

# Package ‘MLVSBM’

April 2, 2022

**Type** Package

**Title** A Stochastic Block Model for Multilevel Networks

**Version** 0.2.3

**Description** Simulation, inference and clustering of multilevel networks using a Stochastic Block Model framework as described in Chabert-Liddell, Barbillon, Donnet and Lazega (2021) <[doi:10.1016/j.csda.2021.107179](https://doi.org/10.1016/j.csda.2021.107179)>. A multilevel network is defined as the junction of two interaction networks, the upper level or inter-organizational level and the lower level or inter-individual level. The inter-level represents an affiliation relationship.

**License** GPL-3

**Encoding** UTF-8

**URL** <https://github.com/Chabert-Liddell/MLVSBM>

**BugReports** <https://github.com/Chabert-Liddell/MLVSBM/issues>

**RoxygenNote** 7.1.2

**Depends** R (>= 3.5.0)

**Imports** R6, blockmodels, ape, magrittr, cluster

**Suggests** testthat (>= 2.1.0), covr, knitr, rmarkdown, ggplot2 (>= 3.3.2), ggforce, spelling, cowplot, reshape2, dplyr

**VignetteBuilder** knitr

**Language** en-US

**NeedsCompilation** no

**Author** Saint-Clair Chabert-Liddell [aut, cre]  
(<<https://orcid.org/0000-0001-5604-7308>>)

**Maintainer** Saint-Clair Chabert-Liddell <[academic@chabert-liddell.com](mailto:academic@chabert-liddell.com)>

**Repository** CRAN

**Date/Publication** 2022-04-01 23:50:07 UTC

**R topics documented:**

|                         |           |
|-------------------------|-----------|
| ARI                     | 2         |
| build_fold_matrix       | 3         |
| coef.FitMLVSBM          | 3         |
| FitMLVSBM               | 4         |
| FitSBM                  | 8         |
| hierarClust             | 11        |
| merge_clust             | 11        |
| MLVSBM                  | 12        |
| mlvsbm_create_network   | 16        |
| mlvsbm_estimate_network | 17        |
| mlvsbm_log_likelihood   | 18        |
| mlvsbm_simulate_network | 19        |
| plot.FitMLVSBM          | 20        |
| predict.FitMLVSBM       | 21        |
| simulate_adjacency      | 21        |
| simulate_affiliation    | 22        |
| spcClust                | 23        |
| split_clust             | 23        |
| <b>Index</b>            | <b>24</b> |

ARI

*Compare two clustering with the Adjusted Rand Index***Description**

Compare two clustering with the Adjusted Rand Index

**Usage**

```
ARI(x, y)
```

**Arguments**

x                    A vector of integers, the clusters labels  
y                    A vector of integers of the same length as x, the clusters labels

**Value**

A number between 0 (random clustering) and 1 (identical clustering)

**Examples**

```
ARI(x = c(1, 2, 1), y = c(2, 2, 1))
```

---

|                   |              |
|-------------------|--------------|
| build_fold_matrix | <i>Title</i> |
|-------------------|--------------|

---

**Description**

Title

**Usage**

```
build_fold_matrix(X, K)
```

**Arguments**

|   |                                 |
|---|---------------------------------|
| X | An adjacency matrix             |
| K | An integer, the number of folds |

**Value**

A matrix of the same size than X with class integer as coefficient

---

|                |                                   |
|----------------|-----------------------------------|
| coef.FitMLVSBM | <i>Extract model coefficients</i> |
|----------------|-----------------------------------|

---

**Description**

Extracts model coefficients from objects with class [FitMLVSBM](#)

**Usage**

```
## S3 method for class 'FitMLVSBM'  
coef(object, ...)
```

**Arguments**

|        |  |
|--------|--|
| object | an R6 object of class FitMLVSBM                      |
| ...    | additional parameters for S3 compatibility. Not used |

**Value**

List of parameters.

