Package ‘sbm’

June 9, 2021

Title Stochastic Blockmodels
Version 0.4.3
URL https://grosssbm.github.io/sbm/
BugReports https://github.com/GrossSBM/sbm/issues
License GPL (>= 3)
Encoding UTF-8
LazyData true
RoxygenNote 7.1.1
Suggests testthat, spelling, knitr, rmarkdown, aricode, covr
Language en-US
Imports alluvial, magrittr, dplyr, purrr, blockmodels, R6, Rcpp, igraph, ggplot2, GREMLINS, stringr, rlang, reshape2, prodlim
Collate 'R6Class-SBM.R' 'R6Class-BipartiteSBM.R'
 'R6Class-BipartiteSBM_fit.R' 'R6Class-MultipartiteSBM.R'
 'R6Class-MultipartiteSBM_fit.R' 'R6Class-MultiplexSBM_fit.R'
 'R6Class-SimpleSBM.R' 'R6Class-SimpleSBM_fit.R' 'RcppExports.R'
 'defineSBM.R' 'estimate.R' 'fungusTreeNetwork.R'
 'multipartiteEcologicalNetwork.R' 'plotAlluvial.R'
 'plotMyMatrix.R' 'plotMyMultipartiteMatrix.R' 'plotMyMultiplexMatrix.R'
 'plotPipe.R' 'utils.R' 'utils_plot.R' 'war.R'
LinkingTo Rcpp, RcppArmadillo
VignetteBuilder knitr
Depends R (>= 3.5.0)
NeedsCompilation yes

Author Julien Chiquet [aut, cre] (<https://orcid.org/0000-0002-3629-3429>),
Sophie Donnet [aut] (<https://orcid.org/0000-0003-4370-7316>),
großBM team [ctb].
Pierre Barbillon [aut] (<https://orcid.org/0000-0002-7766-7693>)

Maintainer Julien Chiquet <julien.chiquet@inrae.fr>

Repository CRAN

Date/Publication 2021-06-09 17:20:02 UTC

R topics documented:

- BipartiteSBM ........................................ 3
- BipartiteSBM_fit .................................... 5
- coef.SBM ............................................ 7
- defineSBM .......................................... 8
- estimateBipartiteSBM ................................. 9
- estimateMultipartiteSBM .............................. 11
- estimateMultiplexSBM ................................. 12
- estimateSimpleSBM .................................. 15
- fitted.SBM .......................................... 17
- fungusTreeNetwork .................................. 18
- is_SBM .............................................. 18
- multipartiteEcologicalNetwork ...................... 19
- MultipartiteSBM .................................... 19
- MultipartiteSBM_fit ................................ 21
- MultiplexSBM_fit .......................... 23
- plot.SBM ........................................... 25
- plotAlluvial ......................................... 27
- plotMyMatrix ....................................... 28
- plotMyMultipartiteMatrix ......................... 29
- plotMyMultiplexMatrix ............................. 30
- predict.SBM ........................................ 32
- sampleBipartiteSBM ................................ 32
- sampleMultipartiteSBM ............................... 34
- sampleMultiplexSBM ................................ 36
- sampleSimpleSBM ................................... 38
- SBM ............................................... 40
- SimpleSBM .......................................... 42
- SimpleSBM_fit ..................................... 45
- war ................................................. 47

Index 48
Description

R6 class for Bipartite SBM
R6 class for Bipartite SBM

Super class

`sbm::SBM` -> `BipartiteSBM`

Active bindings

- `dimLabels`: vector of two characters giving the label of each connected dimension (row, col)
- `blockProp`: list of two vectors of block proportions (aka prior probabilities of each block)
- `connectParam`: parameters associated to the connectivity of the SBM, e.g. matrix of inter/inter
  block probabilities when model is Bernoulli
- `probMemberships`: matrix of estimated probabilities for block memberships for all nodes
- `nbBlocks`: vector of size 2: number of blocks (rows, columns)
- `nbDyads`: number of dyads (potential edges in the network)
- `nbConnectParam`: number of parameter used for the connectivity
- `memberships`: list of size 2: vector of memberships in row, in column.
- `indMemberships`: matrix for clustering memberships

Methods

Public methods:

- `BipartiteSBM$new()`
- `BipartiteSBM$rMemberships()`
- `BipartiteSBM$rEdges()`
- `BipartiteSBM$predict()`
- `BipartiteSBM$show()`
- `BipartiteSBM$plot()`
- `BipartiteSBM$clone()`

Method `new()`: constructor for SBM

Usage:

```r
BipartiteSBM$new(model, nbNodes, blockProp, connectParam,)
```
```
BipartiteSBM

dimLabels = c(row = "row", col = "col"),
covarParam = numeric(length(covarList)),
covarList = list()
)

Arguments:
model character describing the type of model
nbNodes number of nodes in each dimension of the network
blockProp parameters for block proportions (vector of list of vectors)
connectParam list of parameters for connectivity with a matrix of means 'mean' and an optional scalar for the variance 'var'. The dimensions of mu must match blockProp lengths
dimLabels optional labels of each dimension (in row, in column)
covarParam optional vector of covariates effect
covarList optional list of covariates data

Method rMemberships(): a method to sample new block memberships for the current SBM
Usage:
BipartiteSBM$rMemberships(store = FALSE)

Arguments:
store should the sampled blocks be stored (and overwrite the existing data)? Default to FALSE
Returns: the sampled blocks

Method rEdges(): a method to sample a network data (edges) for the current SBM
Usage:
BipartiteSBM$rEdges(store = FALSE)

Arguments:
store should the sampled edges be stored (and overwrite the existing data)? Default to FALSE
Returns: the sampled network

Method predict(): prediction under the current parameters
Usage:
BipartiteSBM$predict(covarList = self$covarList, theta_p0 = 0)

Arguments:
covarList a list of covariates. By default, we use the covariates with which the model was estimated.
theta_p0 double for thresholding...

Method show(): show method
Usage:
BipartiteSBM$show(type = "Bipartite Stochastic Block Model")

Arguments:
type character used to specify the type of SBM

Method plot(): basic matrix plot method for BipartiteSBM object or mesoscopic plot
Usage:
BipartiteSBM$plot(
  type = c("data", "expected", "meso"),
  ordered = TRUE,
  plotOptions = list()
)

Arguments:
type character for the type of plot: either 'data' (true connection), 'expected' (fitted connection) or 'meso' (mesoscopic view). Default to 'data'.
ordered logical: should the rows and columns be reordered according to the clustering? Default to TRUE.
plotOptions list with the parameters for the plot. See help of the corresponding S3 method for details.

Returns: a ggplot2 object for the 'data' and 'expected', a list with the igraph object g, the layout and the plotOptions for the 'meso'

Method clone(): The objects of this class are cloneable with this method.

Usage:
BipartiteSBM$clone(deep = FALSE)

Arguments:
deep Whether to make a deep clone.

BipartiteSBM_fit  R6 Class definition of an Bipartite SBM fit

Description
R6 Class definition of an Bipartite SBM fit
R6 Class definition of an Bipartite SBM fit

Details
This class is designed to give a representation and adjust an LBM fitted with blockmodels.
The list of parameters estimOptions essentially tunes the optimization process and the variational EM algorithm, with the following parameters:

- "nbCores" integer for number of cores used. Default is 2
- "verbosity" integer for verbosity (0, 1). Default is 1
- "plot" boolean, should the ICL by dynamically plotted or not. Default is TRUE
- "exploreFactor" control the exploration of the number of groups
- "exploreMin" explore at least until exploreMin even if the exploration factor rule is achieved. Default 4. See the package blockmodels for details.
- "exploreMin" stop exploration at exploreMax even if the exploration factor rule is not achieved. Default Inf. See the package blockmodels for details.
- "fast" logical: should approximation be used for Bernoulli model with covariates. Default to TRUE
Super classes

  sbm::SBM -> sbm::BipartiteSBM -> BipartiteSBM_fit

Active bindings

  loglik  double: approximation of the log-likelihood (variational lower bound) reached
  ICL    double: value of the integrated classification log-likelihood
  penalty double, value of the penalty term in ICL
  entropy double, value of the entropy due to the clustering distribution
  storedModels data.frame of all models fitted (and stored) during the optimization

Methods

  Public methods:

    • BipartiteSBM_fit$new()
    • BipartiteSBM_fit$optimize()
    • BipartiteSBM_fit$setModel()
    • BipartiteSBM_fit$reorder()
    • BipartiteSBM_fit$show()
    • BipartiteSBM_fit$clone()

Method new(): constructor for a Bipartite SBM fit

  Usage:
  BipartiteSBM_fit$new(incidenceMatrix, model,
                       dimLabels = c(row = "rowName", col = "colName"),
                       covarList = list())

  Arguments:
  incidenceMatrix rectangular (weighted) matrix
  model character('bernoulli', 'poisson', 'gaussian')
  dimLabels labels of each dimension (in row, in columns)
  covarList and optional list of covariates, each of whom must have the same dimension as incidenceMatrix

Method optimize(): function to perform optimization

  Usage:
  BipartiteSBM_fit$optimize(estimOptions = list())

  Arguments:
  estimOptions a list of parameters controlling the inference algorithm and model selection.
              See details.

Method setModel(): method to select a specific model among the ones fitted during the optimization. Fields of the current SBM_fit will be updated accordingly.
Usage:
BipartiteSBM_fit$setModel(index)

Arguments:
index  integer, the index of the model to be selected (row number in storedModels)

Method reorder(): permute group labels by order of decreasing probability

Usage:
BipartiteSBM_fit$reorder()

Method show(): show method

Usage:
BipartiteSBM_fit$show(type = "Fit of a Bipartite Stochastic Block Model")

Arguments:
type  character used to specify the type of SBM

Method clone(): The objects of this class are cloneable with this method.

Usage:
BipartiteSBM_fit$clone(deep = FALSE)

Arguments:
deep  Whether to make a deep clone.

---

**coef.SBM**

*Extract model coefficients*

---

**Description**

Extracts model coefficients from objects with class `SBM` and children (`SimpleSBM_fit, BipartiteSBM_fit`)

**Usage**

