.integer(c(rep(1, 34), rep(2, 35)))
learn.birds <- mixmodLearn(data = birds, knownLabels = birds.partition)
## get summary
summary(learn.birds)

## A composite example with a heterogeneous data set
data(heterodatatrain)
learn.hetero <- mixmodLearn(heterodatatrain[-1], knownLabels = heterodatatrain$V1)
## get summary
summary(learn.hetero)
```

---

mixmodMultinomialModel

Create an instance of the [MultinomialModel] class

Description

Define a list of multinomial model to test in MIXMOD.

Usage

```
mixmodMultinomialModel(
  listModels = NULL,
  free.proportions = TRUE,
  equal.proportions = TRUE,
  variable.independency = NULL,
  component.independency = NULL
)
```

Arguments

listModels	a list of characters containing a list of models. It is optional.
free.proportions	logical to include models with free proportions. Default is TRUE.

`equal.proportions`
: logical to include models with equal proportions. Default is FALSE.

`variable.independency`
: logical to include models where $[\varepsilon_k^j]$ is independent of the variable j. optional.

`component.independency`
: logical to include models where $[\varepsilon_k^j]$ is independent of the component k. optional.

Details

In the multinomial mixture model, the multinomial distribution is associated to the jth variable of the kth component is reparameterized by a center a_k^j and the dispersion ε_k^j around this center. Thus, it allows us to give an interpretation similar to the center and the variance matrix used for continuous data in the Gaussian mixture context. In the following, this model will be denoted by $[\varepsilon_k^j]$. In this context, three other models can be easily deduced. We note $[\varepsilon_k]$ the model where ε_k^j is independent of the variable j, $[\varepsilon^j]$ the model where ε_k^j is independent of the component k and, finally, $[\varepsilon]$ the model where ε_k^j is independent of both the variable \$j\$ and the component k. In order to maintain some unity in the notation, we will denote also $[\varepsilon_k^{jh}]$ the most general model introduced at the previous section.

Value

an object of [`MultinomialModel`] containing some of the 10 Binary Models:

Model	Prop.	Var.	Comp.
Binary_p_E	Equal	TRUE	TRUE
Binary_p_Ej		FALSE	TRUE
Binary_p_Ek		TRUE	FALSE
Binary_p_Ekj		FALSE	FALSE
Binary_p_Ekjh		FALSE	FALSE
Binary_pk_E	Free	TRUE	TRUE
Binary_pk_Ej		FALSE	TRUE
Binary_pk_Ek		TRUE	FALSE
Binary_pk_Ekj		FALSE	FALSE
Binary_pk_Ekjh		FALSE	FALSE

Author(s)

Florent Langrognet and Remi Lebret and Christian Poli ans Serge Iovleff, with contributions from C. Biernacki and G. Celeux and G. Govaert <contact@mixmod.org>

References

C. Biernacki, G. Celeux, G. Govaert, F. Langrognet. "Model-Based Cluster and Discriminant Analysis with the MIXMOD Software". Computational Statistics and Data Analysis, vol. 51/2, pp. 587-600. (2006)

Examples

```
mixmodMultinomialModel()
# multinomial models with equal proportions
mixmodMultinomialModel(equal.proportions = TRUE, free.proportions = FALSE)
# multinomial models with a pre-defined list
mixmodMultinomialModel(listModels = c("Binary_pk_E", "Binary_p_E"))
# multinomial models with equal proportions and independent of the variable
mixmodMultinomialModel(free.proportions = FALSE, variable.independency = TRUE)
```

`mixmodPredict` *mixmodPredict*

Description

TODO: describe

Usage

```
mixmodPredict(...)
```

Arguments

`...`	...

Value

A MixmodPredict object

`MixmodPredict-class` *Constructor of [`MixmodPredict`] class*

Description

This is a class to run discriminant analysis with mixmod.

Details

data numeric vector, matrix, or data frame of observations. Either qualitative or quantitative.

dataType character. It defines whether data are quantitative or qualitative.

nbVariable integer. The number of variables.

nbSample integer. The number of observations.

error a character. The mixmod error.

classificationRule a [`MixmodResults`] object containing the classification rule.

partition a matrix containing observations to predict.

proba a matrix of probabilities.

Examples