---

 FitMLVSBM

*An R6 Class object, a fitted multilevel network once \$dovem() is done*


---

### Description

An R6 Class object, a fitted multilevel network once \$dovem() is done

An R6 Class object, a fitted multilevel network once \$dovem() is done

### Public fields

vbound The vector of variational bound for monitoring convergence

### Active bindings

affiliation\_matrix Get the affiliation matrix

adjacency\_matrix Get the list of adjacency matrices

nb\_nodes Get the list of the number of nodes

nb\_clusters Get the list of the number of blocks

parameters Get the list of the model parameters

membership Get the list of the variational parameters

independent Are the levels independent?

distribution Emission distribution of each level

directed Are the levels directed?

entropy Get the entropy of the model

bound Get the variational bound of the model

df\_mixture Get the degrees of freedom of the mixture parameters

df\_connect Get the degrees of freedom of the connection parameters

connect Get the number of possible observed connections

ICL Get the ICL model selection criterion of the model

full\_penalty Get the penalty used to compute the ICL

Z Get the list of block memberships (vector form)

X\_hat Get the list of the matrices of probability connection predictions

map Get the list of block memberships (matrix form)

penalty Get the ICL penalty

likelihood Compute the likelihood of both levels

complete\_likelihood Get the complete likelihood of the model

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

- `FitMLVSBM$new()`
- `FitMLVSBM$update_alpha()`
- `FitMLVSBM$update_pi()`
- `FitMLVSBM$update_gamma()`
- `FitMLVSBM$init_clustering()`
- `FitMLVSBM$clear()`
- `FitMLVSBM$m_step()`
- `FitMLVSBM$ve_step()`
- `FitMLVSBM$do_vem()`
- `FitMLVSBM$permute_empty_class()`
- `FitMLVSBM$plot()`
- `FitMLVSBM$show()`
- `FitMLVSBM$print()`
- `FitMLVSBM$clone()`

**Method** `new()`: Constructor for the FitMLVSBM class

*Usage:*

```
FitMLVSBM$new(
  Q = list(I = 1, O = 1),
  A = NA,
  X = NA,
  M = list(I = NA, O = NA),
  directed = NA,
  distribution = list("bernoulli", "bernoulli"),
  independent = FALSE
)
```

*Arguments:*

Q List of number of blocks  
 A Affiliation matrix  
 X List of adjacency matrices  
 M List of Mask matrices  
 directed List of boolean  
 distribution List of string  
 independent Boolean

*Returns:* A FitMLVSBM object

**Method** `update_alpha()`: Update the connection parameters for the M step

*Usage:*

```
FitMLVSBM$update_alpha(safeguard = 2 * .Machine$double.eps)
```

*Arguments:*

safeguard Parameter live in a compact [safeguard, 1-safeguard]

**Method** `update_pi()`: Update the upper level mixture parameter for the M step

*Usage:*

```
FitMLVSBM$update_pi(safeguard = 0.001)
```

*Arguments:*

`safeguard` Parameter live in a compact [`safeguard`, 1-`safeguard`]

**Method** `update_gamma()`: Update the lower level mixture parameter for the M step

*Usage:*

```
FitMLVSBM$update_gamma(safeguard = 1e-06)
```

*Arguments:*

`safeguard` Parameter live in a compact [`safeguard`, 1-`safeguard`]

**Method** `init_clustering()`: `init_clustering` Initial clustering for VEM algorithm

*Usage:*

```
FitMLVSBM$init_clustering(
  safeguard = 2 * .Machine$double.eps,
  method = "hierarchical",
  Z = NULL
)
```

*Arguments:*

`safeguard` Parameter live in a compact [`safeguard`, 1-`safeguard`]

`method` Algorithm used to initiate the clustering, either "spectral", "hierarchical" or "merge\_split"  
(if Z is provided)

`Z` Initial clustering if provided

**Method** `clear()`: Reset all parameters

*Usage:*

```
FitMLVSBM$clear()
```

**Method** `m_step()`: `m_step` Compute the M step of the VEM algorithm

*Usage:*

```
FitMLVSBM$m_step(safeguard = 1e-06)
```

*Arguments:*

`safeguard` Parameter live in a compact [`safeguard`, 1-`safeguard`]

**Method** `ve_step()`: Compute the VE step of the VEM algorithm

*Usage:*

```
FitMLVSBM$ve_step(threshold = 1e-06, fixPointIter = 10, safeguard = 1e-06)
```

