Package 'SBMSplitMerge'

January 20, 2025

Title Inference for a Generalised SBM with a Split Merge Sampler

Version 1.1.1

Description Inference in a Bayesian framework for a generalised stochastic block model. The generalised stochastic block model (SBM) can capture group structure in network data without requiring conjugate priors on the edge-states. Two sampling methods are provided to perform inference on edge parameters and block structure: a split-merge Markov chain Monte Carlo algorithm and a Dirichlet process sampler. Green, Richardson (2001) <doi:10.1111/1467-9469.00242>; Neal (2000) <doi:10.1080/10618600.2000.10474879>; Ludkin (2019) <arXiv:1909.09421>.

Depends R (>= 3.1.0)

License MIT + file LICENSE

Language en-GB

LazyData true

RoxygenNote 7.1.0

Imports ggplot2, scales, reshape2

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author Matthew Ludkin [aut, cre, cph]

Maintainer Matthew Ludkin <m.ludkin1@lancaster.ac.uk>

Repository CRAN

Date/Publication 2020-06-04 13:30:05 UTC

Contents

`accept` *accept* `propsbm` *with the acceptance probability alpha*

Description

accept `propsbm` with the acceptance probability alpha

Usage

```
accept(currsbm, propsbm, edges, sbmmod, logjac = 0, logu = 0, ...)
```

Arguments

currsbm	current sbm state
propsbm	proposed sbm state
edges	an edges object
sbmmod	an sbmmod model
logjac	log Jacobian of transformation of variables
logu	log density for auxiliary variables
...	additional arguments to pass to dedges

Value

updated sbm object

---

addblock *Add a block move*

Description

proposes adding an empty block labelled kappa+1 to sbm

Usage

```
addblock(sbm, edges, sbmmod, rho = 1)
```

Arguments

sbm	the current state of the sampler
edges	an edges object
sbmmod	an sbmmod model
rho	probability of choosing to add a block

Value

an updated sbm object

ARI *Adjusted Rand Index*

Description

Calculate the Adjusted Rand Index between two clusterings

Usage

```
ARI(z, truez)
```

Arguments

z	input vector
truez	reference vector

Value

Adjusted Rand Index of z against truez

Examples

```
ARI(c(1,1,2,2,3,3), c(2,2,1,1,3,3)) ## 1 - doesn't care for labels
ARI(c(1,1,2,2,3,3), c(1,1,1,1,2,2)) ## 0.444
```

blockmat *Block matrix*

Description

converts x to a matrix of block assignments

Usage

```
blockmat(x, ...)
```

Arguments

x	object for dispatch
...	additional arguments for method

Value

matrix of block assignment indicators

See Also

blockmat.sbm blockmat.blocks blockmat.numeric

blockmat.blocks *Block matrix*

Description

converts block assignments of a `blocks` object to a matrix of block assignments

Usage

```
## S3 method for class 'blocks'
blockmat(blocks, kappa)
```

Arguments

`blocks`	a `blocks` object
`kappa`	number of blocks in matrix

Value

matrix with `kappa` rows and a 1 at `(k,i)` if node `i` is in block `k` under `blocks`

blockmat.numeric *Block matrix*

Description

converts a vector of block assignments to a matrix of block assignments

Usage

```
## S3 method for class 'numeric'
blockmat(x, kappa)

## S3 method for class 'factor'
blockmat(x, kappa)
```

Arguments

`x`	a numeric-vector of node-to-block assignments
`kappa`	number of blocks

Value

matrix with `kappa` rows and a 1 at `(k,i)` if node `i` is in block `k` under `x`

blockmat.sbm *Block matrix*

Description

converts block assignments of an sbm object to a matrix of block assignments

Usage

```
## S3 method for class 'sbm'
blockmat(SBM, kappa)
```

Arguments

SBM	an sbm object
kappa	number of blocks in matrix

Value

matrix with kappa rows and a 1 at (k,i) if node i is in block k under SBM

blockmod *Block Model*

Description

create a blockmod object

Usage

```
blockmod(fixkappa, logd, dcond, r, ...)
```

Arguments

fixkappa	Logical - is kappa fixed or can it vary under the model?
logd	function(blocks) - log density for blocks
dcond	function(blocks, i) - conditional density for the block assignment i in blocks
r	function(n), sorted=FALSE - samples a blocks object from the model
...	parameters of the model for use in r, logd, dcond

Details

A block model is a probability model for a blocks object. This class creates a closure with three functions: - a random method for sampling block a structure from the model with n nodes; a - a log-density method for computing the log-density of a given block structure in a blocks object - a conditional density function that takes a blocks object and a node i

Value

a blockmod object

See Also

multinom dma crp blocks

---

blocks *Blocks object*

Description

create a blocks object

Usage

```
blocks(z, kappa)
```

Arguments

z	vector of block labels for each node
kappa	maximum number of blocks

Details

stores the block allocations and total number of blocks for a stochastic block model

Value

a blocks object

Examples

```
## Assign six nodes to four blocks:
b <- blocks(c(1,1,2,3,4,4), 4)
print(b)
plot(b) ## shows id two nodes are members of the same block
```

blocktrace *plot a trace of the blocks from MCMC samples*

Description

plot a trace of the blocks from MCMC samples

Usage

```
blocktrace(postz, burnin)
```

Arguments

postz	output from sampler
burnin	which iterations to plot? defaults to all.

