# Package 'shinySbm'

July 17, 2026

**Title** 'shiny' Application to Use the Stochastic Block Model

**Version** 0.2.1

**Description** A 'shiny' interface for a simpler use of the 'sbm' R package. It also contains useful functions to easily explore the 'sbm' package results. With this package you should be able to use the stochastic block model without any knowledge in R, get automatic reports and nice visuals, as well as learning the basic functions of 'sbm'.

**License** MIT + file LICENSE

**URL** https://shinysbm-theodore-vanrenterghem-b12616c23cfbfb3f0fe520178bcb95a.pages-forge.inrae.fr/, https://cran.r-project.org/package=shinySbm

**BugReports** https://github.com/Jo-Theo/shinySbm/issues

**Depends** R (>= 3.5.0), sbm

**Imports** colourpicker, config (>= 0.3.1), data.table, dplyr, DT, flextable, fresh, ggplot2, golem (>= 0.3.5), grDevices, magrittr, parallel, patchwork, purrr, rmarkdown, shiny (>= 1.7.2), shinyalert, shinydashboard, stats, stringr, utils, visNetwork

**Suggests** htmltools, knitr, spelling, testthat (>= 3.0.0)

**VignetteBuilder** knitr

**Config/roxygen2/version** 8.0.0

**Config/testthat/edition** 3

**Encoding** UTF-8

**Language** en-US

**LazyData** true

**NeedsCompilation** no

**RoxygenNote** 8.0.0

**Author** Theodore Vanrenterghem [aut], Julie Aubert [aut, cre] (ORCID: <https://orcid.org/0000-0001-5203-5748>),

Saint-Clair Chabert-Liddell [aut] (ORCID:
<https://orcid.org/0000-0001-5604-7308>),
großBM team [ctb],
INRAE [cph] (ROR: <https://ror.org/003vg9w96>)

**Maintainer** Julie Aubert <julie.aubert@inrae.fr>

**Repository** CRAN

**Date/Publication** 2026-07-17 08:30:02 UTC

# Contents

FungusTreeNetwork . . . . . 2
get_adjacency . . . . . 3
get_adjacency.default . . . . . 4
get_block . . . . . 5
get_block.BipartiteSBM_fit . . . . . 7
get_block.SimpleSBM_fit . . . . . 8
get_flextable . . . . . 9
plotSbm . . . . . 10
plotSbm.BipartiteSBM_fit . . . . . 12
plotSbm.default . . . . . 13
plotSbm.matrix . . . . . 14
plotSbm.SimpleSBM_fit . . . . . 16
shinySbmApp . . . . . 17
visSbm . . . . . 18
visSbm.BipartiteSBM_fit . . . . . 20
visSbm.default . . . . . 22
visSbm.SimpleSBM_fit . . . . . 23

**Index** **25**

---

`FungusTreeNetwork` *FungusTreeNetwork*

---

## Description

fungus-tree interaction network

This data set provides information about 154 fungi sampled on 51 tree species. Composed of nodes and edges lists build based on **sbm** data package.

## Usage

```
FungusTreeNetwork
```

### Format

A list of two elements, networks and sbmResults.

networks is itself a list with the following entries:

**tee_names:** (character) tree names

**fungus_names:** (character) fungus names

**tree_tree:** a list describing the unipartite tree_tree networks with nodes (data.frame), edges (data.frame) and type (character specifying the type of the network)

**fungus_tree:** a list describing the bipartite fungus_tree networks with nodes (data.frame), edges (data.frame) and type (character)

sbmResults is a list with the following entries:

**tree_tree** Results of `sbm::estimateSimpleSBM` applied on tree_tree data with a Poisson model.

**fungus_tree** Results of `sbm::estimateBipartiteSBM` applied on fungus_tree data with a Bernoulli model.

### 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.

---

## `get_adjacency` *get_adjacency*

---

### Description

A fct that build an adjacency matrix from a list of edges.

### Usage

```
get_adjacency(edges, type = c("unipartite", "bipartite"), directed = FALSE)
```

### Arguments

| | |
|---|---|
| `edges` | Can be a table which is a list pair of nodes (nodes ids are one the two first columns) a numerical third column can be associated will be the connections values. |
| `type` | network type can be ‘’bipartite’‘ or ‘’unipartite’‘ |
| `directed` | whether or not connections are directed (‘TRUE‘) or symmetrical (‘FALSE‘) (default is set to ‘TRUE‘) |