*Arguments:*

`threshold` The convergence threshold

`fixPointIter` The maximum number of fixed point iterations

`safeguard` Parameter live in a compact [`safeguard`, 1-`safeguard`]

**Method** `do_vem()`: Launch a Variational EM algorithm

*Usage:*

```
FitMLVSBM$do_vem(
  init = "hierarchical",
  threshold = 1e-06,
  maxIter = 1000,
  fixPointIter = 100,
  safeguard = 1e-06,
  Z = NULL
)
```

*Arguments:*

`init` The method for `self$init_clustering`  
`threshold` The convergence threshold  
`maxIter` The max number of VEM iterations  
`fixPointIter` The max number of fixed point iterations for VE step  
`safeguard` Parameter live in a compact [`safeguard`, `1-safeguard`]  
`Z` Initial clustering if provided

**Method** `permute_empty_class()`: `permute_empty_class` Put empty blocks numbers at the end

*Usage:*

```
FitMLVSBM$permute_empty_class()
```

**Method** `plot()`: Plot of FitMLVSBM objects

*Usage:*

```
FitMLVSBM$plot(type = c("matrix"), ...)
```

*Arguments:*

`type` A string for the type of plot, just "matrix" for now

*Returns:* a ggplot2 object

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

*Usage:*

```
FitMLVSBM$show(type = "Multilevel Stochastic Block Model")
```

*Arguments:*

`type` character to tune the displayed name

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

*Usage:*

```
FitMLVSBM$print()
```

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

*Usage:*

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

*Arguments:*

`deep` Whether to make a deep clone.

**Description**

a fitted level of a unilevel network once `$do_vem()` is done

**Public fields**

`vbound` vector of variational bound for convergence monitoring

**Active bindings**

`adjacency` Get the adjacency matrix

`mask` Get the mask matrix for dealing with NA

`nb_nodes` Get the number of nodes of the level

`nb_clusters` Get the number of blocks

`distribution` Get the distribution used for the connections

`directed` Get if the level is directed or not

`mixture_parameter` Access the block proportions

`connectivity_parameter` Access the connectivity matrix

`membership` Access the variational parameters

`entropy` Get the entropy of the model

`bound` Get the variational bound of the model

`df_mixture` Get the degree of freedom of the block proportion

`df_connect` Get the degree of freedom of the connection parameters

`connect` Get the number of observed dyads

`ICL` Get the ICL model selection criterion

`penalty` Get the penalty used for computing the ICL

`Z` Access the vector of block membership (clustering)

`X_hat` Get the connection probability matrix

`X_likelihood` adjacency part of the log likelihood

`Z_likelihood` block part of the log likelihood

`likelihood` complete log likelihood



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

- `FitSBM$new()`
- `FitSBM$update_alpha()`
- `FitSBM$update_pi()`
- `FitSBM$init_clustering()`
- `FitSBM$m_step()`
- `FitSBM$ve_step()`
- `FitSBM$do_vem()`
- `FitSBM$permute_empty_class()`
- `FitSBM$clear()`
- `FitSBM$clone()`

**Method** `new()`: Constructor for FitSBM R6 class

*Usage:*

```
FitSBM$new(
  Q = 1,
  X = NULL,
  M = NULL,
  directed = FALSE,
  distribution = "bernoulli"
)
```

*Arguments:*

Q Number of blocks  
 X Adjacency matrix  
 M Mask matrix  
 directed boolean  
 distribution string (only "bernoulli")

*Returns:* A new FitSBM object

**Method** `update_alpha()`: Update the connection parameter for the M step

*Usage:*

```
FitSBM$update_alpha(safeguard = 1e-06)
```

*Arguments:*

safeguard Parameter live in a compact [safeguard, 1-safeguard]

**Method** `update_pi()`: Update the upper level mixture parameter for the M step

*Usage:*

```
FitSBM$update_pi(safeguard = 1e-06)
```

*Arguments:*

safeguard Parameter live in a compact [safeguard, 1-safeguard]

**Method** `init_clustering()`: `init_clustering` Initial clustering for VEM algorithm

*Usage:*

```
FitSBM$init_clustering(safeguard = 1e-06, method = "hierarchical", Z = NULL)
```

