

# Package ‘sbm’

October 12, 2020

**Title** Stochastic Blockmodels

**Version** 0.2.2

**Description** A collection of tools and function to adjust a variety of stochastic blockmodels (SBM). Support at the moment Simple and Bipartite SBM (undirected and directed) for Bernoulli, Poisson and Gaussian emission laws of the edges as described in Léger, 2016 <arxiv:1602.07587>.

**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, alluvial, igraph, aricode, covr

**Language** en-US

**Imports** magrittr, blockmodels, R6, Rcpp, ggplot2

**Collate** 'R6Class-SBM.R' 'R6Class-SBM\_fit.R'  
'R6Class-BipartiteSBM\_fit.R' 'R6Class-SBM\_sampler.R'  
'R6Class-BipartiteSBM\_sampler.R' 'R6Class-SimpleSBM\_fit.R'  
'R6Class-SimpleSBM\_sampler.R' 'RcppExports.R' 'estimate.R'  
'fungus\_tree\_network.R' 'plotMyMatrix.R' 'sample.R'  
'sbm-package.R' 'utils-pipe.R' 'utils.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],  
großBM team [ctb],  
Pierre Barbillon [aut] (<<https://orcid.org/0000-0002-7766-7693>>)

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

**Repository** CRAN

**Date/Publication** 2020-10-12 19:20:02 UTC

## R topics documented:

BipartiteSBM_fit . . . . .	2
BipartiteSBM_sampler . . . . .	4
coef.SBM . . . . .	6
estimateBipartiteSBM . . . . .	7
estimateSimpleSBM . . . . .	8
fitted.SBM_fit . . . . .	11
fungus_tree_network . . . . .	11
plot.SBM . . . . .	12
plotMyMatrix . . . . .	13
predict.SBM . . . . .	13
sampleBipartiteSBM . . . . .	14
sampleSimpleSBM . . . . .	16
SBM . . . . .	17
SBM_fit . . . . .	19
SBM_sampler . . . . .	21
SimpleSBM_fit . . . . .	22
SimpleSBM_sampler . . . . .	24
<b>Index</b>	<b>27</b>

---

BipartiteSBM_fit	<i>R6 Class definition of an Bipartite SBM fit</i>
------------------	--

---

### 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.

### Super classes

`sbm::SBM` -> `sbm::SBM_fit` -> `BipartiteSBM_fit`

**Active bindings**

nbNodes vector of size 2: number of nodes (rows, columns)  
 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.  
 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\\$predict\(\)](#)
- [BipartiteSBM\\_fit\\$reorder\(\)](#)
- [BipartiteSBM\\_fit\\$show\(\)](#)
- [BipartiteSBM\\_fit\\$clone\(\)](#)

**Method** `new()`: constructor for a Bipartite SBM fit

*Usage:*

```
BipartiteSBM_fit$new(
  incidenceMatrix,
  model,
  dimLabels = list(row = "rowLabel", col = "colLabel"),
  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(
  verbosity = 3,
  plot = FALSE,
  explorFactor = 1.5,
  nbBlocksRange = c(4, Inf),
  nbCores = 2,
  fast = TRUE
)
```

*Arguments:*

verbosity integer, the level of verbosity. Default to 3  
 plot logical, if TRUE plotting is done dynamically on the screen. Default to TRUE  
 explorFactor double factor for exploring successive model  
 nbBlocksRange 2-size vector: range of exploration  
 nbCores integer, the number of cores to use. Default is 2.  
 fast logical: should approximation be used for Bernoulli model with covariates. Default to TRUE

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

*Usage:*

```
BipartiteSBM_fit$predict(covarList = self$covarList)
```

*Arguments:*

covarList a list of covariates. By default, we use the covariates with which the model was estimated.

