

# Package ‘ppsbm’

March 19, 2018

**Type** Package

**Title** Clustering in Longitudinal Networks

**Version** 0.2.2

**Author** D. Giorgi, C. Matias, T. Rebafka, F. Villers

**Maintainer** Daphné Giorgi <daphne.giorgi@sorbonne-universite.fr>

**Description** Stochastic block model used for dynamic graphs represented by Poisson processes.

To model recurrent interaction events in continuous time, an extension of the stochastic block model is proposed where every individual belongs to a latent group and interactions between two individuals follow a conditional inhomogeneous Poisson process with intensity driven by the individuals’ latent groups. The model is shown to be identifiable and its estimation is based on a semiparametric variational expectation-maximization algorithm. Two versions of the method are developed, using either a nonparametric histogram approach (with an adaptive choice of the partition size) or kernel intensity estimators. The number of latent groups can be selected by an integrated classification likelihood criterion.

Y. Baraud and L. Birgé (2009). <doi:10.1007/s00440-007-0126-6>.

C. Biernacki, G. Celeux and G. Govaert (2000). <doi:10.1109/34.865189>.

M. Corneli, P. Latouche and F. Rossi (2016). <doi:10.1016/j.neucom.2016.02.031>.

J.-J. Daudin, F. Picard and S. Robin (2008). <doi:10.1007/s11222-007-9046-7>.

A. P. Dempster, N. M. Laird and D. B. Rubin (1977). <http://www.jstor.org/stable/2984875>.

G. Grégoire (1993). <http://www.jstor.org/stable/4616289>.

L. Hubert and P. Arabie (1985). <doi:10.1007/BF01908075>.

M. Jordan, Z. Ghahramani, T. Jaakkola and L. Saul (1999). <doi:10.1023/A:1007665907178>.

C. Matias, T. Rebafka and F. Villers (2018). <doi:10.1093/biomet/asy016>.

C. Matias and S. Robin (2014). <doi:10.1051/proc/201447004>.

H. Ramlau-Hansen (1983). <doi:10.1214/aos/1176346152>.

P. Reynaud-Bouret (2006). <doi:10.3150/bj/1155735930>.

**License** GPL (>= 2)

**Imports** Rfast, clue, gtools, parallel

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 6.0.1

**URL** <https://cran.r-project.org>

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2018-03-19 16:37:08 UTC

## R topics documented:

ARI . . . . .	3
bootstrap_and_CI . . . . .	3
classInd . . . . .	5
confidenceInterval . . . . .	5
convertGroupPair . . . . .	6
convertNodePair . . . . .	7
correctTau . . . . .	8
find_ql . . . . .	9
find_ql_diff . . . . .	9
generateDynppsbm . . . . .	10
generateDynppsbmConst . . . . .	11
generated_Q3 . . . . .	12
generated_Q3_n20 . . . . .	13
generated_sol_hist . . . . .	13
generated_sol_kernel . . . . .	14
generatePP . . . . .	14
generatePPConst . . . . .	15
JEvalMstep . . . . .	15
kernelIntensities . . . . .	16
listNodePairs . . . . .	17
mainVEM . . . . .	17
mainVEMPar . . . . .	20
modelSelection_Q . . . . .	21
modelSelec_QPlot . . . . .	22
Mstep_hist . . . . .	23
Mstep_kernel . . . . .	23
permuteZEst . . . . .	24
sortIntensities . . . . .	25
statistics . . . . .	25
tauDown_Q . . . . .	26
tauInitial . . . . .	27
tauKmeansSbm . . . . .	28
taurhoInitial . . . . .	29
tauUpdate . . . . .	30
tauUp_Q . . . . .	30
VEstep . . . . .	31

**Index**

**32**

---

ARI                                      *Adjusted Rand Index (ARI)*

---

**Description**

Compute the Adjusted Rand Index (ARI) between the true latent variables and the estimated latent variables

**Usage**

ARI(z, hat.z)

**Arguments**

z                                      Matrix of size  $Q \times n$  with entries = 0 or 1 : 'true' latent variables  
hat.z                                    Matrix of  $Q \times n$  with  $0 < \text{entries} < 1$  : estimated latent variables

**Examples**

```
z <- matrix(c(1,1,0,0,0,0, 0,0,1,1,0,0, 0,0,0,0,1,1), nrow = 3, byrow = TRUE)
hat.z <- matrix(c(0,0,1,1,0,0, 1,1,0,0,0,0, 0,0,0,0,1,1), nrow = 3, byrow = TRUE)

ARI(z, hat.z)
```

---

bootstrap\_and\_CI                      *Bootstrap and Confidence Interval*

---

**Description**

Not for sparse models and only for histograms

**Usage**

```
bootstrap_and_CI(sol, Time, R, alpha = 0.05, nbcores = 1, d_part = 5,
  n_perturb = 10, perc_perturb = 0.2, directed, filename = NULL)
```

**Arguments**

sol                                    sol  
Time                                    time  
R                                        Number of bootstrap samples  
alpha                                    Level of confidence :  $1 - \alpha$

nbcors	Number of cores for parallel execution If set to 1 it does sequential execution Beware: parallelization with fork (multicore) : doesn't work on Windows!
d_part	Maximal level for finest partitions of time interval [0,T], used for kmeans initializations. <ul style="list-style-type: none"> <li>• Algorithm takes partition up to depth <math>2^d</math> with <math>d = 1, \dots, d_{part}</math></li> <li>• Explore partitions <math>[0, T], [0, T/2], [T/2, T], \dots, [0, T/2^d], \dots, [(2^d-1)T/2^d, T]</math></li> <li>• Total number of partitions <math>n_{part} = 2^{(d_{part}+1)} - 1</math></li> </ul>
n_perturb	Number of different perturbations on k-means result
perc_perturb	Percentage of labels that are to be perturbed (= randomly switched)
directed	Boolean for directed (TRUE) or undirected (FALSE) case
filename	filename