*Arguments:*

safeguard Parameter live in a compact [safeguard, 1-safeguard]

method Algorithm used to initiate the clustering, either "spectral", "hierarchical" or "merge\_split"  
(if Z is provided)

Z Initial clustering if provided

**Method** `m_step()`: `m_step` Compute the M step of the VEM algorithm

*Usage:*

```
FitSBM$m_step(safeguard = 1e-06)
```

*Arguments:*

safeguard Parameter live in a compact [safeguard, 1-safeguard]

**Method** `ve_step()`: Compute the VE step of the VEM algorithm

*Usage:*

```
FitSBM$ve_step(threshold = 1e-06, fixPointIter = 100, safeguard = 1e-06)
```

*Arguments:*

threshold The convergence threshold

fixPointIter The maximum number of fixed point iterations

safeguard Parameter live in a compact [safeguard, 1-safeguard]

**Method** `do_vem()`: Launch a Variational EM algorithm

*Usage:*

```
FitSBM$do_vem(
  init = "hierarchical",
  threshold = 1e-06,
  maxIter = 1000,
  fixPointIter = 100,
  safeguard = 1e-06,
  Z = NULL
)
```

*Arguments:*

init The method for `self$init_clustering`

threshold The convergence threshold

maxIter The max number of VEM iterations

fixPointIter The max number of fixed point iterations for VE step

safeguard Parameter live in a compact [safeguard, 1-safeguard]

Z Initial clustering if provided

**Method** `permute_empty_class()`: `permute_empty_class` Put empty blocks numbers at the end

*Usage:*

```
FitSBM$permute_empty_class()
```

**Method** `clear()`: Reset all parameters

*Usage:*

`FitSBM$clear()`

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

*Usage:*

`FitSBM$clone(deep = FALSE)`

*Arguments:*

`deep` Whether to make a deep clone.

---

hierarClust                      *Perform a Hierarchical Clustering*

---

### Description

Perform a Hierarchical Clustering

### Usage

`hierarClust(X, K)`

### Arguments

`X`                      An Adjacency Matrix  
`K`                      the number of wanted clusters

### Value

A vector : The clusters labels

---

merge\_clust                      *Merge a list of clusters*

---

### Description

Merge a list of clusters

### Usage

`merge_clust(Z, Q)`

### Arguments

`Z`                      a vector of cluster memberships  
`Q`                      the number of original clusters

**Value**

A list of  $Q(Q-1)/2$  clustering of  $Q-1$  clusters

---

MLVSBM

*R6Class for multilevel object*

---

**Description**

Store all simulation parameters and list of fittedmodels. Methods for global inference and model selection are included.

**Active bindings**

nb\_nodes List of the umber of nodes for each levels  
simulation\_parameters List of parameters of the MLVSBM  
affiliation\_matrix Access the affiliation matrix  
adjacency\_matrix Access the list of adjacency\_matrix  
memberships Access the list of the clusterings  
fittedmodels Get the list of selected fitted FitMLVSBM objects  
ICL A summary table of selected fitted models and ICL model selection criterion  
ICL\_sbm Summary table of ICL by levels  
tmp\_fittedmodels A list of all fitted FitMLVSBM objects  
fittedmodels\_sbm A list of selected fitted FitSBM objects of each levels  
max\_clusters Access the list of maximum model size  
min\_clusters Access the list of minimum model size  
directed Access the list of boolean for levels direction  
directed Access the list of the distribution used for each levels

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

- `MLVSBM$estimate_level()`
- `MLVSBM$estimate_sbm_neighbours()`
- `MLVSBM$estimate_sbm_from_neighbours()`
- `MLVSBM$estimate_sbm()`
- `MLVSBM$mceestimate()`
- `MLVSBM$estimate_from_neighbours()`
- `MLVSBM$estimate_neighbours()`
- `MLVSBM$merge_split_membership()`
- `MLVSBM$mc_ms_estimate()`
- `MLVSBM$estimate_one()`

- `MLVSBM$estimate_all_bm()`
- `MLVSBM$new()`
- `MLVSBM$findmodel()`
- `MLVSBM$clearmodels()`
- `MLVSBM$addmodel()`
- `MLVSBM$simulate()`
- `MLVSBM$clone()`

