Package 'ppsbm'

May 9, 2026

Type Package

Title Clustering in Longitudinal Networks

Version 1.0.0

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>.

Depends R (>= 3.5.0)

License GPL (>= 2)

Imports Rfast, clue, gtools, parallel

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

URL https://cran.r-project.org

NeedsCompilation no

Author Daphné Giorgi [aut, cre],
Catherine Matias [aut],
Tabea Rebafka [aut],
Fanny Villers [aut]

Repository CRAN

Date/Publication 2025-01-15 21:30:02 UTC

Contents

ARI . . . 2
bootstrap_and_CI . . . 3
convertGroupPair . . . 5
convertNodePair . . . 6
find_ql . . . 7
generateDynppsbm . . . 8
generateDynppsbmConst . . . 9
generated_Q3 . . . 10
generated_Q3_n20 . . . 11
generated_sol_hist . . . 12
generated_sol_kernel . . . 13
generatePP . . . 14
generatePPConst . . . 15
kernelIntensities . . . 15
listNodePairs . . . 16
mainVEM . . . 17
modelSelection_Q . . . 21
modelSelec_QPlot . . . 22
sortIntensities . . . 23
statistics . . . 24

Index **26**

`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<entries<1: estimated latent variables

References

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

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 Bands*

Description

Plots confidence bands for estimated intensities between pairs of groups obtained by bootstrap.

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	One list (for one value of Q) output by mainVEM with hist method.
Time	Positive real number. [0,Time] is the total time interval of observation.
R	Number of bootstrap samples.
alpha	Level of confidence: $1 - \alpha$.

nbcores	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 on the bootstrap samples • Algorithm takes partition up to depth 2^d with $d = 1, ..., d_{part}$ • Explore partitions $[0,T], [0,T/2], [T/2,T], ...[0,T/2^d], ...[(2^d-1)T/2^d, T]$ • Total number of partitions $npart = 2^{(d_{part}+1)} - 1$
n_perturb	Number of different perturbations on k-means result on the bootstrap samples.
perc_perturb	Percentage of labels that are to be perturbed (= randomly switched) on the bootstrap samples.
directed	Boolean for directed (TRUE) or undirected (FALSE) case.
filename	Name of the file where to save the results.

Details

Not for sparse models and only for histogram method.

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

K <- 2^3

# VEM-algo hist:
sol.hist <- mainVEM(list(Nijk=statistics(data,n,K,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,nbcores=1,d_part=1,n_perturb=0,
     directed=FALSE)

# plot confidence bands
alpha.hat <- exp(sol.hist$logintensities.ql)
vec.x <- (0:K)*Time/K
ind.ql <- 0
par(mfrow=c(2,3))
for (q in 1:Q){
  for (l in q:Q){
    ind.ql <- ind.ql+1
    ymax <- max(c(boot$CI.limits[ind.ql,2,],alpha.hat[ind.ql,]))
    plot(vec.x,c(alpha.hat[ind.ql,],alpha.hat[ind.ql,K]),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,K]),col='blue',
        type='s',lty=3)
    lines(vec.x,c(boot$CI.limits[ind.ql,2,],boot$CI.limits[ind.ql,2,K]),col='blue',
        type='s',lty=3)
  }
}
```

`convertGroupPair` *Convert group pair* (q, l)

Description

Gives the index in $1, \ldots, Q^2$ (directed) or $1, \ldots, Q(Q+1)/2$ (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 $(2Q - q + 2) * (q-1)/2 + l - q + 1$

Value

Index corresponding to the group pair (q, l)

Examples

```
# Convert the group pair (3,2) into an index, where the total number of groups 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` *Convert node pair* (i, j)

Description

Convert node pair (i, j) into an index in $\{1, \ldots, N\}$ where $N = n(n-1)$ (directed case) or $N = n(n-1)/2$ (undirected case).

Usage

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

Arguments

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

Details

Interacting individuals (i, j) must be encoded into integer values in $\{1, \ldots, N\}$ in describing the data.

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 $(2 * n - i) * (i-1)/2 + j - i$

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 range of values for `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 interactions

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

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

---

find_ql *Convert index into group pair*

Description

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

`generateDynppsbm` *Dynppsbm data generator*

Description

Generates data under the Dynamic Poisson Process Stochastic Blockmodel (dynppsbm).