```r
## S3 method for class 'SBM'
coef(object, type = c("connectivity", "block", "covariates"), ...)
```

**Arguments**

- **object** an R6 object inheriting from class `SBM_fit` (like `SimpleSBM_fit` or `BipartiteSBM_fit`)
- **type** type of parameter that should be extracted. Either 'block' for $\pi$
  - 'connectivity' for $\theta$
  - or "covariates" for $\beta$
  - Default is 'connectivity'.
- **...** additional parameters for S3 compatibility. Not used
Define a network

Usage

```r
defineSBM(
  netMat,
  model = "bernoulli",
  type = ifelse(ncol(netMat) == nrow(netMat), "simple", "bipartite"),
  directed = !isSymmetric(netMat),
  dimLabels = c(row = "row", col = "col"),
  covariates = list()
)
```

Arguments

- `netMat` a matrix describing the network: either an adjacency (square) or incidence matrix with possibly weighted entries.
- `model` character describing the model for the relation between nodes ("bernoulli", "poisson", "gaussian", ...). Default is "bernoulli".
- `type` Type of the matrix, choice between 'simple' and 'bipartite'
- `directed` logical: is the network directed or not? Only relevant when `type` is 'Simple'. Default is TRUE if `netMat` is symmetric, FALSE otherwise
- `dimLabels` an optional vector of labels for each dimension (in row, in column). Default value = c('row' = row,'col'= col)
- `covariates` a list of matrices with same dimension as `mat` describing covariates at the edge level. No covariate per Default.

Value

an object SimpleSBM or BipartiteSBM with the informations required to define a future multipartite network

Examples

```r
A <- matrix(rbinom(100,1,.2), 10, 10)
myNet <- defineSBM(A, "poisson", "simple", TRUE, "Actor")
```
**Description**

This function performs variational inference of bipartite Stochastic Block Models, with various model for the distribution of the edges: Bernoulli, Poisson, or Gaussian models.

**Usage**

```r
estimateBipartiteSBM(
  netMat,
  model = "bernoulli",
  dimLabels = c(row = "row", col = "col"),
  covariates = list(),
  estimOptions = list()
)
```

**Arguments**

- `netMat`: a matrix describing the network: either an adjacency (square) or incidence matrix with possibly weighted entries.
- `model`: character describing the model for the relation between nodes ("bernoulli", "poisson", "gaussian", ...). Default is "bernoulli".
- `dimLabels`: an optional vector of labels for each dimension (in row, in column)
- `covariates`: a list of matrices with same dimension as mat describing covariates at the edge level. No covariate per Default.
- `estimOptions`: a list of parameters controlling the inference algorithm and model selection. See details.

**Details**

The list of parameters `estimOptions` essentially tunes the optimization process and the variational EM algorithm, with the following parameters:

- "nbCores" integer for number of cores used. Default is 2
- "verbosity" integer for verbosity (0, 1). Default is 1
- "plot" boolean, should the ICL by dynamically plotted or not. Default is TRUE
- "exploreFactor" control the exploration of the number of groups
- "exploreMax" explore at least until exploreMin even if the exploration factor rule is achieved. Default 4. See the package blockmodels for details.
- "exploreMin" stop exploration at exploreMax even if the exploration factor rule is not achieved. Default Inf. See the package blockmodels for details.
- "fast" logical: should approximation be used for Bernoulli model with covariates. Default to TRUE
Value

a list with the estimated parameters. See details...

Examples

### BIPARTITE BINARY SBM (Bernoulli model)

```r
## Graph parameters and Sampling
nbNodes <- c(60, 80)
blockProp <- list(c(.5, .5), c(1/3, 1/3, 1/3)) # group proportions
means <- matrix(runif(6), 2, 3) # connectivity matrix
# In Bernoulli SBM, parameters is a list with a
# matrix of means 'mean' which are probabilities of connection
connectParam <- list(mean = means)
mySampler <- sampleBipartiteSBM(nbNodes, blockProp, connectParam, model = 'bernoulli')
```

```r
## Estimation
myBipartiteSBM <- estimateBipartiteSBM(mySampler$networkData, estimOptions = list(plot = FALSE))
plot(myBipartiteSBM, 'expected')
```

### BIPARTITE POISSON SBM

```r
## Graph parameters & Sampling
nbNodes <- c(60, 80)
blockProp <- list(c(.5, .5), c(1/3, 1/3, 1/3)) # group proportions
means <- matrix(rbinom(6, 30, 0.25), 2, 3) # connectivity matrix
connectParam <- list(mean = means)
mySampler <- sampleBipartiteSBM(nbNodes, blockProp, connectParam, model = 'poisson')
```

```r
## Estimation
myBipartiteSBM <-
  estimateBipartiteSBM(mySampler$networkData, 'poisson', estimOptions = list(plot = FALSE))
plot(myBipartiteSBM, 'expected')
```

### BIPARTITE GAUSSIAN SBM

```r
## Graph parameters & sampling
nbNodes <- c(60, 80)
blockProp <- list(c(.5, .5), c(1/3, 1/3, 1/3)) # group proportions
means <- 20 * matrix(runif(6), 2, 3) # connectivity matrix
connectParam <- list(mean = means, var = 1)
mySampler <- sampleBipartiteSBM(nbNodes, blockProp, connectParam, model = 'gaussian')
```

```r
## Estimation
myBipartiteSBM <-
  estimateBipartiteSBM(mySampler$networkData, 'gaussian', estimOptions = list(plot = FALSE))
plot(myBipartiteSBM, 'expected')
```
estimateMultipartiteSBM

*Estimation for multipartite SBM*

**Description**

Estimation for multipartite SBM

**Usage**

```r
estimateMultipartiteSBM(listSBM, estimOptions = list())
```

**Arguments**

- `listSBM`: list of networks that were defined by the `defineSBM` function
- `estimOptions`: options for the inference procedure

**Details**

The list of parameters `estimOptions` essentially tunes the optimization process and the variational EM algorithm, with the following parameters:

- "nbCores" integer for number of cores used. Default is 2
- "verbosity" integer for verbosity (0, 1). Default is 1
- "nbBlocksRange" List of length the number of functional groups, each element supplying the minimal and maximal number of blocks to be explored. The names of the list must be the names of the functional groups. Default value is from 1 to 10
- "initBM" Boolean. True if using simple and bipartite SBM as initialisations. Default value = TRUE
- "maxiterVEM" Number of max. number of iterations in the VEM. Default value = 100
- "maxiterVE" Number of max. number of iterations in the VE. Default value = 100

**Value**

a `MultipartiteSBM_fit` object with the estimated parameters and the blocks in each Functional Group

**Examples**

```r
## Not run:
## About the Parts/Functional Groups (FG)
blockProp <- list(c(0.16, 0.40, 0.44), c(0.3, 0.7)) # prop of blocks in each FG
archiMultipartite <- rbind(c(1, 2), c(2, 2), c(1, 1)) # architecture of the multipartite net.

