Package ‘RMixtCompUtilities’

November 13, 2020

Type Package
Title Utility Functions for 'MixtComp' Outputs
Version 4.1.4
Date 2020-11-12
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 (>= 4.0.4), Rmixmod, blockcluster

RoxygenNote 7.1.1
Encoding UTF-8
NeedsCompilation no

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

Maintainer Julien Vandaele <julien.vandaele@inria.fr>

Repository CRAN
Date/Publication 2020-11-13 06:30:13 UTC
**Description**

MixtComp (Mixture Composer, [https://github.com/modal-inria/MixtComp](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
availableModels

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/](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

<table>
<thead>
<tr>
<th>Available models</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
</tr>
<tr>
<td>Get information about models implemented in MixtComp</td>
</tr>
<tr>
<td>Usage</td>
</tr>
<tr>
<td>availableModels()</td>
</tr>
<tr>
<td>Value</td>
</tr>
<tr>
<td>a data.frame containing models implemented in MixtComp</td>
</tr>
<tr>
<td>model</td>
</tr>
<tr>
<td>data.type</td>
</tr>
<tr>
<td>format</td>
</tr>
<tr>
<td>missing.formats</td>
</tr>
<tr>
<td>hyperparameter</td>
</tr>
<tr>
<td>comments</td>
</tr>
<tr>
<td>reference</td>
</tr>
</tbody>
</table>
Author(s)
Quentin Grimonprez

See Also
mixtCompLearn

Examples
availableModels()

completeAlgo
Add the missing element to algo parameter Add the missing element to algo parameter with default values

Description
Add the missing element to algo parameter
Add the missing element to algo parameter with default values

Usage
completeAlgo(algo)

Arguments
algo
a list with the different algo parameters for rmc function

Value
algo parameter with all required elements (see createAlgo function)

Author(s)
Quentin Grimonprez
**computeDiscrimPowerVar**

*Discriminative power*

**Description**

Compute the discriminative power of each variable or classe

**Usage**

```r
computeDiscrimPowerVar(outMixtComp, class = NULL)
```

```r
computeDiscrimPowerClass(outMixtComp)
```

**Arguments**

- `outMixtComp` object of class `MixtCompLearn` or `MixtComp` obtained using `mixtCompLearn` or `mixtCompPredict` functions from `RMixtComp` package or `rmcMultiRun` from `RMixtCompIO` package.
- `class` NULL or a number of classes. If NULL, return the discriminative power of variables globally otherwise return the discriminative power of variables in the given class.

**Details**

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

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

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 variable j in class k is defined by 1 - C(j)

\[
C(j) = - \sum_{k=1}^{n} P(Z_i! = k|x_{ij}) \log(P(Z_i! = k|x_{ij}))/(n \times \log(2))
\]

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

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

**Value**

the discriminative power
computeSimilarityVar

Author(s)
Matthieu Marbac

See Also
plotDiscrimClass plotDiscrimVar

Examples

```r
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)

discVar <- computeDiscrimPowerVar(resLearn)
discVarInClass1 <- computeDiscrimPowerVar(resLearn, class = 1)
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)
computeSimilarityVar

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 $\Delta(j,h)$

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

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

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

Value

a similarity matrix

Author(s)

Quentin Grimonprez

See Also

heatmapVar heatmapClass

Examples

```r
require(RMixtCompIO) # for learning a mixture model
dataLearn <- list(var1 = as.character(c(rnorm(50, -2, 0.8), rnorm(50, 2, 0.8), rnorm(50, 0, 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,
   ...)```
convertFunctionalToVector

*Convert a mixtcomp string into a list of 2 vectors*

**Description**
Convert a mixtcomp string into a list of 2 vectors

**Usage**

```
convertFunctionalToVector(x)
```

**Arguments**

- `x` a string containing a functional observation (cf example)

**Value**

a list of 2 vectors: time and value

**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 = 50,
  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.
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  
Create a functional in MixtComp format

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])
**formatData**

*Format the data parameter required by rmc*

**Description**

Format data.frame or matrix in list of character

**Usage**

formatData(data)

**Arguments**

- **data**
  
  data parameter as data.frame, matrix or list

**Value**

data as a list of characters

**Author(s)**

Quentin Grimonprez

---

**formatModel**

*Format the model parameter*

**Description**

Format the model list for rmc/rmcMultiRun functions: - add paramStr when missing - ensure the list format of each element

