

# Package ‘RMixtCompUtilities’

December 19, 2019

**Type** Package

**Title** Utility Functions for 'MixtComp' Outputs

**Version** 4.1.0

**Date** 2019-12-16

**Copyright** Inria - Université de Lille - CNRS

**License** AGPL-3

**Description** Mixture Composer <<https://github.com/modal-inria/MixtComp>> is a project to build mixture models with heterogeneous data sets and partially missing data management. This package contains graphical, getter and some utility functions to facilitate the analysis of 'MixtComp' output.

**URL** <https://github.com/modal-inria/MixtComp>,  
<https://massiccc.lille.inria.fr/>

**BugReports** <https://github.com/modal-inria/MixtComp/issues>

**Imports** plotly, ggplot2, scales

**Suggests** testthat, xml2, RMixtCompIO, Rmixmod, blockcluster

**RoxygenNote** 7.0.2

**Encoding** UTF-8

**NeedsCompilation** no

**Author** Vincent Kubicki [aut],  
Christophe Biernacki [aut],  
Quentin Grimonprez [aut, cre],  
Matthieu Marbac-Lourdelle [aut],  
Étienne Goffinet [ctb],  
Serge Iovleff [ctb]

**Maintainer** Quentin Grimonprez <[quentin.grimonprez@inria.fr](mailto:quentin.grimonprez@inria.fr)>

**Repository** CRAN

**Date/Publication** 2019-12-19 17:50:22 UTC

**R topics documented:**

RMixtCompUtilities-package . . . . .	2
availableModels . . . . .	3
computeDiscrimPowerVar . . . . .	4
computeSimilarityVar . . . . .	6
convertFunctionalToVector . . . . .	7
createAlgo . . . . .	8
createFunctional . . . . .	9
getBIC . . . . .	10
getCompletedData . . . . .	11
getEmpiricTik . . . . .	12
getParam . . . . .	14
getPartition . . . . .	15
getType . . . . .	16
heatmapClass . . . . .	18
heatmapTikSorted . . . . .	19
heatmapVar . . . . .	20
histMisclassif . . . . .	22
plot.MixtComp . . . . .	23
plotConvergence . . . . .	24
plotDataBoxplot . . . . .	26
plotDataCI . . . . .	27
plotDiscrimClass . . . . .	29
plotDiscrimVar . . . . .	30
plotParamConvergence . . . . .	31
plotProportion . . . . .	33
print.MixtComp . . . . .	34
refactorCategorical . . . . .	35
summary.MixtComp . . . . .	36
<b>Index</b>	<b>38</b>

---

 RMixtCompUtilities-package

*RMixtCompUtilities*


---

**Description**

MixtComp (Mixture Composer, <https://github.com/modal-inria/MixtComp>) is a model-based clustering package for mixed data originating from the Modal team (Inria Lille).

It has been engineered around the idea of easy and quick integration of all new univariate models, under the conditional independence assumption. Five basic models (Gaussian, Multinomial, Poisson, Weibull, NegativeBinomial) are implemented, as well as two advanced models (Func\_CS and Rank\_ISR). MixtComp has the ability to natively manage missing data (completely or by interval). MixtComp is used as an R package, but its internals are coded in C++ using state of the art libraries for faster computation.

Online SaaS version (not up-to-date): <https://massiccc.lille.inria.fr/>

This package contains plots, getters and format functions to simplify the use of RMixtComp and RMixtCompIO packages. It is recommended to use RMixtComp (instead of RMixtCompIO) which is more user-friendly.

## Details

[createAlgo](#) gives you default values for required parameters.

[convertFunctionalToVector](#), [createFunctional](#) and [refactorCategorical](#) functions help to transform data to the required format.

Getters are available to easily access some results: [getBIC](#), [getICL](#), [getCompletedData](#), [getParam](#), [getTik](#), [getEmpiricTik](#), [getPartition](#), [getType](#), [getModel](#), [getVarNames](#).

You can compute discriminative powers and similarities with functions: [computeDiscrimPowerClass](#), [computeDiscrimPowerVar](#), [computeSimilarityClass](#), [computeSimilarityVar](#).

Graphics functions are [plot.MixtComp](#), [heatmapClass](#), [heatmapTikSorted](#), [heatmapVar](#), [histMisclassif](#), [plotConvergence](#), [plotDataBoxplot](#), [plotDataCI](#), [plotDiscrimClass](#), [plotDiscrimVar](#), [plotProportion](#).

## See Also

RMixtComp RMixtCompIO Rmixmod, blockcluster packages

---

availableModels	<i>Available models</i>
-----------------	-------------------------

---

## Description

Get information about models implemented in MixtComp

## Usage

```
availableModels()
```

## Value

a data.frame containing models implemented in MixtComp

**model** model name

**data.type** data type

**format** Special format required for individuals

**missing.formats** accepted formats (separated by a ;) for missing values

**hyperparameter** Required hyperparameters in the paramStr elements of model object

**comments** comments about the model

**reference** link to article

**Author(s)**

Quentin Grimonprez

**See Also**

mixtCompLearn

**Examples**

availableModels()

---

computeDiscrimPowerVar

*Discriminative power*

---

**Description**

Compute the discriminative power of each variable or classe

**Usage**

computeDiscrimPowerVar(outMixtComp)

computeDiscrimPowerClass(outMixtComp)

**Arguments**

outMixtComp      object of class *MixtCompLearn* or *MixtComp* obtained using `mixtCompLearn` or `mixtCompPredict` functions from `RMixtComp` package or `rmcMultiRun` from `RMixtCompIO` package.

**Details**

The discriminative power of variable  $j$  is defined by  $1 - C(j)$

$$C(j) = - \sum_{k=1}^K \sum_{i=1}^n P(Z_i = k|x_{ij}) \log(P(Z_i = k|x_{ij})) / (n * \log(K))$$

A high value (close to one) means that the variable is highly discriminating. A low value (close to zero) means that the variable is poorly discriminating.

The discriminative power of class  $k$  is defined by  $1 - D(k)$

$$D(k) = - \sum_{i=1}^n P(Z_i = k|x_i) \log(P(Z_i = k|x_i)) / (n * \exp(-1))$$

**Value**

the discriminative power

**Author(s)**

Matthieu Marbac

**See Also**

[plotDiscrimClass](#) [plotDiscrimVar](#)

**Examples**

```
require(RMixture) # for learning a mixture model
dataLearn <- list(var1 = as.character(c(rnorm(50, -2, 0.8), rnorm(50, 2, 0.8))),
                 var2 = as.character(c(rnorm(50, 2), rpois(50, 8))))

model <- list(var1 = list(type = "Gaussian", paramStr = ""),
             var2 = list(type = "Poisson", paramStr = ""))

algo <- list(
  nClass = 2,
  nInd = 100,
  nbBurnInIter = 100,
  nbIter = 100,
  nbGibbsBurnInIter = 100,
  nbGibbsIter = 100,
  nInitPerClass = 3,
  nSemTry = 20,
  confidenceLevel = 0.95,
  ratioStableCriterion = 0.95,
  nStableCriterion = 10,
  mode = "learn"
)

resLearn <- rmcMultiRun(algo, dataLearn, model, nRun = 3)

discVar <- computeDiscrimPowerVar(resLearn)
discClass <- computeDiscrimPowerClass(resLearn)

# graphic representation of discriminant variables
plotDiscrimVar(resLearn)
# graphic representation of discriminant classes
plotDiscrimClass(resLearn)
```

---

computeSimilarityVar    *Similarity*

---

### Description

Compute the similarity between variables (or classes)

### Usage

```
computeSimilarityVar(outMixtComp)
```

```
computeSimilarityClass(outMixtComp)
```

### Arguments

outMixtComp    object of class *MixtCompLearn* or *MixtComp* obtained using `mixtCompLearn` or `mixtCompPredict` functions from *RMixtComp* package or `rmcMultiRun` from *RMixtCompIO* package.