## About the connection matrices
directed <- c(NA, TRUE, FALSE) # type of each network
model <- c("gaussian", "bernoulli", "poisson")
C1 <-
```
library(M grape)

# Simulate a multiplex SBM with three layers

## Parameter Generation

C1 <- list(mean = matrix(c(6.1, 8.9, 6.6, 9.8, 2.6, 1.0), 3, 2),
            var = matrix(c(1.6, 1.6, 1.8, 1.7, 2.3, 1.5),3, 2))
C2 <- list(mean = 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)
C3 <- list(mean = .5 * (m3 + t(m3)))
connectParam <- list(C1, C2, C3)

## Graph Sampling

mySampleMSBM <- sampleMultipartiteSBM(nbNodes, blockProp,
                                       archiMultipartite, connectParam, model,
                                       directed, dimLabels = c('A', 'B'), seed = 2)

listSBM <- mySampleMSBM$listSBM
estimOptions <- list(initBM = FALSE, nbCores = 2)
myMSBM <- estimateMultipartiteSBM(listSBM, estimOptions)
plot(myMSBM, type = "data")
plot(myMSBM, type = "expected")
plot(myMSBM, type = "meso")

## End(Not run)
estimateMultiplexSBM

- "initBM"Boolean. True if using simple and bipartite SBM as initialisations. Default value = TRUE
- "maxiterVEM"Number of max. number of iterations in the VEM. Default value = 100
- "maxiterVE"Number of max. number of iterations in the VE. Default value = 100
- "plot"boolean, should the ICL by dynamically plotted or not. Default is TRUE. For dependent networks
- "exploreFactor"control the exploration of the number of groups. For dependent networks
- "exploreMin"explore at least until exploreMin even if the exploration factor rule is achieved. Default 4. See the package blockmodels for details. For dependent networks
- "exploreMax"Stop exploration at exploreMax even if the exploration factor rule is not achieved. Default Inf. See the package blockmodels for details. For dependent networks
- "nbBlocksRange"minimal and maximal number or blocks explored. For dependent networks
- "fast"logical: should approximation be used for Bernoulli model with covariates. Default to TRUE. For dependent networks

Value

a MultiplexSBM_fit object with the estimated parameters and the blocks

Examples

```r
## Not run:
### =======================================
### MULTIPLEX SBM without dependence between layers
###
Nnodes <- 40
dockProp <- c(.4,.6)
dbLayers <- 2
dockParam <- list(list(mean=matrix(rbeta(4,.5,.5),2,2)),list(mean=matrix(rexp(4,.5),2,2)))
dock <- c("bernoulli","poisson")
docktype <- "directed"
dockSampleMultiplexSBM <-
sampleMultiplexSBM(
  nbNodes = Nnodes,
  blockProp = blockProp,
  dbLayers = dbLayers,
  dockParam = dockParam,
  dock = dock,
  type = type)
docklist <- dockSampleMultiplexSBM$docklist
dockestsOptions <- list(initBM = FALSE, nbCores = 2)
dockMultiplexSBM <- estimateMultiplexSBM(docklist,dockestsOptions,dependent=FALSE)
### =======================================
### MULTIPLEX SBM Gaussian with dependence
###
Q <- 3
dbLayers <- 2
dockParam <- list()
dockParam$mu <- vector("list",dbLayers)
```
connectParam$mu[[1]] <- matrix(.1,Q,Q) + diag(1:Q)
connectParam$mu[[2]] <- matrix(-2,Q,Q) + diag(rev(Q:1))
connectParam$Sigma <- matrix(c(2,1,1,4),nbLayers,nbLayers)
model <- rep("gaussian",2)
type <- "directed"
Nnodes <- 80
blockProp <- c(.3,.3,.4)
mySampleMultiplexSBM <- sampleMultiplexSBM(
    nbNodes = Nnodes,
    blockProp = blockProp,
    nbLayers = nbLayers,
    connectParam = connectParam,
    model=model,
    type="undirected",
    dependent=TRUE)
listSBM <- mySampleMultiplexSBM$listSBM
myMultiplexSBM <- estimateMultiplexSBM(listSBM,estimOptions,dependent=TRUE)
## MultiplexSBM Bernoulli with dependence
Q <- 2
P00<-matrix(runif(Q*Q),Q,Q)
P10<-matrix(runif(Q*Q),Q,Q)
P01<-matrix(runif(Q*Q),Q,Q)
P11<-matrix(runif(Q*Q),Q,Q)
SumP<-P00+P10+P01+P11
P00<-P00/SumP
P01<-P01/SumP
P10<-P10/SumP
P11<-P11/SumP
connectParam <- list()
connectParam$prob00 <- P00
connectParam$prob01 <- P01
connectParam$prob10 <- P10
connectParam$prob11 <- P11
model <- rep("bernoulli",2)
type <- "directed"
NbLayers <- 2
Nnodes <- 40
blockProp <- c(.6,.4)
mySampleMultiplexSBM <- sampleMultiplexSBM(
    nbNodes = Nnodes,
    blockProp = blockProp,
    nbLayers = nbLayers,
    connectParam = connectParam,
    model=model,
    type=type,
    dependent=TRUE)
listSBM <- mySampleMultiplexSBM$listSBM
myMultiplexSBM <- estimateMultiplexSBM(listSBM,estimOptions,dependent=TRUE)
## End(Not run)
Description

This function performs variational inference of simple Stochastic Block Models, with various model for the distribution of the edges: Bernoulli, Poisson, or Gaussian models.

Usage

```r
estimateSimpleSBM(
  netMat,  # a matrix describing the network: either an adjacency (square) or incidence matrix with possibly weighted entries.
  model = "bernoulli",  # character describing the model for the relation between nodes ('bernoulli', 'poisson', 'gaussian', ...). Default is 'bernoulli'.
  directed = !isSymmetric(netMat),  # logical: is the network directed or not? Only relevant when type is 'Simple'. Default is TRUE if netMat is symmetric, FALSE otherwise.
  dimLabels = c(node = "nodeName"),  # an optional label for referring to the nodes
  covariates = list(),  # a list of matrices with same dimension as mat describing covariates at the edge level. No covariate per default.
  estimOptions = list()  # a list of parameters controlling the inference algorithm and model selection. See details.
)
```

Arguments

- **netMat**: a matrix describing the network: either an adjacency (square) or incidence matrix with possibly weighted entries.
- **model**: character describing the model for the relation between nodes ('bernoulli', 'poisson', 'gaussian', ...). Default is 'bernoulli'.
- **directed**: logical: is the network directed or not? Only relevant when type is 'Simple'. Default is TRUE if netMat is symmetric, FALSE otherwise.
- **dimLabels**: an optional label for referring to the nodes.
- **covariates**: a list of matrices with same dimension as mat describing covariates at the edge level. No covariate per default.
- **estimOptions**: a list of parameters controlling the inference algorithm and model selection. See details.

Details

The list of parameters `estimOptions` essentially tunes the optimization process and the variational EM algorithm, with the following parameters:

- "nbCores"integer for number of cores used. Default is 2
- "verbosity"integer for verbosity (0, 1). Default is 1
- "plot"boolean, should the ICL by dynamically plotted or not. Default is TRUE
- "exploreFactor"control the exploration of the number of groups
- "exploreMin"explore at least until exploreMin even if the exploration factor rule is achieved. Default 4. See the package blockmodels for details.
• "exploreMax"  Stop exploration at exploreMax even if the exploration factor rule is not achieved. Default Inf. See the package blockmodels for details.
• "nbBlocksRange"  minimal and maximal number or blocks explored
• "fast"  logical: should approximation be used for Bernoulli model with covariates. Default to TRUE

Value

a list with the estimated parameters. See details...

Examples

### SIMPLE BINARY SBM (Bernoulli model)

#### Graph parameters & Sampling

```r
nbNodes <- 90
blockProp <- c(.5, .25, .25) # group proportions
means <- diag(.4, 3) + 0.05 # connectivity matrix: affiliation network
connectParam <- list(mean = means)
mySampler <- sampleSimpleSBM(nbNodes, blockProp, connectParam)
adjacencyMatrix <- mySampler$networkData
```

#### Estimation

```r
mySimpleSBM <- estimateSimpleSBM(adjacencyMatrix, 'bernoulli', estimOptions = list(plot = FALSE))
plot(mySimpleSBM, 'data', ordered = FALSE)
plot(mySimpleSBM, 'data')
plot(mySimpleSBM, 'expected', ordered = FALSE)
plot(mySimpleSBM, 'expected')
plot(mySimpleSBM, 'meso')
```

### SIMPLE POISSON SBM

#### Graph parameters & Sampling

```r
nbNodes <- 90
blockProp <- c(.5, .25, .25) # group proportions
means <- diag(15., 3) + 5 # connectivity matrix: affiliation network
connectParam <- list(mean = means)
mySampler <- sampleSimpleSBM(nbNodes, blockProp, list(mean = means), model = "poisson")
adjacencyMatrix <- mySampler$networkData
```

#### Estimation

```r
mySimpleSBM <- estimateSimpleSBM(adjacencyMatrix, 'poisson', estimOptions = list(plot = FALSE))
plot(mySimpleSBM, 'data', ordered = FALSE)
plot(mySimpleSBM, 'data')
plot(mySimpleSBM, 'expected', ordered = FALSE)
plot(mySimpleSBM, 'expected')
```

### =======================================
### SIMPLE GAUSSIAN SBM

## Graph parameters & Sampling

nbNodes <- 90
blockProp <- c(.5, .25, .25)  # group proportions
means <- diag(15., 3) + 5  # connectivity matrix: affiliation network
connectParam <- list(mean = means, var = 2)
mySampler <- sampleSimpleSBM(nbNodes, blockProp, connectParam, model = "gaussian")

## Estimation

mySimpleSBM <-
estimateSimpleSBM(mySampler$networkData, 'gaussian', estimOptions = list(plot = FALSE))
plot(mySimpleSBM, 'data', ordered = FALSE)
plot(mySimpleSBM, 'expected', ordered = FALSE)
plot(mySimpleSBM, 'expected')

<table>
<thead>
<tr>
<th>fitted.SBM</th>
<th>Extract model fitted values</th>
</tr>
</thead>
</table>

**Description**

Extracts fitted values for object with class `SimpleSBM_fit, BipartiteSBM_fit` or `multipartiteSBM_fit`

**Usage**

