

Package ‘mixer’

November 19, 2009

Type Package

Title Random graph clustering

Version 1.0

Date 2009-10-15

Author Christophe Ambroise, Gilles Grasseau, Mark Hoebeke, Pierre Latouche, Vincent Miele,
Franck Picard

Maintainer Gilles Grasseau <mixer@genopole.cnrs.fr>

Description Routines for the analysis (unsupervised clustering) of networks using MIXtures of
Erdos-Renyi random graphs

URL <http://ssbgroup.fr/mixnet/mixer.html>, <http://ssbgroup.fr>

License GPL (>= 2)

Depends R (>= 2.5.0), methods

Encoding UTF-8

Repository CRAN

Date/Publication 2009-11-19 18:33:11

R topics documented:

blog	2
getModel	3
getModel.mixer	3
graph.affiliation	4
macaque	5
mixer	6
plot.mixer	8
setSeed	10

Index	12
--------------	-----------

blog

French Political Blogosphere network

Description

French Political Blogosphere network dataset consists of a single day snapshot of over 1,100 political blogs automatically extracted the 14 October 2006 and manually classified by the "Observatoire Presidentielle" project. This project is the result of a collaboration between RTGI SAS and Exalead and aims at analyzing the French presidential campaign on the web.

Usage

```
data(blog)
```

Format

196 political blogs described by their political origin and connections. There are 2 items in the list `blog`:

links adjacency matrix describing if there exist an hyperlink between two blogs.

politicalParty political group of the blogs.

Details

In this data set, nodes represent hostnames (a hostname contains a set of pages) and edges represent hyperlinks between different hostnames. If several links exist between two different hostnames, they were collapsed into a single one. Note that intra domain links can be considered if hostnames are not identical. Finally, in this experimentation we consider that edges are not oriented which is not realistic but which does not affect the interpretation of the groups. This network presents an interesting communities organization due to the existence of several political parties and commentators. We assume that authors of these blogs tend to link, by political affinities, blogs with similar political positions. Six known communities compose this network : **Gauche** ("french democrat"), **Divers Centre** (Moderate party), **Droite** (french republican), **Ecologiste** (green), **Liberal** (supporters of economic-liberalism) and finally **Analysts**.

References

<http://www.observatoire-presidentielle.fr/>

Examples

```
data(blog)
mixer(x=blog$links, qmin=2, qmax=12) -> xout
## Not run: plot(xout)
```

getModel	<i>Get the parameters of a model</i>
----------	--------------------------------------

Description

Generic method to get the parameters of a model

Usage

```
getModel( object, ...)
```

Arguments

object	an object representing a model
...	adding arguments (depending on the object type)

Value

Return parameters of the model.

getModel.mixer	<i>Get the model parameters</i>
----------------	---------------------------------

Description

Given a number of classes, get the model parameters (alphas, Pis, Taus).

Usage

```
## S3 method for class 'mixer':
getModel( object, ...)
```

Arguments

object	a mixer object (output of the mixer function).
...	adding arguments (depending on the object type)

q selects a q-class model. If NULL, q value is chosen to maximise the criterion (ICL criterion or ILvb criterion) (see [plot.mixer](#) and [mixer](#)). Default NULL.

Value

Return a list with the following attributes:

q	q-class model selected.
criterion	criterion value (ICL criterion or ILvb criterion).
alphas	vector of class proportion.
Pis	connectivity matrix of classes.
Taus	matrix of posterior probabilities (of the hidden colour knowing the graph structure).

Author(s)

G. Grasseau

Examples

```
graph.affiliation(n=100,c(1/3,1/3,1/3),0.8,0.2)->g
mixer(g$x,qmin=2,qmax=6)->xout
getModel(xout)
```

graph.affiliation *Simulation of an Affiliation Graph*

Description

Simulate an affiliation Graph with a given number of clusters, specific class proportions and within/between connection probabilities.