```
# start by extract 10 observations from iris data set
remaining.obs <- sample(1:nrow(iris), 10)

# then run a mixmodLearn() analysis without those 10 observations
learn <- mixmodLearn(iris[-remaining.obs, 1:4], iris$Species[-remaining.obs])
# create a MixmodPredict to predict those 10 observations
new("MixmodPredict", data = iris[remaining.obs, 1:4], classificationRule = learn["bestResult"])

getSlots("MixmodPredict")
```

MixmodResults-class *Constructor of [MixmodResults] class*

Description

This is a class to contain results from MIXMOD library.

Details

nbCluster integer. It indicates the number of components.

model character. Name of the model.

criterion list of character. This option permits to select the criterion giving the best configuration of an execution.

criterionValue numeric. Values of the criterion.

parameters a S4 [Parameter] object. The best model parameters.

likelihood numeric. The model likelihood.

partition vector of integers defining the partition.

proba a matrix of probabilities.

error a character. The mixmod error.

Examples

```
getSlots("MixmodResults")
```

`mixmodStrategy` *Create an instance of [Strategy] class*

Description

This class will contain all the parameters needed by the estimation algorithms.

Usage

```
mixmodStrategy(...)
```

Arguments

`...` all arguments are transfered to the Strategy constructor. Valid arguments are:

algo: list of character string with the estimation algorithm. Possible values: "EM", "SEM", "CEM", c("EM","SEM"). Default value is "EM".

nbTry: integer defining the number of tries. Default value: 1.

initMethod: a character string with the method of initialization of the algorithm specified in the `algo` argument. Possible values: "random", "smallEM", "CEM", "SEMMax", "parameter", "label". Default value: "smallEM".

nbTryInInit: integer defining number of tries in `initMethod` algorithm. Default value: 50.

nbIterationInInit: integer defining the number of "EM" or "SEM" iterations in `initMethod`. Default values: 5 if `initMethod` is "smallEM" and 100 if `initMethod` is "SEMMax".

nbIterationInAlgo: list of integers defining the number of iterations if user want to use nbIteration as rule to stop the algorithm(s). Default value: 200.

epsilonInInit: real defining the epsilon value in the initialization step. Only available if `initMethod` is "smallEM". Default value: 0.001.

epsilonInAlgo: list of reals defining the epsilon value for the algorithm. Warning: epsilonInAlgo doesn't have any sense if `algo` is SEM, so it needs to be set as NaN in that case. Default value: 0.001.

seed: integer defining the seed of the random number generator. Setting a particular seed allows the user to (re)-generate a particular sequence of random numbers. Default value is NULL, i.e. a random seed.

parameter: instance of "Parameter" subclass. Required if initMethod is "parameter", forbidden otherwise.

labels: vector of integers containing labels. Required if initMethod is "label", forbidden otherwise.

Details

There are different ways to initialize an algorithm :

random Initialization from a random position is a standard way to initialize an algorithm. This random initial position is obtained by choosing at random centers in the data set. This simple strategy is repeated 5 times (the user can choose the number of times) from different random positions and the position that maximises the likelihood is selected.

smallEM A maximum of 50 iterations of the EM algorithm according to the process : n_i numbers of iterations of EM are done (with random initialization) until the `smallEM` stop criterion value has been reached. This action is repeated until the sum of n_i

reaches 50 iterations (or if in one action 50 iterations are reached before the stop criterion value).\ It appears that repeating runs of EM is generally profitable since using a single run of EM can often lead to suboptimal solutions.

CEM 10 repetitions of 50 iterations of the CEM algorithm are done. One advantage of initializing an algorithm with CEM lies in the fact that CEM converges generally in a small number of iterations. Thus, without consuming a large amount of CPU times, several runs of CEM are performed. Then EM is run with the best solution among the 10 repetitions.

SEMMax A run of 500 iterations of SEM. The idea is that an SEM sequence is expected to enter rapidly in the neighbourhood of the global maximum of the likelihood function.

Defining the algorithms used in the strategy, the stopping rule and when to stop.

- Algorithms :

  EM Expectation Maximisation

  CEM Classification EM

  SEM Stochastic EM

- Stopping rules for the algorithm :

  nbIterationInAlgo Sets the maximum number of iterations

  epsilonInAlgo Sets relative increase of the log-likelihood criterion

- Default values are 200 `nbIterationInAlgo` of `EM` with an `epsilonInAlgo` value of $10 - 3$.

Value

a [`Strategy`] object

Author(s)

Florent Langrognet and Remi Lebret and Christian Poli ans Serge Iovleff, with contributions from C. Biernacki and G. Celeux and G. Govaert `<contact@mixmod.org>`

References

Biernacki, C., Celeux, G., Govaert, G., 2003. "Choosing starting values for the EM algorithm for getting the highest likelihood in multivariate gaussian mixture models". Computational Statistics and Data Analysis 41, 561-575.

Examples