```r
## S3 method for class 'SBM'
fitted(object, ...)
```

**Arguments**

- `object`: an R6 object inheriting from SimpleSBM_fit, BipartiteSBM_fit or MultipartiteSBM_fit
- `...`: additional parameters for S3 compatibility. Not used

**Value**

a matrix of expected fitted values for each dyad
fungusTreeNetwork  \quad  fungus-tree interaction network

Description

This data set provides information about 154 fungi sampled on 51 tree species.

Usage

fungusTreeNetwork

Format

A list with the following entries:

- **fungi_list**  \quad  list of the fungus species names
- **tree_list**  \quad  list of the tree species names
- **fungus_tree**  \quad  binary fungus-tree interactions
- **tree_tree**  \quad  weighted tree-tree interactions (number of common fungal species two tree species host)
- **covar_tree**  \quad  covariates associated to pairs of trees (namely genetic, taxonomic and geographic distances)

Source


is_SBM  \quad  Auxiliary function to check the given class of an object

Description

Auxiliary function to check the given class of an object

Usage

is_SBM(RObject)

Arguments

- **RObject**  \quad  an R6 object inheriting from class SBM

Value

TRUE or FALSE
Ecological multipartite interaction network

Description

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

Usage

multpartiteEcologicalNetwork

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 with 141 rows and 46 columns
- **Inc_plant_flovis** Interactions between plants (rows) and pollinators (cols). Matrix with 141 rows and 173 columns ...

Source


MultipartiteSBM  

*R6 Class definition of a Multipartite SBM*

Description

R6 Class definition of a Multipartite SBM

Super class

`sbm::SBM` -> `MultipartiteSBM`
Active bindings

- `dimLabels` vector of characters giving the label of each connected dimension
- `blockProp` list of two vectors of block proportions (aka prior probabilities of each block)
- `connectParam` parameters associated to the connectivity of the SBM, e.g., matrix of inter/inter
  block probabilities when model is Bernoulli
- `probMemberships` matrix of estimated probabilities for block memberships for all nodes
- `nbBlocks` vector with the number of blocks in each FG
- `nbConnectParam` number of parameter used for the connectivity
- `architecture` organization of the multipartite network
- `nbNetworks` number of networks in the multipartite network
- `memberships` list of size 2: vector of memberships in all parts of the network
- `indMemberships` matrix for clustering memberships

Methods

**Public methods:**

- `MultipartiteSBM$new()`
- `MultipartiteSBM$show()`
- `MultipartiteSBM$print()`
- `MultipartiteSBM$plot()`
- `MultipartiteSBM$clone()`

**Method new():** constructor for Multipartite SBM

*Usage:*

```r
MultipartiteSBM$new(
  model = character(0),
  architecture = matrix(NA, 0, 2),
  directed = logical(0),
  nbNodes = numeric(0),
  dimLabels = character(0),
  blockProp = list(),
  connectParam = list()
)
```

*Arguments:*

- `model` character describing the type of model
- `architecture` a 2-column matrix describing interactions between the networks
- `directed` vector of logical: are the network directed or not?
- `nbNodes` number of nodes in each dimension/part of the network
- `dimLabels` labels of each par of the network
- `blockProp` parameters for block proportions (vector of list of vectors)
- `connectParam` parameters of connectivity (vector of list of vectors)

**Method show():** print method
Usage:
MultipartiteSBM$show(type = "Multipartite Stochastic Block Model")

Arguments:
type character to tune the displayed name

Method print(): print method

Usage:
MultipartiteSBM$print()

Method plot(): plot Multipartite Network

Usage:
MultipartiteSBM$plot(
  type = c("data", "expected", "meso"),
  ordered = TRUE,
  plotOptions = list()
)

Arguments:
type character for the type of plot: either 'data' (true connection), 'expected' (fitted connection)
or 'meso' (mesoscopic view). Default to 'data'.
ordered TRUE is the matrices are plotted after reorganization with the blocks. Default value = TRUE
plotOptions list of plot options for the mesoscopic view or matrix view

Method clone(): The objects of this class are cloneable with this method.

Usage:
MultipartiteSBM$clone(deep = FALSE)

Arguments:
deep Whether to make a deep clone.

Description
R6 Class definition of a Multipartite SBM fit
R6 Class definition of a Multipartite SBM fit
Details
This class is designed to give a representation and adjust a Multipartite SBM fitted with GREMLIN. The list of parameters estimOptions essentially tunes the optimization process and the variational EM algorithm, with the following parameters

- "nbCores": integer for number of cores used. Default is 2
- "verbosity": integer for verbosity (0, 1). Default is 1
- "nbBlocksRange": list of length the number of functional groups, each element supplying the minimal and maximal number of blocks to be explored. The names of the list must be the names of the functional groups. Default value is from 1 to 10
- "initBM": Boolean. True if using simple and bipartite SBM as initialisations. Default value = TRUE
- "maxiterVEM": number of max. number of iterations in the VEM. Default value = 100
- "maxiterVE": number of max. number of iterations in the VE. Default value = 100

Super classes

- sbm::SBM -> sbm::MultipartiteSBM -> MultipartiteSBM_fit

Active bindings

- loglik: double: approximation of the log-likelihood (variational lower bound) reached
- ICL: double: value of the integrated classification log-likelihood
- storedModels: data.frame of all models fitted (and stored) during the optimization

Methods

Public methods:

- MultipartiteSBM_fit$new()
- MultipartiteSBM_fit$optimize()
- MultipartiteSBM_fit$predict()
- MultipartiteSBM_fit$setModel()
- MultipartiteSBM_fit$show()
- MultipartiteSBM_fit$clone()

Method new(): constructor for Multipartite SBM

Usage:
MultipartiteSBM_fit$new(netList)

Arguments:
netList list of SBM objects

Method optimize(): estimation of multipartiteSBM via GREMLINS

Usage:
MultipartiteSBM_fit$optimize(estimOptions)
**Arguments:**
estimOptions  options for MultipartiteBM

**Method** predict(): prediction under the currently estimated model

*Usage:*
MultipartiteSBM_fit$predict()

*Returns:* a list of matrices matrix of expected values for each dyad

**Method** setModel(): method to select a specific model among the ones fitted during the optimization. Fields of the current MultipartiteSBM_fit will be updated accordingly.

*Usage:*
MultipartiteSBM_fit$setModel(index)

*Arguments:*
index  integer, the index of the model to be selected (row number in storedModels)

**Method** show(): show method

*Usage:*
MultipartiteSBM_fit$show(type = "Fit of a Multipartite Stochastic Block Model")

*Arguments:*
type  character used to specify the type of SBM

**Method** clone(): The objects of this class are cloneable with this method.

*Usage:*
MultipartiteSBM_fit$clone(deep = FALSE)

*Arguments:*
deep  Whether to make a deep clone.

---

**MultiplexSBM_fit**  
*R6 Class definition of a Multiplex SBM fit*

**Description**

R6 Class definition of a Multiplex SBM fit

**Details**

This class is designed to give a representation and adjust a Multiplex SBM fitted with GREMLIN. The list of parameters estimOptions essentially tunes the optimization process and the variational EM algorithm, with the following parameters:

- "nbCores" integer for number of cores used. Default is 2
- "verbosity" integer for verbosity (0, 1). Default is 1
• "nbBlocksRange" List of length the number of functional groups, each element supplying the minimal and maximal number of blocks to be explored. The names of the list must be the names of the functional groups. Default value is from 1 to 10
• "initBM" Boolean. True if using simple and bipartite SBM as initialisations. Default value = TRUE
• "maxiterVEM" Number of max. number of iterations in the VEM. Default value = 100
• "maxiterVE" Number of max. number of iterations in the VE. Default value = 100

Super classes

```
sbm::SBM -> sbm::MultipartiteSBM -> sbm::MultipartiteSBM_fit -> MultiplexSBM_fit
```

Active bindings

```
nbBlocks  vector of size 2: number of blocks (rows, columns)
dependentNetwork : connection parameters in each network
storedModels  data.frame of all models fitted (and stored) during the optimization
namesLayers : names of the various Networks
```

Methods

**Public methods:**

- `MultiplexSBM_fit$new()`
- `MultiplexSBM_fit$optimize()`
- `MultiplexSBM_fit$plot()`
- `MultiplexSBM_fit$show()`
- `MultiplexSBM_fit$predict()`
- `MultiplexSBM_fit$clone()`

**Method new():** constructor for Multiplex SBM

**Usage:**

