Package ‘RMixtCompIO’

June 18, 2020

Type Package

Title Mixture Models with Heterogeneous and (Partially) Missing Data

Version 4.0.5

Date 2020-06-16

Copyright Inria - Université de Lille - CNRS; Patrick Wieschollek, Tobias Wood & the respective contributors for CppOptimizationLibrary

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. It includes models for real, categorical, counting, functional and ranking data. This package contains the minimal R interface of the C++ 'MixtComp' library.

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

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

Imports Rcpp, doParallel, foreach

Suggests Rmixmod, blockcluster, testthat, RInside, xml2

LinkingTo Rcpp, RcppEigen, BH

RoxygenNote 7.1.0

Encoding UTF-8

NeedsCompilation yes

Author Vincent Kubicki [aut],
Christophe Biernacki [aut],
Quentin Grimonprez [aut, cre],
Serge Iovleff [ctb],
Matthieu Marbac-Lourdelle [ctb],
Étienne Goffinet [ctb],
Patrick Wieschollek [ctb] (for CppOptimizationLibrary),
Tobias Wood [ctb] (for CppOptimizationLibrary)

Maintainer Quentin Grimonprez <quentin.grimonprez@inria.fr>

Repository CRAN

Date/Publication 2020-06-18 11:10:17 UTC

R topics documented:

RMixtCompIO-package *RMixtCompIO*

Description

MixtComp (Mixture Composer) 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. This package contains the minimal R interface of the C++ library.

Online SaaS version: https://massiccc.lille.inria.fr/

Details

The main function is rmcMultiRun that runs a SEM algorithm to learn a mixture model.

See Also

rmcMultiRun. Other clustering packages: Rmixmod, blockcluster

Examples

```
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))))

dataPredict <- list(var1 = as.character(c(rnorm(10, -2, 0.8), rnorm(10, 2, 0.8))),
                    var2 = as.character(c(rnorm(10, 2), rpois(10, 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"
)


# run RMixtComp in unsupervised clustering mode + data as matrix
resLearn <- rmcMultiRun(algo, dataLearn, model, nRun = 3)


# run RMixtComp in predict mode + data as list
algo$nInd = 20
algo$mode = "predict"
resPredict <- rmcMultiRun(algo, dataPredict, model, resLearn)
```

rmcMultiRun *Learn and predict a Mixture Model*

Description

Estimate the parameter of a mixture model or predict the cluster of new samples. It manages heterogeneous data as well as missing and incomplete data.

Usage

```
rmcMultiRun(
  algo,
  data,
  model,
  resLearn = list(),
  nRun = 1,
  nCore = 1,
  verbose = FALSE
)
```

Arguments

algo	a list containing the parameters of the SEM-Gibbs algorithm (see *Details*).

`data` a data.frame, a matrix or a named list containing the data (see *Details Data format* sections).

`model` a named list containing models and hyperparameters (see *Details* section).

`resLearn` output of *rmcMultiRun* (only for predict mode).

`nRun` number of runs for every given number of class. If >1, SEM is run `nRun` times for every number of class, and the best according to observed likelihood is kept.

`nCore` number of cores used for the parallelization of the *nRun* runs.

`verbose` if TRUE, print some informations.

Details

The *data* object is a list where each element correponds to a variable, each element must be named. Missing and incomplete data are managed, see section *Data format* for how to format them.

The *model* object is a named list containing the variables to use in the model. All variables listed in the *model* object must be in the *data* object. *model* can contain less variables than *data*. An element of the list corresponds to a model which is described by a list of 2 elements: type containing the model name and paramStr containing the hyperparameters. For example: `model <-list(real1 = list(type = "Gaussian",paramStr = ""),func1 = list(type = "Func_CS",paramStr = "nSub: 4,nCoeff: 2"))`.