**Method** `estimate_level()`:

*Usage:*

```
MLVSBM$estimate_level(
  level = "lower",
  Q_min = 1,
  Q_max = 10,
  Z = NULL,
  init = "hierarchical",
  depth = 1,
  nb_cores = NULL
)
```

**Method** `estimate_sbm_neighbours()`:

*Usage:*

```
MLVSBM$estimate_sbm_neighbours(
  level = "lower",
  Q = NULL,
  Q_min = 1,
  Q_max = 10,
  fit = NULL,
  nb_cores = NULL,
  init = NULL
)
```

**Method** `estimate_sbm_from_neighbours()`:

*Usage:*

```
MLVSBM$estimate_sbm_from_neighbours(
  level = "lower",
  Q = NULL,
  fits = NULL,
  nb_cores = NULL
)
```

**Method** `estimate_sbm()`:

*Usage:*

```
MLVSBM$estimate_sbm(level = "lower", Q = Q, Z = NULL, init = "hierarchical")
```

**Method** `mceestimate()`:

*Usage:*

```
MLVSBM$mceestimate(Q, Z = NULL, init = "hierarchical", independent = FALSE)
```

**Method** estimate\_from\_neighbours():

*Usage:*

```
MLVSBM$estimate_from_neighbours(
  Q,
  models = NULL,
  independent = FALSE,
  nb_cores = nb_cores
)
```

**Method** estimate\_neighbours():

*Usage:*

```
MLVSBM$estimate_neighbours(
  Q,
  fit = NULL,
  independent = independent,
  nb_cores = NULL
)
```

**Method** merge\_split\_membership():

*Usage:*

```
MLVSBM$merge_split_membership(
  fitted = private$fitted[[length(private$fitted)]]
)
```

**Method** mc\_ms\_estimate():

*Usage:*

```
MLVSBM$mc_ms_estimate(Z = NA, independent = FALSE, nb_cores = NULL)
```

**Method** estimate\_one():

*Usage:*

```
MLVSBM$estimate_one(
  Q,
  Z = NULL,
  independent = FALSE,
  init = "hierarchical",
  nb_cores = NULL
)
```

**Method** estimate\_all\_bm():

*Usage:*

```
MLVSBM$estimate_all_bm(
  Q = NULL,
  Z = NULL,
  independent = FALSE,
  clear = TRUE,
  nb_cores = NULL
)
```

**Method** `new()`: Constructor for R6 class MLVSBM

*Usage:*

```
MLVSBM$new(
  n = NULL,
  X = NULL,
  A = NULL,
  Z = NULL,
  directed = NULL,
  sim_param = NULL,
  distribution = list("bernoulli", "bernoulli")
)
```

*Arguments:*

`n` A list of size 2, the number of nodes

`X` A list of 2 adjacency matrices

`A` The affiliation matrix

`Z` A list of 2 vectors, the blocks membership

`directed` A list of 2 booleans

`sim_param` A list of MLVSBM parameters for simulating networks

`distribution` The distributions of the interactions ("bernoulli")

*Returns:* A MLVSBM object

**Method** `findmodel()`: Find a fitted model of a given size

*Usage:*

```
MLVSBM$findmodel(nb_clusters = NA, fit = NA)
```

*Arguments:*

`nb_clusters` A list of the size of the model

`fit` if `fit = "best"` return the best model according to the ICL

*Returns:* A FitMLVSBM object

**Method** `clearmodels()`: delete all fitted models

*Usage:*

```
MLVSBM$clearmodels()
```

**Method** `addmodel()`: Added a FitMLVSBM object to the list of fitted model

*Usage:*

```
MLVSBM$addmodel(fit)
```

*Arguments:*

`fit` The FitMLVSBM object to be added

**Method** `simulate()`:

*Usage:*

```
MLVSBM$simulate()
```

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

*Usage:*

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

*Arguments:*

deep Whether to make a deep clone.