```
MultiplexSBM_fit$new(netList, dependentNet = FALSE)
```

**Arguments:**

- `netList` list of SBM object with
- `dependentNet` boolean indicating whether dependence is assumed between networks beyond the common dependence on the latent variables

**Method optimize():** estimation of multipartiteSBM via GREMLINS

**Usage:**

```
MultiplexSBM_fit$optimize(estimOptions)
```

**Arguments:**

- `estimOptions` options for MultipartiteBM

**Method plot():** plot Multiplex Network
plot.SBM

Usage:
MultiplexSBM_fit$plot(
  type = c("data", "expected"),
  ordered = TRUE,
  plotOptions = list()
)

Arguments:
type character for the type of plot: either 'data' (true connection), 'expected' (fitted connection). Default to 'data'.
ordered TRUE is the matrices are plotted after reorganization with the blocks. Default value = TRUE
plotOptions list of plot options for the matrix view

Method show(): show method

Usage:
MultiplexSBM_fit$show(type = "Fit of a Multiplex Stochastic Block Model")

Arguments:
type character used to specify the type of SBM

Method predict(): prediction under the currently estimated model

Usage:
MultiplexSBM_fit$predict()

Returns: a list of matrices matrix of expected values for each dyad

Method clone(): The objects of this class are cloneable with this method.

Usage:
MultiplexSBM_fit$clone(deep = FALSE)

Arguments:
deep Whether to make a deep clone.

---

plot.SBM  

SBM Plot

Description

Basic matrix plot method for SBM object or mesoscopic view

Usage

```r
## S3 method for class 'SBM'
plot(
  x,
  type = c("data", "expected", "meso"),
  ordered = TRUE,
  plotOptions = list(),
  ...
)
```
Arguments

- **x**: an object inheriting from class SBM
- **type**: character for the type of plot: either 'data' (true connection), 'expected' (fitted connection) or 'meso' (mesoscopic). Default to 'data'.
- **ordered**: logical: should the rows and columns be ordered according to the clustering? Default to TRUE (not taken into account for 'meso').
- **plotOptions**: list with parameters for 'meso' type plot and data type plot. Details are given below
- **...**: additional parameters for S3 compatibility. Not used

Details

The list of parameters plotOptions for the mesoscopic plot is:

- "seed": seed to control the layout
- "title": character string for the title. Default value is NULL
- "layout": Default value = NULL
- "vertex.color": Default value is "salmon2"
- "vertex.frame.color": Node border color. Default value is "black"
- "vertex.shape": One of "none", "circle", "square", "csquare", "rectangle" "crectangle", "vrectangle", "pie", "raster", or "sphere". Default value = "circle"
- "vertex.size": Size of the node (default is 2)
- "vertex.size2": The second size of the node (e.g. for a rectangle)
- "vertex.label.name": Names of the vertices. Default value is the label of the nodes
- "vertex.label.color": Default value is "black"
- "vertex.label.font": Default value is 2. Font: 1 plain, 2 bold, 3 italic, 4 bold italic, 5 symbol
- "vertex.label.cex": Font size (multiplication factor, device-dependent). Default value is 0.9.
- "vertex.label.dist": Distance between the label and the vertex. Default value is 0
- "vertex.label.degree": The position of the label in relation to the vertex. Default value is 0
- "edge.threshold": Threshold under which the edge is not plotted. Default value is = -Inf
- "edge.color": Default value is "gray"
- "edge.width": Factor parameter. Default value is 10
- "edge.arrow.size": Default value is 1
- "edge.arrow.width": Default value is 2
- "edge.lty": Line type, could be 0 or "blank", 1 or "solid", 2 or "dashed", 3 or "dotted", 4 or "dotdash", 5 or "longdash", 6 or "twodash". Default value is "solid"
- "edge.curved": Default value is = 0.3.

For type = 'data' or 'expected plot', the list of parameters plotOptions is

- "legend": Boolean. Set TRUE if you want to see the legend. Default value is FALSE
plotAlluvial

- "legend.title": Boolean. Set TRUE if you want to print the title of the legend. Default value is FALSE
- "legend.position": Position of the legend. Possible values are 'bottom', 'top', 'left', 'right'. Default value is 'bottom'
- "rowNames": Set true if the rownames must be plotted. Default value is FALSE
- "colNames": Set true if the colNames must be plotted. Default value is FALSE
- "line.color": Chain of character. The color of the lines to separate groups if a clustering is provided. Default value is red
- "line.width": Numeric. Width of the lines to separate groups. Default value is NULL, automatically chosen
- "title": Chain of character. Title of the plot. Default value is NULL

Value

a ggplot2 object for the 'data' and 'expected', a list with the igraph object g and the layout for the 'meso'

plotAlluvial

Plot an alluvial plot between clusterings

Description

Plot an alluvial plot between clusterings

Usage

plotAlluvial(listMemberships, plotOptions = list())

Arguments

- listMemberships: a list vectors containing the memberships
- plotOptions: a list containing the options for Alluvial plots

Details

The list of parameters plotOptions provides the following options

- "curvy": numeric, controls the curvature of the alluvial. Default value = 0.3
- "alpha": numeric, vector of transparency of the stripes. Default value = 0.8
- "gap.width": numeric, relative width of inter-category gaps. Default value = 0.1
- "col": vector of colors of the stripes. Default value = "darkolivegreen3"
- "border": vector of border colors for the stripes. Default is white
Value

display the alluvial plot, returns the plotOptions as a list

Examples

```r
listMemberships <- list(C1 = rep(c('A','B','C'),each=10),C2 = rep(c(1,2,4),10))
plotAlluvial(listMemberships)
```

plotMyMatrix

Plot an adjacency or incidence Matrix

Description

Plot an adjacency or incidence Matrix

Usage

```r
plotMyMatrix(
  Mat,
  dimLabels = list(row = "row", col = "col"),
  clustering = NULL,
  plotOptions = NULL
)
```

Arguments

- `Mat`: a matrix representing the network
- `dimLabels`: a list of length 2 specifying the types of nodes in row and col (functional group)
  (Default is `NULL`)
- `clustering`: a list of length 2 specifying a clustering on row and col
- `plotOptions`: a list providing options. See details below.

Details

The list of parameters `plotOptions` for the matrix plot is

- "legend": Boolean. Set TRUE if you want to see the legend. Default value is FALSE
- "legend.title": Boolean. Set TRUE if you want to print the title of the legend. Default value is FALSE
- "legend.position": Position of the legend. Possible values are 'bottom', 'top', 'left', 'right'. Default value is 'bottom'
- "rowNames": Set true if the rownames must be plotted. Default value is FALSE
- "colNames": Set true if the colNames must be plotted. Default value is FALSE
- "line.color": Chain of character. The color of the lines to separate groups if a clustering is provided. Default value is red
- "line.width": Numeric. Width of the lines to separate groups. Default value is NULL, automatically chosen
- "title": Chain of character. Title of the plot. Default value is NULL
Value

a ggplot object corresponding to the plot

Examples

M <- matrix(sample(c(0,1),900,replace=TRUE),30,30)
plotMyMultipartiteMatrix(M, plotOptions= list(legend = FALSE))
M2 <- matrix( rpois(800,10),40,20)
plotMyMultipartiteMatrix(M2, dimLabels=list(row= 'reader',col = 'book'), plotOptions = list(legend = TRUE))

plotMyMultipartiteMatrix

Plot the matrices corresponding to a Multipartite Network

Description

Plot the matrices corresponding to a Multipartite Network

Usage

plotMyMultipartiteMatrix(listSBM, memberships = NULL, plotOptions = list())

Arguments

listSBM : a list of objects representing the multipartite network (see)
memberships : a list of length equal to the number of Functional Groups providing the clusterings inside each group.
plotOptions : a list containing the options. See details.

Details

plotOptions is a list containing the following items

- "normalized": Boolean. TRUE if the various matrices are presented in the same scale (between 0 and 1). FALSE otherwise. Default value FALSE
- "compact": Boolean. Default value is TRUE if you ask for the matrices to be transposed to have a more compact view
- "legend": Boolean. Set TRUE if you want to see the legend. Default value is FALSE
- "legend.title": Boolean. Set TRUE if you want to print the title of the legend. Default value is FALSE
- "legend.position": Position of the legend. Possible values are 'bottom', 'top', 'left', 'right'. Default value is 'bottom'
- "nodeNames": Set true if the node Names must be plotted. Default value is FALSE
- "line.color": The color of the lines to separate groups. Default value is red
- "line.width": Width of the lines to separate groups. Default value is NULL, automatically chosen
- "title": Title of the plot. Default value is NULL
plotMyMultiplexMatrix

Plot the matrices corresponding to a Multiplex Network

Description

Plot the matrices corresponding to a Multiplex Network

Usage

plotMyMultiplexMatrix(listSBM, memberships = NULL, plotOptions = list())

Arguments

- `listSBM`: a list of objects representing the multiplex network (see)
- `memberships`: a list of length equal to the number of Functional Groups providing the clusterings inside each group.
- `plotOptions`: a list containing the options. See details.
plotMyMultiplexMatrix

Details

plotOptions is a list containing the following items

- "normalized": Boolean. TRUE if the various matrices are presented in the same scale (between 0 and 1). FALSE otherwise. Default value FALSE
- "compact": Boolean. Default value is TRUE if you ask for the matrices to be transposed to have a more compact view
- "legend": Boolean. Set TRUE if you want to see the legend. Default value is FALSE
- "legend.title": Boolean. Set TRUE if you want to print the title of the legend. Default value is FALSE
- "legend.position": Position of the legend. Possible values are 'bottom', 'top', 'left', 'right'. Default value is 'bottom'
- "nodeNames": Set true if the node Names must be plotted. Default value is FALSE
- "line.color": The color of the lines to separate groups. Default value is red
- "line.width": Width of the lines to separate groups. Default value is NULL, automatically chosen
- "title": Title of the plot. Default value is NULL

Value

a ggplot object corresponding to the plot

Examples