Usage

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

Arguments

`intens`	List of intensity functions $\alpha^{(q,l)}$, each one is a list of 2 components: • `intens` : a positive function. The intensity function $\alpha^{(q,l)}$ • `max` : positive real number. An upper bound on $\alpha^{(q,l)}$
`Time`	Positive real number. [0,Time] is the total time interval of observation.
`n`	Total number of nodes, $1 \leq i \leq n$.
`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 then `intens` should be of length Q^2, else of length $Q * (Q+1)/2$.

Value

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

References

MATIAS, C., REBAFKA, T. & VILLERS, F. (2018). A semiparametric extension of the stochastic block model for longitudinal networks. Biometrika. 105(3): 665-680.

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,l) = (1,1)
intens[[2]] <- list(intens= function(x) exp(3*x)*(sin(6*pi*x-pi/2)+1)/2,max=13)
    # (q,l) = (1,2)
intens[[3]] <- list(intens= function(x) 8.1*(exp(-6*abs(x-1/2))-.049),max=8)
    # (q,l) = (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 the Dynamic Poisson Process Stochastic Blockmodel (dynppsbm) with piecewise constant intensity functions.

Usage

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

Arguments

`intens`	Matrix with piecewise constant intensities $\alpha^{(q,l)}$. Each row gives the constant values of the piecewise constant intensity for a group pair (q, l) on a regular partition of the time interval [0,Time].
`Time`	Positive real number. [0,Time] is the total time interval of observation.
`n`	Total number of nodes, $1 \leq i \leq n$.
`prop.groups`	Vector of group proportions, should be of length Q.
`directed`	Boolean for directed (TRUE) or undirected (FALSE) case. If directed then `intens` should be of length Q^2, else of length $Q * (Q + 1)/2$.

References

MATIAS, C., REBAFKA, T. & VILLERS, F. (2018). A semiparametric extension of the stochastic block model for longitudinal networks. Biometrika. 105(3): 665-680.

Examples

```
# Define 2 different piecewise constant intensity functions
# on a 3 parts regular partition of time interval [0,Time]
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(0.2,0.8)
obs <- generateDynppsbmConst(intens,Time,n,prop.groups,directed=TRUE)
```

`generated_Q3` *Example dataset*

Description

Example of undirected dataset with $n = 50$ individuals in $Q = 3$ clusters and final observation Time=1.

Usage

```
generated_Q3
```

Format

A list of 3 components:

`data` Observed data is itself a list of 3 components:

- `time.seq` - Vector containing the times (in [0,1]) of the events (length M).
- `type.seq` - Vector containing the types in $\{1, \ldots, N\}$ of the events (length M). Here, $N = n(n-1)/2$ (undirected).
- `Time` - Positive real number. [0,Time] is the total time interval of observation.

`z` Latent variables. A matrix with size $Q \times n$ and entries 1 (cluster q contains node i) or 0 (else).

`intens` Intensities used to simulate data. A list of $Q(Q+1)/2$ intensity functions. Each one is given as a list of 2 components:

- `intens` - a positive function. The intensity function $\alpha^{(q,l)}$
- `max` - positive real number. An upper bound on function $\alpha^{(q,l)}$

Details

This random datatset was obtained using the following code

```
intens <- list(NULL)
intens[[1]] <- list(intens=function(x) return (rep(4,length(x))), max=4.1)
intens[[2]] <- list(intens=function(x){
   y <- rep(0,length(x))
   y[x<.25] <- 4
   y[x>.75] <- 10
   return(y)
 }, max=10.1)
intens[[3]] <- list(intens=function(x) return(8*(1/2-abs(x-1/2))), max=4.1)
intens[[4]] <- list(intens=function(x) return(100*x*exp(-8*x)), max=4.698493)
intens[[5]] <- list(intens=function(x) return(exp(3*x)*(sin(6*pi*x-pi/2)+1)/2), max=12.59369)
intens[[6]] <- list(intens=function(x) return(8.1*(exp(-6*abs(x-1/2))-.049)), max=7.8031)

generated_Q3 <- generateDynppsbm(intens,Time=1,n=50,prop.groups=rep(1/3,3),directed=F)
```

References

MATIAS, C., REBAFKA, T. & VILLERS, F. (2018). A semiparametric extension of the stochastic block model for longitudinal networks. Biometrika. 105(3): 665-680.

