Package 'GREMLINS'

January 20, 2025

Type Package

Title Generalized Multipartite Networks

Version 0.2.1

Description We define generalized multipartite networks as the joint observation of several networks implying some common pre-specified groups of individuals. The aim is to fit an adapted version of the popular stochastic block model to multipartite networks, as described in Bar-hen, Barbillon and Donnet (2020) <arXiv:1807.10138>.

URL `https://GrossSBM.github.io/GREMLINS/`

BugReports `https://github.com/GrossSBM/GREMLINS/issues`

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

Suggests testthat (>= 2.1.0), spelling, knitr, rmarkdown

Language en-US

Imports R6, parallel, stats, utils, igraph, blockmodels, aricode, pbmcapply

Collate 'GREMLINS_package.R' 'BMPOO.R' 'defineNetwork.R' 'multipartiteBM.R' 'VEMPOOVersion.R' 'multipartiteBMFixedModel.R' 'initializeVEM.R' 'searchVKPOOVersion.R' 'utils.R' 'compLikICL.R' 'extractClustersMBM.R' 'rMBM.R' 'comparClassif.R' 'MPEcoNetwork.R' 'predictMBM.R'

VignetteBuilder knitr

Depends R (>= 3.5.0)

NeedsCompilation no

Author Sophie Donnet [aut, cre] (<`https://orcid.org/0000-0003-4370-7316`>), Pierre Barbillon [aut] (<`https://orcid.org/0000-0002-7766-7693`>)

Maintainer Sophie Donnet <`sophie.donnet@inrae.fr`>

Repository CRAN

Date/Publication 2023-03-10 18:20:02 UTC

Contents

`comparClassif` *Compare two classifications on all the Functional groups*

Description

Compare two classifications on all the Functional groups

Usage

```
comparClassif(classif1, classif2)
```

Arguments

`classif1`	: list a length n_FG.
`classif2`	: list a length n_FG.

Value

Adjusted Rand Index (ARI) for each Functional Group.

Examples

```
nFG <- 3;
vK <- c(4,5,2) ;
vNQ <- c(100,40,50);
classif1 <- lapply(1:nFG,function(q){sample(1:vK[q],vNQ[q],replace=TRUE)})
classif2 <- classif1
classif2[[2]] <-  sample(1:vK[2],vNQ[2],replace=TRUE)
resCompar <- comparClassif (classif1,classif2)
```

compLikICL — *compute the Integrated likeilhood and the ICL criteria for the MBM*

Description

compute the Integrated likeilhood and the ICL criteria for the MBM

Usage

```
compLikICL(paramEstim, list_Net, v_distrib = NULL)
```

Arguments

`paramEstim`	Estimated parameters of MBM
`list_Net`	A list of network
`v_distrib`	Type of proababilistic distributions in each network : if 0/1 then Bernoulli, if counting then Poisson. My default = Bernoulli. Must give a vector whose length is the number of networks in list_Net

Value

Pseudo-Likelihood and penalty

defineNetwork — *Define a network providing its matrix of interactions and specifying the functions groups in row and col.*

Description

Define a network providing its matrix of interactions and specifying the functions groups in row and col.

Usage

```
defineNetwork(mat, typeInter, rowFG, colFG)
```

Arguments

`mat`	An adjacency matrix (symmetric or not) or an incidence matrix
`typeInter`	Type of the matrix, choice between "inc" (incidence), "adj" (adjacency) and "diradj" (directed adjacency)
`rowFG`	Name of the functional group in row
`colFG`	Name of the function group in column

Value

a list object formatted for the GREMLINS package

Examples

```
A <- matrix(rbinom(100,1,.2),10,10)
type <- "diradj"
defineNetwork(A,"diradj","FG1","FG1")
```

`extractClustersMBM` *Extract the clusters in each functional group*

Description

Extract the clusters in each functional group

Usage

```
extractClustersMBM(resMBM, whichModel = 1)
```

Arguments

`resMBM`	A fitted Generalized BlockModel
`whichModel`	The index corresponding to the model to plot (default is 1, the best model)

Value

a list a length the number of Functional Groups. Each element is a list of length the number of blocks composed of the index of the individuals in each block of each cluster.