### Examples

```
# data of a synthetic graph with 50 individuals and 3 clusters

n <- 50
Q <- 3

Time <- generated_Q3$data$Time
data <- generated_Q3$data
z <- generated_Q3$z

Dmax <- 2^3

# VEM-algo hist
sol.hist <- mainVEM(list(Nijk=statistics(data,n,Dmax,directed=FALSE),Time=Time),
  n,Qmin=3,directed=FALSE,method='hist',d_part=1,n_perturb=0)[[1]])

# compute bootstrap confidence bands
boot <- bootstrap_and_CI(sol.hist,Time,R=10,alpha=0.1,nbcors=1,d_part=1,n_perturb=0,
  directed=FALSE)

# plot confidence bands
alpha.hat <- exp(sol.hist$logintensities.ql)
vec.x <- (0:Dmax)*Time/Dmax
ind.ql <- 0
par(mfrow=c(2,3))
for (q in 1:Q){
  for (l in q:Q){
    ind.ql <- ind.ql+1
    ymax <- max(c(boot$CI.limits[ind.ql,2,],alpha.hat[ind.ql,]))
    plot(vec.x,c(alpha.hat[ind.ql,],alpha.hat[ind.ql,Dmax]),type='s',col='black',
      ylab='Intensity',xaxt='n',xlab= paste('(',q,',',l,')'),sep=""),
      cex.axis=1.5,cex.lab=1.5,ylim=c(0,ymax),main='Confidence bands')
    lines(vec.x,c(boot$CI.limits[ind.ql,1,],boot$CI.limits[ind.ql,1,Dmax]),col='blue',
      type='s',lty=3)
  }
}
```

```

        lines(vec.x,c(boot$CI.limits[ind.q1,2,],boot$CI.limits[ind.q1,2,Dmax]),col='blue',
              type='s',lty=3)
    }
}

```

---

classInd	<i>Function for k-means</i>
----------	-----------------------------

---

**Description**

Function for k-means

**Usage**

```
classInd(c1)
```

**Arguments**

c1                      Label list of nodes

**Value**

x : class indicator matrix

---

confidenceInterval	<i>Confidence Interval</i>
--------------------	----------------------------

---

**Description**

Compute confidence bands for all pair of groups  $(q, l)$

**Usage**

```
confidenceInterval(boot.sol, alpha = 0.05)
```

**Arguments**

boot.sol                Bootstrap list of estimators

alpha                    Level of confidence :  $1 - \alpha$

**Examples**

```

# data of a synthetic graph with 50 individuals and 3 clusters

n <- 50
Q <- 3

Time <- generated_Q3$data$Time
data <- generated_Q3$data
z <- generated_Q3$z

Dmax <- 2^3

# VEM-algo hist
sol.hist <- mainVEM(list(Nijk=statistics(data,n,Dmax,directed=FALSE),Time=Time),
  n,Qmin=3,directed=FALSE,method='hist',d_part=1,n_perturb=0)[[1]])

# compute bootstrap confidence bands
boot <- bootstrap_and_CI(sol.hist,Time,R=5,alpha=0.1,nbcores=1,d_part=1,n_perturb=0,
  directed=FALSE)

boot.sol <- boot$boot.sol

confidenceInterval(boot.sol)

```

---

convertGroupPair	<i>Convert group pair (q, l)</i>
------------------	----------------------------------

---

**Description**

Gives the index in  $1, \dots, Q^2$  (directed) or  $1, \dots, Q * (Q + 1) / 2$  (undirected) that corresponds to group pair  $(q, l)$ . Works also for vectors of indices  $q$  and  $l$ .

**Usage**

```
convertGroupPair(q, l, Q, directed = TRUE)
```

**Arguments**

q	Group index $q$
l	Group index $l$
Q	Total number of groups $Q$
directed	Boolean for directed (TRUE) or undirected (FALSE) case

**Details**

Relations between groups  $(q, l)$  are stored in vectors, whose indexes depend on whether the graph is directed or undirected.

**Directed case :** • The  $(q, l)$  group pair is converted into the index  $(q - 1) * Q + l$

**Undirected case :** • The  $(q, l)$  group pair with  $q \leq l$  is converted into the index  $(2 * Q - q + 2) * (q - 1) / 2 + l - q + 1$

**Value**

Index corresponding to the group pair  $(q, l)$

**Examples**

```
# Convert the group pair (3,2) into an index, where the total number of group is 3,
# for directed and undirected graph
```

```
q <- 3
l <- 2
Q <- 3
```

```
directedIndex <- convertGroupPair(q,l,Q)
undirectedIndex <- convertGroupPair(q,l,Q, FALSE)
```

---

convertNodePair	<i>Convert node pair (i, j)</i>
-----------------	---------------------------------

---

**Description**

Convert node pair  $(i, j)$  into an index

**Directed case :** • The node pair  $(i, j)$  with  $(i \neq j)$  is converted into the index  $(i - 1) * (n - 1) + j - (i < j)$

**Undirected case :** • The node pair  $(i, j)$  with  $(i \neq j)$  is converted into the index  $(2 * n - i) * (i - 1) / 2 + j - i$

**Usage**

```
convertNodePair(i, j, n, directed)
```

**Arguments**

i	Node $i$ : $i \in 1, \dots, n$
j	Node $j$ : $j \in 1, \dots, n$
n	Total number of nodes : $i, j \in 1, \dots, n$
directed	Boolean for directed (TRUE) or undirected (FALSE) case

**Details**

The number of possible node pairs is

- $N = n * (n - 1)$  for the directed case
- $N = n * (n - 1)/2$  for the undirected case

which corresponds to the cardinality of data\$type.seq

**Value**

Index corresponding to the node pair

**Examples**

```
# Convert the node pair (3,7) into an index, where the total number of nodes is 10,  
# for directed and undirected graph
```

```
i <- 3  
j <- 7  
n <- 10
```

```
directedIndex <- convertNodePair(i,j,n,TRUE)  
undirectedIndex <- convertNodePair(i,j,n,FALSE)
```

---

correctTau

*Handling of values of  $\tau$*

---

**Description**

Avoid values of  $\tau$  to be exactly 0 and exactly 1.