`generated_Q3_n20` *Example dataset*

Description

Example of undirected dataset with $n = 20$ individuals in $Q = 3$ clusters and observation Time=1

Usage

```
generated_Q3_n20
```

Format

A list of 3 components:

`data` Observed data is itself a list of 3 components:

- `time.seq` - Vector containing the times of the events (length M)
- `type.seq` - Vector containing the types of the events (length M)
- `Time` - Positive real number. [0,Time] is the total time interval of observation

`z` Latent variables. A matrix with size $Q \times n$ and entries 1 (cluster q contains node i) or 0 (else).

`intens` Intensities used to simulate data. A list of $Q(Q+1)/2$ intensity functions. Each one is given as a list of 2 components:

- `intens` - a positive function. The intensity function $\alpha^{(q,l)}$
- `max` - positive real number. An upper bound on function $\alpha^{(q,l)}$

Details

This random datatset was obtained using the following code

```
intens <- list(NULL)
intens[[1]] <- list(intens=function(x) return (rep(4,length(x))), max=4.1)
intens[[2]] <- list(intens=function(x){
   y <- rep(0,length(x))
   y[x<.25] <- 4
   y[x>.75] <- 10
   return(y)
 }, max=10.1)
intens[[3]] <- list(intens=function(x) return(8*(1/2-abs(x-1/2))), max=4.1)
intens[[4]] <- list(intens=function(x) return(100*x*exp(-8*x)), max=4.698493)
intens[[5]] <- list(intens=function(x) return(exp(3*x)*(sin(6*pi*x-pi/2)+1)/2), max=12.59369)
intens[[6]] <- list(intens=function(x) return(8.1*(exp(-6*abs(x-1/2))-.049)), max=7.8031)

generated_Q3_n20 <- generateDynppsbm(intens,Time=1,n=20,prop.groups=rep(1/3,3),directed=F)
```

References

MATIAS, C., REBAFKA, T. & VILLERS, F. (2018). A semiparametric extension of the stochastic block model for longitudinal networks. Biometrika. 105(3): 665-680.

---

generated_sol_hist *Output example of mainVEM*

Description

Output of mainVEM obtained on dataset `generated_Q3` with `hist` method and Qmin=1, Qmax=5.

Usage

```
generated_sol_hist
```

Format

List of 5 components. Each one is the output of the algorithm with a different value of the number of clusters Q for $1 \leq Q \leq 5$ and given as a list of 8 components:

`tau` Matrix with size $Q \times n$ containing the estimated probability in $(0, 1)$ that cluster q contains node i.

`rho` Sparsity parameter - 1 in this case (non sparse method).

`beta` Sparsity parameter - 1 in this case (non sparse method).

`logintensities.ql` Matrix with size $Q(Q+1)/2 \times K$. Each row contains estimated values of the log of the intensity function $\log(\alpha^{(q,l)})$ on a regular partition (in K parts) of the time interval [0,Time].

`best.d` Vector with length $Q(Q+1)/2$ (undirected case) with estimated value for the exponent of the best partition to estimate intensity $\alpha^{(q,l)}$. The best number of parts is $K = 2^d$.

`J` Estimated value of the ELBO

`run` Which run of the algorithm gave the best solution. A run relies on a specific initialization of the algorithm. A negative value maybe obtained in the decreasing phase (for Q) of the algorithm.

`converged` Boolean. If TRUE, the algorithm stopped at convergence. Otherwise it stopped at the maximal number of iterations.

Details

This solution was (randomly) obtained using the following code

```
Nijk <- statistics(generated_Q3$data,n=50,K=8,directed=FALSE)
generated_sol_hist <- mainVEM(list(Nijk=Nijk,Time=1),n=50,Qmin=1,Qmax=5,directed=FALSE,method='hist'
```

References

MATIAS, C., REBAFKA, T. & VILLERS, F. (2018). A semiparametric extension of the stochastic block model for longitudinal networks. Biometrika. 105(3): 665-680.

---

generated_sol_kernel *Output example of mainVEM*

Description

Output of mainVEM obtained on dataset `generated_Q3` with `kernel` method and Qmin=Qmax=5.