### Value

an adjacency/incidence matrix (data.frame) representing the network

## Examples

```
# For unipartite network
data_uni <- FungusTreeNetwork$networks$tree_tree

# If the network is symmetric:
my_mat <- get_adjacency(data_uni$edges,
  type = data_uni$type,
  directed = FALSE
)
# If the network is directed:
my_mat <- get_adjacency(data_uni$edges,
  type = data_uni$type,
  directed = TRUE
)

# For bipartite network
data_bi <- FungusTreeNetwork$networks$fungus_tree

my_mat <- get_adjacency(data_bi$edges, type = data_bi$type)

# In any case with a 2 columns data.frames the network is considered binary and each line is a 1.
binary_net <- FungusTreeNetwork$fungus_tree$edges[, -3]

my_mat <- get_adjacency(binary_net, type = data_bi$type)
```

---

`get_adjacency.default` *get_adjacency.default*

---

## Description

A fct that build an adjacency matrix from a list of edges

## Usage

```
## Default S3 method:
get_adjacency(edges, type = c("unipartite", "bipartite"), directed = FALSE)
```

## Arguments

| | |
|---|---|
| `edges` | Can be a table which is a list pair of nodes (nodes ids are one the two first columns) a numerical third column can be associated will be the connections values. |
| `type` | network type can be ‘’bipartite’‘ or ‘’unipartite’‘ |
| `directed` | whether or not connections are directed (‘TRUE‘) or symmetrical (‘FALSE‘) (default is set to ‘TRUE‘) |

## Value

an adjacency/incidence matrix (data.frame) representing the network

## Examples

```
# For unipartite network
data_uni <- FungusTreeNetwork$networks$tree_tree

# If the network is symmetric:
my_mat <- get_adjacency(data_uni$edges,
  type = data_uni$type,
  directed = FALSE
)
# If the network is directed:
my_mat <- get_adjacency(data_uni$edges,
  type = data_uni$type,
  directed = TRUE
)

# For bipartite network
data_bi <- FungusTreeNetwork$networks$fungus_tree

my_mat <- get_adjacency(data_bi$edges, type = data_bi$type)

# In any case with a 2 columns data.frames the network is considered binary and each line is a 1.
binary_net <- FungusTreeNetwork$fungus_tree$edges[, -3]

my_mat <- get_adjacency(binary_net, type = data_bi$type)
```

---

# get_block *get_block generic*

---

## Description

A fct that return blocks attribution or probabilities for each nodes in a Sbm fit from the **sbm** package.

## Usage

```
get_block(
  x,
  labels = "default",
  node_names = NULL,
  attribution = TRUE,
  proportion = FALSE
)
```

## Arguments

| | |
|---|---|
| x | Sbm model of class 'BipartiteSBM_fit', 'SimpleSBM_fit'. |
| labels | labels for nodes. If it's simple sbm it should be a single character ("default" -> c("nodes")). If sbm is bipartite a named character (names are row and col) ("default" -> c(row = 'row', col = 'col')). |
| node_names | **"bipartite case":** named list ("row","col"), row is a character vector containing names of nodes in rows, and respectively for columns<br>**"unipartite case":** character: node names |
| attribution | Boolean indicating whether or not the produced tables should contain a block attribution column. This column shows the block in which each nodes is the most likely to be. |
| proportion | Boolean indicating whether or not the produced tables should contain the probabilities to belong in each blocks. These columns shows for every nodes and every blocks the probabilities that the node belong to the block. |

## Value

**"bipartite case":** A list containing two data.frames with block attributions and/or proportions one for the row blocks and one for the column blocks

**"unipartite case":** A data.frame with block attributions and/or proportions

## Examples

```
# my_sbm_bi <- sbm::estimateBipartiteSBM(sbm::fungusTreeNetwork$fungus_tree,
#                                        model = 'bernoulli')
my_sbm_bi <- FungusTreeNetwork$sbmResults$fungus_tree

node_names_bi <- list(
  row = FungusTreeNetwork$networks$fungus_names,
  col = FungusTreeNetwork$networks$tree_names
)

my_blocks_bi <- get_block(my_sbm_bi,
  labels = c(row = "Fungus", col = "Tree"),
  node_names = node_names_bi
)
my_blocks_bi$row
my_blocks_bi$col

# my_sbm_uni <- sbm::estimateSimpleSBM(sbm::fungusTreeNetwork$tree_tree,
#                                      model = "poisson")
my_sbm_uni <- FungusTreeNetwork$sbmResults$tree_tree

node_names_uni <- list(FungusTreeNetwork$networks$tree_names)

my_blocks_uni <- get_block(my_sbm_uni,
  labels = c("Tree"),
  node_names = node_names_uni
)
```

```
my_blocks_uni
```

---

`get_block.BipartiteSBM_fit` *get_block.BipartiteSBM_fit method*

---

## Description

A fct that return blocks attribution or probabilities for each nodes in a Sbm fit from the **sbm** package.