**Usage**

```
correctTau(tau)
```

**Arguments**

tau                     $\tau$



---

find_ql	<i>Convert index into group pair</i>
---------	--------------------------------------

---

**Description**

This function is the inverse of the conversion  $(q, l), q, l$  into  $1, \dots, Q^2$  for the directed case  $(q, l), q \leq l$  into  $1, \dots, Q * (Q + 1)/2$  for the undirected case. It takes the integer index corresponding to  $(q, l)$  and returns  $(q, l)$ .

**Usage**

```
find_ql(ind_ql, Q, directed = TRUE)
```

**Arguments**

ind_ql	Converted $(q, l)$ index
Q	Total number of groups $Q$
directed	Boolean for directed (TRUE) or undirected (FALSE) case

**Value**

Group pair  $(q, l)$  corresponding to the given index

**Examples**

```
# Convert the index 5 into a group pair for undirected graph
# and the index 8 into a group pair for directed graph
# where the total number of group is 3

ind_ql_dir <- 8
ind_ql_undir <- 5

Q <- 3

directedIndex <- find_ql(ind_ql_dir, Q)
undirectedIndex <- find_ql(ind_ql_undir, Q, FALSE)
```

---

find_ql_diff	<i>Convert index into group pair in tauDown_Q</i>
--------------	---

---

**Description**

This function is the inverse of the conversion  $(q, l), q < l$  into  $1, \dots, Q * (Q - 1)/2$ . Used only in tauDown\_Q.

**Usage**

```
find_ql_diff(ind_ql, Q)
```

**Arguments**

ind_ql	Converted $(q, l)$ index
Q	Total number of groups $Q$

**Value**

Group pair  $(q, l)$  corresponding to the given index

---

generateDynppsbm	<i>Data under dynppsbm</i>
------------------	----------------------------

---

**Description**

Generate data under dynppsbm

**Usage**

```
generateDynppsbm(intens, Time, n, prop.groups, directed = TRUE)
```

**Arguments**

intens	List containing intensity functions $\alpha^{(q,l)}$ and upper bounds of intensities
Time	Final time
n	Total number of nodes
prop.groups	Vector of group proportions (probability to belong to a group), should be of length $Q$
directed	Boolean for directed (TRUE) or undirected (FALSE) case. If directed=TRUE then intens should be of length $Q^2$ and if directed =FALSE then length $Q * (Q + 1)/2$

**Value**

Simulated data, latent group variables and intensities  $\alpha^{(q,l)}$

## References

- ANDERSEN, P. K., BORGAN, Ø., GILL, R. D. & KEIDING, N. (1993). Statistical models based on counting processes. Springer Series in Statistics. Springer-Verlag, New York.
- DAUDIN, J.-J., PICARD, F. & ROBIN, S. (2008). A mixture model for random graphs. *Statist. Comput.* 18, 173–183.
- MATIAS, C., REBAFKA, T. & VILLERS, F. (2018). A semiparametric extension of the stochastic block model for longitudinal networks. *Biometrika*.
- MATIAS, C. & ROBIN, S. (2014). Modeling heterogeneity in random graphs through latent space models: a selective review. *Esaim Proc. & Surveys* 47, 55–74.

## Examples

```
# Generate data from an undirected graph with n=10 individuals and Q=2 clusters

# equal cluster proportions
prop.groups <- c(0.5,0.5)

# 3 different intensity functions :
intens <- list(NULL)
intens[[1]] <- list(intens= function(x) 100*x*exp(-8*x),max=5)
  # (q,1) = (1,1)
intens[[2]] <- list(intens= function(x) exp(3*x)*(sin(6*pi*x-pi/2)+1)/2,max=13)
  # (q,1) = (1,2)
intens[[3]] <- list(intens= function(x) 8.1*(exp(-6*abs(x-1/2))-.049),max=8)
  # (q,1) = (2,2)

# generate data :
obs <- generateDynppsbm(intens,Time=1,n=10,prop.groups,directed=FALSE)

# latent variables (true clustering of the individuals)
obs$z

# number of time events :
length(obs$data$time.seq)

# number of interactions between each pair of individuals:
table(obs$data$type.seq)
```

---

generateDynppsbmConst *Data under dynppsbm with piecewise constant intensities*

---

## Description

Generate data under dynppsbm with piecewise constant intensities

## Usage

```
generateDynppsbmConst(intens, Time, n, prop.groups, directed = TRUE)
```

**Arguments**

<code>intens</code>	Matrix with piecewise constant intensities $\alpha^{(q,l)}$ (each row gives the constants of the piecewise constant intensity for a group pair $(q, l)$ )
<code>Time</code>	Time
<code>n</code>	Total number of nodes
<code>prop.groups</code>	Vector of group proportions, should be of length $Q$
<code>directed</code>	Boolean for directed (TRUE) or undirected (FALSE) case If directed then <code>intens</code> should be of length $Q^2$ and if undirected then length $Q * (Q + 1)/2$

**Examples**

```
intens1 <- c(1,3,8)
intens2 <- c(2,3,6)

intens <- matrix(c(intens1,intens2,intens1,intens2),4,3)

Time <- 10
n <- 20
prop.groups <- c(0.2,0.3)
dynpsbm <- generateDynpsbmConst(intens,Time,n,prop.groups,directed=TRUE)
```