```r
Nnodes <- c(40,30)
blockProp <- list(c(.4,.6),c(0.5,0.5))
nbLayers <- 2
connectParam <- list(list(mean=matrix(rbeta(4,.5,.5),2,2)),list(mean=matrix(rexp(4,.5),2,2)))
names(connectParam) <- c('Read','Score')
model <- c('bernoulli','poisson')
type <- "bipartite"
mySampleMultiplexSBM <- sampleMultiplexSBM(
  nbNodes = Nnodes,
  blockProp = blockProp,
  nbLayers = nbLayers,
  connectParam = connectParam,
  model=model,
  dimLabels = c('readers','books'),
  type=type)
listNet <- mySampleMultiplexSBM$listSBM
names(listNet) <- c("Read","Affinity")
plotMyMultiplexMatrix(listNet,plotOptions=list(legend = TRUE))
```
**predict.SBM**  
*Model Predictions*

**Description**

Make predictions from an SBM.

**Usage**

```r
## S3 method for class 'SBM'
predict(object, covarList = object$covarList, theta_p0 = 0, ...)
```

**Arguments**

- `object`: an R6 object inheriting from class SBM_fit (like SimpleSBM_fit or BipartiteSBM_fit)
- `covarList`: a list of covariates. By default, we use the covariates associated with the model.
- `theta_p0`: a threshold...
- `...`: additional parameters for S3 compatibility. Not used

**Value**

a matrix of expected values for each dyad

---

**sampleBipartiteSBM**  
*Sampling of Bipartite SBMs*

**Description**

This function samples a simple Stochastic Block Models, with various model for the distribution of the edges: Bernoulli, Poisson, or Gaussian models, and possibly with covariates.

**Usage**

```r
sampleBipartiteSBM(  
  nbNodes,  
  blockProp,  
  connectParam,  
  model = "bernoulli",  
  dimLabels = c(row = "rowName", col = "colName"),  
  covariates = list(),  
  covariatesParam = numeric(0)  
)  
```
sampleBipartiteSBM

Arguments

- **nbNodes**: number of nodes in the network
- **blockProp**: parameters for block proportions: list of size two with row and column block proportions
- **connectParam**: list of parameters for connectivity with a matrix of means 'mean' and an optional matrix of variances 'var', the sizes of which must match blockProp length (in row, respectively in column)
- **model**: character describing the model for the relation between nodes ('bernoulli', 'poisson', 'gaussian', 'ZIgaussian'). Default is 'bernoulli'.
- **dimLabels**: an optional list of labels for each dimension (in row, in column)
- **covariates**: a list of matrices with same dimension as mat describing covariates at the edge level. No covariate per Default.
- **covariatesParam**: optional vector of covariates effect. A zero length numeric vector by default.

Value

an object with class `BipartiteSBM`

Examples

```r
### =======================================
### BIPARTITE BERNOULLI SBM
## Graph parameters
nbNodes <- c(100, 120)
blockProp <- list(c(.5, .5), c(1/3, 1/3, 1/3)) # group proportions
means <- matrix(runif(6), 2, 3) # connectivity matrix
# In Bernoulli SBM, parameters is a list with
# a matrix of means 'mean' which are probabilities of connection
connectParam <- list(mean = means)

## Graph Sampling
dimLabels = c(row="Reader",col="Book")
mySampler <- sampleBipartiteSBM(nbNodes, blockProp, connectParam, model = 'bernoulli',dimLabels)
plot(mySampler)
plot(mySampler,type='meso',plotOptions = list(vertex.label.name=list(row='Reader',col='Book')))
mySampler$rMemberships() # sample new memberships
mySampler$rEdges() # sample new edges
mySampler$rNetwork() # sample a new network (blocks and edges)
### =======================================
### BIPARTITE POISSON SBM
## Graph parameters
nbNodes <- c(100, 120)
blockProp <- list(c(.5, .5), c(1/3, 1/3, 1/3)) # group proportions
means <- matrix(rbinom(6, 30, 0.25), 2, 3) # connectivity matrix
# In Poisson SBM, parameters is a list with a matrix of
# means 'mean' which are a mean integer value taken by edges
connectParam <- list(mean = means)
```
## Graph Sampling

dimLabels = c(row = 'Ind', col = 'Service')
mySampler <- sampleBipartiteSBM(nbNodes, blockProp, connectParam, model = 'poisson', dimLabels)
plot(mySampler, type='expected')
plotOptions = list(vertex.label.name=c('U','V'), vertex.size = c(1.4,1.3))
plot(mySampler, type='meso', plotOptions = plotOptions)
hist(mySampler$networkData)

### BIPARTITE GAUSSIAN SBM

### Graph parameters

nbNodes <- c(100, 120)
blockProp <- list(c(.5, .5), c(1/3, 1/3, 1/3))  # group proportions
means <- 20 * matrix(runif(6), 2, 3)  # connectivity matrix
# In Gaussian SBM, parameters is a list with a matrix
# of means 'mean' and a matrix of variances 'var'
connectParam <- list(mean = means, var = 1)

mySampler <- sampleBipartiteSBM(nbNodes, blockProp, connectParam, model = 'gaussian')
plot(mySampler)
hist(mySampler$networkData)

### Sampling of Multipartite SBMs

This function samples a Multipartite Stochastic Block Models, with various model for the distribution of the edges: Bernoulli, Poisson, or Gaussian models

### Usage

```r
sampleMultipartiteSBM(
  nbNodes,
  blockProp,
  archiMultipartite,
  connectParam,
  model,
  directed,
  dimLabels = NULL,
  seed = NULL
)
```

### Arguments

- `nbNodes`: number of nodes in each functional group involved in the multipartite network
blockProp  a list of parameters for block proportions in each functional group
archiMultipartite a matrix with two columns and nbNetworks lines, each line specifying the index of the functional groups in interaction.
connectParam list of parameters for connectivity (of length nbNetworks). Each element is a list of one or two elements: a matrix of means 'mean' and an optional matrix of variances 'var', the sizes of which must match blockProp length
model a vector of characters describing the model for each network of the Multipartite relation between nodes ('bernoulli', 'poisson', 'gaussian', ....). Default is 'bernoulli'.
directed a vector of logical, directed network or not for each network. Default is FALSE.
dimLabels an optional list of labels for functional group involved in the network
seed numeric to set the seed.

Value
a list of two elements : simulatedMemberships are the clustering of each node in each Functional Group, multipartiteNetwork is the list of the simulated networks (each one being a simple or bipartite network)

Examples

```r
### =======================================
### MULTIPARTITE SBM : 4 networks between 3 Functional Groups
### Graph parameters
# About the Functional Groups (FG)
nbNodes <- c(100,50,40)
blockProp <- vector("list", 3) # parameters of clustering in each functional group
blockProp[[1]] <- c(0.4,0.3,0.3) # in Functional Group 1
blockProp[[2]] <- c(0.6,0.4)  # in Functional Group 2
blockProp[[3]] <- c(0.6,0.4)  # in Functional Group 3
# About the interactions between the FG
archiMultipartite <- rbind(c(1,2),c(2,3),c(2,2),c(1,3)) #
model <- c('bernoulli','poisson','gaussian','gaussian') # type of distribution in each network
# for each network : directed or not (not required for an interaction between two different FG)
directed <- c(NA,NA,FALSE,NA)
connectParam <- list()
connectParam[[1]] <- list(mean = matrix(c(0.3, 0.3, 0.5, 0.2, 0.6, 0.6),3,2))
connectParam[[2]] <- list(mean = matrix(c(1000, 500, 400, 950),2,2))
connectParam[[3]] <- list(mean = matrix(c(10, -10, 20), 2,2), var = matrix(1,2,2))
connectParam[[4]] <- list(mean = matrix(c(3, 23, 11, 16, 2, 25), 3,2))
connectParam[[4]]$var <- matrix(c(10,20,1,5,0.1,10), 3,2)
dimLabels <- c("A","B","C")
### Graph Sampling
mySampleMBM <- sampleMultipartiteSBM(nbNodes, blockProp,
archiMultipartite,
connectParam, model, directed,
dimLabels,seed = 3)
listSBM <- mySampleMBM$listSBM
memberships <- mySampleMBM$memberships
```
sampleMultiplexSBM

Sampling of Multiplex SBMs

Description

This function samples a Multiplex Stochastic Block Models, with various model for the distribution of the edges: Bernoulli, Poisson, or Gaussian models.

Usage