Value

'ggplot2' object

crp *Chinese Restaurant Process*

Description

A blockmod for the Chinese restaurant process (CRP)

Usage

```
crp(gamma)
```

Arguments

gamma	concentration parameter

Details

The CRP posits that each node arrives in turn. The first node joins the first block. Each subsequent node starts a new block with probability 'gamma' or joins an existing block proportional to the block size.

Value

a block model representing a CRP(gamma) distribution

Examples

```
## simulate from a CRP(5) prior
m <- crp(5)
print(m)
m$r(10)
```

`ddirichlet` *Dirichlet distribution*

Description

Density of Dirichlet distribution

Usage

```
ddirichlet(x, gam, log = FALSE)
```

Arguments

`x`	random variable in the d-dimensional simplex
`gam`	a length K concentration parameter
`log`	return the log-probability instead?

Value

the density

Examples

```
g <- rep(2,5)
p <- rdirichlet(1, g) ## a length-5 probability vector
ddirichlet(p, g)
```

`dedges` *Density of edges*

Description

Compute the probability density for an `edges` object

Usage

```
dedges(x, edges, edgemod, na.rm = TRUE, ...)
```

Arguments

x	an R object for dispatch
edges	an edges object
edgemod	an edgemod object
na.rm	remove NAs when calculating?
...	additional arguments

Value

matrix same size as edges$E with density of each edge

See Also

dedges.sbm dedges.sbm

---

dedges.numeric *likelihood of edges*

Description

likelihood of edges

Usage

```
## S3 method for class 'numeric'
dedges(x, edges, edgemod, na.rm = na.rm, ...)
```

Arguments

x	a matrix of parameters (with same size as edges$E)
edges	an edges object
edgemod	an edgemod object
na.rm	remove NAs when calculating?
...	additional arguments passed to edgemod$logd

Value

likelihood of edges under the edgemod using parameters in matrix pmat

dedges.sbm *Density of edges*

Description

Compute the probability density for an edges object under an sbm object

Usage

```
## S3 method for class 'sbm'
dedges(x, edges, edgemod, na.rm = TRUE, ...)
```

Arguments

x	an sbm object
edges	an edges object
edgemod	an edgemod object
na.rm	remove NAs when calculating?
...	additional arguments for dedges.params

Value

matrix same size as edges$E with density of each edge

Examples

```
## make an sbm model, sample data then plot and print:
model <- sbmmod(dma(2,5), param_beta(1,1,1,1), edges_bern())
s <- model$r(100)
e <- redges(s, model$edge)
dedges(s, e, model$edge)
```

delblock *Delete a block move*

Description

proposes deleting an empty block (chosen at random among empty Blocks)

Usage

```
delblock(sbm, edges, sbmmod, rho = 1)
```

Arguments

sbm	the current state of the sampler
edges	an edges object
sbmmod	an sbmmod model
rho	probability of choosing to add a block

Value

an updated sbm object

---

dma *Dirichlet Multinomial Allocation*

Description

A blockmod for Dirichlet Multinomial Allocation (DMA)

Usage

```
dma(gamma, delta)
```

Arguments

gamma	parameter for Dirichlet component
delta	parameter for Poison component

Details

This model posits:

$$kappa - 1 \; Pois(delta)$$

$$omega|kappa, gamma \; Dirichlet(gamma)$$

$$Z_i|omega \; Multinomial(omega) for i = 1..n$$

Value

a block model representing a dma(gamma, delta) distribution

Examples

```
## simulate from a DMA(2, 5) prior
## This models the `number of blocks-1` as Poisson(5)
## and block assignments as Dirichlet-Multinomial(2, 2, ...)
m <- dma(2, 5)
print(m)
m$r(10)
```

drawblock.dp *Draw block membership*

Description

Draw block membership in a Dirichlet process sampler

Usage

```
drawblock.dp(i, currsbm, edges, sbmmod)
```

Arguments

i	node to update
currsbm	current sbm object
edges	an edges object
sbmmod	an sbmmod object

Details

sample a new block assignment for i under a Dirichlet process. Care needs to be taken with singleton blocks to update the parameter model in currsbm.