---

generated\_Q3

*Generated graph with 50 individuals and 3 clusters*

---

**Description**

Generated graph with 50 individuals and 3 clusters

**Usage**

```
generated_Q3
```

**Format**

A data frame

**data** List of 3

**z** Latent variables

**intens** Intensities

---

generated\_Q3\_n20      *Generated graph with 20 individuals and 3 clusters*

---

**Description**

Generated graph with 20 individuals and 3 clusters

**Usage**

generated\_Q3\_n20

**Format**

A data frame

**data** List of 3

**z** Latent variables

**intens** Intensities

---

generated\_sol\_hist      *Generated solution with histogram method*

---

**Description**

Generated solution with histogram method

**Usage**

generated\_sol\_hist

**Format**

List of 5 iterations of the algorithm, each one containing

**List of 8** tau, rho, beta, logintensities.ql, best.d, J, run, converged

---

generated\_sol\_kernel    *Generated solution with kernel method*

---

**Description**

Generated solution with kernel method

**Usage**

generated\_sol\_kernel

**Format**

Solution containing

**List of 8** tau, logintensities.ql.ij, J, run, converged

---

generatePP                *Poisson process*

---

**Description**

Generate realizations of an inhomogeneous Poisson process with an intensity function

**Usage**

generatePP(intens, Time, max.intens)

**Arguments**

intens	Intensity function defined on [0,Time] (needs to be positive)
Time	Final time
max.intens	Upper bound of intensity on [0,Time]

**Value**

Vector of realizations of the PP

**Examples**

```
# Generate a Poisson Process with intensity function
# intens= function(x) 100*x*exp(-8*x)
# and max.intens = 5

intens <- function(x) 100*x*exp(-8*x)

poissonProcess <- generatePP(intens, Time=30, max.intens=1)
```

---

generatePPConst	<i>Poisson process with piecewise constant intensities</i>
-----------------	--

---

**Description**

Generate realizations of a Poisson process with piecewise constant intensities

**Usage**

```
generatePPConst(intens, Time)
```

**Arguments**

intens	Vector with the constants of the intensities (defined on a regular partition of interval [0,Time])
Time	Time

**Examples**

```
intens <- c(1,3,8)
constpp <- generatePPConst(intens, 10)
```

---

JEvalMstep	<i>Evaluation of criterion J</i>
------------	----------------------------------

---

**Description**

Evaluation of the criterion J to verify the convergence of the VEM algorithm

**Usage**

```
JEvalMstep(VE, mstep, data, directed, sparse, method = "hist")
```

**Arguments**

VE	Results of the previous VE for iterative computation
mstep	Results of the previous mstep for iterative computation <ul style="list-style-type: none"> <li>• mstep\$sum_rhotau : N_Q vector (not needed in the function)</li> <li>• mstep\$sum_rhotau_obs : N_Q vector</li> <li>• mstep\$logintensities.ql : N_Q x Dmax matrix</li> <li>• m.step\$beta : N_Q vector</li> </ul>
data	Data same of <a href="#">mainVEM</a>
directed	Boolean for directed (TRUE) or undirected (FALSE) case
sparse	Boolean for sparse (TRUE) or not sparse (FALSE) case
method	List of string. Can be "hist" for histogram method or "kernel" for kernel method

---

kernelIntensities      *Direct kernel estimator intensities*

---

### Description

Compute smooth intensities with direct kernel estimation of intensities relying on a classification  $\tau$ . This can be used with the values  $\tau$  obtained on a dataset with mainVEM function run with 'hist' method.

### Usage

```
kernelIntensities(data, tau, Q, n, directed, rho = 1, sparse = FALSE,
  nb.points = 1000)
```

### Arguments

data	List with 3 components: <ul style="list-style-type: none"> <li>• data\$time.seq : sequence of observed time points of the m-th event (M-vector)</li> <li>• data\$type.seq : sequence of observed values convertNodePair(i,j,n,directed) (auxiliary.R) of process that produced the mth event (M-vector)</li> <li>• \$Time - [0,data\$Time] is the total time interval of observation</li> </ul>
tau	$\tau$
Q	Total number of groups
n	Total number of nodes
directed	Boolean for directed (TRUE) or undirected (FALSE) case
rho	$\rho$
sparse	Boolean for sparse (TRUE) or not sparse (FALSE) case
nb.points	Number of points

### Details

Warning : sparse case not implemented !!!

### Examples

```
# The generated_sol_kernel was generated calling mainVEM with kernel method on the generated_Q3 data
# (50 individuals and 3 clusters)

data <- generated_Q3$data

n <- 50
Q <- 3
```



```
# compute smooth intensity estimators
sol.kernel.intensities <- kernelIntensities(data,generated_sol_kernel$tau,Q,n,directed=FALSE)
```

---

listNodePairs	<i>List node pairs</i>
---------------	------------------------

---

### Description

Create the list of all node pairs

### Usage

```
listNodePairs(n, directed = TRUE)
```

### Arguments

n	Total number of nodes
directed	Boolean for directed (TRUE) or undirected (FALSE) case

### Value

Matrix with two columns which lists all the possible node pairs. Each row is a node pair.

### Examples

```
# List all the node pairs with 10 nodes, for directed and undirected graphs

n <- 10
listNodePairs(n, TRUE)
listNodePairs(n, FALSE)
```

---

mainVEM	<i>Adaptative VEM algorithm</i>
---------	---------------------------------

---

### Description

Principal adaptative VEM algorithm for histogram with model selection or for kernel method.