## Usage

```
## S3 method for class 'BipartiteSBM_fit'
get_block(
  x,
  labels = "default",
  node_names = NULL,
  attribution = TRUE,
  proportion = FALSE
)
```

## Arguments

| | |
|---|---|
| `x` | Sbm model of class 'BipartiteSBM_fit'. |
| `labels` | labels for nodes. If it's simple sbm it should be a single character ("default" -> c("nodes")). If sbm is bipartite a named character (names are row and col) ("default" -> c(row = 'row', col = 'col')). |
| `node_names` | named list ("row","col"), row is a character vector containing names of nodes in rows, and respectively for columns |
| `attribution` | Boolean indicating whether or not the produced tables should contain a block attribution column. This column shows the block in which each nodes is the most likely to be. |
| `proportion` | Boolean indicating whether or not the produced tables should contain the probabilities to belong in each blocks. These columns shows for every nodes and every blocks the probabilities that the node belong to the block. |

## Value

A list containing two data.frames with block attributions and/or proportions one for the row blocks and one for the column blocks

## Examples

```
# my_sbm_bi <- sbm::estimateBipartiteSBM(sbm::fungusTreeNetwork$fungus_tree,
#                                        model = 'bernoulli')
my_sbm_bi <- FungusTreeNetwork$sbmResults$fungus_tree

node_names_bi <- list(
  row = FungusTreeNetwork$networks$fungus_names,
  col = FungusTreeNetwork$networks$tree_names
)

my_blocks_bi <- get_block(my_sbm_bi,
  labels = c(row = "Fungus", col = "Tree"),
  node_names = node_names_bi
)
my_blocks_bi$row
my_blocks_bi$col
```

---

`get_block.SimpleSBM_fit` *get_block.SimpleSBM_fit method*

---

## Description

A fct that return blocks attribution or probabilities for each nodes in a Sbm fit from the **sbm** package.

## Usage

```
## S3 method for class 'SimpleSBM_fit'
get_block(
  x,
  labels = "default",
  node_names = NULL,
  attribution = TRUE,
  proportion = FALSE
)
```

## Arguments

| | |
|---|---|
| x | Sbm model of class 'SimpleSBM_fit'. |
| labels | labels for nodes. If it's simple sbm it should be a single character ("default" -> c("nodes")). If sbm is bipartite a named character (names are row and col) ("default" -> c(row = 'row', col = 'col')). |
| node_names | character: node names |
| attribution | Boolean indicating whether or not the produced tables should contain a block attribution column. This column shows the block in which each nodes is the most likely to be. |

proportion — Boolean indicating whether or not the produced tables should contain the probabilities to belong in each blocks. These columns shows for every nodes and every blocks the probabilities that the node belong to the block.

## Value

A data.frame with block attributions and/or proportions

## Examples

```
# my_sbm_uni <- sbm::estimateSimpleSBM(sbm::fungusTreeNetwork$tree_tree,
#                                      model = "poisson")
my_sbm_uni <- FungusTreeNetwork$sbmResults$tree_tree

node_names_uni <- list(FungusTreeNetwork$networks$tree_names)

my_blocks_uni <- get_block(my_sbm_uni,
  labels = c("Tree"),
  node_names = node_names_uni
)
my_blocks_uni
```

---

# get_flextable *get_flextable*

---

## Description

A fct that build a flextable from an sbm object

## Usage

```
get_flextable(
  sbm,
  labels = "default",
  type = c("blockProp", "connectParam", "storedModels"),
  settings = list()
)
```

## Arguments

sbm — an sbm model product of **sbm** estimation (simple or bipartite)

labels — labels for nodes. If it's simple sbm it should be a single character ("default" -> c("nodes")). If sbm is bipartite a named character (names are row and col) ("default" -> c(row = 'row', col = 'col')).

type — the type of table wanted.

settings — a list of settings

## Details

Values of `type`

**'blockProp':** gives the block proportions.

**'connectParam':** gives the block connectivity.

**'storedModels':** gives the stored modems summary.

The list of parameters `settings` for the flextable

**"caption":** Caption is the flextable title (character)

**"digits":** nb of digits wanted to be shown in the table

**"selected_col":** Color highlighting the selected model

**"best_col":** Color of text for the best model

## Value

Return the selected flextable

## Examples

```
# my_sbm <- sbm::estimateBipartiteSBM(sbm::fungusTreeNetwork$fungus_tree,
#                                     model = 'bernoulli')
my_sbm <- FungusTreeNetwork$sbmResults$fungus_tree

get_flextable(my_sbm,
  labels = c(row = "Fungus", col = "Trees"),
  type = "blockProp"
)

get_flextable(my_sbm,
  labels = c(row = "Fungus", col = "Trees"),
  type = "connectParam", settings = list(digits = 5)
)

get_flextable(my_sbm,
  labels = "default",
  type = "storedModels", settings = list(caption = "New Title")
)
```