Usage

```
graph.affiliation( n=100, alphaVect=c(1/2,1/2), lambda=0.7,
                  epsilon=0.05, directed=FALSE )
```

Arguments

n	number of nodes of the simulated Graph.
alphaVect	vector of cluster proportions.
lambda	within-cluster probability of edge.
epsilon	between-clusters probability of edge.
directed	TRUE for directed graphs.

Details

`graph.affiliation` simulates a simple Erdos-Renyi Mixture of Graph model, using the same within-cluster edge probability for all clusters and a unique between-cluster edge probability.

Value

`graph.affiliation` returns a list of 2 objects:

`x` an adjacency matrix of size n by n,
`cluster` a vector of integers indicating the cluster to which each node is allocated.

Author(s)

Christophe Ambroise

Examples

```
graph.affiliation(n=100,c(1/3,1/3,1/3),0.8,0.2)->g
str(g)
```

macaque

Connection of macaque brain cortical regions

Description

The dataset consists in 47 brain cortical regions connected by 505 inter-regional pathways in the Macaque Cortex.

Usage

```
data(macaque)
```

Format

A data frame describing the adjacency matrix of the connection of the 47 brain cortical regions of the Macaque Cortex

Details

As brain function is based on inter-regional connections, studying the way cortical regions interact may offer new perspectives in the comprehension of information flows within the brain. It appears that particular brain regions may play different roles: some regions can be at the "center" of a particular part of the network, meaning that a lot of information will pass through them, whereas other parts of the network may be more "peripherica". Consequently, identifying central zones would be important, as their lesion may compromise the integrity of the whole network. From a topological view, finding those "hubs" as focused much attention, with a definition based on degree only. However, there exists many ways for a node to be a hub, and degree is one criteria. As there is no definition of what a hub is, there are many different hubs (provincial and central). This is why [1]

developed a multi-criteria strategy to find nodes that can be called "hubs". From a methodological point of view, this approach seems to be limited as the resulting hubs will be criteria-dependent. The gain of Mixer is that the model can be used to find those hubs. Indeed, using the underlying missing data framework, MixNet will find nodes that connect heavily to other nodes in the network, and that share this connectivity pattern (a class of hubs for instance).

References

[1] Sporns O., Honey C., and Kotter R. Identification and classification of hubs in brain networks. PLOS one, 10:1-14, 2007.

Examples

```
data(macaque)
mixer(macaque, qmin=8) -> xout
## Not run: plot(xout)
```

mixer *MIXtures of Erdős-Rényi random graphs*

Description

Estimate the parameters, the hidden class variables, as well as the number of classes q of a MIXture of Erdős Rényi random graphs. The estimation is performed for binary graphs (edges are assumed to be drawn from Bernoulli distributions).

Usage

```
mixer(x, qmin=2, qmax=NULL, method="variational", directed = NULL,
      nbiter=10, fpnbiter=5, improve=FALSE)
```

Arguments

<code>x</code>	an adjacency matrix or a matrix of edges (each column gives the two node indexes defining an edge) or a spm file name (a <code>.spm</code> file describes the network as a sparse matrix).
<code>qmin</code>	minimum number of classes.
<code>qmax</code>	maximum number of classes (if <code>NULL</code> , only $q=qmin$ is considered).
<code>method</code>	strategy used for the estimation: "variational", "classification", or "bayesian"
<code>directed</code>	TRUE/FALSE for directed/undirected graph. Default is <code>NULL</code> , i.e. according to the input array <code>x</code> , <code>mixer</code> identifies whether the graph is directed or undirected.
<code>nbiter</code>	maximum number of EM iterations (Default: 10).
<code>fpnbiter</code>	maximum number of internal iterations for the E step (Default: 5).
<code>improve</code>	selects between improved or basic strategies (Default: <code>FALSE</code>).

Details

`mixer` implements the Erdős-Rényi mixture model for graphs (called MixNet) which has been proposed by Daudin et. al (2008) with an associated EM estimation algorithm. The MixNet model is well suited to capture the structure of a network and in particular to detect communities.