Value

updated sbm object

See Also

For full algorithm details see http://doi.org/10.17635/lancaster/thesis/296

drawblock.gibbs *Gibbs-like reassignment of nodes to the current set of blocks*

Description

Reassign node ‘i‘ to the current set of blocks given the current number of blocks and the other block assignments

Usage

```
drawblock.gibbs(i, currsbm, edges, sbmmod)
```

Arguments

i	the node to reassign
currsbm	an sbm object
edges	an edges object
sbmmod	an sbmmod object

Value

updated sbm object with new block assignment for i

---

drawblocks.dp *Draw block memberships*

Description

Draw block memberships in a Dirichlet process sampler

Usage

```
drawblocks.dp(currsbm, edges, sbmmod)
```

Arguments

currsbm	current sbm object
edges	an edges object
sbmmod	an sbmmod object

Details

iteratively updates the block assignment of each node using a Dirichlet process update move

Value

updated sbm object

drawblocks.gibbs *Gibbs-like reassignment of nodes to the current set of blocks*

Description

Sweep through the set of nodes and reassign to the current set of blocks given the current number of blocks

Usage

```
drawblocks.gibbs(currsbm, edges, sbmmod)
```

Arguments

currsbm	an sbm object
edges	an edges object
sbmmod	an sbmmod object

Value

updated sbm object with new block assignments

drawparams *Metropolis updates by drawing parameters*

Description

Simulate parameters for the given model with a Metropolis-Hastings step

Usage

```
drawparams(sbm, edges, sbmmod, sigma = 0.1)
```

Arguments

sbm	current sbm object
edges	an edges
sbmmod	an sbmmod
sigma	parameter for drawparam

Details

iterate through the parameters in currsbm and update.

Value

updated sbm object

edgemod *Class for edge models*

Description

A class with a random and density method for edges objects

Usage

```
edgemod(logd, r, ...)
```

Arguments

logd	function(e, p) to calculate likelihood of edge an edge e given parameter array p
r	function(p) - simulate an edge given a parameter p (optional)
...	additional arguments to append to edgemod internal list

Value

an edgemod object

Note

the parameter for logd is an array of c(dimension of theta, dim(E)) e.g. from parammat

See Also

edges_bern edges_pois edges_norm

edges *Class for edge data*

Description

A class to hold edge data

Usage

```
edges(e, sym, loops, ...)
```

Arguments

e	a matrix or array representing the raw edge-state data
sym	is the network symmetric? (e[ji] = e[ji])
loops	does the network contain self-loops? (edges from node i to i)
...	additional arguments to append to edges internal list

Value

an edges object

Examples

```
## make an sbm model, sample data then plot and print:
model <- sbmmod(dma(2,5), param_beta(1,1,1,1), edges_bern())
s <- model$r(100)
e <- redges(s, model$edge)
plot(e)
plot(e, s)
print(e)
```

---

edges_bern *Bernoulli edge model*

Description

Make an edgemod model with Bernoulli edge-states

Usage

```
edges_bern(...)
```

Arguments

...	additional parameters to pass to rbinom

Value

an edgemod

Examples

```
eb <- edges_bern() ## makes `eb` an edgemod for Bernoulli edge-states
```

edges_nbin *Negative-Binomial edge model*

Description

Make an edgemod model with Negative-Binomial edge-states

Usage

```
edges_nbin(...)
```

Arguments

...	additional parameters to pass to rnbinom

Value

an edgemod

Examples

```
enb <- edges_nbin() ## makes `enb` an edgemod for Negative-Binomial edge-states
```

edges_norm *Normal edge model*

Description

Make an edgemod model with Normal edge-states

Usage

```
edges_norm(...)
```

Arguments

...	additional parameters to pass to rnorm

Value

an edgemod

Examples

```
en <- edges_norm() ## makes `en` an edgemod for Normal edge-states
```

edges_pois *Poisson edge model*

Description

Make an edgemod model with Poisson edge-states

Usage

```
edges_pois(...)
```

Arguments

...	additional parameters to pass to rpois

Value

an edgemod

Examples

```
ep <- edges_pois() ## makes `ep` an edgemod for Poisson edge-states
```

Enron *The Enron data set as extracted from* igraph *using the script in data-raw*

Description

A data set of counts of emails between email addresses This is a non-symmetric network. Nodes represent email address. The edge-state ij between two email addresses i and j is the number of emails sent from i to j The Groups vector is the node label from the igraph attribute "notes"

Usage

```
Enron
```

Format

A list containing

Edges an edges object with each edge-state representing the number of emails between two email addresses

Groups A vector giving a group name to which the email address belong. The order matches the edges such that Edges[i,j] is the edge-state between the nodes i and nodes j who are members of Groups[i] and Groups[j] respectively

Source

https://cran.r-project.org/package=igraphdata

`eval_plots` *get a set of evaluation plots from MCMC samples*

Description

get a set of evaluation plots from MCMC samples

Usage

```
eval_plots(output, burnin, theta_index)
```

Arguments

`output`	from sampler
`burnin`	burn-in period (a vector of iteration numbers to subset outputs)
`theta_index`	which set of thetas to plot?

