

# Package ‘nett’

May 9, 2026

**Title** Network Analysis and Community Detection

**Version** 1.0.0

**Description** Features tools for the network data analysis and community detection.

Provides multiple methods for fitting, model selection and goodness-of-fit testing in degree-corrected stochastic blocks models.

Most of the computations are fast and scalable for sparse networks, esp. for Poisson versions of the models.

Implements the following:

Amini, Chen, Bickel and Levina (2013) <[doi:10.1214/13-AOS1138](https://doi.org/10.1214/13-AOS1138)>

Bickel and Sarkar (2015) <[doi:10.1111/rssb.12117](https://doi.org/10.1111/rssb.12117)>

Lei (2016) <[doi:10.1214/15-AOS1370](https://doi.org/10.1214/15-AOS1370)>

Wang and Bickel (2017) <[doi:10.1214/16-AOS1457](https://doi.org/10.1214/16-AOS1457)>

Zhang and Amini (2020) <[doi:10.48550/arXiv.2012.15047](https://doi.org/10.48550/arXiv.2012.15047)>

Le and Levina (2022) <[doi:10.1214/21-EJS1971](https://doi.org/10.1214/21-EJS1971)>.

**License** MIT + file LICENSE

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.2.1

**Suggests** testthat, knitr, rmarkdown, igraph, ggplot2, dplyr, tidyr, tibble, mixtools, EnvStats, purrr, RSpectra

**Depends** R (>= 2.10)

**Imports** magrittr, Rcpp, Matrix, foreach, methods, stats, grDevices, graphics

**LinkingTo** Rcpp, RcppArmadillo

**VignetteBuilder** knitr

**URL** <https://github.com/aaamini/nett>

**BugReports** <https://github.com/aaamini/nett/issues>

**NeedsCompilation** yes

**Author** Arash A. Amini [aut, cre] (ORCID: <<https://orcid.org/0000-0002-2808-8310>>), Linfan Zhang [aut]

**Maintainer** Arash A. Amini <aaamini@ucla.edu>

**Repository** CRAN

**Date/Publication** 2022-11-09 10:50:05 UTC

## Contents

adj_spec_test . . . . .	3
bethe_hessian_select . . . . .	4
compute_block_sums . . . . .	5
compute_confusion_matrix . . . . .	5
compute_mutual_info . . . . .	6
estim_dcsbm . . . . .	6
eval_dcsbm_bic . . . . .	7
eval_dcsbm_like . . . . .	8
eval_dcsbm_loglr . . . . .	9
extract_largest_cc . . . . .	10
extract_low_deg_comp . . . . .	10
fast_cpl . . . . .	11
fast_sbm . . . . .	12
gen_rand_conn . . . . .	12
get_dcsbm_exav_deg . . . . .	13
label_mat2vec . . . . .	14
label_vec2mat . . . . .	14
nac_test . . . . .	15
plot_deg_dist . . . . .	16
plot_net . . . . .	16
plot_roc . . . . .	17
plot_smooth_profile . . . . .	18
polblogs . . . . .	18
pp_conn . . . . .	19
printf . . . . .	20
rsymperm . . . . .	20
sample_dcer . . . . .	21
sample_dclvm . . . . .	21
sample_dcpp . . . . .	22
sample_dcsbm . . . . .	23
sample_tdcsm . . . . .	23
simulate_roc . . . . .	24
sinkhorn_knopp . . . . .	25
snac_resample . . . . .	26
snac_select . . . . .	26
snac_test . . . . .	27
spec_clust . . . . .	29
spec_repr . . . . .	30

**Index**

**31**

---

adj_spec_test	<i>Adjusted spectral test</i>
---------------	-------------------------------

---

### Description

The adjusted spectral goodness-of-fit test based on Poisson DCSBM.

The test is a natural extension on Lei's work of testing goodness-of-fit for SBM. The residual matrix  $\tilde{A}$  is computed from the DCSBM estimation expectation of A. To speed up computation, the residual matrix uses Poisson variance instead. Specifically,

$$\tilde{A}_{ij} = (A_{ij} - \hat{P}_{ij}) / (n\hat{P}_{ij})^{1/2}, \quad \hat{P}_{ij} = \hat{\theta}_i \hat{\theta}_j \hat{B}_{\hat{z}_i, \hat{z}_j} \cdot 1\{i \neq j\}$$

where  $\hat{\theta}$  and  $\hat{B}$  are computed using [estim\\_dcsbm](#) if not provided.