### Details

The similarities between variables  $j$  and  $h$  is defined by  $\Delta(j,h)$

$$\Delta(j, h)^2 = 1 - \sqrt{(1/n) * \sum_{i=1}^n \sum_{k=1}^K (P(Z_i = k|x_{ij}) - P(Z_i = k|x_{ih}))^2}$$

The similarities between classes  $k$  and  $g$  is defined by  $1 - \Sigma(k,g)$

$$\Sigma(k, g)^2 = (1/n) * \sum_{i=1}^n (P(Z_i = k|x_i) - P(Z_i = g|x_i))^2$$

### Value

a similarity matrix

### Author(s)

Quentin Grimonprez

### See Also

[heatmapVar](#) [heatmapClass](#)

**Examples**

```
require(RMixtureCompIO) # for learning a mixture model
dataLearn <- list(var1 = as.character(c(rnorm(50, -2, 0.8), rnorm(50, 2, 0.8))),
                 var2 = as.character(c(rnorm(50, 2), rpois(50, 8))))

model <- list(var1 = list(type = "Gaussian", paramStr = ""),
             var2 = list(type = "Poisson", paramStr = ""))

algo <- list(
  nClass = 2,
  nInd = 100,
  nbBurnInIter = 100,
  nbIter = 100,
  nbGibbsBurnInIter = 100,
  nbGibbsIter = 100,
  nInitPerClass = 3,
  nSemTry = 20,
  confidenceLevel = 0.95,
  ratioStableCriterion = 0.95,
  nStableCriterion = 10,
  mode = "learn"
)

resLearn <- rmcMultiRun(algo, dataLearn, model, nRun = 3)

simVar <- computeSimilarityVar(resLearn)
simClass <- computeSimilarityClass(resLearn)
```

---

convertFunctionalToVector

*Convert a mixtcomp string into 2-columns matrix*

---

**Description**

Convert a mixtcomp string into 2-columns matrix

**Usage**

```
convertFunctionalToVector(x)
```

**Arguments**

x a string containing a functional observation (cf example)

**Value**

matrix

**Author(s)**

Quentin Grimonprez

**Examples**

```
convertFunctionalToVector("1:5,1.5:12,1.999:2.9")
```

---

createAlgo

*Create algo object*

---

**Description**

create an algo object required by mixtCompLearn and mixtCompPredict from RMixtComp.

**Usage**

```
createAlgo(
  nbBurnInIter = 50,
  nbIter = 50,
  nbGibbsBurnInIter = 50,
  nbGibbsIter = 50,
  nInitPerClass = 10,
  nSemTry = 20,
  confidenceLevel = 0.95,
  ratioStableCriterion = 0.99,
  nStableCriterion = 20
)
```

**Arguments**

nbBurnInIter	Number of iterations of the burn-in part of the SEM algorithm.
nbIter	Number of iterations of the SEM algorithm.
nbGibbsBurnInIter	Number of iterations of the burn-in part of the Gibbs algorithm.
nbGibbsIter	Number of iterations of the Gibbs algorithm.
nInitPerClass	Number of individuals used to initialize each cluster (default = 10).
nSemTry	Number of try of the algorithm for avoiding an error.
confidenceLevel	confidence level for confidence bounds for parameter estimation
ratioStableCriterion	stability partition required to stop earlier the SEM
nStableCriterion	number of iterations of partition stability to stop earlier the SEM



**Value**

a list with the parameters values

**Author(s)**

Quentin Grimonprez

**Examples**

```
# default values
algo <- createAlgo()

# change some values
algo <- createAlgo(nbIter = 200)
```

---

createFunctional	<i>Create a functional in MixtComp format</i>
------------------	---

---

**Description**

Create a functional in MixtComp format

**Usage**

```
createFunctional(time, value)
```

**Arguments**

time	vector containing the time of the functional
value	vector containing the value of the functional

**Value**

The functional data formatted to the mixtcomp standard

**Author(s)**

Quentin Grimonprez

**Examples**

```
mat <- matrix(c(1, 2, 3, 9, 1, 1.5, 15, 1000), ncol = 2)
createFunctional(mat[,1], mat[,2])
```

---

getBIC	<i>Get criterion value</i>
--------	----------------------------

---

**Description**

Get criterion value

**Usage**

```
getBIC(outMixtComp)
```

```
getICL(outMixtComp)
```

**Arguments**

outMixtComp      object of class *MixtCompLearn* or *MixtComp* obtained using `mixtCompLearn` or `mixtCompPredict` functions from `RMixtComp` package or `rmcMultiRun` from `RMixtCompIO` package.

**Value**

value of the criterion

**Author(s)**

Quentin Grimonprez

**See Also**

Other getter: [getCompletedData\(\)](#), [getEmpiricTik\(\)](#), [getParam\(\)](#), [getPartition\(\)](#), [getType\(\)](#)

**Examples**

```
require(RMixtCompIO) # for learning a mixture model
dataLearn <- list(var1 = as.character(c(rnorm(50, -2, 0.8), rnorm(50, 2, 0.8))),
                 var2 = as.character(c(rnorm(50, 2), rpois(50, 8))))
```

```
model <- list(var1 = list(type = "Gaussian", paramStr = ""),
              var2 = list(type = "Poisson", paramStr = ""))
```

```
algo <- list(
  nClass = 2,
  nInd = 100,
  nbBurnInIter = 100,
  nbIter = 100,
  nbGibbsBurnInIter = 100,
  nbGibbsIter = 100,
  nInitPerClass = 3,
  nSemTry = 20,
```

```
    confidenceLevel = 0.95,  
    ratioStableCriterion = 0.95,  
    nStableCriterion = 10,  
    mode = "learn"  
  )  
  
  resLearn <- rmcMultiRun(algo, dataLearn, model, nRun = 3)  
  
  # get criterion  
  bic <- getBIC(resLearn)  
  icl <- getICL(resLearn)
```

---

getCompletedData	<i>Get the completed data from MixtComp object</i>
------------------	--

---

### Description

Get the completed data from MixtComp object

### Usage

```
getCompletedData(outMixtComp, var = NULL, with.z_class = FALSE)
```

### Arguments

outMixtComp	object of class <i>MixtCompLearn</i> or <i>MixtComp</i> obtained using <code>mixtCompLearn</code> or <code>mixtCompPredict</code> functions from <code>RMixtComp</code> package or <code>rmcMultiRun</code> from <code>RMixtCompIO</code> package.
var	Name of the variables for which to extract the completed data. Default is <code>NULL</code> (all variables are extracted)
with.z_class	if <code>TRUE</code> , <code>z_class</code> is returned with the data.