---

`plotSbm` *plotSbm*

---

## Description

A fct that plot a beautiful matrix from an sbm object or a network matrix it does have suitable parameters to get the wanted plots. This is the generic function: it does have one method Bipartite and one for Simple Sbm. The ‘x‘ object need to be construct by one of the ‘estimate***SBM‘ function from the ‘sbm‘ package.

## Usage

```
plotSbm(
  x,
  ordered = FALSE,
  transpose = FALSE,
  labels = NULL,
  plotOptions = list()
)
```

## Arguments

| | |
|---|---|
| x | Sbm model of class ‘BipartiteSBM_fit‘, ‘SimpleSBM_fit‘ or simple numeric ‘matrix‘. |
| ordered | Boolean. Set TRUE if the matrix should be reordered (Default is FALSE) |
| transpose | Boolean. Set TRUE to invert columns and rows to flatten a long matrix (Default is FALSE) |
| labels | a named list (names should be: ‘"col"‘ and ‘"row"‘) of characters describing columns and rows component (Default is NULL) |
| plotOptions | a list providing options. See details below. |

## Details

The list of parameters plotOptions for the matrix plot is

**"showValues":** Boolean. Set TRUE to see the real values. Default value is TRUE

**"showPredictions":** Boolean. Set TRUE to see the predicted values. Default value is TRUE

**"title":** Title in characters. Will be printed at the bottom of the matrix. Default value is NULL

**"colPred":** Color of the predicted values, the small values will be more transparent. Default value is "red"

**"colValue":** Color of the real values, the small values will close to white. Default value is "black"

**"showLegend":** Should a legend be printed ? TRUE or FALSE, default: FALSE

**"interactionName":** Name of connection in legend default: "Connection"

## Value

a ggplot object corresponding to the matrix plot inside the app. Groups the network matrix is organized by blocks, the small tiles are for individuals connections. The big tiles between red lines are for block connectivity

## Examples

```
# my_sbm_bi <- sbm::estimateBipartiteSBM(sbm::fungusTreeNetwork$fungus_tree,
#                                        model = 'bernoulli')
my_sbm_bi <- FungusTreeNetwork$sbmResults$fungus_tree

plotSbm(my_sbm_bi,
```

```
  ordered = TRUE, transpose = TRUE,
  plotOptions = list(title = "An example Matrix")
)


# my_sbm_uni <- sbm::estimateSimpleSBM(sbm::fungusTreeNetwork$tree_tree,
#                                      model = "poisson")
my_sbm_uni <- FungusTreeNetwork$sbmResults$tree_tree

plotSbm(my_sbm_uni,
  ordered = TRUE,
  plotOptions = list(title = "An example Matrix")
)

n_col <- 100
n_row <- 90
mat <- matrix(sample(0:10, n_col * n_row, replace = TRUE), n_col, n_row)
plotSbm(mat,
  transpose = TRUE,
  labels = list(col = "Columns", row = "Rows"),
  plotOptions = list(colValue = "blue")
)
```

---

`plotSbm.BipartiteSBM_fit` *plotSbm.BipartiteSBM_fit Method*

---

## Description

plotSbm method for BipartiteSBM_fit object

## Usage

```
## S3 method for class 'BipartiteSBM_fit'
plotSbm(
  x,
  ordered = FALSE,
  transpose = FALSE,
  labels = NULL,
  plotOptions = list()
)
```

## Arguments

| | |
|---|---|
| `x` | an Sbm model of class ‘"BipartiteSBM_fit"‘ |
| `ordered` | Boolean. Set `TRUE` if the matrix should be reordered (Default is `FALSE`) |
| `transpose` | Boolean. Set `TRUE` to invert columns and rows to flatten a long matrix (Default is `FALSE`) |

| | |
|---|---|
| labels | named list (names should be: ‘"col"‘ and ‘"row"‘) of characters describing columns and rows component (Default is NULL) |
| plotOptions | list providing options. See details below. |