**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.

---

BipartiteSBM\_sampler *R6 class for Bipartite SBM sampler*

---

**Description**

R6 class for Bipartite SBM sampler

R6 class for Bipartite SBM sampler

**Super classes**

`sbm::SBM` -> `sbm::SBM_sampler` -> `BipartiteSBM_sampler`

**Active bindings**

nbNodes vector of size 2: number of nodes (rows, columns)  
 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 list of 2 matrix for clustering memberships  
 expectation expected values of connection under the current model

**Methods****Public methods:**

- [BipartiteSBM\\_sampler\\$new\(\)](#)
- [BipartiteSBM\\_sampler\\$rMemberships\(\)](#)
- [BipartiteSBM\\_sampler\\$rIncidence\(\)](#)
- [BipartiteSBM\\_sampler\\$show\(\)](#)
- [BipartiteSBM\\_sampler\\$clone\(\)](#)

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

*Usage:*

```

BipartiteSBM_sampler$new(
  model,
  nbNodes,
  blockProp,
  connectParam,
  dimLabels = list(row = "rowLabel", col = "colLabel"),
  covarParam = numeric(0),
  covarList = list()
)

```

*Arguments:*

model character describing the type of model  
 nbNodes number of nodes in 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 generate a vector of block indicators

*Usage:*

```
BipartiteSBM_sampler$rMemberships()
```

**Method** `rIncidence()`: a method to sample an adjacency matrix for the current SBM

*Usage:*

```
BipartiteSBM_sampler$rIncidence()
```

*Returns:* nothing (sampled matrix is store in the current object, accessible via \$netMatrix)

**Method show():** show method

*Usage:*

```
BipartiteSBM_sampler$show(
  type = "Sampler for 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_sampler$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

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

## Arguments

object	an R6 object inheriting from class <code>SBM_fit</code> (like <code>SimpleSBM_fit</code> or <code>BipartiteSBM_fit</code> )
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

## Value

vector or list of parameters.

---

estimateBipartiteSBM *Estimation of Bipartite SBMs*

---

## 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

```
estimateBipartiteSBM(  
  netMat,  
  model = "bernoulli",  
  dimLabels = list(row = "rowLabel", col = "colLabels"),  
  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 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.
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
- "nbBlocksRange"minimal and maximal number or blocks explored

## Value

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

**Examples**

```

### =====
### BIPARTITE BINARY SBM (Bernoulli model)

## 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')

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

### =====
### BIPARTITE POISSON SBM

## 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')

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

### =====
### BIPARTITE GAUSSIAN SBM
## 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')

## Estimation
myBipartiteSBM <-
  estimateBipartiteSBM(mySampler$netMatrix, 'gaussian', estimOptions = list(plot = FALSE))
plot(myBipartiteSBM, 'expected')

```



## 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

```
estimateSimpleSBM(
  netMat,
  model = "bernoulli",
  directed = !isSymmetric(netMat),
  dimLabels = list(row = "rowLabel", col = "colLabels"),
  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'.
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 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.
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
- "nbBlocksRange"minimal and maximal number or blocks explored

## Value

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

**Examples**

```

### =====
### SIMPLE BINARY SBM (Bernoulli model)

## Graph parameters & Sampling
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$netMatrix

## Estimation
mySimpleSBM <-
  estimateSimpleSBM(adjacencyMatrix, 'bernoulli', estimOptions = list(plot = FALSE))
plot(mySimpleSBM, 'data', ordered = FALSE)
plot(mySimpleSBM, 'data')
plot(mySimpleSBM, 'expected', ordered = FALSE)
plot(mySimpleSBM, 'expected')

### =====
### SIMPLE POISSON 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)
mySampler <- sampleSimpleSBM(nbNodes, blockProp, list(mean = means), model = "poisson")
adjacencyMatrix <- mySampler$netMatrix

## Estimation
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$netMatrix, 'gaussian', estimOptions = list(plot = FALSE))
plot(mySimpleSBM, 'data', ordered = FALSE)
plot(mySimpleSBM, 'data')

```

```
plot(mySimpleSBM, 'expected', ordered = FALSE)
plot(mySimpleSBM, 'expected')
```

---

fitted.SBM_fit	<i>Extract model fitted values</i> Extracts fitted values for object with class <code>SBM_fit</code> and children ( <code>SimpleSBM_fit</code> , <code>BipartiteSBM_fit</code> )
----------------	--

---

### Description

Extract model fitted values Extracts fitted values for object with class `SBM_fit` and children (`SimpleSBM_fit`, `BipartiteSBM_fit`)

### Usage

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

### Arguments

object	an R6 object inheriting from class <code>SBM_fit</code> (like <code>SimpleSBM_fit</code> or <code>BipartiteSBM_fit</code> )
...	additional parameters for S3 compatibility. Not used

### Value

a matrix of expected fitted values for each dyad

---

fungus_tree_network	<i>fungus-tree interaction network</i>
---------------------	--

---

### Description

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

### Usage