### Value

a matrix with the data completed by `MixtComp` (`z_class` is in the first column and then variables are sorted in alphabetic order, it may differ from the original order of the data).

### Author(s)

Quentin Grimonprez

### See Also

Other getter: [getBIC\(\)](#), [getEmpiricTik\(\)](#), [getParam\(\)](#), [getPartition\(\)](#), [getType\(\)](#)

**Examples**

```

require(RMixtCompIO) # for learning a mixture model
dataLearn <- list(var1 = as.character(c(rnorm(50, -2, 0.8), rnorm(50, 2, 0.8))),
                 var2 = as.character(c(rnorm(50, 2), rpois(50, 8))))

# add missing values
dataLearn$var1[12] = "?"
dataLearn$var2[72] = "?"

model <- list(var1 = list(type = "Gaussian", paramStr = ""),
              var2 = list(type = "Poisson", paramStr = ""))

algo <- list(
  nClass = 2,
  nInd = 100,
  nbBurnInIter = 100,
  nbIter = 100,
  nbGibbsBurnInIter = 100,
  nbGibbsIter = 100,
  nInitPerClass = 3,
  nSemTry = 20,
  confidenceLevel = 0.95,
  ratioStableCriterion = 0.95,
  nStableCriterion = 10,
  mode = "learn"
)

resLearn <- rmcMultiRun(algo, dataLearn, model, nRun = 3)

# get completedData
completedData <- getCompletedData(resLearn)
completedData2 <- getCompletedData(resLearn, var = "var1")

```

---

getEmpiricTik

*Get the tik*


---

**Description**

Get the a posteriori probability to belong to each class for each individual

**Usage**

```
getEmpiricTik(outMixtComp)
```

```
getTik(outMixtComp, log = TRUE)
```

**Arguments**

outMixtComp	object of class <i>MixtCompLearn</i> or <i>MixtComp</i> obtained using <code>mixtCompLearn</code> or <code>mixtCompPredict</code> functions from <code>RMixtComp</code> package or <code>rmcMultiRun</code> from <code>RMixtCompIO</code> package.
log	if TRUE, <code>log(tik)</code> are returned

**Details**

`getTik` returns a posteriori probabilities computed with the returned parameters. `getEmpiricTik` returns an estimation based on the sampled `z_i` during the algorithm.

**Value**

a matrix containing the tik for each individuals (in row) and each class (in column).

**Author(s)**

Quentin Grimonprez

**See Also**

[heatmapTikSorted](#)

Other getter: `getBIC()`, `getCompletedData()`, `getParam()`, `getPartition()`, `getType()`

**Examples**

```
require(RMixtCompIO) # for learning a mixture model
dataLearn <- list(var1 = as.character(c(rnorm(50, -2, 0.8), rnorm(50, 2, 0.8))),
                var2 = as.character(c(rnorm(50, 2), rpois(50, 8))))

model <- list(var1 = list(type = "Gaussian", paramStr = ""),
             var2 = list(type = "Poisson", paramStr = ""))

algo <- list(
  nClass = 2,
  nInd = 100,
  nbBurnInIter = 100,
  nbIter = 100,
  nbGibbsBurnInIter = 100,
  nbGibbsIter = 100,
  nInitPerClass = 3,
  nSemTry = 20,
  confidenceLevel = 0.95,
  ratioStableCriterion = 0.95,
  nStableCriterion = 10,
  mode = "learn"
)

resLearn <- rmcMultiRun(algo, dataLearn, model, nRun = 3)
```

```
# get tik
tikEmp <- getEmpiricTik(resLearn)
tik <- getTik(resLearn, log = FALSE)
```

---

getParam	<i>Get the estimated parameter</i>
----------	------------------------------------

---

### Description

Get the estimated parameter

### Usage

```
getParam(outMixtComp, var)
```

### Arguments

outMixtComp	object of class <i>MixtCompLearn</i> or <i>MixtComp</i> obtained using <code>mixtCompLearn</code> or <code>mixtCompPredict</code> functions from <code>RMixtComp</code> package or <code>rncMultiRun</code> from <code>RMixtCompIO</code> package.
var	name of the variable to get parameter

### Value

the parameter of the variable

### Author(s)

Quentin Grimonprez

### See Also

[plotDataBoxplot](#) [plotDataCI](#)

Other getter: [getBIC\(\)](#), [getCompletedData\(\)](#), [getEmpiricTik\(\)](#), [getPartition\(\)](#), [getType\(\)](#)

### Examples

```
require(RMixtCompIO) # for learning a mixture model
dataLearn <- list(var1 = as.character(c(rnorm(50, -2, 0.8), rnorm(50, 2, 0.8))),
                 var2 = as.character(c(rnorm(50, 2), rpois(50, 8))))

model <- list(var1 = list(type = "Gaussian", paramStr = ""),
              var2 = list(type = "Poisson", paramStr = ""))

algo <- list(
  nClass = 2,
  nInd = 100,
  nbBurnInIter = 100,
```

```
nbIter = 100,
nbGibbsBurnInIter = 100,
nbGibbsIter = 100,
nInitPerClass = 3,
nSemTry = 20,
confidenceLevel = 0.95,
ratioStableCriterion = 0.95,
nStableCriterion = 10,
mode = "learn"
)

resLearn <- rmcMultiRun(algo, dataLearn, model, nRun = 3)

# get estimated parameters for variable var1
param <- getParam(resLearn, "var1")
```

---

getPartition

*Get the estimated class from MixtComp object*

---

### Description

Get the estimated class from MixtComp object

### Usage

```
getPartition(outMixtComp, empiric = TRUE)
```

### Arguments

outMixtComp	object of class <i>MixtCompLearn</i> or <i>MixtComp</i> obtained using <code>mixtCompLearn</code> or <code>mixtCompPredict</code> functions from <code>RMixtComp</code> package or <code>rmcMultiRun</code> from <code>RMixtCompIO</code> package.
empiric	if TRUE, use the partition obtained at the end of the gibbs algorithm. If FALSE, use the partition obtained with the observed probabilities.

### Value

a vector containing the estimated class for each individual.