## Details

The list of parameters plotOptions for the matrix plot is

**"showValues":** Boolean. Set TRUE to see the real values. Default value is TRUE

**"showPredictions":** Boolean. Set TRUE to see the predicted values. Default value is TRUE

**"title":** Title in characters. Will be printed at the bottom of the matrix. Default value is NULL

**"colPred":** Color of the predicted values, the small values will be more transparent. Default value is "red"

**"colValue":** Color of the real values, the small values will close to white. Default value is "black"

**"showLegend":** Should a legend be printed ? TRUE or FALSE, default: FALSE

**"interactionName":** Name of connection in legend default: "Connection"

## Value

a ggplot object corresponding to the matrix plot inside the app. Groups the network matrix is organized by blocks, the small tiles are for individuals connections. The big tiles between red lines are for block connectivity

## Examples

```
# my_sbm_bi <- sbm::estimateBipartiteSBM(sbm::fungusTreeNetwork$fungus_tree,
#                                        model = 'bernoulli')
my_sbm_bi <- FungusTreeNetwork$sbmResults$fungus_tree

plotSbm(my_sbm_bi,
  ordered = TRUE, transpose = TRUE,
  plotOptions = list(title = "An example Matrix")
)
```

---

plotSbm.default *plotSbm.default Method*

---

## Description

plotSbm method for unknown object

### Usage

```
## Default S3 method:
plotSbm(
  x,
  ordered = FALSE,
  transpose = FALSE,
  labels = NULL,
  plotOptions = list()
)
```

### Arguments

| | |
|---|---|
| x | any object |
| ordered | isn't used in default method |
| transpose | isn't used in default method |
| labels | isn't used in default method |
| plotOptions | isn't used in default method |

### Value

default plot for x

---

## plotSbm.matrix *plotSbm.matrix Method*

---

### Description

plotSbm method for matrix object

### Usage

```
## S3 method for class 'matrix'
plotSbm(
  x,
  ordered = FALSE,
  transpose = FALSE,
  labels = NULL,
  plotOptions = list()
)
```

## Arguments

| | |
|---|---|
| x | numeric matrix |
| ordered | Boolean. Set TRUE if the matrix should be reordered (Default is FALSE) |
| transpose | Boolean. Set TRUE to invert columns and rows to flatten a long matrix (Default is FALSE) |
| labels | named list (names should be: ‘"col"‘ and ‘"row"‘) of characters describing columns and rows component (Default is NULL) |
| plotOptions | list providing options. See details below. |

## Details

The list of parameters plotOptions for the matrix plot is

**"showValues":** Boolean. Set TRUE to see the real values. Default value is TRUE

**"showPredictions":** Boolean. Set TRUE to see the predicted values. Default value is TRUE

**"title":** Title in characters. Will be printed at the bottom of the matrix. Default value is NULL

**"colPred":** Color of the predicted values, the small values will be more transparent. Default value is "red"

**"colValue":** Color of the real values, the small values will close to white. Default value is "black"

**"showLegend":** Should a legend be printed ? TRUE or FALSE, default: FALSE

**"interactionName":** Name of connection in legend default: "Connection"

## Value

a ggplot object corresponding to the matrix plot inside the app. Here because there no ’sbm’ information and only a matrix describing a network, The matrix isn’t organized and the tiles are only showing individuals connections.

## Examples

```
n_col <- 100
n_row <- 90
mat <- matrix(sample(0:10, n_col * n_row, replace = TRUE), n_col, n_row)
plotSbm(mat,
  transpose = TRUE,
  labels = list(col = "Columns", row = "Rows"),
  plotOptions = list(colValue = "blue")
)
```

`plotSbm.SimpleSBM_fit` *plotSbm.SimpleSBM_fit Method*

## Description

plotSbm method for SimpleSBM_fit object

## Usage

```
## S3 method for class 'SimpleSBM_fit'
plotSbm(
  x,
  ordered = FALSE,
  transpose = FALSE,
  labels = NULL,
  plotOptions = list()
)
```

## Arguments

| | |
|---|---|
| `x` | Sbm model of class ‘"SimpleSBM_fit"‘ |
| `ordered` | Boolean. Set `TRUE` if the matrix should be reordered (Default is `FALSE`) |
| `transpose` | isn’t used in this method |
| `labels` | named list (names should be: ‘"col"‘ and ‘"row"‘) of characters describing columns and rows component (Default is `NULL`) |
| `plotOptions` | list providing options. See details below. |