### Usage

```
mainVEM(data, n, Qmin, Qmax = Qmin, directed = TRUE, sparse = FALSE,
method = c("hist", "kernel"), init.tau = NULL, cores = 1, d_part = 5,
n_perturb = 10, perc_perturb = 0.2, n_random = 0, nb.iter = 50,
fix.iter = 10, epsilon = 1e-06, filename = NULL)
```

**Arguments**

data	Data format depends on the estimation method used!! 1. Data with <b>hist</b> method - list with 2 components: <b>data\$Time</b> [0,data\$Time] is the total time interval of observation <b>data\$Nijk</b> Data matrix with counts per process $N_{ij}$ and sub-intervals ; matrix of size $N * Dmax$ where $N = n(n-1)$ or $n(n-1)/2$ is the number of possible node pairs in the graph and $Dmax = 2^{dmax}$ is the size of the finest partition in the histogram approach Counts are pre-computed - Obtained through function 'statistics' (auxiliary.R) on data with second format 2. Data with <b>kernel</b> method - list with 3 components: <b>data\$time.seq</b> Sequence of observed time points of the m-th event (M-vector) <b>data\$type.seq</b> Sequence of observed values convertNodePair(i,j,n,directed) (auxiliary.R) of process that produced the mth event (M-vector). <b>data\$Time</b> [0,data\$Time] is the total time interval of observation
n	Total number of nodes
Qmin	Minimum number of groups
Qmax	Maximum number of groups
directed	Boolean for directed (TRUE) or undirected (FALSE) case
sparse	Boolean for sparse (TRUE) or not sparse (FALSE) case
method	List of string. Can be "hist" for histogram method or "kernel" for kernel method
init.tau	List of initial values of $\tau$ - all tau's are matrices with size $Q \times n$ (might be with different values of Q)
cores	Number of cores for parallel execution If set to 1 it does sequential execution Beware: parallelization with fork (multicore) : doesn't work on Windows!
d_part	Maximal level for finest partition of time interval [0,T] used for k-means initializations. <ul style="list-style-type: none"> <li>• Algorithm takes partition up to depth <math>2^d</math> with <math>d = 1, \dots, d_{part}</math></li> <li>• Explore partitions <math>[0, T], [0, T/2], [T/2, T], \dots [0, T/2^d], \dots [(2^d-1)T/2^d, T]</math></li> <li>• Total number of partitions <math>n_{part} = 2^{(d_{part}+1)} - 1</math></li> </ul>
n_perturb	Number of different perturbations on k-means result When $Qmin < Qmax$ , number of perturbations on the result with $Q - 1$ or $Q + 1$ groups
perc_perturb	Percentage of labels that are to be perturbed (= randomly switched)
n_random	Number of completely random initial points. The total number of initializations for the VEM is $n_{part} * (1 + n_{perturb}) + n_{random}$
nb.iter	Number of iterations of the VEM algorithm
fix.iter	Maximum number of iterations of the fixed point into the VE step
epsilon	Threshold for the stopping criterion of VEM and fixed point iterations
filename	Name of the file where to save the results along the computation (increasing steps for $Q$ , these are the longest). The file will contain a list of 'best' results.

## Details

The sparse version works only for the histogram approach.

## References

- DAUDIN, J.-J., PICARD, F. & ROBIN, S. (2008). A mixture model for random graphs. *Statist. Comput.* 18, 173–183.
- DEMPSTER, A. P., LAIRD, N. M. & RUBIN, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *J. Roy. Statist. Soc. Ser. B* 39, 1–38.
- JORDAN, M., GHAHRAMANI, Z., JAAKKOLA, T. & SAUL, L. (1999). An introduction to variational methods for graphical models. *Mach. Learn.* 37, 183–233.
- MATIAS, C., REBAFKA, T. & VILLERS, F. (2018). A semiparametric extension of the stochastic block model for longitudinal networks. *Biometrika*.
- MATIAS, C. & ROBIN, S. (2014). Modeling heterogeneity in random graphs through latent space models: a selective review. *Esaim Proc. & Surveys* 47, 55–74.

## Examples

```
# load data of a synthetic graph with 50 individuals and 3 clusters
n <- 20
Q <- 3

Time <- generated_Q3_n20$data$Time
data <- generated_Q3_n20$data
z <- generated_Q3_n20$z

step <- .001
x0 <- seq(0,Time,by=step)
intens <- generated_Q3_n20$intens

# VEM-algo kernel
sol.kernel <- mainVEM(data,n,Q,directed=FALSE,method='kernel', d_part=0,
  n_perturb=0)[[1]]
# compute smooth intensity estimators
sol.kernel.intensities <- kernelIntensities(data,sol.kernel$tau,Q,n,directed=FALSE)
# eliminate label switching
intensities.kernel <- sortIntensities(sol.kernel.intensities,z,sol.kernel$tau,
  directed=FALSE)

# VEM-algo hist
# compute data matrix with precision d_max=3
Dmax <- 2^3
Nijk <- statistics(data,n,Dmax,directed=FALSE)
sol.hist <- mainVEM(list(Nijk=Nijk,Time=Time),n,Q,directed=FALSE, method='hist',
  d_part=0,n_perturb=0,n_random=0)[[1]]
log.intensities.hist <- sortIntensities(sol.hist$logintensities.q1,z,sol.hist$tau,
  directed=FALSE)

# plot estimators
par(mfrow=c(2,3))
```

```

ind.q1 <- 0
for (q in 1:Q){
  for (l in q:Q){
    ind.q1 <- ind.q1 + 1
    true.val <- intens[[ind.q1]]$intens(x0)
    values <- c(intensities.kernel[ind.q1,],exp(log.intensities.hist[ind.q1,]),true.val)
    plot(x0,true.val,type='l',xlab=paste0("(q,l)=(",q,",",l,")"),ylab='',
        ylim=c(0,max(values)+.1))
    lines(seq(0,1,by=1/Dmax),c(exp(log.intensities.hist[ind.q1,]),
        exp(log.intensities.hist[ind.q1,Dmax])),type='s',col=2,lty=2)
    lines(seq(0,1,by=.001),intensities.kernel[ind.q1,],col=4,lty=3)
  }
}

```

---

mainVEMPar

*VEM step for parallel version*


---

### Description

VEM step for parallel version

### Usage

```
mainVEMPar(init.point, n, Q, data, directed, sparse, method, nb.iter, fix.iter,
epsilon)
```

### Arguments

init.point	Initial point
n	Total number of nodes
Q	Total number of groups
data	Data same of <a href="#">mainVEM</a>
directed	Boolean for directed (TRUE) or undirected (FALSE) case
sparse	Boolean for sparse (TRUE) or not sparse (FALSE) case
method	List of string. Can be "hist" for histogram method or "kernel" for kernel method
nb.iter	Number of iterations
fix.iter	Maximum number of iterations of the fixed point
epsilon	Threshold for the stopping criterion of VEM and fixed point iterations