### Author(s)

Quentin Grimonprez

### See Also

Other getter: [getBIC\(\)](#), [getCompletedData\(\)](#), [getEmpiricTik\(\)](#), [getParam\(\)](#), [getType\(\)](#)

**Examples**

```

require(RMixtureCompIO) # for learning a mixture model
dataLearn <- list(var1 = as.character(c(rnorm(50, -2, 0.8), rnorm(50, 2, 0.8))),
                 var2 = as.character(c(rnorm(50, 2), rpois(50, 8))))

model <- list(var1 = list(type = "Gaussian", paramStr = ""),
             var2 = list(type = "Poisson", paramStr = ""))

algo <- list(
  nClass = 2,
  nInd = 100,
  nbBurnInIter = 100,
  nbIter = 100,
  nbGibbsBurnInIter = 100,
  nbGibbsIter = 100,
  nInitPerClass = 3,
  nSemTry = 20,
  confidenceLevel = 0.95,
  ratioStableCriterion = 0.95,
  nStableCriterion = 10,
  mode = "learn"
)

resLearn <- rmcMultiRun(algo, dataLearn, model, nRun = 3)

# get class
estimatedClass <- getPartition(resLearn)

```

---

 getType

*Names and Types Getters*


---

**Description**

getType returns the type output of a *MixtComp* object, getModel returns the model object, getVarNames returns the name for each variable

**Usage**

```

getType(outMixtComp, with.z_class = FALSE)

getModel(outMixtComp, with.z_class = FALSE)

getVarNames(outMixtComp, with.z_class = FALSE)

```

**Arguments**

outMixtComp      object of class *MixtCompLearn* or *MixtComp* obtained using `mixtCompLearn` or `mixtCompPredict` functions from *RMixtComp* package or `rmcMultiRun` from *RMixtCompIO* package.



with `z_class` if TRUE, the type of `z_class` is returned.

**Value**

a vector containing the type of models, names associated with each individual.

**Author(s)**

Quentin Grimonprez

**See Also**

Other getter: [getBIC\(\)](#), [getCompletedData\(\)](#), [getEmpiricTik\(\)](#), [getParam\(\)](#), [getPartition\(\)](#)

**Examples**

```
require(RMixtureCompIO) # for learning a mixture model
dataLearn <- list(var1 = as.character(c(rnorm(50, -2, 0.8), rnorm(50, 2, 0.8))),
                 var2 = as.character(c(rnorm(50, 2), rpois(50, 8))))

model <- list(var1 = list(type = "Gaussian", paramStr = ""),
             var2 = list(type = "Poisson", paramStr = ""))

algo <- list(
  nClass = 2,
  nInd = 100,
  nbBurnInIter = 100,
  nbIter = 100,
  nbGibbsBurnInIter = 100,
  nbGibbsIter = 100,
  nInitPerClass = 3,
  nSemTry = 20,
  confidenceLevel = 0.95,
  ratioStableCriterion = 0.95,
  nStableCriterion = 10,
  mode = "learn"
)

resLearn <- rmcMultiRun(algo, dataLearn, model, nRun = 3)

# get type
type <- getType(resLearn)

# get model object
model <- getModel(resLearn)

# get variable names
varNames <- getVarNames(resLearn)
```

---

heatmapClass	<i>Heatmap of the similarities between classes about clustering</i>
--------------	---

---

**Description**

Heatmap of the similarities between classes about clustering

**Usage**

```
heatmapClass(output, pkg = c("ggplot2", "plotly"), ...)
```

**Arguments**

output	object returned by <i>mixtCompLearn</i> function from <i>RMixtComp</i> or <i>rmcMultiRun</i> function from <i>RMixtCompIO</i>
pkg	"ggplot2" or "plotly". Package used to plot
...	arguments to be passed to <code>plot_ly</code> . For <code>pkg = "ggplot2"</code> , <code>addValues = TRUE</code> prints similarity values on the heatmap

**Details**

The similarities between classes  $k$  and  $g$  is defined by  $1 - \text{Sigma}(k,g)$

$$\text{Sigma}(k, g)^2 = (1/n) * \sum_{i=1}^n (P(Z_i = k|x_i) - P(Z_i = g|x_i))^2$$

**Author(s)**

Matthieu MARBAC

**See Also**

[computeSimilarityClass](#)

Other plot: [heatmapTikSorted\(\)](#), [heatmapVar\(\)](#), [histMisclassif\(\)](#), [plot.MixtComp\(\)](#), [plotConvergence\(\)](#), [plotDataBoxplot\(\)](#), [plotDataCI\(\)](#), [plotDiscrimClass\(\)](#), [plotDiscrimVar\(\)](#), [plotParamConvergence\(\)](#), [plotProportion\(\)](#)

**Examples**

```
require(RMixtCompIO) # for learning a mixture model
dataLearn <- list(var1 = as.character(c(rnorm(50, -2, 0.8), rnorm(50, 2, 0.8))),
                 var2 = as.character(c(rnorm(50, 2), rpois(50, 8))))

model <- list(var1 = list(type = "Gaussian", paramStr = ""),
              var2 = list(type = "Poisson", paramStr = ""))

algo <- list(
```

```

nClass = 2,
nInd = 100,
nbBurnInIter = 100,
nbIter = 100,
nbGibbsBurnInIter = 100,
nbGibbsIter = 100,
nInitPerClass = 3,
nSemTry = 20,
confidenceLevel = 0.95,
ratioStableCriterion = 0.95,
nStableCriterion = 10,
mode = "learn"
)

resLearn <- rmcMultiRun(algo, dataLearn, model, nRun = 3)

# plot
heatmapClass(resLearn)

```

---

heatmapTikSorted      *Heatmap of the tik =  $P(Z_i=k|x_i)$*

---

### Description

Heatmap of the tik =  $P(Z_i=k|x_i)$

### Usage

```
heatmapTikSorted(output, pkg = c("ggplot2", "plotly"), ...)
```

### Arguments

output	object returned by <i>mixtCompLearn</i> function from <i>RMixtComp</i> or <i>rmcMultiRun</i> function from <i>RMixtCompIO</i>
pkg	"ggplot2" or "plotly". Package used to plot
...	arguments to be passed to plot_ly

### Details

Observation are sorted according to the hard partition then for each component they are sorted by decreasing order of their tik's

### Author(s)

Matthieu MARBAC

**See Also**[getTik](#)

Other plot: [heatmapClass\(\)](#), [heatmapVar\(\)](#), [histMisclassif\(\)](#), [plot.MixtComp\(\)](#), [plotConvergence\(\)](#), [plotDataBoxplot\(\)](#), [plotDataCI\(\)](#), [plotDiscrimClass\(\)](#), [plotDiscrimVar\(\)](#), [plotParamConvergence\(\)](#), [plotProportion\(\)](#)

**Examples**

```
require(RMixtCompIO) # for learning a mixture model
dataLearn <- list(var1 = as.character(c(rnorm(50, -2, 0.8), rnorm(50, 2, 0.8))),
                 var2 = as.character(c(rnorm(50, 2), rpois(50, 8))))

model <- list(var1 = list(type = "Gaussian", paramStr = ""),
             var2 = list(type = "Poisson", paramStr = ""))

algo <- list(
  nClass = 2,
  nInd = 100,
  nbBurnInIter = 100,
  nbIter = 100,
  nbGibbsBurnInIter = 100,
  nbGibbsIter = 100,
  nInitPerClass = 3,
  nSemTry = 20,
  confidenceLevel = 0.95,
  ratioStableCriterion = 0.95,
  nStableCriterion = 10,
  mode = "learn"
)

resLearn <- rmcMultiRun(algo, dataLearn, model, nRun = 3)

# plot
heatmapTikSorted(resLearn)
```