```
mixmodStrategy()
mixmodStrategy(algo = "CEM", initMethod = "random", nbTry = 10, epsilonInInit = 0.00001)
mixmodStrategy(
  algo = c("SEM", "EM"), nbIterationInAlgo = c(200, 100),
  epsilonInAlgo = c(NA, 0.000001)
)
```

`mixmodXmlCheck` *mixmodXmlCheck*

Description

TODO: describe...

Usage

```
mixmodXmlCheck(...)
```

Arguments

`...`	...

Value

Object of type MixmodXmlCheck

`MixmodXmlCheck-class` *Constructor of [MixmodXmlCheck] class*

Description

This is a class to handle XML files (TODO: describe...)

`mixmodXmlInput` *mixmodXmlInput*

Description

TODO: describe..

Usage

```
mixmodXmlInput(...)
```

Arguments

`...`	...

MixmodXmlInput-class *Constructor of [MixmodXmlInput] class*

Description

This is ...

mixmodXmlLoad *mixmodXmlLoad*

Description

TODO: describe...

Usage

```
mixmodXmlLoad(xmlFile, numFormat = "humanReadable")
```

Arguments

xmlFile	...
numFormat	...

Value

XML output of mixmod methods

Model-class *Constructor of [Model] class*

Description

This class defines the Mixmod models.

Details

listModels character containing a list of models.

free.proportions logical to include models with free proportions. Default is TRUE.

equal.proportions logical to include models with equal proportions. Default is FALSE.

Examples

```
getSlots("Model")
```

MultinomialModel-class

MultinomialModel

Description

Constructor of [MultinomialModel] class

Details

This class defines a multinomial Model. Inherits the [Model] class.

variable.independency logical

component.independency logical

Examples

```
new("MultinomialModel")
new("MultinomialModel", listModels = c("Binary_pk_E", "Binary_p_E"))
new("MultinomialModel", free.proportions = FALSE, variable.independency = TRUE)

getSlots("MultinomialModel")
```

MultinomialParameter-class

Constructor of [MultinomialParameter] class

Description

This class defines parameters of a Multinomial Mixture Model. Inherits the [Parameter] class.

Details

center a numeric vector containing center of each cluster.

scatter a vector of matrix containing dispersion matrix of each cluster.

factor a character vector containing the modalities.

Examples

```
new("MultinomialParameter")

getSlots("MultinomialParameter")
```

nbFactorFromData — *Get the number of modalities for each column of a categorical data set*

Description

Get the number of modalities for each column of a categorical data set

Usage

```
nbFactorFromData(x)
```

Arguments

x	a vector or a matrix or a data frame containing factors

Value

a vector containing the number of modalities for each column

oldmixmodPredict — *Create an instance of the [MixmodPredict] class*

Description

This function computes the second step of a discriminant analysis. The aim of this step is to assign remaining observations to one of the groups.

Usage

```
oldmixmodPredict(data, classificationRule, ...)
```

Arguments

data	matrix or data frame containing quantitative,qualitative or composite data. Rows correspond to observations and columns correspond to variables.
classificationRule	a [MixmodResults] object which contains the classification rule computed in the mixmodLearn() or mixmodCluster() step.
...	...

Value

Returns an instance of the [MixmodPredict] class which contains predicted partition and probabilities.

Author(s)

Florent Langrognet and Remi Lebret and Christian Poli ans Serge Iovleff, with contributions from C. Biernacki and G. Celeux and G. Govaert <contact@mixmod.org>

Examples