`GREMLINS` *Adjusting an extended SBM to Multipartite networks*

Description

Generalized multipartite networks consist in the joint observation of several networks implying some common pre-specified groups of individuals. GREMLIM adjusts an adapted version of the popular stochastic block model to multipartite networks, as described in Bar-hen, Barbillon and Donnet (2020) The GREMLINS package provides the following top-level major functions:

- `defineNetwork` a function to define carefully a single network.
- `rMBM` a function to simulate a collection of networks involving common functional groups of entities (with various emission distributions).
- `multipartiteBM` a function to perform inference (model selection and estimation) of SBM for a multipartite network.
- `multipartiteBMFixedModel` a function to estimate the parameters of SBM for a multipartite network for fixed numbers of blocks

Details

We also provide some additional functions useful to analyze the results:

- `extractClustersMBM` a function to extract the clusters in each functional group
- `comparClassif` a function to compute the Adjusted Rand Index (ARI) between two classifications
- `predictMBM` a function to compute the predictions once the model has been fitted
- `compLikICL` a function to compute the Integrated Likelihood and the ICL criteria for the MBM

Author(s)

Pierre Barbillon, Sophie Donnet

References

Bar-Hen, A. and Barbillon, P. & Donnet S. (2020), "Block models for multipartite networks. Applications in ecology and ethnobiology. Journal of Statistical Modelling (to appear)

`MPEcoNetwork` *Multipartite network of mutualistic interactions between plants and pollinators, plants and birds and plants and ants.*

Description

Multipartite network of mutualistic interactions between plants and pollinators, plants and birds and plants and ants.

Usage

```
MPEcoNetwork
```

Format

A list a 3 binary incidence matrices

Inc_plant_ant Interactions between plants (rows) and ants (cols). Matrix with 141 rows and 30 columns

Inc_plant_bird Interactions between plants (rows) and birds (cols). Matrix with141 rows and 46 columns

Inc_plant_flovis Interactions between plants (rows) and pollinators (cols). Matrix with 141 rows and 173 columns ...

Source

Dataset compiled and conducted at Centro de Investigaciones Costeras La Mancha (CICOLMA), located on the central coast of the Gulf of Mexico, Veracruz, Mexico. https://royalsocietypublishing.org/doi/full/10.1098/rspb.2016.1564 https://github.com/lucaspdmedeiros/multi-network_core_removal/tree/master/data

multipartiteBM *Model selection and parameter estimation of MBM*

Description

Select the number of blocks and identify the blocks per functional group using a variational EM algorithm

Usage

```
multipartiteBM(
  list_Net,
  v_distrib = NULL,
  namesFG = NULL,
  v_Kmin = 1,
  v_Kmax = 10,
  v_Kinit = NULL,
  initBM = TRUE,
  keep = FALSE,
  verbose = TRUE,
  nbCores = NULL,
  maxiterVE = NULL,
  maxiterVEM = NULL
)
```

Arguments

`list_Net`	a list of networks (defined via the function defineNetwork) i.e. a multipartite network
`v_distrib`	an optional vector of characters of length the number of networks and specifying the distribution used in each network (possible values `bernoulli,poisson,gaussian,laplace`). If not provided, the model will be 'bernoulli' for all the interactions matrices.
`namesFG`	an optional vector of characters containing the names of functional groups (FG) (If Specified, must correspond to the names in `list_Net`).
`v_Kmin`	an optional vector of integers, specifying the minimal number of blocks per functional group (must be provided in the same order as in `namesFG`). v_Kmin may be a single value (same minimal number of blocks for all the FGs) or a vector with size equal to the number of FGs. Default value = `1`.
`v_Kmax`	an optional vector of integers specifying the maximal number of blocks per functional group provided in the same order as in `namesFG`. v_Kmax may be a single value (same maximal number of blocks for all the FGs) or a vector with size equal to the number of FGs. Default value = `10`.
`v_Kinit`	an optional vector of integers specifying initial numbers of blocks per FG provided in the same order as in `namesFG`. if `v_Kinit` is not specified, then `v_Kinit = v_Kmin`

`initBM`	an optional boolean. If initBM = TRUE an aditional initialisation is done using simple LBM or SBM on each network separatly. Default value = `TRUE`
`keep`	an optional boolean. If TRUE return the estimated parameters for intermediate visited models. Otherwise, only the better model (in ICL sense) is the ouput. Default value = `FALSE`.
`verbose`	an optional boolean. If TRUE, display the current step of the search algorithm
`nbCores`	an optional integer specifying the number or cores used for the estimation. Not parallelized on windows. If `ncores = NULL`, then half of the cores are used.
`maxiterVE`	an optional integer specifying the maximum number of iterations in the VE step of the VEM algorithm. If NULL then default value = `1000`
`maxiterVEM`	an optional integer specifying the maximum number of iterations of the VEM algorithm. If NULL then default value Default value = `1000`

Details

The function `multipartiteBM` selects the better numbers of blocks in each FG (with a penalized likelihood criterion). The model selection is performed with a forward backward strategy and the likelihood of each model is maximized with a variational EM).