Adjusted spectral test

### Usage

```
adj_spec_test(
  A,
  K,
  z = NULL,
  DC = TRUE,
  theta = NULL,
  B = NULL,
  cluster_fct = spec_clust,
  ...
)
```

### Arguments

A	adjacency matrix.
K	number of communities.
z	label vector for rows of adjacency matrix. If not given, will be calculated by the spectral clustering.
DC	whether or not include degree correction in the parameter estimation.
theta	give the propensity parameter directly.
B	give the connectivity matrix directly.
cluster_fct	community detection function to get z , by default using <a href="#">spec_clust</a> .
...	additional arguments for cluster_fct.

### Value

Adjusted spectral test statistics.

## References

Details of modification can be seen at [Adjusted chi-square test for degree-corrected block models](#), Linfan Zhang, Arash A. Amini, arXiv preprint arXiv:2012.15047, 2020.

The original spectral test is from [A goodness-of-fit test for stochastic block models](#) Lei, Jing, Ann. Statist. 44 (2016), no. 1, 401–424. doi:10.1214/15-AOS1370.

---

bethe\_hessian\_select *Beth-Hessian model selection*

---

## Description

Estimate the number of communities under block models by using the spectral properties of network Beth-Hessian matrix with moment correction.

## Usage

```
bethe_hessian_select(A, Kmax)
```

## Arguments

A	adjacency matrix.
Kmax	the maximum number of communities to check.

## Value

A list of result	
K	estimated the number of communities
rho	eigenvalues of the Beth-Hessian matrix

## References

[Estimating the number of communities in networks by spectral methods](#), Can M. Le, Elizaveta Levina, arXiv preprint arXiv:1507.00827, 2015

---

compute\_block\_sums      *Block sum of an adjacency matrix*

---

**Description**

Compute the block sum of an adjacency matrix given a label vector.

**Usage**

```
compute_block_sums(A, z)
```

**Arguments**

A                    adjacency matrix.  
z                    label vector.

**Value**

A K x L matrix with (k,l)-th element as  $\sum_{i,j} A_{i,j} 1_{z_i = k, z_j = l}$

---

compute\_confusion\_matrix  
                          *Compute confusion matrix*

---

**Description**

Compute confusion matrix

**Usage**

```
compute_confusion_matrix(z, y, K = NULL)
```

**Arguments**

z                    a label vector  
y                    a label vector  
K                    number of labels in both z and y

**Value**

A KxK confusion matrix between z and y

---

compute\_mutual\_info      *Compute normalized mutual information (NMI)*

---

### Description

Compute the NMI between two label vectors with the same cluster number

### Usage

compute\_mutual\_info(z, y)

### Arguments

z                      a label vector  
y                      a label vector

### Value

NMI between z and y

---

estim\_dcsbm              *Estimate model parameters of a DCSBM*

---

### Description

Compute the block sum of an adjacency matrix given a label vector.

### Usage

estim\_dcsbm(A, z)

### Arguments

A                      adjacency matrix.  
z                      label vector.

### Details

$$\hat{B}_{k\ell} = \frac{N_{k\ell}(\hat{z})}{m_{k\ell}(\hat{z})}, \quad \hat{\theta}_i = \frac{n_{\hat{z}_i}(\hat{z})d_i}{\sum_{j:\hat{z}_j=\hat{z}_i} d_i}$$

where  $N_{k\ell}(\hat{z})$  is the sum of the elements of A in block  $(k, \ell)$  specified by labels  $\hat{z}$ ,  $n_k(\hat{z})$  is the number of nodes in community  $k$  according to  $\hat{z}$  and  $m_{k\ell}(\hat{z}) = n_k(\hat{z})(n_\ell(\hat{z}) - 1\{k = \ell\})$

**Value**

A list of result

B                    estimated connectivity matrix.

theta                estimated node propensity parameter.

---

eval_dcsbm_bic	<i>Compute BIC score</i>
----------------	--------------------------

---

**Description**

compute BIC score when fitting a DCSBM to network data

**Usage**

```
eval_dcsbm_bic(A, z, K, poi)
```

**Arguments**

A                    adjacency matrix

z                    label vector

K                    number of community in z

poi                  whether to use Poisson version of likelihood

**Details**

the BIC score is calculated by  $-2 \times \log(\text{likelihood}) - K \times (K + 1) \times \log(n)$

**Value**

BIC score

**References**

BIC score is originally proposed in [Likelihood-based model selection for stochastic block models](#) Wang, YX Rachel, Peter J. Bickel, The Annals of Statistics 45, no. 2 (2017): 500-528.

The details of modified implementation can be found in [Adjusted chi-square test for degree-corrected block models](#), Linfan Zhang, Arash A. Amini, arXiv preprint arXiv:2012.15047, 2020.