---

mlvsbm\_create\_network *Create a MLVSBM object from observed data*

---

**Description**

Create a MLVSBM object from observed data

**Usage**

```
mlvsbm_create_network(  
  X,  
  A,  
  directed = NULL,  
  distribution = list("bernoulli", "bernoulli")  
)
```

**Arguments**

|              |   |
|--------------|---|
| X            | A list of 2 squares binary matrices, the first one being the individual or lower level the second one being the organizational or upper level |
| A            | A matrix the affiliation matrix with individuals in rows and organizations in columns   |
| directed     | A list of 2 boolean are the upper and lower level directed or not. Default will check if the matrix are symmetric or not.                     |
| distribution | A list for the distribution of X, only "bernoulli" is implemented   |

**Value**

An unfitted MLVSBM object corresponding to the multilevel network

**Examples**

```
ind_adj <- matrix(stats::rbinom(n = 10**2, size = 1, prob = .2),  
                 nrow = 10, ncol = 10)  
org_adj <- matrix(stats::rbinom(n = 10**2, size = 1, prob = .3),  
                 nrow = 10, ncol = 10)  
affiliation <- diag(1, 10)  
my_mlvsbm <- mlvsbm_create_network(X = list(I = ind_adj, O = org_adj),  
                                  directed = list(I = FALSE, O = FALSE),  
                                  A = affiliation)
```



---

mlvsbm\_estimate\_network

*Infer a multilevel network (MLVSBM object), the original object is modified*

---

## Description

The inference use a greedy algorithm to navigate between model size. For a given model size, the inference is done via a variational EM algorithm. The returned model is the one with the highest ICL criterion among all visited models.

By default the algorithm fits a single level SBM for each level, before inferring the multilevel network. This step can be skipped by specifying an initial clustering with the `init_clustering`. Also, a given model size can be force by setting the parameters `nb_clusters` to a given value.

## Usage

```
mlvsbm_estimate_network(  
  mlv,  
  nb_clusters = NULL,  
  init_clustering = NULL,  
  nb_cores = NULL,  
  init_method = "hierarchical"  
)
```

## Arguments

|                              |  |
|------------------------------|--|
| <code>mlv</code>             | A MLVSBM object, the network to be inferred.   |
| <code>nb_clusters</code>     | A list of 2 integers, the model size. If left to NULL, the algorithm will navigate freely. Otherwise it will navigate between the specified model size and its neighbors.      |
| <code>init_clustering</code> | A list of 2 vectors of integers of the same length as the number of node of each level. If specified, the algorithm will start from this clustering, then navigate freely.     |
| <code>nb_cores</code>        | An integer, the number of cores to use. Default to 1 for Windows and <code>detectCores()/2</code> for Linux and MacOS  |
| <code>init_method</code>     | One of "hierarchical" (the default) or "spectral", "spectral" might be more efficient but can lead to some numeric errors. Not used when <code>int_clustering</code> is given. |

## Value

A FitMLVSBM object, the best inference of the network

**Examples**

```
my_mlvsbm <- MLVSBM::mlvsbm_simulate_network(
  n = list(I = 10, O = 20), # Number of nodes for the lower level and the upper level
  Q = list(I = 2, O = 2), # Number of blocks for the lower level and the upper level
  pi = c(.3, .7), # Block proportion for the upper level, must sum to one
  gamma = matrix(c(.9, .2, # Block proportion for the lower level,
    .1, .8), # each column must sum to one
    nrow = 2, ncol = 2, byrow = TRUE),
  alpha = list(I = matrix(c(.8, .2,
    .2, .1),
    nrow = 2, ncol = 2, byrow = TRUE), # Connection matrix
  O = matrix(c(.99, .3,
    .3, .1),
    nrow = 2, ncol = 2, byrow = TRUE)), # between blocks
  directed = list(I = FALSE, O = FALSE), # Are the upper and lower level directed or not ?
  affiliation = "preferential") # How the affiliation matrix is generated
fit <- MLVSBM::mlvsbm_estimate_network(mlv = my_mlvsbm, nb_cores = 1)
```

---

`mlvsbm_log_likelihood` *Compute the complete log likelihood of a multilevel network for a given clustering of the nodes.*

---

**Description**

This function is useful to compute the likelihood for clusters obtained by different methods.

**Usage**

```
mlvsbm_log_likelihood(mlv, clustering)
```

**Arguments**

|                         |   |
|-------------------------|---|
| <code>mlv</code>        | A MLVSBM object, the network data   |
| <code>clustering</code> | A list of 2 vectors of integers of the same length as the number of node of each level. |

**Value**

A numeric, the log likelihood of the multilevel network for the given clustering.