Value

a list of estimated parameters for the different models ordered by decreasing ICL. If keep = `FALSE`, the length is of length 1 (only the better model is returned).

`fittedModel` contains the results of the inference. `res$fittedModel[[1]]` is a list with fields

- `paramEstim` a MBMfit object.
- `ICL` the penalized likelihood criterion ICL.
- `vJ` the sequence of the varational bound of the likelihood through iterations of the VEM.
- `convergence` TRUE if the VEM reached convergence.

`list_Net` contains the data.

Examples

```
namesFG <- c('A','B')
list_pi <- list(c(0.5,0.5),c(0.3,0.7)) # prop of blocks in each FG
E  <-  rbind(c(1,2),c(2,2)) # architecture of the multipartite net.
typeInter <- c( "inc","diradj")
v_distrib <- c('gaussian','bernoulli')
list_theta <- list()
list_theta[[1]] <- list()
list_theta[[1]]$mean  <- matrix(c(6.1, 8.9, 6.6, 3), 2, 2)
list_theta[[1]]$var  <-  matrix(c(1.6, 1.6, 1.8, 1.5),2, 2)
list_theta[[2]] <- matrix(c(0.7,1.0, 0.4, 0.6),2, 2)
list_Net <- rMBM(v_NQ = c(30,30),E , typeInter, v_distrib, list_pi,
                list_theta, namesFG = namesFG, seed = 2)$list_Net
res_MBMsimu <- multipartiteBM(list_Net, v_distrib,
                              namesFG = c('A','B'), v_Kinit = c(2,2),
                              nbCores = 2,initBM = FALSE)
```

multipartiteBMFixedModel

Model selection and estimation of multipartite blockmodels

Description

Estimate the parameters and give the clustering for given numbers of blocks

Usage

```
multipartiteBMFixedModel(
  list_Net,
  v_distrib,
  namesFG,
  v_K,
  classifInit = NULL,
  nbCores = NULL,
  maxiterVE = NULL,
  maxiterVEM = NULL,
  verbose = TRUE
)
```

Arguments

`list_Net`	A list of network (defined via the function DefineNetwork)
`v_distrib`	Type of proababilistic distributions in each network : if 0/1 then bernoulli, if counting then poisson, gaussian or Zero Inflated Gaussian (ZIgaussian) My default = Bernoulli. Must give a vector whose length is the number of networks in list_Net
`namesFG`	Names of functional groups (must correspond to names in listNet)
`v_K`	A vector with the numbers of blocks per functional group
`classifInit`	A list of initial classification for each functional group in the same order as in namesFG
`nbCores`	Number or cores used for the estimation. Not parallelized on windows. By default : half of the cores
`maxiterVE`	Maximum number of iterations in the VE step of the VEM algorithm. Default value = 1000
`maxiterVEM`	Maximum number of iterations of the VEM algorithm. Default value = 1000
`verbose`	Set to TRUE to display the current step of the search algorithm

Value

Estimated parameters and a classification

Examples

```
namesFG <- c('A','B')
list_pi <- list(c(0.5,0.5),c(0.3,0.7)) # prop of blocks in each FG
E  <-  rbind(c(1,2),c(2,2)) # architecture of the multipartite net.
typeInter <- c( "inc","diradj")
v_distrib <- c('poisson','bernoulli')
list_theta <- list()
list_theta[[1]]   <- matrix(c(6.1, 8.9, 6.6, 3), 2, 2)
list_theta[[2]] <- matrix(c(0.7,1.0, 0.4, 0.6),2, 2)
list_Net <- rMBM(v_NQ = c(20,20),E , typeInter, v_distrib, list_pi,
                list_theta, namesFG = namesFG, seed = 2)$list_Net
#res_MBMsimu_fixed <- multipartiteBMFixedModel(list_Net, v_distrib,
#                                                namesFG = namesFG,
#                                                v_K = c(1,2),
#                                                nbCores = 2)
```

---

predictMBM *Predict NAs in a Collection of Networks from a fitted MBM*

Description

Predict NAs in a Collection of Networks from a fitted MBM

Usage

```
predictMBM(RESMBM, whichModel = 1)
```

Arguments

RESMBM	a fitted multipartite blockmodel
whichModel	The index corresponding to the model used for prediction (default is 1, the best model)