**Usage**

formatModel(model)

**Arguments**

- **model**
  
  description of model used per variable

**Value**

model as a list where each element is the model applied to a variable (list with elements type and paramStr)

**Author(s)**

Quentin Grimonprez
**getBIC**

Get criterion value

---

### Description

Get criterion value

### Usage

```r
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()`, `getMixtureDensity()`, `getParam()`, `getPartition()`, `getType()`

### Examples

```r
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,
)```
getCompletedData

```r
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**  
*Get the completed data from MixtComp object*

**Description**

Get the completed data from MixtComp object (does not manage functional models)

**Usage**

```r
getCompletedData(outMixtComp, var = NULL, 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.
- `var`: Name of the variables for which to extract the completed data. Default is `NULL` (all variables are extracted)
- `with.z_class`: if `TRUE`, `z_class` 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 getters: `getBIC()`, `getEmpiricTik()`, `getMixtureDensity()`, `getParam()`, `getPartition()`, `getType()`
getEmpiricTik

**Examples**

```r
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**

```r
getEmpiricTik(outMixtComp)

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

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

log  if TRUE, log(tik) 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 individual (in row) and each class (in column).

Author(s)

Quentin Grimonprez

See Also

heatmapTikSorted

Other getter: getBIC(), getCompletedData(), getMixtureDensity(),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)
getMixtureDensity

Get the mixture density

d Description
Get the mixture density for each individual

Usage
getMixtureDensity(outMixtComp)

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

Details
\[ d(x_i) = \sum_k \pi_k P(x_i; \theta_k) \]

Value
a vector containing the mixture density for each individual.

Author(s)
Quentin Grimonprez

See Also
Other getter: getBIC(), getCompletedData(), getEmpiricTik(), getParam(), getPartition(), getType()
getParam

Examples

```r
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)
d <- getMixtureDensity(resLearn)
```

getParam

Get the estimated parameter

Description

Get the estimated parameter

Usage

```r
getParam(outMixtComp, var)

getProportion(outMixtComp)
```

Arguments

- `outMixtComp` object of class `MixtCompLearn` or `MixtComp` obtained using `mixtCompLearn` or `mixtCompPredict` functions from `RMixtComp` package or `rmcMultiRun` from `RMixtCompIO` package.
- `var` name of the variable to get parameter
getPartition

Value

the parameter of the variable

Author(s)

Quentin Grimonprez

See Also

plotDataBoxplot plotDataCI

Other getter: getBIC(), getCompletedData(), getEmpiricTik(), getMixtureDensity(), 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")
prop <- getProportion(resLearn)

getPartition

Get the estimated class from MixtComp object

Description

Get the estimated class from MixtComp object
Usage

getPartition(outMixtComp, empiric = FALSE)

Arguments

outMixtComp object of class `MixtCompLearn` or `MixtComp` obtained using `mixtCompLearn` or `mixtCompPredict` functions from `RMixtComp` package or `rmcMultiRun` from `RMixtCompIO` 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()`, `getMixtureDensity()`, `getParam()`, `getType()`

Examples

```r
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 class
```
getType

estimatedClass <- getPartition(resLearn)

---

**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**

```r
getype(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()`, `getMixtureDensity()`, `getParam()`, `getPartition()`

**Examples**

```r
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 = ""))
```
heatmapClass

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

Heatmap of the similarities between classes about clustering

Description

Heatmap of the similarities between classes about clustering

Usage

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

Arguments

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

pkg "ggplot2" or "plotly". Package used to plot

... arguments to be passed to plot_ly. For pkg = "ggplot2", addValues = TRUE prints similarity values on the heatmap
Details

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

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

Author(s)

Matthieu MARBAC

See Also

computeSimilarityClass

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

Examples

```r
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)
```
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 `mixtCompLearn` function from `RMixtComp` or `rmcMultiRun` function from `RMixtCompIO`
- **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

```r
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,
            ...)
```
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 mixtCompLearn function from RMixtComp or rmcMultiRun function from RMixtCompIO
pkg "ggplot2" or "plotly". Package used to plot
... arguments to be passed to plot_ly. For pkg = "ggplot2", addValues = TRUE
prints similarity values on the heatmap

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

\[
Delta(j, h) = 1 - \sqrt{\frac{1}{n} \sum_{i=1}^{n} \sum_{k=1}^{K} \left( P(Z_i = k|x_{ij}) - P(Z_i = k|x_{ih}) \right)^2}
\]

Author(s)
Matthieu MARBAC
histMisclassif

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, 0.8))));

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 `mixtCompLearn` function from `RMixtComp` or `rmcMultiRun` function from `RMixtCompIO`

pkg: "ggplot2" or "plotly". Package used to plot arguments to be passed to `plot_ly`

Details

Missclassification probability of observation i is denoted err_i err_i = 1 - \max_{k=1,\ldots,K} P(Z_i=k|x_i)

Histograms of err_i's can be plotted 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