MixNet must not be confused with Exponential Random Graph Models for Network Data (ERGM) which considers distributions ensuing from the exponential family to model the edge distribution.

There exists a strong connection between Mixnet and block clustering. Block clustering searches for homogeneous blocks in a data matrix by simultaneous clustering of rows and columns.

The `mixer` package implements three different estimation strategies which were developed to deal with directed and undirected graphs:

variational refers to the paper of Daudin et. al (2008). It is the default method.

classification implements the method described in Zanghi et. al (2008). This method is faster than the variational approach and is able to deal with bigger networks but can produce biased estimates.

bayesian implements the method described in Latouche et. al (2008).

The implementation of the two first methods consists of an R wrapper of the c++ software package `mixnet` developed by Vincent Miele (2006).

The `mixer` routine uses the estimation strategy described in `method` and computes a model selection criterion for each value of q (the number of classes) between `qmin` and `qmax`. The ICL criterion is used for the `variational` and `classification` methods. It corresponds to an asymptotic approximation of the Integrated Classification Likelihood. The other criterion, so called ILvb (Integrated Likelihood variational Bayes), is used for the `bayesian` method. It is based on a variational (non-asymptotic) approximation of the Integrated observed Likelihood.

`mixer` is an user-friendly package with a reduced number of functions. For R-developers in statistical networks a more complete set, called `mixer-dev`, is provided (see below).

Value

`mixer` returns an object of class `mixer`. Below the main attributes of this class:

<code>nnodes</code>	number of connected nodes.
<code>map</code>	mapping from connected nodes to the whole set of nodes.
<code>edges</code>	edge list.
<code>qmin, qmax</code>	number of classes.
<code>output</code>	output list of <code>qmax-qmin+1</code> items. Each item contains the result of the estimation for a given number of class q . Details of output field:
<code>output[[i]]\$criterion</code>	ICL criterion or ILvb criterion used for model selection (see details section for more).
<code>output[[i]]\$alphas</code>	vector of proportion, whose length is the number of component.
<code>output[[i]]\$Pis</code>	class connectivity matrix.

```
output[[i]]$Taus
      matrix of posterior probabilities (of the hidden color knowing the graph structure).
```

Author(s)

Christophe Ambroise, Gilles Grasseau, Mark Hoebeke, Pierre Latouche, Vincent Miele, Franck Picard

References

Jean-Jacques Daudin, Franck Picard and Stephane Robin June (2008), *A mixture model for random graphs*. *Statistics and Computing*, 18, 2, 151–171.

Hugo Zanghi, Christophe Ambroise and Vincent Miele (2008), *Fast online graph clustering via Erdős-Rényi mixture*. *Pattern Recognition*, 41, 3592-3599.

Hugo Zanghi, Franck Picard, Vincent Miele, and Christophe Ambroise (2008), *Strategies for Online Inference of Network Mixture*,

<http://arxiv.org/abs/0910.2034v1>

Pierre Latouche, Etienne Birmele, and Christophe Ambroise (2008), *Bayesian methods for graph clustering*,

<http://genome.jouy.inra.fr/ssb/preprint/SSB-RR-17.bayesianMixNet.pdf>

Vincent Miele, MixNet C++ package,

<http://stat.genopole.cnrs.fr/sg/software/mixer/>.

mixer-dev tool: see <http://ssbgroup.fr/mixnet/mixer.html>

Examples

```
graph.affiliation(n=100,c(1/3,1/3,1/3),0.8,0.2)->g
mixer(g$x,qmin=2,qmax=6)->xout
## Not run: plot(xout)
```

```
graph.affiliation(n=50,c(1/3,1/3,1/3),0.8,0.2)->g
mixer(g$x,qmin=2,qmax=5,method="bayesian")->xout
## Not run: plot(xout)
```

```
data(blog)
mixer(x=blog$links,qmin=2,qmax=12)->xout
## Not run: plot(xout)
```

plot.mixer

Plot of mixer object

Description

plot.mixer can display five kinds of figure: Integrated Classification Criterion curve, the adjacency matrix map, the degree distribution histogram, the connectivity matrix graph and the adjacency matrix graph. By default the four first plots are displayed.