## Details

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

**"showValues":** Boolean. Set TRUE to see the real values. Default value is TRUE

**"showPredictions":** Boolean. Set TRUE to see the predicted values. Default value is TRUE

**"title":** Title in characters. Will be printed at the bottom of the matrix. Default value is NULL

**"colPred":** Color of the predicted values, the small values will be more transparent. Default value is "red"

**"colValue":** Color of the real values, the small values will close to white. Default value is "black"

**"showLegend":** Should a legend be printed ? TRUE or FALSE, default: FALSE

**"interactionName":** Name of connection in legend default: "Connection"

## Value

a ggplot object corresponding to the matrix plot inside the app. Groups the network matrix is organized by blocks, the small tiles are for individuals connections. The big tiles between red lines are for block connectivity

## Examples

```
# my_sbm_uni <- sbm::estimateSimpleSBM(sbm::fungusTreeNetwork$tree_tree,
#                                      model = "poisson")
my_sbm_uni <- FungusTreeNetwork$sbmResults$tree_tree

plotSbm(my_sbm_uni,
  ordered = TRUE,
  plotOptions = list(title = "An example Matrix")
)
```

---

shinySbmApp *Run the Shiny Application*

---

## Description

Run the Shiny Application

## Usage

```
shinySbmApp(
  nbCore_control = TRUE,
  console_verbosity = TRUE,
  onStart = NULL,
  options = list(launch.browser = TRUE),
  enableBookmarking = NULL,
  uiPattern = "/",
  ...
)
```

## Arguments

| | |
|---|---|
| nbCore_control | Allow to control the number of Cores when running an 'sbm' |
| console_verbosity | boolean boolean should the console be printing 'sbm' outputs |
| onStart | A function that will be called before the app is actually run. This is only needed for shinyAppObj, since in the shinyAppDir case, a global.R file can be used for this purpose. |
| options | Named options that should be passed to the runApp call (these can be any of the following: "port", "launch.browser", "host", "quiet", "display.mode" and "test.mode"). You can also specify width and height parameters which provide a hint to the embedding environment about the ideal height/width for the app. |
| enableBookmarking | Can be one of "url", "server", or "disable". The default value, NULL, will respect the setting from any previous calls to enableBookmarking(). See enableBookmarking() for more information on bookmarking your app. |

| | |
|---|---|
| uiPattern | A regular expression that will be applied to each GET request to determine whether the ui should be used to handle the request. Note that the entire request path must match the regular expression in order for the match to be considered successful. |
| ... | arguments to pass to golem_opts. See '?golem::get_golem_options' for more details. |

## Value

No return value, called to launch the 'shiny' application

---

visSbm *visSbm*

---

## Description

A fct that plot a **visNetwork** plot of a adjacency matrix or an Sbm fit from the **sbm** package.

## Usage

```
visSbm(
  x,
  labels = "default",
  node_names = NULL,
  directed = "default",
  settings = list()
)
```

## Arguments

| | |
|---|---|
| x | Sbm model of class 'BipartiteSBM_fit', 'SimpleSBM_fit' or simple numeric 'matrix'. |
| labels | labels for nodes. If it's simple sbm it should be a single character ("default" -> c("nodes")). If sbm is bipartite a named character (names are row and col) ("default" -> c(row = 'row', col = 'col')). |
| node_names | if NULL do nothing specific, but list of nodes are given the graph get interactive and nodes names are showed by clicking on a block. In bipartite case a named list:<br>**"row":** character: node names in rows<br>**"col":** character: node names in columns<br>In unipartite case a single character vector containing the nodes names (Default = NULL). |
| directed | Boolean indicating whether or not the network is directed by default, a asymmetrical matrix will be seen as directed. |
| settings | list of settings |

## Details

List of parameters

**"edge_threshold":** "default" erases as many small edges as it can without isolating any nodes (no connection). It can also be a numeric value between 0 and 1, relative (between min and max) filter for small edges value

**"edge_color":** character: color of edges (default: "lightblue")

**"arrows":** boolean: should edges be arrows

**"arrow_thickness":** numeric: arrows size

**"arrow_start":** character: "row" or "col" or labels value according to row or columns. The arrow will start from selected to the the other value

**"node_color":** named character: Bipartite case c(row = "row_color", col = "col_color"). Unipartite case c("node_color")

**"node_shape":** named character: Bipartite case c(row = "row_shape", col = "col_shape"). Unipartite case c("node_shape"). Value from visNetwork shape argument of visEdges function ("triangle","dot","square",etc...)