---

modelSelection\_Q      *Selects the number of groups with ICL*

---

### Description

Selects the number of groups with Integrated Classification Likelihood Criterion

### Usage

```
modelSelection_Q(data, n, Qmin = 1, Qmax, directed = TRUE, sparse = FALSE,
  sol.hist.sauv)
```

### Arguments

data	List with 2 components: <ul style="list-style-type: none"> <li>• \$Time - [0,data\$Time] is the total time interval of observation</li> <li>• \$Nijk - data matrix with the statistics per process <math>N_{ij}</math> and sub-intervals <math>k</math></li> </ul>
n	Total number of nodes $n$
Qmin	Minimum number of groups
Qmax	Maximum number of groups
directed	Boolean for directed (TRUE) or undirected (FALSE) case
sparse	Boolean for sparse (TRUE) or not sparse (FALSE) case
sol.hist.sauv	List of size Qmax-Qmin+1 obtained from running mainVEM(data,n,Qmin,Qmax,method='hist')

### References

BIERNACKI, C., CELEUX, G. & GOVAERT, G. (2000). Assessing a mixture model for clustering with the integrated completed likelihood. *IEEE Trans. Pattern Anal. Machine Intel.* 22, 719–725.

CORNELI, M., LATOUCHE, P. & ROSSI, F. (2016). Exact ICL maximization in a non-stationary temporal extension of the stochastic block model for dynamic networks. *Neurocomputing* 192, 81–91.

DAUDIN, J.-J., PICARD, F. & ROBIN, S. (2008). A mixture model for random graphs. *Statist. Comput.* 18, 173–183.

MATIAS, C., REBAFKA, T. & VILLERS, F. (2018). A semiparametric extension of the stochastic block model for longitudinal networks. *Biometrika*.

### Examples

```
# load data of a synthetic graph with 50 individuals and 3 clusters
n <- 50

# compute data matrix with precision d_max=3
Dmax <- 2^3
data <- list(Nijk=statistics(generated_Q3$data,n,Dmax,directed=FALSE),
  Time=generated_Q3$data$Time)
```

```
# ICL-model selection
sol.selec_Q <- modelSelection_Q(data,n,Qmin=1,Qmax=4,directed=FALSE,
  sparse=FALSE,generated_sol_hist)

# best number Q of clusters:
sol.selec_Q$Qbest
```

---

modelSelec\_QPlot      *Plots for model selection*

---

## Description

Plots for model selection

## Usage

```
modelSelec_QPlot(model.selec_Q)
```

## Arguments

model.selec\_Q    Output from modelSelection\_Q()

## Examples

```
# load data of a synthetic graph with 50 individuals and 3 clusters
n <- 50

# compute data matrix with precision d_max=3
Dmax <- 2^3
data <- list(Nijk=statistics(generated_Q3$data,n,Dmax,directed=FALSE),
  Time=generated_Q3$data$Time)

# ICL-model selection
sol.selec_Q <- modelSelection_Q(data,n,Qmin=1,Qmax=4,directed=FALSE,
  sparse=FALSE,generated_sol_hist)

# plot ICL
modelSelec_QPlot(sol.selec_Q)
```

---

Mstep_hist	<i>M step for histograms</i>
------------	------------------------------

---

**Description**

M step for histograms estimator

**Usage**

Mstep\_hist(data, VE, directed, sparse)

**Arguments**

data	Data same of <a href="#">mainVEM</a>
VE	Results of the previous VE for iterative computation
directed	Boolean for directed (TRUE) or undirected (FALSE) case
sparse	Boolean for sparse (TRUE) or not sparse (FALSE) case

**References**

BARAUD, Y. & BIRGÉ, L. (2009). Estimating the intensity of a random measure by histogram type estimators. *Probab. Theory Related Fields* 143, 239–284.

MATIAS, C., REBAFKA, T. & VILLERS, F. (2018). A semiparametric extension of the stochastic block model for longitudinal networks. *Biometrika*.

REYNAUD -BOURET, P. (2006). Penalized projection estimators of the Aalen multiplicative intensity. *Bernoulli* 12, 633–661.

---

Mstep_kernel	<i>M step for kernel</i>
--------------	--------------------------

---

**Description**

M step for kernel estimator

**Usage**

Mstep\_kernel(data, VE, directed)

**Arguments**

data	Data same of <a href="#">mainVEM</a>
VE	Results of the previous VE for iterative computation
directed	Boolean for directed (TRUE) or undirected (FALSE) case

## References

- GRÉGOIRE , G. (1993). Least squares cross-validation for counting process intensities. *Scand. J. Statist.* 20, pp. 343–360.
- MATIAS, C., REBAFKA, T. & VILLERS, F. (2018). A semiparametric extension of the stochastic block model for longitudinal networks. *Biometrika*.
- RAMLAU-HANSEN, H. (1983). Smoothing counting process intensities by means of kernel functions. *Ann. Statist.* 11, pp. 453–466.

---

 permuteZEst

*Optimal matching between 2 clusterings*


---

## Description

Compute the permutation of the rows of `hat.z` that has to be applied to obtain the "same order" as `z`.  
 Compute optimal matching between 2 clusterings using Hungarian algorithm

## Usage

```
permuteZEst(z, hat.z)
```

## Arguments

<code>z</code>	Matrice of size $Q \times n$
<code>hat.z</code>	Matrice of size $Q \times n$

## References

- HUBERT, L. & ARABIE, P. (1985). Comparing partitions. *J. Classif.* 2, 193–218.
- MATIAS, C., REBAFKA, T. & VILLERS, F. (2018). A semiparametric extension of the stochastic block model for longitudinal networks. *Biometrika*.