Usage

```
generated_sol_kernel
```

Format

Solution for Q=5 clusters, containing 5 components:

`tau` Matrix with size $Q \times n$ containing the estimated probability in $(0, 1)$ that cluster q contains node i.

`J` Estimated value of the ELBO

`run` Which run of the algorithm gave the best solution. A run relies on a specific initialization of the algorithm. A negative value maybe obtained in the decreasing phase (for Q) of the algorithm.

`converged` Boolean. If TRUE, the algorithm stopped at convergence. Otherwise it stopped at the maximal number of iterations.

Details

This solution was (randomly) obtained using the following code

```
# WARNING - This is very long
generated_sol_kernel <- mainVEM(generated_Q3$data,n=50,Qmin=5,directed=FALSE,method='kernel')[[1]]
```

References

MATIAS, C., REBAFKA, T. & VILLERS, F. (2018). A semiparametric extension of the stochastic block model for longitudinal networks. Biometrika. 105(3): 665-680.

---

generatePP *Poisson process generator*

Description

Generates one realization of an inhomogeneous Poisson process (PP) with given intensity function (using the thinning method).

Usage

```
generatePP(intens, Time, max.intens)
```

Arguments

intens	Intensity function defined on [0,Time] (needs to be positive).
Time	Positive real number. [0,Time] is the total time interval of observation.
max.intens	Positive real number. Upper bound of intensity on [0,Time].

Value

Vector with the values of one realization of the PP.

Examples

```
# Generate a Poisson Process on time interval [0,30] with intensity function
# intens= function(x) 100*x*exp(-8*x)
# using max.intens = 5

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

generatePP(intens, Time=30, max.intens=5)
```

generatePPConst *Poisson process with piecewise constant intensity*

Description

Generates one realization of a Poisson process (PP) with a piecewise constant intensity function.

Usage

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

Arguments

intens	Vector with the constant values of the intensity, defined on a regular partition of the time interval [0,Time].
Time	Positive real number. [0,Time] is the total time interval of observation.

Examples

```
# On time interval [0,T], partitioned into 3 regular parts: [0,T/3], [T/3,2T/3] and [2T/3,T],
# define a piecewise constant intensity function
intens <- c(1,2,8)

# generate a PP with total observation time T=10
generatePPConst(intens, 10)
```

kernelIntensities *Direct kernel estimator intensities*

Description

Compute smooth intensities with direct kernel estimation of intensities relying on a classification τ. This can be used with the values τ obtained on a dataset with mainVEM function.

Usage

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

Arguments

`data`	List with 3 components: • `time.seq` - Vector of observed time points of the events (length M). • `type.seq` - Vector of observed types of node pairs (as encoded through convertNodePair of the events (length M)). • `Time` - [0,Time] is the total time interval of observation.
`tau`	Matrix with size $Q \times n$ and values in $(0, 1)$, containing the (estimated) probability that cluster q contains node i.
`Q`	Total number of groups.
`n`	Total number of nodes, $1 \leq i \leq n$.
`directed`	Boolean for directed (TRUE) or undirected (FALSE) case
`rho`	Either 1 (non sparse case) or vector with length $Q(Q+1)/2$ (undirected case) or Q^2 (directed case) with (estimated) values for the sparsity parameters $\rho^{(q,l)}$. See Section S6 in the supplementary material paper of Matias et al. (Biometrika, 2018) for more details.
`sparse`	Boolean for sparse (TRUE) or not sparse (FALSE) case.
`nb.points`	Number of points for the kernel estimation.

Details

Warning: sparse case not implemented !!!