**See Also**

[eval\\_dcsbm\\_like](#), [eval\\_dcsbm\\_loglr](#)

---

eval_dcsbm_like	<i>Log likelihood of a DCSBM (fast with poi = TRUE)</i>
-----------------	---

---

### Description

Compute the log likelihood of a DCSBM, using estimated parameters  $B$ ,  $\theta$  based on the given label vector

### Usage

```
eval_dcsbm_like(A, z, poi = TRUE, eps = 1e-06)
```

### Arguments

A	adjacency matrix
z	label vector
poi	whether to use Poisson version of likelihood
eps	truncation threshold for the Bernoulli likelihood, used when parameter $\phi$ is close to 1 or 0.

### Details

The log likelihood is calculated by

$$\ell(\hat{B}, \hat{\theta}, \hat{\pi}, \hat{z} | A) = \sum_i \log \hat{\pi}_{z_i} + \sum_{i < j} \phi(A_{ij}; \hat{\theta}_i \hat{\theta}_j \hat{B}_{\hat{z}_i \hat{z}_j})$$

where  $\hat{B}$ ,  $\hat{\theta}$  is calculated by [estim\\_dcsbm](#),  $\hat{\pi}_k$  is the proportion of nodes in community  $k$ .

### Value

log likelihood of a DCSBM

### See Also

[eval\\_dcsbm\\_loglr](#), [eval\\_dcsbm\\_bic](#)

---

eval_dcsbm_loglr	<i>Log-likelihood ratio of two DCSBMs (fast with poi = TRUE)</i>
------------------	--

---

### Description

Computes the log-likelihood ratio of one DCSBM relative to another, using estimated parameters  $B$  and  $\theta$  based on the given label vectors.

### Usage

```
eval_dcsbm_loglr(A, labels, poi = TRUE, eps = 1e-06)
```

### Arguments

A	adjacency matrix
labels	a matrix with two columns representing two different label vectors
poi	whether to use Poisson version of likelihood (instead of Bernoulli)
eps	truncation threshold for the Bernoulli likelihood, used when parameter $\theta$ is close to 1 or 0.

### Details

The log-likelihood ratio is computed between two DCSBMs specified by the columns of `labels`. The function computes the log-likelihood ratio of the model with `labels[, 2]` w.r.t. the model with `labels[, 1]`. This is often used with two label vectors fitted using different number of communities (say  $K$  and  $K+1$ ).

When `poi` is set to `TRUE`, the function uses fast sparse matrix computations and is scalable to large sparse networks.

### Value

log-likelihood ratio

### See Also

[eval\\_dcsbm\\_like](#), [eval\\_dcsbm\\_bic](#)

---

extract\_largest\_cc     *Extract largest component*

---

**Description**

Extract the largest connected component of a network

**Usage**

```
extract_largest_cc(gr, mode = "weak")
```

**Arguments**

gr	The network as an igraph object
mode	Type of connected component ("weak" "strong")

**Value**

An igraph object

---

extract\_low\_deg\_comp     *Extract low-degree component*

---

**Description**

Extract a low-degree connected component of a network

**Usage**

```
extract_low_deg_comp(g, deg_prec = 0.75, verb = FALSE)
```

**Arguments**

g	The network as an igraph object
deg_prec	The cut-off degree percentile
verb	Whether to be verbose (TRUE FALSE)

**Value**

An igraph object

---

`fast_cpl`*CPL algorithm for community detection (fast)*

---

### Description

The Conditional Pseudo-Likelihood (CPL) algorithm for fitting degree-corrected block models

### Usage

```
fast_cpl(Amat, K, ilabels = NULL, niter = 10)
```

### Arguments

<code>Amat</code>	adjacency matrix of the network
<code>K</code>	desired number of communities
<code>ilabels</code>	initial label vector (if not provided, initial labels are estimated using <a href="#">spec_clust</a> )
<code>niter</code>	number of iterations

### Details

The function implements the CPL algorithm as described in the paper below. It relies on the `mixtools` package for fitting a mixture of multinomials to a block compression of the adjacency matrix based on the estimated labels and then reiterates.

Technically, `fast_cpl` fits a stochastic block model (SBM) conditional on the observed node degrees, to account for the degree heterogeneity within communities that is not modeled well in SBM. CPL can also be used to effectively estimate the parameters of the degree-corrected block model (DCSBM).

The code is an adaptation of the original R code by Aiyu Chen with slight simplifications.

### Value

Estimated community label vector.