Value

list of ggplot objects (with descriptive names)

`is.sbm` *is.sbm*

Description

Logical check if an object is an `sbm` object

Usage

```
is.sbm(x)
```

Arguments

`x`	an R object

Value

Logical indicating if `x` is an `sbm` object

Macaque — *The Macaque data set as extracted from* igraph *using the script in data-raw*

Description

The Macaque data set as extracted from igraph using the script in data-raw

Usage

```
Macaque
```

Format

An edges object of activation counts between brain regions in a Macaque

See Also

igraph

marglike_bern — *Marginal likelihood model for Bernoulli distributed edges*

Description

calculate the marginal likelihood for a node for samplers using conjugate models

Usage

```
marglike_bern(znoi, ei, parammod)
```

Arguments

znoi	a matrix of block assignments without node i
ei	edge-states incident to i
parammod	a parammod object representing the Bernoulli-Beta model

Value

log-probability of node i belonging to each block

marglike_norm *Marginal likelihood model for Normal distributed edges*

Description

calculate the marginal likelihood for a node for samplers using conjugate models

Usage

```
marglike_norm(znoi, ei, parammod)
```

Arguments

znoi	a matrix of block assignments without node i
ei	edge-states incident to i
parammod	a parammod object

Value

log-probability of node i belonging to each block

marglike_pois *Marginal likelihood model for Poisson distributed edges*

Description

calculate the marginal likelihood for a node for samplers using conjugate models

Usage

```
marglike_pois(znoi, ei, parammod)
```

Arguments

znoi	a matrix of block assignments without node i
ei	edge-states incident to i
parammod	a parammod object

Value

log-probability of node i belonging to each block

mergeavg *Merge blocks*

Description

Merge-move using an average to merge parameters

Usage

```
mergeavg(sbm, edges, sbmmod, ...)
```

Arguments

sbm	the current state of the sampler
edges	an edges object
sbmmod	an sbmmod model
...	additional parameter to 'accept'

Details

the blocks are chosen at random, the nodes reassigned to the block with the smallest index, then the parameters are combined using the average on the transformed scale

Value

an updated sbm object

mergeblocks *merge move block merging*

Description

merge move block merging

Usage

```
mergeblocks(currblocks, propparams, edges, sbmmod, k, l)
```

Arguments

currblocks	current blocks
propparams	proposed parameters
edges	an edges object
sbmmod	an sbmmod model
k	Blocks to merge
l	Blocks to merge

Value

list(proposed block structure, log-acceptance-prob)

---

mergeparams *merge parameters*

Description

merge parameters

Usage

```
mergeparams(x, ...)
```

Arguments

x	an object to dispatch on
...	additional arguments for methods

Value

merged parameters from x

See Also

mergeparams.default mergeparams.numeric

---

mergeparams.default *Merge step: parameters*

Description

Merge step: parameters

Usage

```
## Default S3 method:
mergeparams(params, k, l, parammod)
```

Arguments

params	a params object
k	Blocks to merge
l	Blocks to merge
parammod	a parammod object

Value

list(proposed_params, log-acceptance-prob)

---

mergeparams.numeric *Merge step - parameter merging*

Description

Merge step - parameter merging

Usage

```
## S3 method for class 'numeric'
mergeparams(thetak, thetal, x, parammod)
```

Arguments

thetak, thetal	parameters to merge
x	auxiliary parameter
parammod	a parammod object

Value

list(proposed_params, log-acceptance-prob)

---

modeblocks *modal block assignments from MCMC samples*

Description

modal block assignments from MCMC samples

Usage

```
modeblocks(postz)
```

Arguments

postz	output from sampler

Value

a blocks object with the modal block assignments under postz

multinom *Multinomial block assignment*

Description

A blockmod for Multinomial allocation

Usage

```
multinom(gamma, kappa)
```

Arguments

gamma	parameter for Dirichlet component $Dirichlet(gamma, ..., gamma)$
kappa	the number of blocks

Details

This model posits that: for i=1:n

$$Z_i \; Multinomial(omega)$$

where

$$omega \; Dirichlet(gamma)$$

Value

a block model representing a Multinomial(gamma) distribution

Examples

```
## A fixed number of blocks with multinomial assignment of nodes
m <- multinom(1, 4)
print(m)
m$r(10) ## simulate a blocks object with 10 nodes
```

nodelike *Likelihood of node assignment*

Description

Calculate the likelihood of a nod belonging to each of block

Usage

```
nodelike(blocks, params, edges, i, sbmmod, ...)
```

Arguments

blocks	an blocks object
params	an params object
edges	an edges object
i	the node of interest
sbmmod	an sbmmod object
...	additional arguments for nodelike.blocks

Details

the number of blocks considered is either the number of blocks in sbm (kappa) or kappa+1 when sbmmod has a variable number of blocks. care is taken for data which is directed and with loops.

Value

likelihood of edges emanating from node i

---

numblockstrace *plot a trace of the number of blocks from MCMC samples*

Description

plot a trace of the number of blocks from MCMC samples

Usage

```
numblockstrace(postk, burnin)
```

Arguments

postk	output from sampler
burnin	which iterations to plot? defaults to all.