```r
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

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

Arguments

- `x`: `MixtComp` object
- `nVarMaxToPlot`: number of variables to display
- `pkg`: "ggplot2" or "plotly". Package used to plot
- `plotData`: "CI" or "Boxplot". If "CI", uses `plotDataCI` function. If "Boxplot", uses `plotDataBoxplot`
- `...`: extra parameter for `plotDataCI`

Author(s)

Quentin Grimonprez

See Also

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

Examples

```r
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
### Examples

```r
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)
```

---

### Description

Display a boxplot (5

### Usage

```r
plotDataBoxplot(
    output,
    var,
    class = 1:output$algo$nClass,
    gr1 = TRUE,
    ...)
```
pkg = c("ggplot2", "plotly"),
...)

Arguments

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

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,
  grl = FALSE,
  pkg = c("ggplot2", "plotly"),
  ...
)

**Arguments**

- **output**: object returned by `mixtCompLearn` function from `RMixtComp` or `rmcMultiRun` function from `RMixtCompIO`
- **var**: name of the variable
- **class**: class to plot
- **grl**: if TRUE plot the CI for the dataset and not only classes
- **pkg**: "ggplot2" or "plotly". Package used to plot
- **...**: other parameters (see `Details`
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

```r
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 `mixtCompLearn` function from `RMixtComp` or `rmcMultiRun` function from `RMixtCompIO`
- **ylim**: vector of length 2 defining the range of y-axis
- **pkg**: "ggplot2" or "plotly". Package used to plot
- **...**: arguments to be passed to `plot_ly`

Details

The discriminative power of class \( k \) is defined by

\[
D(k) = - \sum_{i=1}^{n} \frac{P(Z_i = k|x_i) \log(P(Z_i = k|x_i))}{n \times \exp(-1)}
\]

Author(s)

Matthieu MARBAC

See Also

- `computeDiscrimPowerClass`
- Other plot: `heatmapClass()`, `heatmapTikSorted()`, `heatmapVar()`, `histMisclassif()`, `plot.MixtComp()`, `plotConvergence()`, `plotDataBoxplot()`, `plotDataCI()`, `plotDiscrimVar()`, `plotParamConvergence()`, `plotProportion()`

Examples

```r
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, class = NULL, ylim = c(0, 1), pkg = c("ggplot2", "plotly"), ...)`

**Arguments**  
- **output**: object returned by `mixtCompLearn` function from `RMixtComp` or `rmcMultiRun` function from `RMixtCompIO`
- **class**: NULL or a number of classes. If NULL, return the discriminative power of variables globally otherwise return the discriminative power of variables in the given class
- **ylim**: vector of length 2 defining the range of y-axis
- **pkg**: "ggplot2" or "plotly". Package used to plot
- **...**: arguments to be passed to `plot_ly`
Details

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

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

Author(s)

Matthieu MARBAC

See Also

`computeDiscrimPowerVar`

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

Examples

```r
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)

plotDiscrimVar(resLearn, class = 1)
```
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 mixtCompLearn function from RMixtComp or rmcMultiRun function from RMixtCompIO
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,
```r
nStableCriterion = 10,
  mode = "learn"
)

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

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

---

**plotProportion**

*Plot the mixture's proportions*

**Description**

Plot the mixture's proportions

**Usage**

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

**Arguments**

- `output`: object returned by `mixtCompLearn` function from `RMixtComp` or `rmcMultiRun` function from `RMixtCompIO`
- `pkg`: "ggplot2" or "plotly". Package used to plot
- `...`: arguments to be passed to `plot_ly`

**Author(s)**

Quentin Grimonprez

**See Also**

Other plot: `heatmapClass()`, `heatmapTikSorted()`, `heatmapsVar()`, `histMisclassif()`, `plot.MixtComp()`, `plotConvergence()`, `plotDataBoxplot()`, `plotDataCI()`, `plotDiscrimClass()`, `plotDiscrimVar()`, `plotParamConvergence()`