```
fungus_tree_network
```

### Format

A list with the following entries:

**fungi\_list** list of the fungus species names

**tree\_list** list of the tree species names

**fungus\_tree** binary fungus-tree interactions

**tree\_tree** weighted tree-tree interactions (number of common fungal species two tree species host)

**covar\_tree** covariates associated to pairs of trees (namely genetic, taxonomic and geographic distances)

**Source**

Vacher, Corinne, Dominique Piou, and Marie-Laure Desprez-Loustau. "Architecture of an antagonistic tree/fungus network: the asymmetric influence of past evolutionary history." PloS one 3.3 (2008): e1740.

---

plot.SBM

*SBM Plot*

---

**Description**

basic matrix plot method for SBM object

**Usage**

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

**Arguments**

x	a object inheriting from class SBM
type	character for the type of plot: either 'data' (true connection) or 'expected' (fitted connection). Default to 'data'.
ordered	logical: should the rows and columns be reordered according to the clustering? Default to TRUE.
...	additional parameters for S3 compatibility. Not used

**Details**

Basic matrix plot method for SBM object

**Value**

a ggplot2 object

---

plotMyMatrix	<i>Plot an adjacency or incidence Matrix</i>
--------------	--

---

**Description**

Plot an adjacency or incidence Matrix

**Usage**

```
plotMyMatrix(Mat, dimLabels = list(row = NULL, col = 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)

**Value**

a ggplot object corresponding to the plot

**Examples**

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

---

predict.SBM	<i>Model Predictions</i>
-------------	--------------------------

---

**Description**

Make predictions from an SBM.

**Usage**

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

**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.  
 ... : 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**

```
sampleBipartiteSBM(
  nbNodes,
  blockProp,
  connectParam,
  model = "bernoulli",
  dimLabels = list(row = "rowLabel", col = "colLabels"),
  covariates = list(),
  covariatesParam = numeric(0)
)
```

**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', ...). 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_sampler`

**Examples**

```

### =====
### 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
mySampler <- sampleBipartiteSBM(nbNodes, blockProp, connectParam, model = 'bernoulli')
plot(mySampler)
mySampler$memberships() # sample new memberships
mySampler$incidence() # sample new incidence matrix
plot(mySampler)
hist(mySampler$netMatrix)

### =====
### 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
mySampler <- sampleBipartiteSBM(nbNodes, blockProp, connectParam, model = 'poisson')
plot(mySampler)
hist(mySampler$netMatrix)

### =====
### 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)

## Graph Sampling
mySampler <- sampleBipartiteSBM(nbNodes, blockProp, connectParam, model = 'gaussian')
plot(mySampler)
hist(mySampler$netMatrix)

```

---

sampleSimpleSBM	<i>Sampling of Simple SBMs</i>
-----------------	--------------------------------

---

**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**

```
sampleSimpleSBM(
  nbNodes,
  blockProp,
  connectParam,
  model = "bernoulli",
  directed = FALSE,
  dimLabels = list(row = "rowLabel", col = "colLabels"),
  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**

an object with class `SimpleSBM_sampler`

**Examples**

```
### =====
### 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)
mySampler$Memberships() # sample new memberships
mySampler$Adjacency() # sample new adjacency matrix
plot(mySampler)
hist(mySampler$netMatrix)

### =====
### 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)
hist(mySampler$netMatrix)

### =====
### 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)
hist(mySampler$netMatrix)

```

---

SBM

*R6 virtual class for SBM representation (mother class of Simple and Bipartite SBM fit and sampler)*


---

### Description

R6 virtual class for SBM representation (mother class of Simple and Bipartite SBM fit and sampler)

R6 virtual class for SBM representation (mother class of Simple and Bipartite SBM fit and sampler)

### Active bindings

`dimension` size-2 vector: dimension of the network  
`modelName` character, the family of model for the distribution of the edges  
`dimLabels` vector of characters, the label of each dimension  
`nbCovariates` integer, the number of covariates  
`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  
`covarParam` vector of regression parameters associated with the covariates.  
`covarList` list of matrices of covariates  
`covarArray` the array of covariates  
`covarEffect` effect of covariates  
`netMatrix` the matrix (adjacency or incidence) encoding the network

### Methods

#### Public methods:

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

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

*Usage:*