---

heatmapVar

*Heatmap of the similarities between variables about clustering*


---

**Description**

Heatmap of the similarities between variables about clustering

**Usage**

```
heatmapVar(output, pkg = c("ggplot2", "plotly"), ...)
```

**Arguments**

output	object returned by <i>mixtCompLearn</i> function from <i>RMixtComp</i> or <i>rmcMultiRun</i> function from <i>RMixtCompIO</i>
pkg	"ggplot2" or "plotly". Package used to plot
...	arguments to be passed to <i>plot_ly</i> . For <i>pkg = "ggplot2"</i> , <i>addValues = TRUE</i> prints similarity values on the heatmap

**Details**

The similarities between variables *j* and *h* is defined by  $\Delta(j,h)$

$$\Delta(j, h) = 1 - \sqrt{(1/n) * \sum_{i=1}^n \sum_{k=1}^K (P(Z_i = k | x_{ij}) - P(Z_i = k | x_{ih}))^2}$$

**Author(s)**

Matthieu MARBAC

**See Also**

[computeSimilarityVar](#)

Other plot: [heatmapClass\(\)](#), [heatmapTikSorted\(\)](#), [histMisclassif\(\)](#), [plot.MixtComp\(\)](#), [plotConvergence\(\)](#), [plotDataBoxplot\(\)](#), [plotDataCI\(\)](#), [plotDiscrimClass\(\)](#), [plotDiscrimVar\(\)](#), [plotParamConvergence\(\)](#), [plotProportion\(\)](#)

**Examples**

```
require(RMixtCompIO) # for learning a mixture model
dataLearn <- list(var1 = as.character(c(rnorm(50, -2, 0.8), rnorm(50, 2, 0.8))),
                 var2 = as.character(c(rnorm(50, 2), rpois(50, 8))))

model <- list(var1 = list(type = "Gaussian", paramStr = ""),
              var2 = list(type = "Poisson", paramStr = ""))

algo <- list(
  nClass = 2,
  nInd = 100,
  nbBurnInIter = 100,
  nbIter = 100,
  nbGibbsBurnInIter = 100,
  nbGibbsIter = 100,
  nInitPerClass = 3,
  nSemTry = 20,
  confidenceLevel = 0.95,
  ratioStableCriterion = 0.95,
  nStableCriterion = 10,
  mode = "learn"
)
```

```
resLearn <- rmcMultiRun(algo, dataLearn, model, nRun = 3)

# plot
heatmapVar(resLearn)
```

---

histMisclassif      *Histogram of the misclassification probabilities*

---

### Description

Histogram of the misclassification probabilities

### Usage

```
histMisclassif(output, pkg = c("ggplot2", "plotly"), ...)
```

### Arguments

output	object returned by <i>mixtCompLearn</i> function from <i>RMixtComp</i> or <i>rmcMultiRun</i> function from <i>RMixtCompIO</i>
pkg	"ggplot2" or "plotly". Package used to plot
...	arguments to be passed to <i>plot_ly</i>

### Details

Missclassification probability of observation  $i$  is denoted  $\text{err}_i$   $\text{err}_i = 1 - \max_{k=1, \dots, K} P(Z_i=k|x_i)$   
 Histograms of  $\text{err}_i$ 's can be plot for a specific class, all classes or every class

### Author(s)

Matthieu MARBAC

### See Also

Other plot: [heatmapClass\(\)](#), [heatmapTikSorted\(\)](#), [heatmapVar\(\)](#), [plot.MixtComp\(\)](#), [plotConvergence\(\)](#), [plotDataBoxplot\(\)](#), [plotDataCI\(\)](#), [plotDiscrimClass\(\)](#), [plotDiscrimVar\(\)](#), [plotParamConvergence\(\)](#), [plotProportion\(\)](#)

### Examples

```
require(RMixtCompIO) # for learning a mixture model
dataLearn <- list(var1 = as.character(c(rnorm(50, -2, 0.8), rnorm(50, 2, 0.8))),
                 var2 = as.character(c(rnorm(50, 2), rpois(50, 8))))

model <- list(var1 = list(type = "Gaussian", paramStr = ""),
              var2 = list(type = "Poisson", paramStr = ""))
```

```

algo <- list(
  nClass = 2,
  nInd = 100,
  nbBurnInIter = 100,
  nbIter = 100,
  nbGibbsBurnInIter = 100,
  nbGibbsIter = 100,
  nInitPerClass = 3,
  nSemTry = 20,
  confidenceLevel = 0.95,
  ratioStableCriterion = 0.95,
  nStableCriterion = 10,
  mode = "learn"
)

resLearn <- rmcMultiRun(algo, dataLearn, model, nRun = 3)

# plot
histMisclassif(resLearn)

```

---

plot.MixtComp

*Plot of a MixtComp object*


---

## Description

Plot of a *MixtComp* object

## Usage

```

## S3 method for class 'MixtComp'
plot(
  x,
  nVarMaxToPlot = 3,
  pkg = c("ggplot2", "plotly"),
  plotData = c("CI", "Boxplot"),
  ...
)

```

## Arguments

x	<i>MixtComp</i> object
nVarMaxToPlot	number of variables to display
pkg	"ggplot2" or "plotly". Package used to plot
plotData	"CI" or "Boxplot". If "CI", uses <a href="#">plotDataCI</a> function. If "Boxplot", uses <a href="#">plot-DataBoxplot</a>
...	extra parameter for <a href="#">plotDataCI</a>

**Author(s)**

Quentin Grimonprez

**See Also**

`mixtCompLearn` `mixtCompPredict`

Other plot: `heatmapClass()`, `heatmapTikSorted()`, `heatmapVar()`, `histMisclassif()`, `plotConvergence()`, `plotDataBoxplot()`, `plotDataCI()`, `plotDiscrimClass()`, `plotDiscrimVar()`, `plotParamConvergence()`, `plotProportion()`

**Examples**

```
require(RMixtCompIO) # for learning a mixture model
dataLearn <- list(var1 = as.character(c(rnorm(50, -2, 0.8), rnorm(50, 2, 0.8))),
                 var2 = as.character(c(rnorm(50, 2), rpois(50, 8))))

model <- list(var1 = list(type = "Gaussian", paramStr = ""),
             var2 = list(type = "Poisson", paramStr = ""))

algo <- list(
  nClass = 2,
  nInd = 100,
  nbBurnInIter = 100,
  nbIter = 100,
  nbGibbsBurnInIter = 100,
  nbGibbsIter = 100,
  nInitPerClass = 3,
  nSemTry = 20,
  confidenceLevel = 0.95,
  ratioStableCriterion = 0.95,
  nStableCriterion = 10,
  mode = "learn"
)

resLearn <- rmcMultiRun(algo, dataLearn, model, nRun = 3)

plot(resLearn)
```

---

plotConvergence

*Convergence of algorithm*

---

**Description**

Plot the evolution of the completed loglikelihood during the SEM algorithm. The vertical line denotes the end of the burn-in phase.