**Examples**

```r
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

**Description**

Print a `MixtComp` object

**Usage**

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

**Arguments**

- `x` *MixtComp* object
- `nVarMaxToPrint` number of variables to display (including z_class)
- `...` parameter of head function

**Author(s)**

Quentin Grimonprez

**See Also**

`mixtCompLearn`, `mixtCompPredict`
## Examples

```r
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,
nBurnIter = 100,
nIter = 100,
nBurnGibbsIter = 100,
nGibbsIter = 100,
nInitPerClass = 3,
SemTry = 20,
ConfidenceLevel = 0.95,
RatioStableCriterion = 0.95,
StableCriterion = 10,
mode = "learn"
)

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

print(resLearn)
```

---

### refactorCategorical

**Rename a categorical value**

#### Description

Rename a categorical value.

#### Usage

```r
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)

Description
Summary of a `MixtComp` object

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

Arguments
- `object`: `MixtComp` object
- `...`: Not used.

Author(s)
Quentin Grimonprez

See Also
- `mixtCompLearn`
- `print.MixtComp`

Examples
```r
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(
```
```r
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

* getter
  getBIC, 12
  getCompletedData, 13
  getEmpiricTik, 14
  getMixtureDensity, 16
  getParam, 17
  getPartition, 18
  getType, 20
* package
  RMixtCompUtilities-package, 2
* plot
  heatmapClass, 21
  heatmapTikSorted, 23
  heatmapVar, 24
  histMisclassif, 25
  plot.MixtComp, 27
  plotConvergence, 28
  plotDataBoxplot, 29
  plotDataCI, 31
  plotDiscrimClass, 33
  plotDiscrimVar, 34
  plotParamConvergence, 36
  plotProportion, 37

availableModels, 3

completeAlgo, 4
computeDiscrimPowerClass, 3, 33
computeDiscrimPowerClass
  (computeDiscrimPowerVar), 5
computeDiscrimPowerVar, 3, 5, 35
computeSimilarityClass, 3, 22
computeSimilarityClass
  (computeSimilarityVar), 6
computeSimilarityVar, 3, 6, 25
convertFunctionalToVector, 8
createAlgo, 3, 4, 9
createFunctional, 10

formatData, 11

formatModel, 11

getBIC, 3, 12, 13, 15, 16, 18–20
getCompletedData, 3, 12, 13, 15, 16, 18–20
getEmpiricTik, 3, 12, 13, 14, 16, 18–20
getICL, 3
getICL (getBIC), 12
getMixtureDensity, 12, 13, 15, 16, 18–20
getModel, 3
getModel (getType), 20
getParam, 3, 12, 13, 15, 16, 17, 19, 20
getPartition, 3, 12, 13, 15, 16, 18, 19, 20
getProportion (getParam), 17
getTik, 3, 23
getTik (getEmpiricTik), 14
getType, 3, 12, 13, 15, 16, 18, 19, 20
getVarNames, 3
getVarNames (getType), 20

heatmapClass, 3, 7, 21, 23, 25–27, 29, 30, 32, 33, 35–37
heatmapTikSorted, 3, 15, 22, 23, 25–27, 29, 30, 32, 33, 35–37
heatmapVar, 3, 7, 22, 23, 24, 26, 27, 29, 30, 32, 33, 35–37
histMisclassif, 3, 22, 23, 25, 27, 29, 30, 32, 33, 35–37

plot.MixtComp, 3, 22, 23, 25, 26, 27, 29, 30, 32, 33, 35–37
plotConvergence, 3, 22, 23, 25–27, 28, 30, 32, 33, 35–37
plotDataBoxplot, 3, 18, 22, 23, 25–27, 29, 29, 32, 33, 35–37
plotDataCI, 3, 18, 22, 23, 25–27, 29, 30, 31, 33, 35–37
plotDiscrimClass, 3, 6, 22, 23, 25–27, 29, 30, 32, 33, 35–37
plotDiscrimVar, 3, 6, 22, 23, 25–27, 29, 30, 32, 33, 34, 36, 37
plotParamConvergence, 22, 23, 25–27, 29, 30, 32, 33, 35, 36, 37
plotProportion, 3, 22, 23, 25–27, 29, 30, 32, 33, 35, 36, 37
print.MixtComp, 38, 40
refactorCategorical, 39
RMixtCompUtilities-package, 2

summary.MixtComp, 40