**Examples**

```
my_mlvsbm <- MLVSBM::mlvsbm_simulate_network(
  n = list(I = 40, O = 20), # Number of nodes for the lower level and the upper level
  Q = list(I = 2, O = 2), # Number of blocks for the lower level and the upper level
  pi = c(.3, .7), # Block proportion for the upper level, must sum to one
  gamma = matrix(c(.9, .2, # Block proportion for the lower level,
    .1, .8), # each column must sum to one
    nrow = 2, ncol = 2, byrow = TRUE),
```

```

alpha = list(I = matrix(c(.8, .2,
                          .2, .1),
                      nrow = 2, ncol = 2, byrow = TRUE), # Connection matrix
            O = matrix(c(.99, .3,
                          .3, .1),
                      nrow = 2, ncol = 2, byrow = TRUE)), # between blocks
directed = list(I = FALSE, O = FALSE), # Are the upper and lower level directed or not ?
affiliation = "preferential" # How the affiliation matrix is generated
mlvsbm_log_likelihood(mlv = my_mlvsbm, clustering = my_mlvsbm$memberships)

```

---

```
mlvsbm_simulate_network
```

*Create a simulated multilevel network (MLVSBM object)*

---

## Description

Create a simulated multilevel network (MLVSBM object)

## Usage

```

mlvsbm_simulate_network(
  n,
  Q,
  pi,
  gamma,
  alpha,
  directed,
  affiliation = "uniform",
  distribution = list("bernoulli", "bernoulli"),
  no_empty_org = FALSE,
  no_isolated_node = FALSE
)

```

## Arguments

|          |  |
|----------|--|
| n        | A list of 2 positive integers, the number of individuals and organizations.  |
| Q        | A list of 2 positive integers, the number of clusters of individuals and organizations.  |
| pi       | A vector of probabilities of length $Q_O$ , the mixture parameter for the organizations.   |
| gamma    | A $Q_I \times Q_O$ matrix with each column summing to one, the mixture parameters for the individuals  |
| alpha    | A list of 2 matrices, a $Q_I \times Q_I$ matrix giving the connectivity probabilities of the individuals and a $Q_O \times Q_O$ matrix giving the connectivity probabilities of the organizations. |
| directed | A list of 2 logical. Is the individual level a directed network ? Is the inter-organizational level a directed network?  |

|                  |   |
|------------------|---|
| affiliation      | The distribution under which the affiliation matrix is simulated in <code>c("uniform", "preferential")</code> .                     |
| distribution     | A list for the distribution of X, only "bernoulli" is implemented.  |
| no_empty_org     | A logical with FALSE as default, should every organizations have at least one affiliated individual? Needs to have $n_I \geq n_O$ . |
| no_isolated_node | A logical, if TRUE then the network is simulated again until all nodes are connected.   |

**Value**

An MLVSBM object, a simulated multilevel network with levels, affiliations and memberships.

**Examples**

```
my_mlvsbm <- MLVSBM::mlvsbm_simulate_network(
  n = list(I = 10, O = 20), # Number of nodes for the lower level and the upper level
  Q = list(I = 2, O = 2), # Number of blocks for the lower level and the upper level
  pi = c(.3, .7), # Block proportion for the upper level, must sum to one
  gamma = matrix(c(.9, .2, # Block proportion for the lower level,
                  .1, .8), # each column must sum to one
                nrow = 2, ncol = 2, byrow = TRUE),
  alpha = list(I = matrix(c(.8, .2,
                           .2, .1),
                          nrow = 2, ncol = 2, byrow = TRUE), # Connection matrix
              O = matrix(c(.99, .3,
                           .3, .1),
                          nrow = 2, ncol = 2, byrow = TRUE)), # between blocks
  directed = list(I = FALSE, O = FALSE)) # Are the upper and lower level directed
```

---

plot.FitMLVSBM

*Multilevel SBM Plot*

---

**Description**

basic matrix plot method for a FitMLVSBM object

**Usage**

```
## S3 method for class 'FitMLVSBM'
plot(x, type = c("matrix"), ...)
```

**Arguments**

|      |   |
|------|---|
| x    | an R6 object of class <code>FitMLVSBM</code>  |
| type | A string for the type of plot, just "matrix" for now  |
| ...  | additional parameters. block ordering with <code>order = c("affiliation", "degree", "natural")</code> |

**Details**

Basic matrix plot method for a FitMLVSBM object

**Value**

a ggplot2 object

---

predict.FitMLVSBM      *Model Predictions*

---

**Description**

Make predictions from an SBM.