```
# start by extract 10 observations from iris data set
remaining.obs <- sample(1:nrow(iris), 10)
# then run a mixmodLearn() analysis without those 10 observations
learn <- mixmodLearn(iris[-remaining.obs, 1:4], iris$Species[-remaining.obs])
# create a MixmodPredict to predict those 10 observations
prediction <- mixmodPredict(
  data = iris[remaining.obs, 1:4],
  classificationRule = learn["bestResult"]
)
# show results
prediction
# compare prediction with real results
paste("accuracy= ", mean(as.integer(iris$Species[remaining.obs]) == prediction["partition"]) * 100,
  "%",
  sep = ""
)

## A composite example with a heterogeneous data set
data(heterodatatrain)
## Learning with training data
learn <- mixmodLearn(heterodatatrain[-1], knownLabels = heterodatatrain$V1)
```

Parameter-class *Constructor of [Parameter] class*

Description

This class defines parameters of a Mixture Model.

Details

proportions a numeric vector containing proportions of the mixture model.

Examples

```
getSlots("Parameter")
```

plot *Plotting of a class [Mixmod]*

Description

Plotting data from a [Mixmod] object using parameters and partition to distinguish the different clusters.

Usage

```
## S4 method for signature 'Mixmod'
plot(x, y, showOnly = NULL, withResult = NULL, hist_x_dim = 10000, ...)
```

Arguments

x	an object of class [Mixmod]
y	a list of variables to plot (subset). Variables names or indices. Only in a quantitative case.
showOnly	show only (...)
withResult	with result (...)
hist_x_dim	Histogram dimension (???)
...	further arguments passed to or from other methods

Details

For quantitative case, ellipsoids (i.e. linear transformations of hyperspheres) centered at the mean are drawn using the parameters computed by MIXMOD. The directions of the principal axes of the ellipsoids are given by the eigenvectors of the covariance matrix Σ. The squared relative lengths of the principal axes are given by the corresponding eigenvalues. A 1-dimensional representation of variables with the densities is drawn on the diagonal.

For qualitative case, a Multiple Correspondence Analysis is performed to get a 2-dimensional representation of the data set. Bigger symbol means that observations are similar.

See Also

plot

Examples

```
## for quantitative case
data(iris)
xem <- mixmodCluster(iris[1:4], 3)
plot(xem)
plot(xem, c(1, 3))
plot(xem, c("Sepal.Length", "Sepal.Width"))
```

```
## for qualitative case
data(birds)
xem2 <- mixmodCluster(birds, 2)
plot(xem2)
legend("bottomleft", c("Cluster1", "Cluster2"), col = c(2, 3), pch = c(1, 2))
```

---

plotCluster *Plotting of a class [MixmodResults]*

Description

Biplot of two variables from a quantitative data set. Use parameters and partition from a [MixmodResults] object to distinguish the different clusters.

Usage

```
plotCluster(
  x,
  data,
  variable1 = colnames(data)[1],
  variable2 = colnames(data)[2],
  col = x@partition + 1,
  pch = x@partition,
  xlab = variable1,
  ylab = variable2,
  add.ellipse = TRUE,
  ...
)
```

Arguments

x	an object of class [MixmodResults]
data	a data frame containing a quantitative data set.
variable1	index or character containing the name of the first variable. First column of data by default.
variable2	index or character containing the name of the second variable. Second column of data by default.
col	a specification for the default plotting color. By default partition is used to separate clusters with different colors.
pch	either an integer specifying a symbol or a single character to be used as the default in plotting points. By default partition is used to separate clusters with different symbols.
xlab	a title for the x axis. Variable1 by default.
ylab	a title for the y axis. Variable2 by default.
add.ellipse	a boolean. Add ellipses to graph. TRUE by default.
...	further arguments passed to or from other methods

Details

Ellipsoids (i.e. linear transformations of hyperspheres) centered at the mean can be drawn using the parameters computed by MIXMOD. The directions of the principal axes of the ellipsoids are given by the eigenvectors of the covariance matrix Σ. The squared relative lengths of the principal axes are given by the corresponding eigenvalues.

See Also

plot

Examples