Eight models are available in RMixtComp: *Gaussian*, *Multinomial*, *Poisson*, *NegativeBinomial*, *Weibull*, *Func_CS*, *Func_SharedAlpha_CS*, *Rank_ISR*. *Func_CS* and *Func_SharedAlpha_CS* models require hyperparameters: the number of subregressions of functional and the number of coefficients of each subregression. These hyperparameters are specified by: *nSub: i, nCoeff: k* in the *paramStr* field of the *model* object. The *Func_SharedAlpha_CS* is a variant of the *Func_CS* model with the alpha parameter shared between clusters. It means that the start and end of each subregression will be the same across the clusters.

To perform a (semi-)supervised clustering, user can add a variable named *z_class* in the data and model objects with *LatentClass* as model in the model object.

The *algo* object is a list containing the different number of iterations for the algorithm. The algorithm is decomposed in a burn-in phase and a normal phase. Estimates from the burn-in phase are not shown in output.

- nClass: number of class
- nInd: number of individuals
- 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

An object of class MixtComp

Data format

- Gaussian data: Gaussian data are real values with the dot as decimal separator. Missing data are indicated by a *?*. Partial data can be provided through intervals denoted by *[a:b]* where *a* (resp. *b*) is a real or *-inf* (resp. *+inf*).

- Categorical Data: Categorical data must be consecutive integer with 1 as minimal value. Missing data are indicated by a *?*. For partial data, a list of possible values can be provided by *a_1,...,a_j*, where *a_i* denotes a categorical value.

- Poisson and NegativeBinomial Data: Poisson and NegativeBinomial data must be positive integer. Missing data are indicated by a *?*. Partial data can be provided through intervals denoted by *[a:b]* where *a* and *b* are positive integers. *b* can be *+inf*.

- Weibull Data: Weibull data are real positive values with the dot as decimal separator. Missing data are indicated by a *?*. Partial data can be provided through intervals denoted by *[a:b]* where *a* and *b* are positive reals. *b* can be *+inf*.

- Rank data: The format of a rank is: *o_1, ..., o_j* where o_1 is an integer corresponding to the number of the object ranked in 1st position. For example: 4,2,1,3 means that the fourth object is ranked first then the second object is in second position and so on. Missing data can be specified by replacing and object by a *?* or a list of potential object, for example: *4, {2 3}, {2 1}, ?* means that the object ranked in second position is either the object number 2 or the object number 3, then the object ranked in third position is either the object 2 or 1 and the last one can be anything. A totally missing rank is spedified by *?,?,...,?*

- Functional data: The format of a fonctional data is: *time_1:value_1,..., time_j:value_j*. Between individuals, functional data can have different length and different time. *i* is the number of subregressions in a functional data and *k* the number of coefficients of each regression (2 = linear, 3 = quadratic, ...). Missing data are not supported.

- z_class: To perform a (semi-)supervised clustering, user can add a variable named ‘z_class‘ (with eventually some missing values) with "LatentClass" as model. Missing data are indicated by a *?*. For partial data, a list of possible values can be provided by *a_1,...,a_j*, where *a_i* denotes a class number.

MixtComp object

A MixtComp object is a result of a single run of MixtComp algorithm. It is a list containing three elements *mixture*, *variable* and *algo*. If MixtComp fails to run, the list contains a single element: warnLog containing error messages.

The *mixture* element contains

- BIC: value of BIC
- ICL: value of ICL
- nbFreeParameters: number of free parameters of the mixture
- lnObservedLikelihood: observed loglikelihood
- lnCompletedLikelihood: completed loglikelihood

- IDClass: entropy used to compute the discriminative power of variable: -$\sum_{i=1}^{n} t_{ikj} log(t_{ikj})/(n * log(K))$
- IDClassBar: entropy used to compute the discriminative power of variable: -$\sum_{i=1}^{n}(1 - t_{ikj}) log((1 - t_{ikj}))/(n * log(K))$
- delta: similarities between variables
- completedProbabilityLogBurnIn: evolution of the completed log-probability during the burn-in period (can be used to check the convergence and determine the ideal number of iteration)
- completedProbabilityLogRun: evolution of the completed log-probability after the burn-in period (can be used to check the convergence and determine the ideal number of iteration)
- runTime: list containing the total execution time in seconds and the execution time of some subpart.
- lnProbaGivenClass: log-proportion + log-probability of x_i for each class

The *algo* list contains a copy of *algo* parameter with extra elements: nInd, nClass, mode ("learn" or "predict").