Value

‘ggplot2‘ object

parammat *Parameter Matrix*

Description

Make a matrix of parameters

Usage

```
parammat(x, ...)
```

Arguments

x	object for dispatch
...	additional arguments for method

Value

a parameter matrix object

parammat.blocks *Parameter Matrix*

Description

Make a matrix of parameters from a blocks and params object

Usage

```
## S3 method for class 'blocks'
parammat(x, params, ...)
```

Arguments

x	a blocks object
params	a params object
...	(unused)

Value

an NxN matrix P, with P[i,j] = the parameter governing edge ij

parammat.matrix *Parameter Matrix*

Description

Make a matrix of parameters from a matrix of block assignments

Usage

```
## S3 method for class 'matrix'
parammat(zleft, zright, params, ...)
```

Arguments

zleft	block assignment matrix on the left
zright	block assignment matrix on the right
params	the parameters object
...	(unused)

Value

a matrix of parameters of size |zleft| x |zright|

parammat.params *Parameter Matrix*

Description

Make a matrix of parameters from a params object

Usage

```
## S3 method for class 'params'
parammat(x, kappa, ...)
```

Arguments

x	a params object
kappa	- number of blocks to compute for matrix (optional)
...	(unused)

Value

a matrix of parameters

parammat.sbm *Parameter Matrix*

Description

Make a matrix of parameters from an sbm object

Usage

```
## S3 method for class 'sbm'
parammat(x, ...)
```

Arguments

x	an sbm object
...	(unused)

Value

a matrix of parameters

parammod *Parameter Model*

Description

create a parammod object

Usage

```
parammod(logd, r, t, invt, loggradt, ...)
```

Arguments

logd	function(params) - log-density function for parameters
r	function(kappa) - random function to draw parameters
t	mapping parameter space to real line
invt	mapping real line to parameter space
loggradt	log of the gradient of mapping t
...	additional arguments to store in the parammod object

Details

A parameter model is a probability model for a params object. This class creates a closure with five functions: - a random method for sampling a params object - a log-density method for computing the log-density of a given params object - a transformation function t that maps a parameter value to the real line - the inverse of t - the log-gradient of t

Value

a parammod object

See Also

param_beta param_gamma param_nbin param_norm

---

params params *S3 object*

Description

make a params object from the between-block parameter theta0 and a vector of within block parameters thetak

Usage

```
params(theta0, thetak)
```

Arguments

theta0	between block parameters - a vector of length 'dimension of theta'
thetak	within block parameters - a matrix with ncol=kappa and nrow=dimension of theta

Value

a params object

Examples

```
p <- params(0.1, c(0.2,0.4,0.5))
p
```

paramtrace *plot a trace of parameter values from MCMC samples*

Description

plot a trace of parameter values from MCMC samples

Usage

```
paramtrace(theta, range, burnin)
```

Arguments

theta	output from sampler
range	which thetas to plot? defaults to all.
burnin	which iterations to plot? defaults to all.

Value

'ggplot2' object

param_beta *Beta parameter model*

Description

A parammod with beta-distributed parameters

Usage

```
param_beta(a0, a1, b0, b1)
```

Arguments

a0	theta_0 ~ Beta(a0,a1)
a1	theta_0 ~ Beta(a0,a1)
b0	theta_k ~ Beta(b0,b1)
b1	theta_k ~ Beta(b0,b1)

Details

This model represents a prior on theta with:

$$theta_0 \; Beta(a0, a1)$$

$$theta_k \; Beta(b0, b1)$$

for k = 1 ... kappa

Value

a parammod

Examples

```
## theta0 ~ Beta(1,9); thetak ~ Beta(9,1)
pb <- param_beta(1,9,9,1)
pb$r(5) ## a draw with 5 within-block parameters
```

---

param_gamma *Gamma parameter model*

Description

A parammod with gamma-distributed parameters

Usage

```
param_gamma(a0, a1, b0, b1)
```

Arguments

a0	theta_0 ~ Gamma(a0,a1)
a1	theta_0 ~ Gamma(a0,a1)
b0	theta_k ~ Gamma(b0,b1)
b1	theta_k ~ Gamma(b0,b1)

Details

This model represents a prior on theta with:

$$theta_0 \; Gamma(a0, a1)$$

$$theta_k \; Gamma(b0, b1)$$

for k = 1 ... kappa

Value

a parammod

Examples

```
## theta0 ~ Gamma(1,1); thetak ~ Gamma(5,5)
pg <- param_gamma(1,1,5,5)
pg$r(5) ## a draw with 5 within-block parameters
```

param_nbin *Parameter model for Negative Binomial*

Description

Negative Binomial parameter model: `theta_0 = (mu0, sigma0) theta_k = (muk, sigmak)`

Usage

```
param_nbin(a0, a1, b0, b1, c0, c1, d0, d1)
```

Arguments

a0, a1	mu0 ~ Gamma(a0,a1)
b0, b1	sig_0 ~ Beta(b0,b1)
c0, c1	muk ~ Gamma(c0, c1)
d0, d1	sig_k ~ Beta(d0,d1)