**"digits":** integer: number of digits to show when numbers are shown (default: 2)

## Value

a visNetwork visual of the x object

## Examples

```
# my_sbm_bi <- sbm::estimateBipartiteSBM(sbm::fungusTreeNetwork$fungus_tree,
#                                        model = 'bernoulli')
my_sbm_bi <- FungusTreeNetwork$sbmResults$fungus_tree

node_names_bi <- list(
  row = FungusTreeNetwork$networks$fungus_names,
  col = FungusTreeNetwork$networks$tree_names
)

visSbm(my_sbm_bi,
  labels = c(row = "Fungus", col = "Tree"),
  node_names = node_names_bi,
  settings = list(
    arrows = TRUE,
    arrow_start = "Fungus",
    node_color = c(row = "pink", col = "green"),
    node_shape = c(row = "dot", col = "square")
  )
)


# my_sbm_uni <- sbm::estimateSimpleSBM(sbm::fungusTreeNetwork$tree_tree,
#                                      model = "poisson")
my_sbm_uni <- FungusTreeNetwork$sbmResults$tree_tree
```

```
node_names_uni <- list(FungusTreeNetwork$networks$tree_names)

visSbm(my_sbm_uni,
  labels = c("Tree"),
  node_names = node_names_uni,
  settings = list(
    edge_threshold = 0.01,
    edge_color = "grey",
    node_color = c("violet")
  )
)
```

---

`visSbm.BipartiteSBM_fit` *visSbm*

---

## Description

A fct that plot a **visNetwork** plot of a adjacency matrix or an Sbm fit from the **sbm** package.

## Usage

```
## S3 method for class 'BipartiteSBM_fit'
visSbm(
  x,
  labels = "default",
  node_names = NULL,
  directed = "default",
  settings = list()
)
```

## Arguments

| | |
|---|---|
| x | Sbm model of class 'BipartiteSBM_fit', 'SimpleSBM_fit' or simple numeric 'matrix'. |
| labels | labels for nodes. If it's simple sbm it should be a single character ("default" -> c("nodes")). If sbm is bipartite a named character (names are row and col) ("default" -> c(row = 'row', col = 'col')). |
| node_names | if NULL do nothing specific, but list of nodes are given the graph get interactive and nodes names are showed by clicking on a block. In bipartite case a named list:<br>**"row":** character: node names in rows<br>**"col":** character: node names in columns<br>In unipartite case a single character vector containing the nodes names (Default = NULL). |

| | |
|---|---|
| directed | Boolean indicating whether or not the network is directed by default, a asymmetrical matrix will be seen as directed. |
| settings | list of settings |

## Details

List of parameters

**"edge_threshold":** "default" erases as many small edges as it can without isolating any nodes (no connection). It can also be a numeric value between 0 and 1, relative (between min and max) filter for small edges value

**"edge_color":** character: color of edges (default: "lightblue")

**"arrows":** boolean: should edges be arrows

**"arrow_thickness":** numeric: arrows size

**"arrow_start":** character: "row" or "col" or labels value according to row or columns. The arrow will start from selected to the the other value

**"node_color":** named character: Bipartite case c(row = "row_color", col = "col_color"). Unipartite case c("node_color")

**"node_shape":** named character: Bipartite case c(row = "row_shape", col = "col_shape"). Unipartite case c("node_shape"). Value from visNetwork shape argument of visEdges function ("triangle","dot","square",etc...)

**"digits":** integer: number of digits to show when numbers are shown (default: 2)

## Value

a visNetwork visual of the x object

## Examples

```
# my_sbm_bi <- sbm::estimateBipartiteSBM(sbm::fungusTreeNetwork$fungus_tree,
#                                        model = 'bernoulli')
my_sbm_bi <- FungusTreeNetwork$sbmResults$fungus_tree

node_names_bi <- list(
  row = FungusTreeNetwork$networks$fungus_names,
  col = FungusTreeNetwork$networks$tree_names
)

visSbm(my_sbm_bi,
  labels = c(row = "Fungus", col = "Tree"),
  node_names = node_names_bi,
  settings = list(
    arrows = TRUE,
    arrow_start = "Fungus",
    node_color = c(row = "pink", col = "green"),
    node_shape = c(row = "dot", col = "square")
  )
)
```

visSbm.default *visSbm*

## Description

A fct that plot a **visNetwork** plot of a adjacency matrix or an Sbm fit from the **sbm** package.