**Usage**

```
plotConvergence(output, ...)
```



**Arguments**

output            object returned by *mixtCompLearn* function from *RMixtComp* or *rmcMultiRun* function from *RMixtCompIO*

...               graphical parameters

**Details**

This function can be used to check the convergence and choose the parameters `nbBurnInIter` and `nbIter` from `mcStrategy`.

**Author(s)**

Quentin Grimonprez

**See Also**

Other plot: [heatmapClass\(\)](#), [heatmapTikSorted\(\)](#), [heatmapVar\(\)](#), [histMisclassif\(\)](#), [plot.MixtComp\(\)](#), [plotDataBoxplot\(\)](#), [plotDataCI\(\)](#), [plotDiscrimClass\(\)](#), [plotDiscrimVar\(\)](#), [plotParamConvergence\(\)](#), [plotProportion\(\)](#)

**Examples**

```
require(RMixtCompIO) # for learning a mixture model
dataLearn <- list(var1 = as.character(c(rnorm(50, -2, 0.8), rnorm(50, 2, 0.8))),
                 var2 = as.character(c(rnorm(50, 2), rpois(50, 8))))

model <- list(var1 = list(type = "Gaussian", paramStr = ""),
              var2 = list(type = "Poisson", paramStr = ""))

algo <- list(
  nClass = 2,
  nInd = 100,
  nbBurnInIter = 100,
  nbIter = 100,
  nbGibbsBurnInIter = 100,
  nbGibbsIter = 100,
  nInitPerClass = 3,
  nSemTry = 20,
  confidenceLevel = 0.95,
  ratioStableCriterion = 0.95,
  nStableCriterion = 10,
  mode = "learn"
)

resLearn <- rmcMultiRun(algo, dataLearn, model, nRun = 3)

# plot
plotConvergence(resLearn)
```

---

plotDataBoxplot      *Boxplot per class*

---

### Description

Display a boxplot (5)

### Usage

```
plotDataBoxplot(output, var, gr1 = TRUE, pkg = c("ggplot2", "plotly"), ...)
```

### Arguments

output	object returned by <i>mixtCompLearn</i> function from <i>RMixtComp</i> or <i>rmcMultiRun</i> function from <i>RMixtCompIO</i>
var	name of the variable
gr1	if TRUE plot the general distribution of the data
pkg	"ggplot2" or "plotly". Package used to plot
...	other parameters (see <i>Details</i> )

### Details

For functional data, three other parameters are available:

**add.obs** if TRUE, observations are added to the plot. Default = FALSE.

**ylim** ylim of the plot.

**xlim** xlim of the plot.

### Author(s)

Matthieu MARBAC

### See Also

Other plot: [heatmapClass\(\)](#), [heatmapTikSorted\(\)](#), [heatmapVar\(\)](#), [histMisclassif\(\)](#), [plot.MixtComp\(\)](#), [plotConvergence\(\)](#), [plotDataCI\(\)](#), [plotDiscrimClass\(\)](#), [plotDiscrimVar\(\)](#), [plotParamConvergence\(\)](#), [plotProportion\(\)](#)

### Examples

```
require(RMixtCompIO) # for learning a mixture model
dataLearn <- list(var1 = as.character(c(rnorm(50, -2, 0.8), rnorm(50, 2, 0.8))),
                 var2 = as.character(c(rnorm(50, 2), rpois(50, 8))))

model <- list(var1 = list(type = "Gaussian", paramStr = ""),
             var2 = list(type = "Poisson", paramStr = ""))
```

```

algo <- list(
  nClass = 2,
  nInd = 100,
  nbBurnInIter = 100,
  nbIter = 100,
  nbGibbsBurnInIter = 100,
  nbGibbsIter = 100,
  nInitPerClass = 3,
  nSemTry = 20,
  confidenceLevel = 0.95,
  ratioStableCriterion = 0.95,
  nStableCriterion = 10,
  mode = "learn"
)

resLearn <- rmcMultiRun(algo, dataLearn, model, nRun = 3)

# plot
plotDataBoxplot(resLearn, "var1")

```

---

plotDataCI

*Mean and 95%-level confidence intervals per class*


---

## Description

Mean and 95%-level confidence intervals per class

## Usage

```

plotDataCI(
  output,
  var,
  class = 1:output$algo$nClass,
  gr1 = FALSE,
  pkg = c("ggplot2", "plotly"),
  ...
)

```

## Arguments

output	object returned by <i>mixtCompLearn</i> function from <i>RMixtComp</i> or <i>rmcMultiRun</i> function from <i>RMixtCompIO</i>
var	name of the variable
class	class to plot
gr1	if TRUE plot the CI for the dataset and not only classes
pkg	"ggplot2" or "plotly". Package used to plot
...	other parameters (see <i>Details</i> )

## Details

For functional data, three other parameters are available:

**add.obs** if TRUE, observations are added to the plot. Default = FALSE.

**add.CI** if FALSE, confidence intervals are removed from the plot. Default = TRUE.

**xlim** xlim of the plot.

**ylim** ylim of the plot.

## Author(s)

Matthieu MARBAC

## See Also

Other plot: [heatmapClass\(\)](#), [heatmapTikSorted\(\)](#), [heatmapVar\(\)](#), [histMisclassif\(\)](#), [plot.MixtComp\(\)](#), [plotConvergence\(\)](#), [plotDataBoxplot\(\)](#), [plotDiscrimClass\(\)](#), [plotDiscrimVar\(\)](#), [plotParamConvergence\(\)](#), [plotProportion\(\)](#)

## Examples

```
require(RMixtCompIO) # for learning a mixture model
dataLearn <- list(var1 = as.character(c(rnorm(50, -2, 0.8), rnorm(50, 2, 0.8))),
                 var2 = as.character(c(rnorm(50, 2), rpois(50, 8))))

model <- list(var1 = list(type = "Gaussian", paramStr = ""),
             var2 = list(type = "Poisson", paramStr = ""))

algo <- list(
  nClass = 2,
  nInd = 100,
  nbBurnInIter = 100,
  nbIter = 100,
  nbGibbsBurnInIter = 100,
  nbGibbsIter = 100,
  nInitPerClass = 3,
  nSemTry = 20,
  confidenceLevel = 0.95,
  ratioStableCriterion = 0.95,
  nStableCriterion = 10,
  mode = "learn"
)

resLearn <- rmcMultiRun(algo, dataLearn, model, nRun = 3)

# plot
plotDataCI(resLearn, "var1")
```

---

plotDiscrimClass      *Barplot of the discriminative power of the classes*

---

### Description

Barplot of the discriminative power of the classes

### Usage

```
plotDiscrimClass(output, ylim = c(0, 1), pkg = c("ggplot2", "plotly"), ...)
```

### Arguments

output	object returned by <i>mixtCompLearn</i> function from <i>RMixtComp</i> or <i>rmcMultiRun</i> function from <i>RMixtCompIO</i>
ylim	vector of length 2 defining the range of y-axis
pkg	"ggplot2" or "plotly". Package used to plot
...	arguments to be passed to <code>plot_ly</code>