```
data(geyser)
xem1 <- mixmodCluster(geyser, 3)
plotCluster(xem1["bestResult"], geyser)

data(iris)
xem2 <- mixmodCluster(iris[1:4], 2:6)
plotCluster(xem2["bestResult"], iris, variable1 = "Sepal.Length", variable2 = "Sepal.Width")
plotCluster(xem2["bestResult"], iris, variable1 = 1, variable2 = 4)
```

---

predictMain *predictMain*

Description

TODO: describe

---

print *Print a Rmixmod class to standard output.*

Description

Print a Rmixmod class to standard output.

Usage

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

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

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

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

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

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

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

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

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

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

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

Arguments

x	a Rmixmod object: a Strategy, a Model, a GaussianParameter, a MultinomialParameter, a MixmodResults, a MixmodCluster, a MixmodLearn or a MixmodPredict.
...	further arguments passed to or from other methods

Value

NULL. Prints to standard out.

See Also

print

Examples

```
## for strategy
strategy <- mixmodStrategy()
print(strategy)

## for Gaussian models
gmodel <- mixmodGaussianModel()
print(gmodel)
## for multinomial models
mmodel <- mixmodMultinomialModel()
print(mmodel)
```

```
## for clustering
data(geyser)
xem <- mixmodCluster(geyser, 3)
print(xem)
## for Gaussian parameters
print(xem["bestResult"]["parameters"])

## for discriminant analysis
# start by extract 10 observations from iris data set
iris.partition <- sample(1:nrow(iris), 10)
# then run a mixmodLearn() analysis without those 10 observations
learn <- mixmodLearn(iris[-iris.partition, 1:4], iris$Species[-iris.partition])
# print learn results
print(learn)
# create a MixmodPredict to predict those 10 observations
prediction <- mixmodPredict(
  data = iris[iris.partition, 1:4],
  classificationRule = learn["bestResult"]
)
# print prediction results
print(prediction)
```

---

show *Show description of a Rmixmod class to standard output.*

Description

Show description of a Rmixmod class to standard output.

Usage

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

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

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

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

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

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

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

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

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

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

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

Arguments

object	a Rmixmod object: a Strategy, a Model, a GaussianParameter, a MultinomialParameter, a MixmodResults, a MixmodCluster, a MixmodLearn or a MixmodPredict.

Value

NULL. Prints to standard out.

See Also

show

Examples

```
## for strategy
strategy <- mixmodStrategy()
show(strategy)

## for Gaussian models
gmodel <- mixmodGaussianModel()
show(gmodel)
## for multinomial models
mmodel <- mixmodMultinomialModel()
show(mmodel)

## for clustering
data(geyser)
xem <- mixmodCluster(geyser, 3)
show(xem)
## for Gaussian parameters
show(xem["bestResult"]["parameters"])

## for discriminant analysis
# start by extract 10 observations from iris data set
```

```
iris.partition <- sample(1:nrow(iris), 10)
# then run a mixmodLearn() analysis without those 10 observations
learn <- mixmodLearn(iris[-iris.partition, 1:4], iris$Species[-iris.partition])
# create a MixmodPredict to predict those 10 observations
prediction <- mixmodPredict(
  data = iris[iris.partition, 1:4],
  classificationRule = learn["bestResult"]
)
# show results
show(prediction)
```

`sortByCriterion` *Sorting results of a [Mixmod] object by a given criterion*

Description

After calling the mixmodCluster() or mixmodLearn() method, results will be sorted into ascending order according to the first given criterion (descending order for CV criterion). This method is able to reorder the list of results according to a given criterion.

Usage

```
sortByCriterion(object, criterion)

## S4 method for signature 'Mixmod,character'
sortByCriterion(object, criterion)
```

Arguments

`object`	a [Mixmod] object
`criterion`	a string containing the criterion name

Value

a modified [Mixmod] object

Examples