Value

parammod representing Negative-Binomial distributed parameters

Examples

```
## theta0 = (r0, p0); r0~Gamma(1,1); p0 ~ Beta(1,1);
## thetak = (rk, pk); rk~Gamma(3,3); pk ~ Beta(5,5);
pn <- param_nbin(1,1,1,1,3,3,5,5)
pn$r(5) ## a draw with 5 within-block parameters
```

param_norm *Parameter model for Normal Model*

Description

Normal parameter model: `theta_0 = (mu0, sigma0) theta_k = (muk, sigmak)`

Usage

```
param_norm(a0, a1, b0, b1, c0, c1, d0, d1)
```

Arguments

a0, a1	mu0 ~ Normal(a0,a1)
b0, b1	sig_0 ~ Gamma(b0,b1)
c0, c1	muk ~ Normal(c0, c1)
d0, d1	sig_k ~ Gamma(d0,d1)

Value

parammod representing Normal distributed parameters

Examples

```
## theta0 = (mu0, sigma0); mu0~Normal(0,5); sigma0 ~ Gamma(1,1);
## thetak = (muk, sigmak); muk~Normal(0,3); sigmak ~ Gamma(5,2);
pn <- param_norm(0,5,1,1,0,3,5,2)
pn$r(5) ## a draw with 5 within-block parameters
```

---

plot.blocks *Plot blocks*

Description

plots a block object

Usage

```
## S3 method for class 'blocks'
plot(x, col, xaxt = "n", yaxt = "n", xlab = "Nodes", ylab = "Nodes", ...)

## S3 method for class 'blocks'
image(x, col, xaxt = "n", yaxt = "n", xlab = "Nodes", ylab = "Nodes", ...)
```

Arguments

x	a blocks object to plot
col	colours for the plot
xaxt	override image parameters
yaxt	override image parameters
xlab	override image parameters
ylab	override image parameters
...	additional parameters for image

Details

plot the block assignments in a blocks object as a matrix, color-coded by block membership

Examples

```
## Assign six nodes to four blocks:
b <- blocks(c(1,1,2,3,4,4), 4)
plot(b)
## note that the lower left corner has one 2x2 red square
## indicating node 1 and 2 belong to the same block
```

plot.edges *Plot*

Description

plots an edges objects

Usage

```
## S3 method for class 'edges'
plot(x, Blocks, sorted = TRUE, xlab = "Node", ylab = "Node", ...)

## S3 method for class 'edges'
image(x, Blocks, sorted = TRUE, xlab = "Node", ylab = "Node", ...)
```

Arguments

x	an edges object
Blocks	a blocks object or sbm object
sorted	sort by block membership in sbm before plotting?
xlab	label for x-axis
ylab	label for y-axis
...	parameters for image

Value

ggplot2 plot of edges in a raster

Examples

```
## make an sbm model, sample data then plot and print:
model <- sbmmod(dma(2,5), param_beta(1,1,1,1), edges_bern())
s <- model$r(100)
e <- redges(s, model$edge)
plot(e)
plot(e, s)
print(e)
```

plot.sbm *Plot for sbm object*

Description

plot an sbm object as an image

Usage

```
## S3 method for class 'sbm'
plot(x, col, ...)

## S3 method for class 'sbm'
image(x, col, ...)
```

Arguments

x	an sbm object
col	colours for each block - if missing, rainbow is used
...	additional arguments for plot

See Also

plot.default

plotpostpairs *helper function for trace plots*

Description

helper function for trace plots

Usage

```
plotpostpairs(mat)
```

Arguments

mat	matrix to plot as an image using ggplot2

Value

'ggplot2' plot objecy

postpairs — *mean proportion of times two nodes were in the same block under MCMC samples*

Description

mean proportion of times two nodes were in the same block under MCMC samples

Usage

```
postpairs(postz)
```

Arguments

postz	output from sampler

Value

matrix P with P[i,j] = proportion of times i and j are in the same block under postz

rcat — *Draw draw Categorical distribution*

Description

Draw draw Categorical distribution

Usage

```
rcat(n, p, replace = TRUE)
```

Arguments

n	number of draws
p	a length-d probability vector
replace	should the categories be replaced? If so n < p required

Value

a draw from Categorical(p)

Examples

```
rcat(1, 1) ## returns 1 with probability 1
rcat(1, rep(1/6,6)) ## a dice roll
```

rdirichlet *Dirichlet distribution*

Description

Draw from Dirichlet distribution

Usage

```
rdirichlet(n, gam)
```

Arguments

n	number of variates to draw
gam	a vector of concentration parameters of length K

Value

matrix dimension n*k of samples

Examples

```
rdirichlet(1, rep(2,5)) ## a length-5 probability vector
```

redges *Simulate edges*

Description

Simulate edges from an sbm object with a given edgemod

Usage

```
redges(SBM, edgemod, sym = TRUE, loops = FALSE, ...)
```

Arguments

SBM	an sbm object
edgemod	an edgemod object
sym	should the network be symmetric?
loops	should the network have self-loops?
...	additional arguments passed to edgemod$r