The *variable* list contains 3 lists : *data*, *type* and *param*. Each of these lists contains a list for each variable (the name of each list is the name of the variable) and for the class of samples (*z_class*). The *type* list contains the model used for each variable.

Each list of the *data* list contains the completed data in the *completed* element and some statistics about them (*stat*).

The estimated parameter can be found in the *stat* element in the *param* list (see Section *View of an output object*). For more details about the parameters of each model, you can refer to rnorm, rpois, rweibull, rnbinom, rmultinom, or references in the *References* section.

View of a MixtComp object

Example of output object with variables named "categorical", "gaussian", "rank", "functional", "poisson", "nBinom" and "weibull" with respectively *Multinomial*, *Gaussian*, *Rank_ISR*, *Func_CS* (or *Func_SharedAlpha_CS*), *Poisson*, *NegativeBinomial* and *Weibull* as model.

```
output
|_______    algo       __ nbBurnInIter
|                      |_ nbIter
|                      |_ nbGibbsBurnInIter
|                      |_ nbGibbsIter
|                      |_ nInitPerClass
|                      |_ nSemTry
|                      |_ ratioStableCriterion
|                      |_ nStableCriterion
|                      |_ confidenceLevel
|                      |_ mode
|                      |_ nInd
|                      |_ nClass
|
|_______    mixture    __ BIC
|                      |_ ICL
```

```
        |                    |_ lnCompletedLikelihood
        |                    |_ lnObservedLikelihood
        |                    |_ IDClass
        |                    |_ IDClassBar
        |                    |_ delta
        |                    |_ runTime
        |                    |_ nbFreeParameters
        |                    |_ completedProbabilityLogBurnIn
        |                    |_ completedProbabilityLogRun
        |                    |_ lnProbaGivenClass


|
|_______    variable  __ type    __ z_class
                      |         |_ categorical
                      |         |_ gaussian
                      |         |_ ...
                      |
                      |_ data   __ z_class     __ completed
                      |         |              |_ stat
                      |         |_ categorical __ completed
                      |         |              |_ stat
                      |         |_ ...
                      |         |_ functional  __ data
                      |                        |_ time
                      |
                      |_ param  __ z_class     __ stat
                                |              |_ log
                                |              |_ paramStr
                                |_ functional  __ alpha       __ stat
                                |              |              |_ log
                                |              |_ beta        __ stat
                                |              |              |_ log
                                |              |_ sd          __ stat
                                |              |              |_ log
                                |              |_ paramStr
                                |_ rank        __ mu          __ stat
                                |              |              |_ log
                                |              |_ pi          __ stat
                                |              |              |_ log
                                |              |_ paramStr
                                |
                                |_ gaussian    __ stat
                                |              |_ log
                                |              |_ paramStr
                                |_ poisson     __ stat
                                |              |_ log
                                |              |_ paramStr
```

|_ ...

Author(s)

Quentin Grimonprez

Examples

```
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))))

dataPredict <- list(var1 = as.character(c(rnorm(10, -2, 0.8), rnorm(10, 2, 0.8))),
                    var2 = as.character(c(rnorm(10, 2), rpois(10, 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"
)


# run RMixtComp in unsupervised clustering mode + data as matrix
resLearn <- rmcMultiRun(algo, dataLearn, model, nRun = 3)


# run RMixtComp in predict mode + data as list
algo$nInd = 20
algo$mode = "predict"
resPredict <- rmcMultiRun(algo, dataPredict, model, resLearn)
```

Index