```
x <- mixmodCluster(iris[1:4], 2:10, criterion = c("BIC", "ICL"))
icl <- sortByCriterion(x, "ICL")
icl["results"]
```

Strategy-class *Constructor of [Strategy] class*

Description

This class defines the Mixmod strategies.

Details

algo: list of character string with the estimation algorithm. Possible values: "EM", "SEM", "CEM", c("EM","SEM"). Default value is "EM".

nbTry: integer defining the number of tries. Default value: 1.

initMethod: a character string with the method of initialization of the algorithm specified in the `algo` argument. Possible values: "random", "smallEM", "CEM", "SEMMax", "parameter", "label". Default value: "smallEM".

nbTryInInit: integer defining number of tries in `initMethod` algorithm. Default value: 50.

nbIterationInInit: integer defining the number of "EM" or "SEM" iterations in `initMethod`. Default values: 5 if `initMethod` is "smallEM" and 100 if `initMethod` is "SEMMax".

nbIterationInAlgo: list of integers defining the number of iterations if user want to use nbIteration as rule to stop the algorithm(s). Default value: 200.

epsilonInInit: real defining the epsilon value in the initialization step. Only available if `initMethod` is "smallEM". Default value: 0.001.

epsilonInAlgo: list of reals defining the epsilon value for the algorithm. Warning: epsilonInAlgo doesn't have any sense if `algo` is SEM, so it needs to be set as NaN in that case. Default value: 0.001.

seed: integer defining the seed of the random number generator. Setting a particular seed allows the user to (re)-generate a particular sequence of random numbers. Default value is NULL, i.e. a random seed.

parameter: instance of "Parameter" subclass. Required if initMethod is "parameter", forbidden otherwise.

labels: vector of integers containing labels. Required if initMethod is "label", forbidden otherwise.

Examples

```
new("Strategy")
new("Strategy", algo = "SEM", initMethod = "SEMMax")

getSlots("Strategy")
```

summary *Produce result summaries of a Rmixmod class*

Description

Produce result summaries of a Rmixmod class

Usage

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

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

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

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

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

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

Arguments

object	An object (???)
...	further arguments passed to or from other methods

Value

NULL. Summaries to standard out.

See Also

summary

Examples

```
data(geyser)
xem <- mixmodCluster(geyser, 3)
summary(xem)
summary(xem["bestResult"])
summary(xem["bestResult"]["parameters"])
```

`titanic` *Qualitative data: Survival of passengers on the Titanic*

Description

For each person on board the fatal maiden voyage of the ocean liner Titanic, this dataset records: sex, age [adult/child], economic status [first/second/third class, or crew] and whether or not that person survived. Values are aligned and delimited by blanks. There are no missing values.

Format

A data frame with 2201 observations on the following 4 variables.

`Class` 0 = crew, 1 = first, 2 = second, 3 = third, which denote the economic status of the subject

`Age` 1 = adult, 0 = child, which denote if the subject is an adult or a child

`Sex` 1 = male, 0 = female, which denote the sex of the subject

`Survived` 1 = yes, 0 = no, which denote if the subject lived through the fatal maiden voyage of the ocean liner Titanic

Details

The sinking of the Titanic is a famous event, and new books are still being published about it. Many well-known facts-from the proportions of first-class passengers to the "women and children first" policy, and the fact that that policy was not entirely successful in saving the women and children in the third class-are reflected in the survival rates for various classes of passenger.

These data were originally collected by the British Board of Trade in their investigation of the sinking. Note that there is not complete agreement among primary sources as to the exact numbers on board, rescued, or lost.

Due in particular to the very successful film "Titanic", the last years saw a rise in public interest in the Titanic. Very detailed data about the passengers is now available on the Internet, at sites such as "Encyclopedia Titanica".

Source

The source provides a data set recording class, sex, age, and survival status for each person on board of the Titanic, and is based on data originally collected by the British Board of Trade and reprinted in: British Board of Trade (1990), "Report on the Loss of the Titanic (S.S.)". British Board of Trade Inquiry Report (reprint). Gloucester, UK: Allan Sutton Publishing.

Examples

```
data(titanic)
```

xMain *xMain*

Description

TODO: describe

[*Extract parts of a Rmixmod class*

Description

Extract parts of a Rmixmod class

Usage

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

## S4 method for signature 'MixmodPredict,ANY,ANY,ANY'
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	TODO...

Index