Details

None

Value

an edges object

Examples

```
## make an sbm model, sample data then plot and print:
model <- sbmmod(dma(2,5), param_beta(1,1,1,1), edges_bern())
s <- model$r(100)
e <- redges(s, model$edge)
plot(e)
plot(e, s)
print(e)
```

---

rw *Random Walk*

Description

performs a random walk on a parameter value with a given parameter model

Usage

```
rw(p, pm, sigma)
```

Arguments

p	a parameter
pm	a parammod object
sigma	- scale of random walk

Value

ist(proposed parameter, locjacobian)

---

sampler *top level sampler function*

Description

top level sampler function

Usage

```
sampler(
  edges,
  sbmmod,
  nSteps = 1000,
  algorithm = "rj",
  sigma = 0.5,
  statusfreq,
  currsbm,
  ...
)
```

Arguments

edges	an edges object
sbmmod	an sbmmod model
nSteps	number of steps to run sampler
algorithm	choice of algorithm options are: "conjugate", "gibbs", "dp", "rj"
sigma	random walk parameter for theta
statusfreq	print the elapsed number of iterations every statusfreq iterations
currsbm	initial state for sbm object (optional - one is drawn from sbmmod if not supplied)
...	additional parameters to pass to step

Value

postz traces for block assignments z

postt traces for theta

postk traces for number of blocks kappa

postn traces for number of occupied blocks

nsteps number of iterations of chain

algorithm choice

Examples

```
## see vignette("Weibull-edges")
```

`sampler.conj` *Conjugate model sampler*

Description

Conjugate model sampler

Usage

```
sampler.conj(currsbm, edges, sbmmod, sigma = NULL, ...)
```

Arguments

`currsbm`	the current state of the sampler
`edges`	an `edges` object
`sbmmod`	an `sbmmod` model
`sigma`	unused
`...`	additional arguments for `sbmmod$marglike`

Value

next state of `currsbm` object

Note

If using the CRP as the block model, then this is the IRM sampler of Schmidt or Morup (Schmidt, M.N. and Morup, M., 2013. Nonparametric Bayesian modeling of complex networks: An introduction. IEEE Signal Processing Magazine, 30(3), pp.110-128.)

Examples

```
model <- sbmmod(crp(3), param_beta(1,1,1,1), edges_bern(), marglike=marglike_bern)
trueSBM <- model$r(100)
Edges <- redges(trueSBM, model$edge)
out <- sampler(Edges, model, 10, "conjugate")
```

sampler.dp *Dirichlet process sampler*

Description

Dirichlet process sampler

Usage

```
sampler.dp(currsbm, edges, sbmmod, sigma)
```

Arguments

currsbm	the current state of the sampler
edges	an edges object
sbmmod	an sbmmod model
sigma	random walk parameter for theta

Value

next state of currsbm object

See Also

For full algorithm details see http://doi.org/10.17635/lancaster/thesis/296

Examples

```
model <- sbmmod(crp(4), param_norm(0,0,1,1,3,3,1,1), edges_norm())
trueSBM <- model$r(100)
Edges <- redges(trueSBM, model$edge)
dp_out <- sampler(Edges, model, 25, "dp", sigma=0.1)
```

sampler.gibbs *Gibbs sampling for node assignments*

Description

Gibbs sampling for node assignments

Usage

```
sampler.gibbs(currsbm, edges, sbmmod, sigma)
```

Arguments

currsbm	the current state of the sampler
edges	an edges object
sbmmod	an sbmmod model
sigma	random walk parameter for theta

Value

next state of currsbm object

Note

This requires a block model with a fixed kappa

Examples

```
model <- sbmmod(multinom(1, 3), param_gamma(1,1,1,1), edges_pois())
trueSBM <- model$r(10)
Edges <- redges(trueSBM, model$edge)
gibbs_out <- sampler(Edges, model, algorithm="gibbs", 10, sigma=0.1)
eval_plots(gibbs_out)
```

---

sampler.rj *reversible jump Markov chain Monte Carlo split-merge sampler*

Description

reversible jump Markov chain Monte Carlo split-merge sampler

Usage

```
sampler.rj(currsbm, edges, sbmmod, sigma, rho = 10)
```

Arguments

currsbm	the current state of the sampler
edges	an edges object
sbmmod	an sbmmod model
sigma	random walk parameter for theta
rho	propensity to add a block

Value

next state of currsbm object

See Also

For full algorithm details see http://doi.org/10.17635/lancaster/thesis/296

Examples

```
model <- sbmmod(dma(1,10), param_nbin(1,1,4,4,0.5,0.5,0.5,0.5), edges_nbin())
trueSBM <- model$r(100)
Edges <- redges(trueSBM, model$edge)
rj_out <- sampler(Edges, model, 10, "rj", sigma=0.1)
```