Usage

```
## S3 method for class 'mixer':
plot(x, q=NULL, frame=1:4, classes=NULL, quantile.val=0.1, ...)
```

Arguments

<code>x</code>	a mixer object (output of the mixer function).
<code>q</code>	the q-class model to display. By default, the <code>q</code> is set to the value which maximizes the criterion (see frame 1).
<code>frame</code>	a vector of frame numbers to display (5 kinds of plots, see details section for more).
<code>classes</code>	an external classification used for frame 4 (pie chart): vector as <code>factor</code> of node elements (the number of external class levels corresponds to the number of levels).
<code>quantile.val</code>	filters the connectivity matrix values (<code>Pis</code>) in frame 4. Display the upper part (specified by <code>quantile.val</code>) of the distribution.
<code>...</code>	further graphical arguments.

Details

Frame values:

- 1 criterion (ICL or ILvb) versus the number of classes (see [mixer](#)).
- 2 adjacency matrix reorganized according to the estimated partition for a given number of classes `q`.
- 3 degree distribution (histogram) and theoretical degree distribution (blue curve) computed from the q-class model parameters (`alphas`, `Pis`).
- 4 matrix connectivity between classes (`Pis`) given a number of classes `q`. The thickest edges identify the highest values of the connectivity probabilities and the largest nodes point out the most populated classes.
Providing external classes (see **classes** argument) each node displays a pie chart pointing out the classification relevance.
- 5 graph display of the adjacency matrix.

Author(s)

C. Ambroise, G. Grasseau

See Also

[mixer](#), [getModel](#)

Examples

```
#
# Simple example : display the 4 frames for the best class number estimation
#
g <- graph.affiliation(n=100,c(1/3,1/3,1/3),0.8,0.2)
xout <- mixer(g$x,qmin=2,qmax=6)
## Not run: plot(xout)

#
# Display the same for 4 classes with no filtering
#
## Not run: plot(xout, q=4, quantile.val=0)

#
# Display a pie chart for 4 classes
#
data(blog)
xout <- mixer(x=blog$links,qmin=2,qmax=12)
# Unconnected nodes have been removed by mixer.
# xout$map contains the mapping from connected nodes to the whole set
ext.classes <- blog$politicalParty[ xout$map ]
## Not run: plot( xout, frame=4, classes=ext.classes )
```

setSeed

Set internal seed

Description

This utility sets the internal random seed used by mixer.

Usage

```
setSeed( seed = 1 )
```

Arguments

seed sets the seed of the internal random generator in the mixer C/C++ libraries (integer value).

Details

Sets the seed of the random generator (`srand`) in the C standard library. This random generator is used inside the mixer initialization stage. This function is useful to generate exactly the same initial conditions before two `mixer` runs.

Author(s)

G. Grasseau

Examples

```
graph.affiliation(n=100,c(1/3,1/3,1/3),0.8,0.2)->g
setSeed(777)
mixer(g$x,qmin=2,qmax=6)->xout
## Not run: plot(xout)

# Produce strictly the same result
setSeed(777)
mixer(g$x,qmin=2,qmax=6)->xout
## Not run: plot(xout)
```

Index

*Topic **cluster**

- getModel, 3
- getModel.mixer, 3
- mixer, 6
- setSeed, 10

*Topic **datasets**

- blog, 2
- macaque, 5

*Topic **graphs**

- getModel, 3
- getModel.mixer, 3
- graph.affiliation, 4
- mixer, 6
- plot.mixer, 8
- setSeed, 10

blog, 2

getModel, 3, 9
getModel.mixer, 3
graph.affiliation, 4

macaque, 5
mixer, 3, 6, 9

plot.mixer, 3, 8

setSeed, 10