## Usage

```
## Default S3 method:
visSbm(
  x,
  labels = "default",
  node_names = NULL,
  directed = "default",
  settings = list()
)
```

## Arguments

| | |
|---|---|
| x | Sbm model of class 'BipartiteSBM_fit', 'SimpleSBM_fit' or simple numeric 'matrix'. |
| labels | labels for nodes. If it's simple sbm it should be a single character ("default" -> c("nodes")). If sbm is bipartite a named character (names are row and col) ("default" -> c(row = 'row', col = 'col')). |
| node_names | if NULL do nothing specific, but list of nodes are given the graph get interactive and nodes names are showed by clicking on a block. In bipartite case a named list:<br>**"row":** character: node names in rows<br>**"col":** character: node names in columns<br>In unipartite case a single character vector containing the nodes names (Default = NULL). |
| directed | Boolean indicating whether or not the network is directed by default, a asymmetrical matrix will be seen as directed. |
| settings | list of settings |

## Details

List of parameters

**"edge_threshold":** "default" erases as many small edges as it can without isolating any nodes (no connection). It can also be a numeric value between 0 and 1, relative (between min and max) filter for small edges value

**"edge_color":** character: color of edges (default: "lightblue")

**"arrows":** boolean: should edges be arrows

**"arrow_thickness":** numeric: arrows size

**"arrow_start":** character: "row" or "col" or labels value according to row or columns. The arrow will start from selected to the the other value

**"node_color":** named character: Bipartite case c(row = "row_color", col = "col_color"). Unipartite case c("node_color")

**"node_shape":** named character: Bipartite case c(row = "row_shape", col = "col_shape"). Unipartite case c("node_shape"). Value from visNetwork shape argument of visEdges function ("triangle","dot","square",etc...)

**"digits":** integer: number of digits to show when numbers are shown (default: 2)

## Value

a visNetwork visual of the x object

---

`visSbm.SimpleSBM_fit` *visSbm*

---

## Description

A fct that plot a **visNetwork** plot of a adjacency matrix or an Sbm fit from the **sbm** package.

## Usage

```
## S3 method for class 'SimpleSBM_fit'
visSbm(
  x,
  labels = "default",
  node_names = NULL,
  directed = "default",
  settings = list()
)
```

## Arguments

| | |
|---|---|
| `x` | Sbm model of class 'BipartiteSBM_fit', 'SimpleSBM_fit' or simple numeric 'matrix'. |
| `labels` | labels for nodes. If it's simple sbm it should be a single character ("default" -> c("nodes")). If sbm is bipartite a named character (names are row and col) ("default" -> c(row = 'row', col = 'col')). |
| `node_names` | if NULL do nothing specific, but list of nodes are given the graph get interactive and nodes names are showed by clicking on a block. In bipartite case a named list:<br>**"row":** character: node names in rows<br>**"col":** character: node names in columns |

| | |
|---|---|
| | In unipartite case a single character vector containing the nodes names (Default = NULL). |
| directed | Boolean indicating whether or not the network is directed by default, a asymmetrical matrix will be seen as directed. |
| settings | list of settings |

## Details

List of parameters

**"edge_threshold":** "default" erases as many small edges as it can without isolating any nodes (no connection). It can also be a numeric value between 0 and 1, relative (between min and max) filter for small edges value

**"edge_color":** character: color of edges (default: "lightblue")

**"arrows":** boolean: should edges be arrows

**"arrow_thickness":** numeric: arrows size

**"arrow_start":** character: "row" or "col" or labels value according to row or columns. The arrow will start from selected to the the other value

**"node_color":** named character: Bipartite case c(row = "row_color", col = "col_color"). Unipartite case c("node_color")

**"node_shape":** named character: Bipartite case c(row = "row_shape", col = "col_shape"). Unipartite case c("node_shape"). Value from visNetwork shape argument of visEdges function ("triangle","dot","square",etc...)

**"digits":** integer: number of digits to show when numbers are shown (default: 2)

## Value

a visNetwork visual of the x object

## Examples

```
# my_sbm_uni <- sbm::estimateSimpleSBM(sbm::fungusTreeNetwork$tree_tree,
#                                      model = "poisson")
my_sbm_uni <- FungusTreeNetwork$sbmResults$tree_tree

node_names_uni <- list(FungusTreeNetwork$networks$tree_names)

visSbm(my_sbm_uni,
  labels = c("Tree"),
  node_names = node_names_uni,
  settings = list(
    edge_threshold = 0.01,
    edge_color = "grey",
    node_color = c("violet")
  )
)
```

# Index

∗ **datasets**
FungusTreeNetwork, 2

enableBookmarking(), *17*

FungusTreeNetwork, 2

get_adjacency, 3
get_adjacency.default, 4
get_block, 5
get_block.BipartiteSBM_fit, 7
get_block.SimpleSBM_fit, 8
get_flextable, 9

plotSbm, 10
plotSbm.BipartiteSBM_fit, 12
plotSbm.default, 13
plotSbm.matrix, 14
plotSbm.SimpleSBM_fit, 16

shinySbmApp, 17

visSbm, 18
visSbm.BipartiteSBM_fit, 20
visSbm.default, 22
visSbm.SimpleSBM_fit, 23