Examples

```
# The generated_sol_kernel solution was generated calling mainVEM
# with kernel method on the generated_Q3$data dataset.
# (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` *List node pairs*

Description

Create the list of all node pairs

Usage

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

Arguments

n	Total number of nodes, $1 \leq i \leq n$
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 *Adaptive VEM algorithm*

Description

Principal adaptive 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 **hist** method - List with 2 components:
   - **data$Time** Positive real number. [0,data$Time] is the total time interval of observation
   - **data$Nijk** Data matrix with counts per process N_{ij} and sub-intervals ; matrix of size $N * K$ where $N = n(n-1)$ or $n(n-1)/2$ is the number of possible node pairs in the graph and $K = 2^{dmax}$ is the size of the finest partition in the histogram approach.
     Counts are pre-computed - Obtained through function statistics on data with second format and using a number of subintervals K as a power of 2.
2. Data with **kernel** method - List with 3 components:
   - **data$time.seq** Vector of the time points of the events (size M).
   - **data$type.seq** Vector of the corresponding node pair indexes in format output by convertNodePair of the events (same size M).
   - **data$Time** [0,data$Time] is the total time interval of observation

`n` Total number of nodes, $1 \leq i \leq n$

`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

`method` Either `hist` for histogram method or `kernel` for kernel method

Beware: `hist` is recommended (much faster). You can obtain smooth estimated intensities by using kernelIntensities on the output of the `hist` method.

`init.tau` List of initial values of τ - 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.

- Algorithm takes partition up to depth 2^d with $d = 1, ..., d_{part}$
- Explore partitions $[0,T], [0,T/2], [T/2,T], ...[0,T/2^d], ...[(2^d-1)T/2^d, T]$
- Total number of partitions $npart = 2^{(d_{part}+1)} - 1$

`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 $npart * (1 + n_{perturb}) + n_{random}$

`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.

Value

The function outputs a list of Qmax-Qmin+1 components. Each component is the solution obtained for a number of clusters Q, with $Qmin \leq Q \leq Qmax$ and is a list of 8 elements:

- `tau` - Matrix with size $Q \times n$ containing the estimated values in $(0, 1)$ that cluster q contains node i.
- `rho` - When method=`hist` only. Either 1 (non sparse method) or a vector with length $Q(Q+1)/2$ (undirected case) or Q^2 (directed case) with estimated values for the sparsity parameters $\rho^{(q,l)}$. See Section S6 in the supplementary material paper of Matias et al. (Biometrika, 2018) for more details.
- `beta` - When method=`hist` only. Vector with length $Q(Q+1)/2$ (undirected case) or Q^2 (directed case) with estimated values for the sparsity parameters $\beta^{(q,l)}$. See Section S6 in the supplementary material paper Matias et al. (Biometrika, 2018) for more details.
- `logintensities.ql` - When method=`hist` only. Matrix with size $Q(Q+1)/2 \times K$ (undirected case) or $Q^2 \times K$ (directed case). Each row contains estimated values of the log intensity function $\log(\alpha^{(q,l)})$ on a regular partition with K parts of the time interval [0,Time].
- `best.d` - When method=`hist` only. Vector with length $Q(Q+1)/2$ (undirected case) or Q^2 (directed case) with estimated value for the exponent of the best partition to estimate intensity $\alpha^{(q,l)}$. The best number of parts is $K = 2^d$.
- `J` - Estimated value of the ELBO.
- `run` - Which run of the algorithm gave the best solution. A run relies on a specific initialization of the algorithm. A negative value maybe obtained in the decreasing phase (for Q) of the algorithm.
- `converged` - Boolean. If TRUE, the algorithm stopped at convergence. Otherwise it stopped at the maximal number of iterations.

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. 105(3): 665-680.

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 (ie nb of parts K=2^{d_max}=8).
K <- 2^3
Nijk <- statistics(data,n,K,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$logintensities.ql,z,sol.hist$tau,
     directed=FALSE)

# plot estimators
par(mfrow=c(2,3))
ind.ql <- 0
for (q in 1:Q){
  for (l in q:Q){
    ind.ql <- ind.ql + 1
    true.val <- intens[[ind.ql]]$intens(x0)
    values <- c(intensities.kernel[ind.ql,],exp(log.intensities.hist[ind.ql,]),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/K),c(exp(log.intensities.hist[ind.ql,]),
        exp(log.intensities.hist[ind.ql,K])),type='s',col=2,lty=2)
    lines(seq(0,1,by=.001),intensities.kernel[ind.ql,],col=4,lty=3)
  }
}
```

`modelSelection_Q` *Selects the number of groups with ICL criterion*

Description

Selects the number of groups with Integrated Classification Likelihood (ICL) criterion.

Usage

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

Arguments

`data`	List with 2 components: • `Time` - Positive real number. [0,Time] is the total time interval of observation. • `Nijk` - Data matrix with the statistics per process N_{ij} and sub-intervals $1 \leq k \leq K$.
`n`	Total number of nodes, $1 \leq i \leq n$.
`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 on the data with method='hist'.