```r
call = sampleMultiplexSBM(
  nbNodes,
  blockProp,
  nbLayers,
  connectParam,
  model,
  type = c("directed", "undirected", "bipartite"),
  dependent = FALSE,
  dimLabels = NULL,
  seed = NULL
)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>nbNodes</code></td>
<td>number of nodes in each functional group involved in the Multiplex network</td>
</tr>
<tr>
<td><code>blockProp</code></td>
<td>a vector for block proportion if the networks are simple, a list of parameters for block proportions for both functional groups if the networks are bipartite</td>
</tr>
<tr>
<td><code>nbLayers</code></td>
<td>a matrix with two columns and nbNetworks lines, each line specifying the index of the functional groups in interaction.</td>
</tr>
<tr>
<td><code>connectParam</code></td>
<td>list of parameters for connectivity (of length nbNetworks). Each element is a list of one or two elements: a matrix of means 'mean' and an optional matrix of variances 'var', the sizes of which must match blockProp length</td>
</tr>
<tr>
<td><code>model</code></td>
<td>a vector of characters describing the model for each network of the Multiplex relation between nodes ('bernoulli', 'poisson', 'gaussian', ...). Default is 'bernoulli'.</td>
</tr>
<tr>
<td><code>type</code></td>
<td>a string of character indicating whether the networks are directed, undirected or bipartite</td>
</tr>
<tr>
<td><code>dependent</code></td>
<td>connection parameters in each network</td>
</tr>
<tr>
<td><code>dimLabels</code></td>
<td>an optional list of labels for functional group involved in the network</td>
</tr>
<tr>
<td><code>seed</code></td>
<td>numeric to set the seed.</td>
</tr>
</tbody>
</table>
Value

a list of two elements: simulatedMemberships are the clustering of each node in each Functional Group, MultiplexNetwork is the list of the simulated networks (each one being a simple or bipartite network)

Examples

```r
nbLayers <- 2

## MultiplexSBM without dependence between layers
Nnodes <- 40
blockProp <- c(.4,.6)
connectParam <- list(list(mean=matrix(rbeta(4,.5,.5),2,2)),list(mean=matrix(rexp(4,.5),2,2)))
model <- c("beroulli","poisson")
type <- "directed"
mySampleMultiplexSBM <-
  sampleMultiplexSBM(
    nbNodes = Nnodes,
    blockProp = blockProp,
    nbLayers = nbLayers,
    connectParam = connectParam,
    model=model,
    type=type)
listSBM <- mySampleMultiplexSBM$listSBM

## MultiplexSBM Gaussian with dependence
Q <- 3
nbLayers <- 2
connectParam <- list()
connectParam$mu <- vector("list",nbLayers)
connectParam$mu[[1]] <- matrix(.1,Q,Q) + diag(1:Q)
connectParam$mu[[2]] <- matrix(-2,Q,Q) + diag(rev(Q:1))
connectParam$Sigma <- matrix(c(2,1,1,4),nbLayers,nbLayers)
model <- rep("gaussian",2)
type <- "directed"
Nnodes <- 80
blockProp <- c(.3,.3,.4)
mySampleMultiplexSBM <-
  sampleMultiplexSBM(
    nbNodes = Nnodes,
    blockProp = blockProp,
    nbLayers = nbLayers,
    connectParam = connectParam,
    model=model,
    type="undirected",
    dependent=TRUE)
listSBM <- mySampleMultiplexSBM$listSBM

## MultiplexSBM Bernoulli with dependence
Q <- 2
P00<-matrix(runif(Q*Q),Q,Q)
P10<-matrix(runif(Q*Q),Q,Q)
P01<-matrix(runif(Q*Q),Q,Q)
```
Sampling of Simple SBMs

Description

This function samples a simple Stochastic Block Models, with various model for the distribution of the edges: Bernoulli, Poisson, or Gaussian models, and possibly with covariates

Usage

```r
sampleSimpleSBM(
  nbNodes,
  blockProp,
  connectParam,
  model = "bernoulli",
  directed = FALSE,
  dimLabels = c(node = "nodeName"),
  covariates = list(),
  covariatesParam = numeric(0)
)
```
Arguments

- **nbNodes** number of nodes in the network
- **blockProp** parameters for block proportions
- **connectParam** list of parameters for connectivity with a matrix of means 'mean' and an optional matrix of variances 'var', the sizes of which must match blockProp length
- **model** character describing the model for the relation between nodes ('bernoulli', 'poisson', 'gaussian', ...). Default is 'bernoulli'.
- **directed** logical, directed network or not. Default is FALSE.
- **dimLabels** an optional list of labels for each dimension (in row, in column)
- **covariates** a list of matrices with same dimension as mat describing covariates at the edge level. No covariate per Default.
- **covariatesParam** optional vector of covariates effect. A zero length numeric vector by default.

Value

a list of matrices with same dimension as mat describing covariates at the edge level. No covariate per Default.

Examples

```r
### SIMPLE BINARY SBM (Bernoulli model)
## Graph parameters
nbNodes <- 90
blockProp <- c(.5, .25, .25) # group proportions
means <- diag(.4, 3) + 0.05 # connectivity matrix: affiliation network
# In Bernoulli SBM, parameters is a list with a
# matrix of means 'mean' which are probabilities of connection
connectParam <- list(mean = means)

## Graph Sampling
mySampler <- sampleSimpleSBM(nbNodes, blockProp, connectParam, model = 'bernoulli')
plot(mySampler)
plot(mySampler)
plot(mySampler, type = 'meso')
hist(mySampler$networkData)

### SIMPLE POISSON SBM
## Graph parameters
nbNodes <- 90
blockProp <- c(.5, .25, .25) # group proportions
means <- diag(15.3) + 5 # connectivity matrix: affiliation network
# In Poisson SBM, parameters is a list with
# a matrix of means 'mean' which are a mean integer value taken by edges
connectParam <- list(mean = means)

## Graph Sampling
mySampler <- sampleSimpleSBM(nbNodes, blockProp, list(mean = means), model = 'poisson')
```

plot(mySampler)
plot(mySampler,type='meso')
hist(mySampler$networkData)

### SIMPLE GAUSSIAN SBM
### Graph parameters
nbNodes <- 90
blockProp <- c(.5, .25, .25)  # group proportions
means <- diag(15., 3) + 5  # connectivity matrix: affiliation network
# In Gaussian SBM, parameters is a list with
# a matrix of means 'mean' and a matrix of variances 'var'
connectParam <- list(mean = means, var = 2)

## Graph Sampling
mySampler <- sampleSimpleSBM(nbNodes, blockProp, connectParam, model = "gaussian")
plot(mySampler)
plot(mySampler,type='meso')
hist(mySampler$networkData)

---

**SBM**  
*R6 virtual class for SBM representation (mother class of SimpleSBM, BipartiteSBM, MultipartiteSBM)*

**Description**
R6 virtual class for SBM representation (mother class of SimpleSBM, BipartiteSBM, MultipartiteSBM)

**Active bindings**
- **modelName** character, the family of model for the distribution of the edges
- **directed** mode of the network data (directed or not or not applicable)
- **dimLabels** vector or list of characters, the label of each dimension
- **nbNodes** vector describing the number of the successive elements connecting the network
- **nbCovariates** integer, the number of covariates
- **blockProp** block proportions (aka prior probabilities of each block)
- **connectParam** parameters associated to the connectivity of the SBM, e.g. matrix of inter/inter block probabilities when model is Bernoulli
- **covarParam** vector of regression parameters associated with the covariates.
- **covarList** list of matrices of covariates
- **covarArray** the array of covariates
- **covarEffect** effect of covariates
- **networkData** the network data (adjacency or incidence matrix or list of such object)
- **expectation** expected values of connection under the current model
Methods

Public methods:

• SBM$new()
• SBM$rNetwork()
• SBM$show()
• SBM$print()
• SBM$clone()

Method new(): constructor for SBM

Usage:
SBM$new(
  model = vector("character", 0),
  directed = vector("logical", 0),
  dimension = vector("numeric", 0),
  dimLabels = vector("character", 0),
  blockProp = vector("numeric", 0),
  connectParam = vector("list", 0),
  covarParam = numeric(length(covarList)),
  covarList = list()
)

Arguments:
model character describing the type of model
directed logical describing if the network data is directed or not
dimension dimension of the network data
dimLabels labels of each dimension
blockProp parameters for block proportions (vector or list of vectors)
connectParam list of parameters for connectivity
covarParam optional vector of covariates effect
covarList optional list of covariates data

Method rNetwork(): a method to sample a network data for the current SBM (blocks and edges)

Usage:
SBM$rNetwork(store = FALSE)