Value

the collection of matrices of predictions (probability for binary, intensity for weighted network) a

Examples

```
namesFG <- c('A','B')
list_pi <- list(c(0.5,0.5),c(0.3,0.7)) # prop of blocks in each FG
E  <-  rbind(c(1,2),c(2,2)) # architecture of the multipartite net.
typeInter <- c( "inc","diradj")
v_distrib <- c('gaussian','bernoulli')
list_theta <- list()
list_theta[[1]] <- list()
list_theta[[1]]$mean  <- matrix(c(6.1, 8.9, 6.6, 3), 2, 2)
list_theta[[1]]$var  <-  matrix(c(1.6, 1.6, 1.8, 1.5),2, 2)
list_theta[[2]] <- matrix(c(0.7,1.0, 0.4, 0.6),2, 2)
```

```
list_Net <- rMBM(v_NQ = c(30,30),E , typeInter, v_distrib, list_pi,
                 list_theta, namesFG = namesFG, seed = 2)$list_Net
res_MBMsimu <- multipartiteBM(list_Net, v_distrib,
                              namesFG = c('A','B'), v_Kinit = c(2,2),
                              nbCores = 2,initBM = FALSE)
pred <- predictMBM(res_MBMsimu)
```

rMBM *Simulate datasets from the multipartite block model (MBM).*

Description

rMBM simulates a collection of networks involving common functional groups of entities. The networks may be directed, undirected or bipartite. The emission distribution of the edges may be Bernoulli, Poisson, Gaussian, Zero-Inflated Gaussian, or Laplace. See the vignette for more information about the model.

Usage

```
rMBM(
  v_NQ,
  E,
  typeInter,
  v_distrib,
  list_pi,
  list_theta,
  namesFG = NULL,
  keepClassif = FALSE,
  seed = NULL
)
```

Arguments

v_NQ	: number of individual in each Functional Group (FG)
E	: define the architecture of the Multipartite.
typeInter	: type of interaction in each network: undirected adjacency (adj), directed adjacency (diradj) or incidence (inc). (vector of size equal to nrow(E))
v_distrib	: vector of the distributions: 'bernoulli', 'poisson', 'gaussian', 'ZIgaussian' (for Zero inflated gaussian) or 'laplace' (vector of size equal to nrow(E))
list_pi	: parameters of the blocks distribution
list_theta	: parameters of the interactions distribution. For Bernoulli a probability, for Poisson positive real number, for Gaussian a list specifying mean and var (plus p0 for ZIgaussian), for Laplace a list with location and scale
namesFG	: names of the FG. (default value = NULL, then the functional groups are labelled FG1, FG2, etc)
keepClassif	: equal to TRUE if you want to keep the simulated blocks/classification (default value = FALSE).
seed	: set the seed for the random simulation (default value = NULL)

Value

A list of lists containing the networks (list_net) and if keepClassif = TRUE the classifications (classif) Each element of list_net corresponds to a network : each network is a list containing the matrix (mat) , the type of network(diradj, adj, inc), the functional group in row (rowFG) and the functional group in columns (colFG)

Examples

```
namesFG <- c('A','B','C')
list_pi = list(c(0.16 ,0.40 ,0.44),c(0.3,0.7),c(0.5,0.5))
E  <-  rbind(c(1,2),c(2,3),c(1,1))
typeInter <- c( "inc","inc", "adj")
v_distrib <- c('ZIgaussian','bernoulli','poisson')
list_theta <- list()
list_theta[[1]] <- list()
list_theta[[1]]$mean  <- matrix(c(6.1, 8.9, 6.6, 9.8, 2.6, 1.0), 3, 2)
list_theta[[1]]$var  <-  matrix(c(1.6, 1.6, 1.8, 1.7 ,2.3, 1.5),3, 2)
list_theta[[1]]$p0  <-  matrix(c(0.4, 0.1, 0.6, 0.5 , 0.2, 0),3, 2)
list_theta[[2]] <- matrix(c(0.7,1.0, 0.4, 0.6),2, 2)
m3 <- matrix(c(2.5, 2.6 ,2.2 ,2.2, 2.7 ,3.0 ,3.6, 3.5, 3.3),3,3 )
list_theta[[3]] <- (m3 + t(m3))/2
dataSim <- rMBM(v_NQ = c(100,50,40) , E = E , typeInter = typeInter,
                v_distrib = v_distrib, list_pi = list_pi,
                list_theta = list_theta, namesFG)
list_net <- dataSim$list_Net
classifTrue <- dataSim$classif
```

Index