### References

For more details, see [Pseudo-likelihood methods for community detection in large sparse networks](#), A. A. Amini, A. Chen, P. J. Bickel, E. Levina, *Annals of Statistics* 2013, Vol. 41 (4), 2097—2122.

### Examples

```
head(fast_cpl(igraph::as_adj(polblogs), 2), 50)
```

---

fast\_sbm

*Sample from a SBM (fast)*


---

**Description**

Samples an adjacency matrix from a stochastic block model (SBM)

**Usage**

```
fast_sbm(z, B)
```

**Arguments**

z	Node labels ( $n * 1$ )
B	Connectivity matrix ( $K * K$ )

**Details**

The function implements a fast algorithm for sampling sparse SBMs, by only sampling the necessary nonzero entries. This function is adapted almost verbatim from the original code by Aiyu Chen.

**Value**

An adjacency matrix following SBM

**Examples**

```
B = pp_conn(n = 10^4, oir = 0.1, lambda = 7, pri = rep(1,3))$B
head(fast_sbm(sample(1:3, 10^4, replace = TRUE), B))
```

---

gen\_rand\_conn

*Generate randomly permuted connectivity matrix*


---

**Description**

Creates a randomly permuted DCPD connectivity matrix with a given average expected degree

**Usage**

```
gen_rand_conn(n, K, lambda, gamma = 0.3, pri = rep(1, K)/K, theta = rep(1, n))
```

**Arguments**

n	number of nodes
K	number of communities
lambda	expected average degree
gamma	a measure of out-in-ratio (convex combination parameter)
pri	the prior on community labels
theta	node connection propensity parameter of DCSBM, by default $E(\theta) = 1$

**Details**

The connectivity matrix is a convex combination of a random symmetric permutation matrix and the matrix of all ones, with weights  $\gamma$  and  $1-\gamma$ .

**Value**

connectivity matrix B of the desired DCSBM.

---

get\_dcsbm\_exav\_deg     *Calculate the expected average degree of a DCSBM*

---

**Description**

Calculate the expected average degree of a DCSBM

**Usage**

```
get_dcsbm_exav_deg(n, pri, B, ex_theta = 1)
```

**Arguments**

n	number of nodes
pri	distribution of node labels ( $K \times 1$ )
B	connectivity matrix ( $K \times K$ )
ex_theta	expected value of theta

**Value**

expected average degree of a DCSBM

---

label_mat2vec	<i>Convert label matrix to vector</i>
---------------	---------------------------------------

---

**Description**

Convert label matrix to vector

**Usage**

```
label_mat2vec(Z)
```

**Arguments**

Z                    a cluster assignment matrix

**Value**

A label vector that follows the assignment matrix

---

label_vec2mat	<i>Convert label vector to matrix</i>
---------------	---------------------------------------

---

**Description**

Convert label vector to matrix

**Usage**

```
label_vec2mat(z, K = NULL, sparse = FALSE)
```

**Arguments**

z                    a label vector  
K                    number of labels in z  
sparse                whether the output should be sparse matrix

**Value**

A cluster assignment matrix that follows from the label vector z

---

nac_test	<i>NAC test</i>
----------	-----------------

---

### Description

The NAC test to measure the goodness-of-fit of the DCSBM to network data.

The function computes the NAC+ or NAC statistics in the paper below. Label vectors, if not provided, are estimated using [spec\\_clust](#) by default but one can also use any other community detection algorithms through `cluster_fct`. Note that the function has to have A and K as its first two arguments, and additional arguments could be provided through ...

### Usage

```
nac_test(A, K, z = NULL, y = NULL, plus = TRUE, cluster_fct = spec_clust, ...)
```

### Arguments

A	adjacency matrix.
K	number of communities.
z	label vector for rows of A. If not provided, will be estimated from <code>cluster_fct</code> .
y	label vector for columns of A. If not provided, will be estimated from <code>cluster_fct</code> .
plus	whether or not use column label vector with (K+1) communities, default is TRUE.
cluster_fct	community detection function to get z and y, by default using <a href="#">spec_clust</a> . The first two arguments have to be A and K.
...	additional arguments for <code>cluster_fct</code> .

### Value

A list of result	
stat	NAC or NAC+ test statistic.
z	row label vector.
y	column label vector.

### References

[Adjusted chi-square test for degree-corrected block models](#), Linfan Zhang, Arash A. Amini, arXiv preprint arXiv:2012.15047, 2020.