## Examples

```
z <- matrix(c(1,1,0,0,0,0, 0,0,1,1,0,0, 0,0,0,0,1,1), nrow = 3, byrow = TRUE)
hat.z <- matrix(c(0,0,1,1,0,0, 1,1,0,0,0,0, 0,0,0,0,1,1), nrow = 3, byrow = TRUE)

perm <- permuteZEst(z,hat.z)
```



---

sortIntensities	<i>Sort intensities</i>
-----------------	-------------------------

---

**Description**

Sort intensities associated with hat.z "in the same way" as the original intensities associated with z by permutation of rows

**Usage**

```
sortIntensities(intensities, z, hat.z, directed)
```

**Arguments**

intensities	Intensities $\alpha$
z	Matrice of size $Q \times n$
hat.z	Matrice of size $Q \times n$
directed	Boolean for directed (TRUE) or undirected (FALSE) case

**References**

HUBERT, L. & ARABIE, P. (1985). Comparing partitions. *J. Classif.* 2, 193–218.

MATIAS, C., REBAFKA, T. & VILLERS, F. (2018). A semiparametric extension of the stochastic block model for longitudinal networks. *Biometrika*.

**Examples**

```
z <- matrix(c(1,1,0,0,0,0, 0,0,1,1,0,0, 0,0,0,0,1,1), nrow = 3, byrow = TRUE)
hat.z <- matrix(c(0,0,1,1,0,0, 1,1,0,0,0,0, 0,0,0,0,1,1), nrow = 3, byrow = TRUE)

intens <- matrix(c(1,1,1,2,2,2,3,3,3),9)

sortIntensities(intens,z,hat.z, TRUE)
```

---

statistics	<i>Compute statistics</i>
------------	---------------------------

---

**Description**

Convert the initial data into the statistics matrix  $N_{ijk}$ , by counting the number of events for the nodes during the subintervals of a particular partition of the time interval.

**Usage**

```
statistics(data, n, K, directed = TRUE)
```

**Arguments**

data	List with \$type.seq, \$time.seq
n	Total number of nodes : $i, j \in 1, \dots, n$
K	Size of the regular partition, i.e. number of subintervals
directed	Boolean for directed (TRUE) or undirected (FALSE) case

**Value**

$N(i,j)k$  = number of events for the node (i,j) during the k-th subinterval

**Examples**

```
# Convert the generated data into the statistics matrix N_ijk with 8 columns

n <- 50
Dmax <- 2^3

obs <- statistics(generated_Q3$data, n, Dmax, directed=FALSE)
```

---

tauDown\_Q

---

*Construct initial  $\tau$  from  $Q + 1$* 


---

**Description**

Construct initial  $\tau$  with  $Q$  groups from value obtained at  $Q + 1$  groups

**Usage**

```
tauDown_Q(tau, n_perturb = 1)
```

**Arguments**

tau	$\tau$
n_perturb	Number of different perturbations on k-means result

**Value**

List of matrixes of initial values for  $\tau$  for  $Q$  groups from value obtained at  $Q + 1$

**Examples**

```
# Generate first initial tau for generated_Q3 data

n <- 50
Dmax <- 2^3
Q <- 3
d_part <- 1 # less than 3 (owing to Dmax)
n_perturb <- 2
perc_perturb <- 0.2
n_random <- 1
directed <- FALSE

data <- list(Nijk = statistics(generated_Q3$data, n, Dmax, directed = FALSE))

tau <- tauInitial(data,n,Q,d_part,n_perturb,perc_perturb,n_random,directed)

tau.list <- tauDown_Q(tau[[1]],1)
```

---

<code>tauInitial</code>	<i>List of initial values for <math>\tau</math></i>
-------------------------	---

---

**Description**

Same function whatever directed or undirected case

**Usage**

```
tauInitial(data, n, Q, d_part, n_perturb, perc_perturb, n_random, directed)
```

**Arguments**

<code>data</code>	Data : only needs the $N_{ijk}$ field of data
<code>n</code>	Total number of nodes
<code>Q</code>	Total number of groups
<code>d_part</code>	Maximal level for finest partitions of time interval $[0,T]$ , used for kmeans initializations. <ul style="list-style-type: none"> <li>• Algorithm takes partition up to depth <math>2^d</math> with <math>d = 1, \dots, d_{part}</math></li> <li>• Explore partitions <math>[0, T], [0, T/2], [T/2, T], \dots, [0, T/2^d], \dots, [(2^d-1)T/2^d, T]</math></li> <li>• Total number of partitions <math>n_{part} = 2^{(d_{part}+1)} - 1</math></li> </ul>
<code>n_perturb</code>	Number of different perturbations on k-means result
<code>perc_perturb</code>	Percentage of labels that are to be perturbed (= randomly switched)
<code>n_random</code>	Number of completely random initial points. If not zero there will be <code>n_random</code> taus uniformly sampled in the initialization.
<code>directed</code>	Boolean for directed (TRUE) or undirected (FALSE) case

**Details**

The (maximal) total number of initializations is  $d_{part} * (1 + n_{perturb}) + n_{random}$

**Value**

List of matrixes of initial values for  $\tau$

**Examples**

```
# Generate initial tau for generated_Q3 data

n <- 50
Dmax <- 2^3
Q <- 3
d_part <- 1 # less than 3 (owing to Dmax)
n_perturb <- 2
perc_perturb <- 0.2
n_random <- 1
directed <- FALSE

data <- list(Nijk = statistics(generated_Q3$data, n, Dmax, directed = FALSE))

tau <- tauInitial(data,n,Q,d_part,n_perturb,perc_perturb,n_random,directed)
```