### Details

The discriminative power of class  $k$  is defined by  $1 - D(k)$

$$D(k) = - \sum_{i=1}^n P(Z_i = k|x_i) \log(P(Z_i = k|x_i)) / (n * \exp(-1))$$

### Author(s)

Matthieu MARBAC

### See Also

[computeDiscrimPowerClass](#)

Other plot: [heatmapClass\(\)](#), [heatmapTikSorted\(\)](#), [heatmapVar\(\)](#), [histMisclassif\(\)](#), [plot.MixtComp\(\)](#), [plotConvergence\(\)](#), [plotDataBoxplot\(\)](#), [plotDataCI\(\)](#), [plotDiscrimVar\(\)](#), [plotParamConvergence\(\)](#), [plotProportion\(\)](#)

### Examples

```
require(RMixtCompIO) # for learning a mixture model
dataLearn <- list(var1 = as.character(c(rnorm(50, -2, 0.8), rnorm(50, 2, 0.8))),
                 var2 = as.character(c(rnorm(50, 2), rpois(50, 8))))

model <- list(var1 = list(type = "Gaussian", paramStr = ""),
              var2 = list(type = "Poisson", paramStr = ""))
```

```

algo <- list(
  nClass = 2,
  nInd = 100,
  nbBurnInIter = 100,
  nbIter = 100,
  nbGibbsBurnInIter = 100,
  nbGibbsIter = 100,
  nInitPerClass = 3,
  nSemTry = 20,
  confidenceLevel = 0.95,
  ratioStableCriterion = 0.95,
  nStableCriterion = 10,
  mode = "learn"
)

resLearn <- rmcMultiRun(algo, dataLearn, model, nRun = 3)

plotDiscrimClass(resLearn)

```

---

plotDiscrimVar      *Barplot of the discriminative power of the variables*

---

### Description

Barplot of the discriminative power of the variables

### Usage

```
plotDiscrimVar(output, ylim = c(0, 1), pkg = c("ggplot2", "plotly"), ...)
```

### Arguments

output	object returned by <i>mixtCompLearn</i> function from <i>RMixtComp</i> or <i>rmcMultiRun</i> function from <i>RMixtCompIO</i>
ylim	vector of length 2 defining the range of y-axis
pkg	"ggplot2" or "plotly". Package used to plot
...	arguments to be passed to <code>plot_ly</code>

### Details

The discriminative power of variable  $j$  is defined by  $1 - C(j)$

$$C(j) = - \sum_{k=1}^K \sum_{i=1}^n P(Z_i = k|x_{ij}) \ln(P(Z_i = k|x_{ij})) / (n * \log(K))$$

### Author(s)

Matthieu MARBAC

**See Also**

[computeDiscrimPowerVar](#)

Other plot: [heatmapClass\(\)](#), [heatmapTikSorted\(\)](#), [heatmapVar\(\)](#), [histMisclassif\(\)](#), [plot.MixtComp\(\)](#), [plotConvergence\(\)](#), [plotDataBoxplot\(\)](#), [plotDataCI\(\)](#), [plotDiscrimClass\(\)](#), [plotParamConvergence\(\)](#), [plotProportion\(\)](#)

**Examples**

```
require(RMixtCompIO) # for learning a mixture model
dataLearn <- list(var1 = as.character(c(rnorm(50, -2, 0.8), rnorm(50, 2, 0.8))),
                 var2 = as.character(c(rnorm(50, 2), rpois(50, 8))))

model <- list(var1 = list(type = "Gaussian", paramStr = ""),
              var2 = list(type = "Poisson", paramStr = ""))

algo <- list(
  nClass = 2,
  nInd = 100,
  nbBurnInIter = 100,
  nbIter = 100,
  nbGibbsBurnInIter = 100,
  nbGibbsIter = 100,
  nInitPerClass = 3,
  nSemTry = 20,
  confidenceLevel = 0.95,
  ratioStableCriterion = 0.95,
  nStableCriterion = 10,
  mode = "learn"
)

resLearn <- rmcMultiRun(algo, dataLearn, model, nRun = 3)

# plot
plotDiscrimVar(resLearn)
```

---

plotParamConvergence *Evolution of parameters*

---

**Description**

Plot the evolution of estimated parameters after the burn-in phase.

**Usage**

```
plotParamConvergence(output, var, ...)
```

**Arguments**

output	object returned by <i>mixtCompLearn</i> function from <i>RMixtComp</i> or <i>rmcMultiRun</i> function from <i>RMixtCompIO</i>
var	name of the variable
...	graphical parameters

**Author(s)**

Quentin Grimonprez

**See Also**

Other plot: [heatmapClass\(\)](#), [heatmapTikSorted\(\)](#), [heatmapVar\(\)](#), [histMisclassif\(\)](#), [plot.MixtComp\(\)](#), [plotConvergence\(\)](#), [plotDataBoxplot\(\)](#), [plotDataCI\(\)](#), [plotDiscrimClass\(\)](#), [plotDiscrimVar\(\)](#), [plotProportion\(\)](#)

**Examples**

```
require(RMixtCompIO) # for learning a mixture model
dataLearn <- list(var1 = as.character(c(rnorm(50, -2, 0.8), rnorm(50, 2, 0.8))),
                 var2 = as.character(c(rnorm(50, 2), rpois(50, 8))))

model <- list(var1 = list(type = "Gaussian", paramStr = ""),
             var2 = list(type = "Poisson", paramStr = ""))

algo <- list(
  nClass = 2,
  nInd = 100,
  nbBurnInIter = 100,
  nbIter = 100,
  nbGibbsBurnInIter = 100,
  nbGibbsIter = 100,
  nInitPerClass = 3,
  nSemTry = 20,
  confidenceLevel = 0.95,
  ratioStableCriterion = 0.95,
  nStableCriterion = 10,
  mode = "learn"
)

resLearn <- rmcMultiRun(algo, dataLearn, model, nRun = 3)

# plot
plotParamConvergence(resLearn, "var1")
plotParamConvergence(resLearn, "var2")
```



---

plotProportion	<i>Plot the mixture's proportions</i>
----------------	---------------------------------------

---

### Description

Plot the mixture's proportions

### Usage

```
plotProportion(output, pkg = c("ggplot2", "plotly"), ...)
```

### Arguments

output	object returned by <i>mixtCompLearn</i> function from <i>RMixtComp</i> or <i>rmcMultiRun</i> function from <i>RMixtCompIO</i>
pkg	"ggplot2" or "plotly". Package used to plot
...	arguments to be passed to <code>plot_ly</code>