**Usage**

```
## S3 method for class 'FitMLVSBM'
predict(object, ...)
```

**Arguments**

object            an R6 object of class `FitMLVSBM`  
 ...              additional parameters for S3 compatibility. Not used

**Value**

A list with the following entries:

**dyads** A list of matrix with the probability of each dyads

**nodes** A list of vectors with the clustering of each nodes

---

simulate\_adjacency      *Simulation an adjacency matrix*

---

**Description**

Simulation an adjacency matrix

**Usage**

```
simulate_adjacency(  
  Z,  
  n,  
  alpha,  
  directed,  
  distribution = "bernoulli",  
  no_isolated_node = FALSE  
)
```

**Arguments**

|                               |  |
|-------------------------------|--|
| <code>Z</code>                | A vector of integer of size $n$ , the label                    |
| <code>n</code>                | An integer, the number of rows or columns of the matrix        |
| <code>alpha</code>            | A $\max(Z) \times \max(Z)$ matrix, the connectivity parameters |
| <code>directed</code>         | A boolean, Is the network directed or not ?                    |
| <code>distribution</code>     | The distribution of the indices: only "bernoulli"              |
| <code>no_isolated_node</code> | A boolean, may row and column of adjacency matrices sum to 0   |

**Value**

A  $n \times n$  adjacency matrix

---

`simulate_affiliation` *Simulate of matrix of affiliation*

---

**Description**

Simulate of matrix of affiliation

**Usage**

```
simulate_affiliation(n, m, affiliation = "uniform", no_empty_org = FALSE)
```

**Arguments**

|                           |  |
|---------------------------|--|
| <code>n</code>            | An integer, the number of individuals                                  |
| <code>m</code>            | An integer, the number of organizations                                |
| <code>affiliation</code>  | The type of affiliation between c("uniform", "preferential")           |
| <code>no_empty_org</code> | A Boolean. Force all columns to have at least a 1. Must have $n > m$ . |

**Value**

A  $n \times m$  affiliation matrix, with a unique 1 on each rows

---

|          |                                      |
|----------|--------------------------------------|
| spcClust | <i>Perform a spectral clustering</i> |
|----------|--------------------------------------|

---

**Description**

Perform a spectral clustering

**Usage**

```
spcClust(X, K)
```

**Arguments**

|   |                        |
|---|------------------------|
| X | an Adjacency matrix    |
| K | the number of clusters |

**Value**

A vector : The clusters labels

---

|             |                                 |
|-------------|---------------------------------|
| split_clust | <i>Merge a list of clusters</i> |
|-------------|---------------------------------|

---

**Description**

Merge a list of clusters

**Usage**

```
split_clust(X, Z, Q)
```

**Arguments**

|   |                                 |
|---|---------------------------------|
| X | an adjacency matrix             |
| Z | a vector of cluster memberships |
| Q | The number of maximal clusters  |

**Value**

A list of Q clustering of Q+1 clusters

# Index

ARI, [2](#)

build\_fold\_matrix, [3](#)

coef.FitMLVSBM, [3](#)

FitMLVSBM, [3](#), [4](#), [20](#), [21](#)

FitSBM, [8](#)

hierarClust, [11](#)

merge\_clust, [11](#)

MLVSBM, [12](#)

mlvsbm\_create\_network, [16](#)

mlvsbm\_estimate\_network, [17](#)

mlvsbm\_log\_likelihood, [18](#)

mlvsbm\_simulate\_network, [19](#)

plot.FitMLVSBM, [20](#)

predict.FitMLVSBM, [21](#)

simulate\_adjacency, [21](#)

simulate\_affiliation, [22](#)

spcClust, [23](#)

split\_clust, [23](#)