---

tauKmeansSbm

*k-means for SBM*


---

**Description**

k-means for SBM

**Usage**

```
tauKmeansSbm(statistics, n, Q, directed)
```

**Arguments**

statistics	Statistics matrix $N_{ijk}$ , counting the events for the nodes pair $(i, j)$ during the subinterval $k$
n	Total number of nodes $n$
Q	Total number of groups $Q$
directed	Boolean for directed (TRUE) or undirected (FALSE) case

**Value**

Initial values for  $\tau$

**Examples**

```

n <- 50
Q <- 3

Dmax <- 2^3

Nijk <- statistics(generated_Q3$data,n,Dmax,directed=FALSE)

tau <- tauKmeansSbm(Nijk,n,Q,FALSE)

```

---

<code>taurhoInitial</code>	<i>Sparse setup - <math>\rho</math> parameter</i>
----------------------------	---

---

**Description**

Sparse setup -  $\rho$  parameter

**Usage**

```
taurhoInitial(tau, data, n, Q, directed = TRUE)
```

**Arguments**

<code>tau</code>	$\tau$	
<code>data</code>		Data : only needs the $N_{ijk}$ field of data
<code>n</code>		Total number of nodes
<code>Q</code>		Total number of groups
<code>directed</code>		Boolean for directed (TRUE) or undirected (FALSE) case

**Value**

Both  $\tau$  and  $\rho$ .

**Examples**

```

# Generate first initial tau for generated_Q3 data

n <- 50
Dmax <- 2^3
Q <- 3
d_part <- 1 # less than 3 (owing to Dmax)
n_perturb <- 2
perc_perturb <- 0.2
n_random <- 1
directed <- FALSE

```

```

data <- list(Nijk = statistics(generated_Q3$data, n, Dmax, directed = FALSE))
tau <- tauInitial(data,n,Q,d_part,n_perturb,perc_perturb,n_random,directed)
taurho <- taurhoInitial(tau[[1]],data,n,Q,directed=FALSE)

```

---

tauUpdate	<i>Update <math>\tau</math></i>
-----------	---------------------------------

---

### Description

One update of  $\tau$  by the fixed point equation

### Usage

```
tauUpdate(tau, pi, mstep, data, directed, sparse, method, rho)
```

### Arguments

tau	Old $\tau$
pi	Estimator of group probabilities $\pi$
mstep	Results of the previous mstep for iterative computation
data	Data same of <a href="#">mainVEM</a>
directed	Boolean for directed (TRUE) or undirected (FALSE) case
sparse	Boolean for sparse (TRUE) or not sparse (FALSE) case
method	List of string. Can be "hist" for histogram method or "kernel" for kernel method
rho	Old $\rho$ (only for sparse model, set to 0 otherwise)

---

tauUp_Q	<i>Construct initial <math>\tau</math> from <math>Q - 1</math></i>
---------	--

---

### Description

Construct initial  $\tau$  with  $Q$  groups from value obtained at  $Q - 1$  groups

### Usage

```
tauUp_Q(tau, n_perturb = 1)
```

### Arguments

tau	$\tau$
n_perturb	Number of different perturbations on k-means result

**Value**

List of matrixes of initial values for  $\tau$  for  $Q$  groups from value obtained at  $Q - 1$

**Examples**

```
# Generate first initial tau for generated_Q3 data

n <- 50
Dmax <- 2^3
Q <- 3
d_part <- 1 # less than 3 (owing to Dmax)
n_perturb <- 2
perc_perturb <- 0.2
n_random <- 1
directed <- FALSE

data <- list(Nijk = statistics(generated_Q3$data, n, Dmax, directed = FALSE))

tau <- tauInitial(data,n,Q,d_part,n_perturb,perc_perturb,n_random,directed)

tau.list <- tauUp_Q(tau[[1]],1)
```

---

VEstep

*VE step*


---

**Description**

VE step

**Usage**

```
VEstep(VE, mstep, directed, sparse, method, epsilon, fix.iter, data)
```

**Arguments**

VE	Results of the previous VE step for iterative computation
mstep	Results of the previous mstep for iterative computation
directed	Boolean for directed (TRUE) or undirected (FALSE) case
sparse	Boolean for sparse (TRUE) or not sparse (FALSE) case
method	List of string. Can be "hist" for histogram method or "kernel" for kernel method
epsilon	Threshold for the stopping criterion of VEM and fixed point iterations
fix.iter	Maximum number of iterations of the fixed point
data	Data same of <a href="#">mainVEM</a>

# Index

## \*Topic **datasets**

- generated\_Q3, [12](#)
- generated\_Q3\_n20, [13](#)
- generated\_sol\_hist, [13](#)
- generated\_sol\_kernel, [14](#)

ARI, [3](#)

bootstrap\_and\_CI, [3](#)

- classInd, [5](#)
- confidenceInterval, [5](#)
- convertGroupPair, [6](#)
- convertNodePair, [7](#)
- correctTau, [8](#)

- find\_ql, [9](#)
- find\_ql\_diff, [9](#)

- generated\_Q3, [12](#)
- generated\_Q3\_n20, [13](#)
- generated\_sol\_hist, [13](#)
- generated\_sol\_kernel, [14](#)
- generateDynppsbm, [10](#)
- generateDynppsbmConst, [11](#)
- generatePP, [14](#)
- generatePPConst, [15](#)

JEvalMstep, [15](#)

kernelIntensities, [16](#)

listNodePairs, [17](#)

- mainVEM, [15](#), [17](#), [20](#), [23](#), [30](#), [31](#)
- mainVEMPar, [20](#)
- modelSelec\_QPlot, [22](#)
- modelSelection\_Q, [21](#)
- Mstep\_hist, [23](#)
- Mstep\_kernel, [23](#)

permuteZEst, [24](#)

- sortIntensities, [25](#)
- statistics, [25](#)

- tauDown\_Q, [26](#)
- tauInitial, [27](#)
- tauKmeansSbm, [28](#)
- taurhoInitial, [29](#)
- tauUp\_Q, [30](#)
- tauUpdate, [30](#)

VEstep, [31](#)