Value

The function outputs a list of 7 components:

- `Qbest` Selected value of the number of groups in [Qmin, Qmax].
- `sol.Qbest` Solution of the mainVEM function for the number of groups Qbest.
- `Qmin` Minimum number of groups used.
- `all.J` Vector of length Qmax-Qmin+1. Each value is the estimated ELBO function J for estimation with Q groups, $Qmin \leq Q \leq Qmax$.

- `all.ICL` Vector of length Qmax-Qmin+1. Each value is the ICL value for estimation with Q groups, $Qmin \leq Q \leq Qmax$.
- `all.compl.log.likelihood` Vector of length Qmax-Qmin+1. Each value is the estimated complete log-likelihood value for estimation with Q groups, $Qmin \leq Q \leq Qmax$.
- `all.pen` Vector of length Qmax-Qmin+1. Each value is the penalty term in ICL for estimation with Q groups, $Qmin \leq Q \leq Qmax$.

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. 105(3): 665-680.

Examples

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

# compute data matrix of counts per subinterval with precision d_max=3
# (ie nb of parts K=2^{d_max}=8).
K <- 2^3
data <- list(Nijk=statistics(generated_Q3$data,n,K,directed=FALSE),
    Time=generated_Q3$data$Time)

# ICL-model selection with groups ranging from 1 to 4
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 the Integrated Classification Likelihood (ICL) criterion, the Complete Log-Likelihood (CLL) and the ELBO (J criterion).

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 of counts per subinterval with precision d_max=3 :
# (ie nb of parts K=2^{d_max}=8).
K <- 2^3
data <- list(Nijk=statistics(generated_Q3$data,n,K,directed=FALSE),
    Time=generated_Q3$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)
```

---

sortIntensities *Sort intensities*

Description

Sort intensities associated with the estimated clustering $\hat{z}$ "in the same way" as the original intensities associated with true clustering z by permutation of rows.

Usage

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

Arguments

intensities	Matrix whose rows contain piecewise constant intensities $\alpha^{(q,l)}$, given as a vector of values on a regular partition of the time interval.
z	Matrix of size $Q \times n$.
hat.z	Matrix 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. 105(3): 665-680.

Examples

```
# True and estimated clusters for n=6 nodes clustered into Q=3 groups
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)

# Set constant intensities for each directed pair (q,l)
intens <- matrix(c(1,1,1,2,2,2,3,3,3),9)

# Permute the rows according to the permutation that "matches" hat.z with z
sortIntensities(intens,z,hat.z, TRUE)
```

`statistics` *Compute statistics*

Description

Convert the initial data into the statistics matrix $N_{(ij),k}$, by counting the number of events for the nodes pair types (i, j) during the k-th subinterval of a regular partition (in K parts) of the time interval.

Usage

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

Arguments

`data`	List with 3 components `$type.seq`, `$time.seq`, `$Time` (see mainVEM for more details).
`n`	Total number of nodes, $1 \leq i \leq n$.
`K`	Size of the regular partition, i.e. number of subintervals of the time interval. When used as input in the VEM algorithm (with `hist` method), K must be a power of 2.
`directed`	Boolean for directed (TRUE) or undirected (FALSE) case.

Value

N[(i,j),k] = matrix with K columns, each row contains the number of events for the node pair (i, j) during the k-th subinterval

Examples

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

n <- 50
K <- 2^3

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

Index

∗ **datasets**
generated_Q3, 10
generated_Q3_n20, 11
generated_sol_hist, 12
generated_sol_kernel, 13

ARI, 2

bootstrap_and_CI, 3

convertGroupPair, 5
convertNodePair, 6, *16*, *18*

find_ql, 7

generated_Q3, 10
generated_Q3_n20, 11
generated_sol_hist, 12
generated_sol_kernel, 13
generateDynppsbm, 8
generateDynppsbmConst, 9
generatePP, 14
generatePPConst, 15

kernelIntensities, 15, *18*

listNodePairs, 16

mainVEM, *3*, *12, 13*, *15*, 17, *21*, *24*
modelSelec_QPlot, 22
modelSelection_Q, 21, *23*

sortIntensities, 23
statistics, *18*, 24