```
SBM$new(
  model = "",
  dimension = numeric(2),
  dimLabels = vector("list", 2),
  blockProp = numeric(0),
  connectParam = list(mean = matrix()),
  covarParam = numeric(length(covarList)),
  covarList = list()
)
```

*Arguments:*

`model` character describing the type of model  
`dimension` dimension of the network matrix  
`dimLabels` labels of each dimension (in row, in columns)  
`blockProp` parameters for block proportions (vector of list of vectors)  
`connectParam` list of parameters for connectivity

covarParam optional vector of covariates effect  
 covarList optional list of covariates data

**Method plot():** basic matrix plot method for SBM object

*Usage:*

SBM\$plot(type = c("data", "expected"), ordered = TRUE)

*Arguments:*

type character for the type of plot: either 'data' (true connection) or 'expected' (fitted connection). Default to 'data'.

ordered logical: should the rows and columns be reordered according to the clustering? Default to TRUE.

*Returns:* a ggplot2 object

**Method show():** print method

*Usage:*

SBM\$show(type = "Stochastic Block Model")

*Arguments:*

type character to tune the displayed name

**Method print():** print method

*Usage:*

SBM\$print()

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

*Usage:*

SBM\$clone(deep = FALSE)

*Arguments:*

deep Whether to make a deep clone.

---

SBM\_fit

*R6 virtual class for SBM fit (mother class of Simple and Bipartite SBM fit)*

---

## Description

R6 virtual class for SBM fit (mother class of Simple and Bipartite SBM fit)

R6 virtual class for SBM fit (mother class of Simple and Bipartite SBM fit)

## Super class

`sbm::SBM` -> SBM\_fit

**Active bindings**

probMemberships matrix – or list of 2 matrices for Bipartite network – of estimated probabilities for block memberships for all nodes

loglik double: approximation of the log-likelihood (variational lower bound) reached

ICL double: value of the integrated classification log-likelihood

expectation expected values of connection under the currently adjusted model

fitted matrix of predicted value of the network

**Methods****Public methods:**

- [SBM\\_fit\\$new\(\)](#)
- [SBM\\_fit\\$show\(\)](#)
- [SBM\\_fit\\$setModel\(\)](#)
- [SBM\\_fit\\$clone\(\)](#)

**Method** `new()`: constructor for SBM fit

*Usage:*

`SBM_fit$new(data, model, dimLabels, covarList)`

*Arguments:*

`data` the data matrix of the network  
`model` character describing the type of model  
`dimLabels` labels of each dimension (in row, in columns)  
`covarList` optional list of matrices for covariates

**Method** `show()`: print/show method

*Usage:*

`SBM_fit$show(type = "Fit of a Stochastic Block Model")`

*Arguments:*

`type` character to tune the displayed name

**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:*

`SBM_fit$setModel(index)`

*Arguments:*

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

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

*Usage:*

`SBM_fit$clone(deep = FALSE)`

*Arguments:*

`deep` Whether to make a deep clone.

---

 SBM\_sampler

*R6 class for SBM sampler*


---

**Description**

R6 class for SBM sampler

R6 class for SBM sampler

**Super class**

`sbm::SBM` -> SBM\_sampler

**Active bindings**

`variance` variance of each dyad under the current model

**Methods****Public methods:**

- `SBM_sampler$new()`
- `SBM_sampler$show()`
- `SBM_sampler$clone()`

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

*Usage:*

```
SBM_sampler$new(
  model,
  nbNodes,
  blockProp,
  dimLabels,
  connectParam,
  covarParam = numeric(0),
  covarList = list()
)
```

*Arguments:*

`model` character describing the type of model

`nbNodes` number of nodes in the network

`blockProp` parameters for block proportions (vector of list of vectors)

`dimLabels` labels of each dimension (in row, in column)

`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

`covarParam` optional vector of covariates effect

`covarList` optional list of covariates data

**Method show():** print/show method

*Usage:*

```
SBM_sampler$show(type = "Sampler for a Stochastic Block Model")
```

*Arguments:*

type character to tune the displayed name

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

*Usage:*

```
SBM_sampler$clone(deep = FALSE)
```

*Arguments:*

deep Whether to make a deep clone.

---

SimpleSBM\_fit

*R6 Class definition of a Simple SBM fit*


---

**Description**

R6 Class definition of a Simple SBM fit

R6 Class definition of a Simple SBM fit

**Details**

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

**Super classes**