### Author(s)

Quentin Grimonprez

### See Also

Other plot: [heatmapClass\(\)](#), [heatmapTikSorted\(\)](#), [heatmapVar\(\)](#), [histMisclassif\(\)](#), [plot.MixtComp\(\)](#), [plotConvergence\(\)](#), [plotDataBoxplot\(\)](#), [plotDataCI\(\)](#), [plotDiscrimClass\(\)](#), [plotDiscrimVar\(\)](#), [plotParamConvergence\(\)](#)

### Examples

```
require(RMixtCompIO) # for learning a mixture model
dataLearn <- list(var1 = as.character(c(rnorm(50, -2, 0.8), rnorm(50, 2, 0.8))),
                 var2 = as.character(c(rnorm(50, 2), rpois(50, 8))))

model <- list(var1 = list(type = "Gaussian", paramStr = ""),
             var2 = list(type = "Poisson", paramStr = ""))

algo <- list(
  nClass = 2,
  nInd = 100,
  nbBurnInIter = 100,
  nbIter = 100,
  nbGibbsBurnInIter = 100,
  nbGibbsIter = 100,
  nInitPerClass = 3,
  nSemTry = 20,
  confidenceLevel = 0.95,
  ratioStableCriterion = 0.95,
```

```

    nStableCriterion = 10,
    mode = "learn"
  )

resLearn <- rmcMultiRun(algo, dataLearn, model, nRun = 3)

# plot
plotProportion(resLearn)

```

---

print.MixtComp      *Print Values*

---

### Description

Print a *MixtComp* object

### Usage

```
## S3 method for class 'MixtComp'
print(x, nVarMaxToPrint = 5, ...)
```

### Arguments

*x*                    *MixtComp* object  
*nVarMaxToPrint*    number of variables to display (including *z\_class*)  
 ...                    Not used.

### Author(s)

Quentin Grimonprez

### See Also

mixtCompLearn mixtCompPredict

### Examples

```

require(RMixtCompIO) # for learning a mixture model
dataLearn <- list(var1 = as.character(c(rnorm(50, -2, 0.8), rnorm(50, 2, 0.8))),
                 var2 = as.character(c(rnorm(50, 2), rpois(50, 8))))

model <- list(var1 = list(type = "Gaussian", paramStr = ""),
             var2 = list(type = "Poisson", paramStr = ""))

algo <- list(
  nClass = 2,
  nInd = 100,

```

```
nbBurnInIter = 100,  
nbIter = 100,  
nbGibbsBurnInIter = 100,  
nbGibbsIter = 100,  
nInitPerClass = 3,  
nSemTry = 20,  
confidenceLevel = 0.95,  
ratioStableCriterion = 0.95,  
nStableCriterion = 10,  
mode = "learn"  
)  
  
resLearn <- rmcMultiRun(algo, dataLearn, model, nRun = 3)  
  
print(resLearn)
```

---

refactorCategorical    *Rename a categorical value*

---

### Description

Rename a categorical value

### Usage

```
refactorCategorical(  
  data,  
  oldCateg = unique(data),  
  newCateg = 1:length(oldCateg)  
)
```

### Arguments

data	matrix/data.frame/vector containing the data
oldCateg	vector containing categories to change
newCateg	vector containing new categorical values

### Value

Data with new categorical values

### Author(s)

Quentin Grimonprez

**Examples**

```
dat <- c("single", "married", "married", "divorced", "single")
refactorCategorical(dat, c("single", "married", "divorced"), 1:3)
```

---

summary.MixtComp

*MixtComp Object Summaries*


---

**Description**

Summary of a *MixtComp* object

**Usage**

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

**Arguments**

object	<i>MixtComp</i> object
...	Not used.

**Author(s)**

Quentin Grimonprez

**See Also**

mixtCompLearn [print.MixtComp](#)

**Examples**

```
require(RMixtCompIO) # for learning a mixture model
dataLearn <- list(var1 = as.character(c(rnorm(50, -2, 0.8), rnorm(50, 2, 0.8))),
                 var2 = as.character(c(rnorm(50, 2), rpois(50, 8))))

model <- list(var1 = list(type = "Gaussian", paramStr = ""),
             var2 = list(type = "Poisson", paramStr = ""))

algo <- list(
  nClass = 2,
  nInd = 100,
  nbBurnInIter = 100,
  nbIter = 100,
  nbGibbsBurnInIter = 100,
  nbGibbsIter = 100,
  nInitPerClass = 3,
  nSemTry = 20,
  confidenceLevel = 0.95,
```

```
    ratioStableCriterion = 0.95,  
    nStableCriterion = 10,  
    mode = "learn"  
  )  
  
  resLearn <- rmcMultiRun(algo, dataLearn, model, nRun = 3)  
  
  summary(resLearn)
```

# Index

- \*Topic **package**
  - RMixtCompUtilities-package, 2
- availableModels, 3
- computeDiscrimPowerClass, 3, 29
- computeDiscrimPowerClass
  - (computeDiscrimPowerVar), 4
- computeDiscrimPowerVar, 3, 4, 31
- computeSimilarityClass, 3, 18
- computeSimilarityClass
  - (computeSimilarityVar), 6
- computeSimilarityVar, 3, 6, 21
- convertFunctionalToVector, 7
- createAlgo, 3, 8
- createFunctional, 9
  
- getBIC, 3, 10, 11, 13–15, 17
- getCompletedData, 3, 10, 11, 13–15, 17
- getEmpiricTik, 3, 10, 11, 12, 14, 15, 17
- getICL, 3
- getICL (getBIC), 10
- getModel, 3
- getModel (getType), 16
- getParam, 3, 10, 11, 13, 14, 15, 17
- getPartition, 3, 10, 11, 13, 14, 15, 17
- getTik, 3, 20
- getTik (getEmpiricTik), 12
- getType, 3, 10, 11, 13–15, 16
- getVarNames, 3
- getVarNames (getType), 16
  
- heatmapClass, 3, 6, 18, 20–22, 24–26, 28, 29, 31–33
- heatmapTikSorted, 3, 13, 18, 19, 21, 22, 24–26, 28, 29, 31–33
- heatmapVar, 3, 6, 18, 20, 20, 22, 24–26, 28, 29, 31–33
- histMisclassif, 3, 18, 20, 21, 22, 24–26, 28, 29, 31–33
  
- plot.MixtComp, 3, 18, 20–22, 23, 25, 26, 28, 29, 31–33
- plotConvergence, 3, 18, 20–22, 24, 24, 26, 28, 29, 31–33
- plotDataBoxplot, 3, 14, 18, 20–25, 26, 28, 29, 31–33
- plotDataCI, 3, 14, 18, 20–26, 27, 29, 31–33
- plotDiscrimClass, 3, 5, 18, 20–22, 24–26, 28, 29, 31–33
- plotDiscrimVar, 3, 5, 18, 20–22, 24–26, 28, 29, 30, 32, 33
- plotParamConvergence, 18, 20–22, 24–26, 28, 29, 31, 31, 33
- plotProportion, 3, 18, 20–22, 24–26, 28, 29, 31, 32, 33
- print.MixtComp, 34, 36
  
- refactorCategorical, 35
- RMixtCompUtilities-package, 2
  
- summary.MixtComp, 36