---

sbm *Class* sbm

Description

Class sbm

Usage

```
sbm(blocks, params)
```

Arguments

blocks	a blocks object
params	a params object

Value

an sbm object

Examples

```
sbm(blocks(c(1,1,2,2,3,3)), params(0.1, c(0.4,0.5,0.6)))
```

---

sbmmod *Stochastic block model object*

Description

A wrapper for a block and parameter model

Usage

```
sbmmod(blockmod, parammod, edgemod, ...)
```

Arguments

blockmod	a blockmod object
parammod	a parammod object
edgemod	an edgemod object
...	additional arguments to store in the sbmmod object

Details

Simple wrapper for the block and parameter model for an sbm object

Value

an sbmmod object with a method 'r(n)' sampling an sbm object with n nodes from the model and a method logd(sbm) computing the log-density of sbm under the model

Author(s)

Matthew Ludkin

See Also

blockmod parammod edgemod

---

splitavg *split move using average to merge parameters*

Description

split move using average to merge parameters

Usage

```
splitavg(sbm, edges, sbmmod, ...)
```

Arguments

sbm	the current state of the sampler
edges	an edges object
sbmmod	an sbmmod model
...	additional parameter to 'accept'

Value

an updated sbm object

splitblocks *split move: blocks*

Description

split move: blocks

Usage

```
splitblocks(currblocks, propparams, edges, sbmmod, k)
```

Arguments

currblocks	current blocks
propparams	proposed parameters
edges	an edges object
sbmmod	a model list
k	block to split

Value

list(proposed block structure, log-acceptance-prob)

splitparams *split move: parameters*

Description

split move: parameters

Usage

```
splitparams(x, ...)
```

Arguments

x	object for dispatch
...	additional arguments for method

Value

list(proposed_params, log-acceptance-prob)

splitparams.numeric *split move:* params

Description

split move: params

Usage

```
## S3 method for class 'numeric'
splitparams(theta, u, x, parammod)
```

Arguments

theta	a parameter to split
u	auxiliary variable
x	auxiliary variable
parammod	parammod object

Value

list(proposed_params, log-acceptance-prob)

splitparams.params *split move:* params

Description

split move: params

Usage

```
## S3 method for class 'params'
splitparams(params, k, parammod)
```

Arguments

params	a params object to split
k	block to split
parammod	parammod object

Value

list(proposed_params, log-acceptance-prob)

StackOverflow *The Stack-Overflow data set as extracted from* igraph *using the script in data-raw Extracted on 27/8/2019 from Kaggle (login required) using:* library(rvest) read_html("https://www.kaggle.com/stackoverflow/stack-overflow-tag-network/dow

---

Description

The Stack-Overflow data set as extracted from igraph using the script in data-raw Extracted on 27/8/2019 from Kaggle (login required) using: library(rvest) read_html("https://www.kaggle.com/stackoverflow/

Usage

```
StackOverflow
```

Format

An edges object of activation counts between brain regions in a Macaque

Source

https://www.kaggle.com/stackoverflow/stack-overflow-tag-network/

See Also

igraph

---

updateblock *Update the block assignment of a node*

Description

change the block assignment in x of a node to a new block

Usage

```
updateblock(x, ...)
```

Arguments

x	object for dispatch
...	additional arguments for method

Value

object like ‘x‘ with updated block structure

See Also

updateblock.blocks updateblock.sbm

---

updateblock.blocks *Update the block assignment of a node*

Description

change the block assignment in an blocks object to a new block

Usage

```
## S3 method for class 'blocks'
updateblock(blocks, i, newblock)
```

Arguments

blocks	a blocks object
i	the node to update
newblock	the new block for node i

Value

new blocks object

---

updateblock.sbm *Update the block assignment of a node*

Description

change the block assignment in an sbm object to a new block

Usage

```
## S3 method for class 'sbm'
updateblock(currsbm, i, newblock, model)
```

Arguments

currsbm	an sbm object
i	the node to update
newblock	the new block for node i
model	an sbmmod object

Value

new sbm object

Note

If adding a new block, this draws from the prior

---

vmeasure *V-measure*

Description

Calculate the V-measure of two clusterings

Usage

```
vmeasure(z, truez, beta = 1)
```

Arguments

z	input vector
truez	reference vector
beta	parameter beta=1 gives equal weight to homogeneity and completeness

Details

An information based measure of similarity between two clusterings

Value

v-measure of z against truez

See Also

Rosenberg, A., & Hirschberg, J. (2007, June). V-measure: A conditional entropy-based external cluster evaluation measure. In Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL) (pp. 410-420).

Examples

```
vmeasure(c(1,1,2,2,3,3), c(2,2,1,1,3,3)) ## 1 - doesn't care for labels
vmeasure(c(1,1,2,2,3,3), c(1,1,2,2,2,2)) ## 0.7333
vmeasure(c(1,1,2,2,3,3), c(1,1,2,2,3,4)) ## 0.904
```

Index