Arguments:
store should the sampled network be stored (and overwrite the existing data)? Default to FALSE

Returns: a list with the sampled block and network

Method show(): print method

Usage:
SBM$show(type = "Stochastic Block Model")

Arguments:
**SimpleSBM**

R6 class for Simple SBM

---

**Description**

R6 class for Simple SBM

**Super class**

`sbm::SBM` -> `SimpleSBM`

**Active bindings**

- `dimLabels` a single character giving the label of the nodes
- `blockProp` vector of block proportions (aka prior probabilities of each block)
- `connectParam` parameters associated to the connectivity of the SBM, e.g. matrix of inter/inter block probabilities when model is Bernoulli
- `probMemberships` matrix of estimated probabilities for block memberships for all nodes
- `nbBlocks` number of blocks
- `nbDyads` number of dyads (potential edges in the network)
- `nbConnectParam` number of parameter used for the connectivity
- `memberships` vector of clustering
- `indMemberships` matrix for clustering memberships
Methods

Public methods:

- SimpleSBM$new()
- SimpleSBM$rMemberships()
- SimpleSBM$rEdges()
- SimpleSBM$predict()
- SimpleSBM$show()
- SimpleSBM$plot()
- SimpleSBM$clone()

Method new(): constructor for SBM

Usage:
SimpleSBM$new(
  model,
  nbNodes,
  directed,
  blockProp,
  connectParam,
  dimLabels = c(node = "nodeName"),
  covarParam = numeric(length(covarList)),
  covarList = list()
)

Arguments:
model character describing the type of model
nbNodes number of nodes in the network
directed logical, directed network or not.
blockProp parameters for block proportions (vector of list of vectors)
connectParam list of parameters for connectivity with a matrix of means 'mean' and an optional scalar for the variance 'var'. The size of mu must match blockProp length
dimLabels optional label for the node (default is "nodeName")
covarParam optional vector of covariates effect
covarList optional list of covariates data

Method rMemberships(): a method to sample new block memberships for the current SBM

Usage:
SimpleSBM$rMemberships(store = FALSE)

Arguments:
store should the sampled blocks be stored (and overwrite the existing data)? Default to FALSE

Returns: the sampled blocks

Method rEdges(): a method to sample a network data (edges) for the current SBM

Usage:
SimpleSBM$rEdges(store = FALSE)
Arguments:
store should the sampled edges be stored (and overwrite the existing data)? Default to FALSE

Returns: the sampled network

Method predict(): prediction under the currently parameters
Usage:
`SimpleSBM$predict(covarList = self$covarList, theta_p0 = 0)`
Arguments:
covarList a list of covariates. By default, we use the covariates with which the model was estimated
theta_p0 a threshold...
Returns: a matrix of expected values for each dyad

Method show(): show method
Usage:
`SimpleSBM$show(type = "Simple Stochastic Block Model")`
Arguments:
type character used to specify the type of SBM

Method plot(): basic matrix plot method for SimpleSBM object or mesoscopic plot
Usage:
`SimpleSBM$plot(
  type = c("data", "expected", "meso"),
  ordered = TRUE,
  plotOptions = list()
)`
Arguments:
type character for the type of plot: either 'data' (true connection), 'expected' (fitted connection) or 'meso' (mesoscopic view). Default to 'data'.
ordered logical: should the rows and columns be reordered according to the clustering? Default to TRUE.
plotOptions list with the parameters for the plot. See help of the corresponding S3 method for details.
Returns: a ggplot2 object for the 'data' and 'expected', a list with the igraph object g, the layout and the plotOptions for the 'meso'

Method clone(): The objects of this class are cloneable with this method.
Usage:
`SimpleSBM$clone(deep = FALSE)`
Arguments:
deep Whether to make a deep clone.
Description

R6 Class definition of a Simple SBM fit

Details

This class is designed to give a representation and adjust an SBM fitted with blockmodels.

The list of parameters estimOptions essentially tunes the optimization process and the variational EM algorithm, with the following parameters:

- "nbCores" integer for number of cores used. Default is 2
- "verbosity" integer for verbosity (0, 1). Default is 1
- "plot" boolean, should the ICL by dynamically plotted or not. Default is TRUE
- "exploreFactor" control the exploration of the number of groups
- "exploreMin" explore at least until exploreMin even if the exploration factor rule is achieved. Default 4. See the package blockmodels for details.
- "exploreMax" stop exploration at exploreMax even if the exploration factor rule is not achieved. Default Inf. See the package blockmodels for details.
- "fast" logical: should approximation be used for Bernoulli model with covariates. Default to TRUE

Super classes

```
sbm::SBM -> sbm::SimpleSBM -> SimpleSBM_fit
```

Active bindings

```
loglik double: approximation of the log-likelihood (variational lower bound) reached
ICL double: value of the integrated classification log-likelihood
penalty double: value of the penalty term in ICL
entropy double, value of the entropy due to the clustering distribution
storedModels data.frame of all models fitted (and stored) during the optimization
```

Methods

Public methods:

- `SimpleSBM_fit$new()`
- `SimpleSBM_fit$optimize()`
- `SimpleSBM_fit$setModel()`
Method new(): constructor for a Simple SBM fit

Usage:
```r
SimpleSBM_fit$new(
  adjacencyMatrix,  
  model,            
  directed,        
  dimLabels = c(node = "nodeName"),  
  covarList = list()
)
```

Arguments:
- `adjacencyMatrix` square (weighted) matrix
- `model` character (`"bernoulli"`, `"poisson"`, `"gaussian"`)
- `directed` logical, directed network or not. In not, `adjacencyMatrix` must be symmetric.
- `dimLabels` list of labels of each dimension (in row, in columns)
- `covarList` and optional list of covariates, each of whom must have the same dimension as `adjacencyMatrix`

Method optimize(): function to perform optimization

Usage:
```r
SimpleSBM_fit$optimize(estimOptions = list())
```

Arguments:
- `estimOptions` a list of parameters controlling the inference algorithm and model selection. See details.

Method setModel(): method to select a specific model among the ones fitted during the optimization. Fields of the current SBM_fit will be updated accordingly.

Usage:
```r
SimpleSBM_fit$setModel(index)
```

Arguments:
- `index` integer, the index of the model to be selected (row number in storedModels)

Method reorder(): permute group labels by order of decreasing probability

Usage:
```r
SimpleSBM_fit$reorder()
```

Method show(): show method

Usage:
```r
SimpleSBM_fit$show(type = "Fit of a Simple Stochastic Block Model")
```

Arguments:
- `type` character used to specify the type of SBM
Method clone(): The objects of this class are cloneable with this method.

Usage:
SimpleSBM_fit$clone(deep = FALSE)

Arguments:
depth Whether to make a deep clone.

---

**Description**

This dataset contains two networks where the nodes are countries and an edge in network "belligerent" means that the two countries have been at least once at war between years 1816 to 2007 while an edge in network "alliance" means that the two countries have had a formal alliance between years 1816 to 2012. The network belligerent have less nodes since countries which have not been at war are not considered.

**Usage**

war

**Format**

A list with 2 two igraph objects, alliance and belligerent. Each graph have three attributes: 'name' (the country name), 'power' (a score related to military power: the higher, the better) and 'trade' (a score related to the trade effort between pairs of countries).

**Source**

networks were extracted from [https://www.correlatesofwar.org/](https://www.correlatesofwar.org/)

**References**


**Examples**

data(war)
class(war$belligerent)
igraph::gorder(war$alliance)
igraph::gorder(war$belligerent)
igraph::edges(war$alliance)
igraph::get.graph.attribute(war$alliance)
Index

* datasets
  fungusTreeNetwork, 18
  multipartiteEcologicalNetwork, 19
  war, 47

BipartiteSBM, 3, 33
BipartiteSBM_fit, 5, 7, 17

coef.SBM, 7

defineSBM, 8

estimateBipartiteSBM, 9
estimateMultipartiteSBM, 11
estimateMultiplexSBM, 12
estimateSimpleSBM, 15

fitted.SBM, 17
fungusTreeNetwork, 18

is_SBM, 18

multipartiteEcologicalNetwork, 19
multipartiteSBM_fit, 17
MultipartiteSBM, 19
MultipartiteSBM_fit, 21
MultiplexSBM_fit, 23

plot.SBM, 25
plotAlluvial, 27
plotMyMatrix, 28
plotMyMultipartiteMatrix, 29
plotMyMultiplexMatrix, 30
predict.SBM, 32

sampleBipartiteSBM, 32
sampleMultipartiteSBM, 34
sampleMultiplexSBM, 36
sampleSimpleSBM, 38
SBM, 7, 40

sbm:::BipartiteSBM, 6

sbm:::MultipartiteSBM, 22, 24
sbm:::MultipartiteSBM_fit, 24
sbm:::SBM, 3, 6, 19, 22, 24, 42, 45
sbm:::SimpleSBM, 45
SimpleSBM, 39, 42
SimpleSBM_fit, 7, 17, 45

war, 47