```
sbm::SBM -> sbm::SBM_fit -> SimpleSBM_fit
```

**Active bindings**

nbNodes number of nodes

nbBlocks number of blocks

nbDyads number of dyads (potential edges in the network)

memberships vector of clustering

nbConnectParam number of parameter used for the connectivity

directed is the network directed or not

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$predict()`
- `SimpleSBM_fit$reorder()`
- `SimpleSBM_fit$show()`
- `SimpleSBM_fit$clone()`

**Method** `new()`: constructor for a Simple SBM fit

*Usage:*

```
SimpleSBM_fit$new(
  adjacencyMatrix,
  model,
  directed,
  dimLabels = list(row = "rowLabel", col = "colLabel"),
  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:*

```
SimpleSBM_fit$optimize(
  verbosity = 3,
  plot = FALSE,
  explorFactor = 1.5,
  nbBlocksRange = c(4, Inf),
  nbCores = 2,
  fast = TRUE
)
```

*Arguments:*

`verbosity` integer, the level of verbosity. Default to 3  
`plot` logical, if TRUE plotting is done dynamically on the screen. Default to TRUE  
`explorFactor` double factor for exploring successive model  
`nbBlocksRange` 2-size vector: range of exploration  
`nbCores` integer, the number of cores to use. Default is 2.  
`fast` logical: should approximation be used for Bernoulli model with covariates. Default to TRUE

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

*Usage:*

```
SimpleSBM_fit$predict(covarList = self$covarList)
```

*Arguments:*

covarList a list of covariates. By default, we use the covariates with which the model was estimated

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

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

*Usage:*

```
SimpleSBM_fit$reorder()
```

**Method** show(): show method

*Usage:*

```
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:*

deep Whether to make a deep clone.

---

SimpleSBM\_sampler      *R6 class for Simple SBM sampler*

---

## Description

R6 class for Simple SBM sampler

R6 class for Simple SBM sampler

## Super classes

`sbm::SBM` -> `sbm::SBM_sampler` -> SimpleSBM\_sampler

## Active bindings

nbNodes number of 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

expectation expected values of connection under the current model

directed is the network directed or not



**Methods****Public methods:**

- `SimpleSBM_sampler$new()`
- `SimpleSBM_sampler$rMemberships()`
- `SimpleSBM_sampler$rAdjacency()`
- `SimpleSBM_sampler$show()`
- `SimpleSBM_sampler$clone()`

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

```
SimpleSBM_sampler$new(
  model,
  nbNodes,
  directed,
  blockProp,
  connectParam,
  dimLabels = list(row = "rowLabel", col = "colLabel"),
  covarParam = numeric(0),
  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 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 generate a vector of block indicators*Usage:*

```
SimpleSBM_sampler$rMemberships()
```

*Returns:* nothing (sampled memberships is stored in the current object)

**Method rAdjacency():** a method to sample an adjacency matrix for the current SBM*Usage:*

```
SimpleSBM_sampler$rAdjacency()
```

*Returns:* nothing (sampled adjacency matrix is stored in the current object)

**Method show():** show method*Usage:*

```
SimpleSBM_sampler$show(type = "Sampler for 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_sampler$clone(deep = FALSE)
```

*Arguments:*

deep Whether to make a deep clone.

# Index

## \* datasets

fungus\_tree\_network, [11](#)

BipartiteSBM\_fit, [2](#), [6](#), [11](#)

BipartiteSBM\_sampler, [4](#), [14](#)

coef.SBM, [6](#)

estimateBipartiteSBM, [7](#)

estimateSimpleSBM, [8](#)

fitted.SBM\_fit, [11](#)

fungus\_tree\_network, [11](#)

plot.SBM, [12](#)

plotMyMatrix, [13](#)

predict.SBM, [13](#)

sampleBipartiteSBM, [14](#)

sampleSimpleSBM, [16](#)

SBM, [6](#), [17](#)

sbm::SBM, [2](#), [4](#), [19](#), [21](#), [22](#), [24](#)

sbm::SBM\_fit, [2](#), [22](#)

sbm::SBM\_sampler, [4](#), [24](#)

SBM\_fit, [11](#), [19](#)

SBM\_sampler, [21](#)

SimpleSBM\_fit, [6](#), [11](#), [22](#)

SimpleSBM\_sampler, [16](#), [24](#)