### See Also

[snac\\_test](#)

**Examples**

```
A <- sample_dcpp(500, 10, 4, 0.1)$adj
nac_test(A, K = 4)$stat
nac_test(A, K = 4, cluster_fct = fast_cpl)$stat
```

---

plot\_deg\_dist                      *Plot degree distribution*

---

**Description**

Plot the degree distribution of a network on log scale

**Usage**

```
plot_deg_dist(gr, logx = TRUE)
```

**Arguments**

gr                      the network as an igraph object  
logx                    whether the degree is in log scale.

**Value**

Histogram of the degree of 'gr'.

---

plot\_net                      *Plot a network*

---

**Description**

Plot a network using degree-modulated node sizes, community colors and other enhancements

**Usage**

```
plot_net(
  gr,
  community = NULL,
  color_map = NULL,
  extract_lcc = TRUE,
  heavy_edge_deg_perc = 0.97,
  coord = NULL,
  vsize_func = function(deg) log(deg + 3) * 1,
  vertex_border = FALSE,
  niter = 1000,
  vertex_alpha = 0.4,
```

```

    remove_loops = TRUE,
    make_simple = FALSE,
    ...
)

```

### Arguments

gr	the network as an igraph object
community	community assignment; vector of node labels
color_map	color palette for clusters in 'gr'
extract_lcc	Extract largest connected component or not
heavy_edge_deg_perc	Degree percentile threshold for determining heavy edges
coord	Optional starting positions for the vertices. If this argument is not NULL then it should be an appropriate matrix of starting coordinates.
vsize_func	function to determine the size of node size
vertex_border	whether to show the border of vertex or not
niter	number of iteration for FR layout computation
vertex_alpha	factor modifying the opacity alpha of vertex; typically in [0,1]
remove_loops	whether to remove loops in the network
make_simple	whether to simplify edge weight calculation
...	other settings

### Value

A network plot

---

plot_roc	<i>Plot ROC curves</i>
----------	------------------------

---

### Description

Plot ROC curves given results from [simulate\\_roc](#).

### Usage

```
plot_roc(roc_results, method_names = NULL, font_size = 16)
```

### Arguments

roc_results	data frame roc from the output list of <a href="#">simulate_roc</a>
method_names	a list of method names
font_size	font size of the plot

### Value

Roc plot based on results from [simulate\\_roc](#)

---

plot\_smooth\_profile     *Plot community profiles*

---

### Description

Plot the smooth community profiles based on a resampled statistic

### Usage

```
plot_smooth_profile(
  tstat,
  net_name = "",
  trunc_type = "none",
  spar = 0.3,
  plot_null_spar = TRUE,
  alpha = 0.3,
  base_font_size = 12
)
```

### Arguments

tstat	dataframe that has a column 'value' as statistic in the plot and a column 'K' as its corresponding community number
net_name	name of network
trunc_type	method to round the dip/elbow point as the estimated community number
spar	the sparsity level of fitting spline to the value of tstat
plot_null_spar	whether to plot the spline with zero sparsity
alpha	transparency of the points in the plot
base_font_size	font size of the plot

### Value

smooth profile plot of a network

---

polblogs     *Political blogs network*

---

### Description

This is a directed network of hyperlinks between political blogs about politics in the United States of America.

**Usage**

```
data(polblogs)
```

**Format**

An igraph data with 1490 nodes and 19090 edges

**References**

**Data source.** Original paper: [The political blogosphere and the 2004 US election: divided they blog](#), Adamic, Lada A., and Natalie Glance. "." Proceedings of the 3rd international workshop on Link discovery. 2005.

---

pp\_conn

*Generate planted partition (PP) connectivity matrix*

---

**Description**

Create a degree-corrected planted partition connectivity matrix with a given average expected degree.

**Usage**

```
pp_conn(
  n,
  oir,
  lambda,
  pri,
  theta = rep(1, n),
  normalize_theta = FALSE,
  d = rep(1, length(pri))
)
```

**Arguments**

n	the number of nodes
oir	out-in-ratio
lambda	the expected average degree
pri	the prior on community labels
theta	node connection propensity parameter of DCSBM
normalize_theta	whether to normalize theta so that $\max(\theta) == 1$
d	diagonal of the connectivity matrix. An all-one vector by default.

**Value**

The connectivity matrix B of the desired DCSBM.

printf                      *The usual "printf" function*

---

**Description**

The usual "printf" function

**Usage**

```
printf(...)
```

**Arguments**

...                      printing object

**Value**

the value of the printing object

---

rsymperm                      *Generate random symmetric permutation matrix*

---

**Description**

Generate a random symmetric permutation matrix (recursively)

**Usage**

```
rsymperm(K)
```

**Arguments**

K                      size of the matrix

**Value**

A random K x K symmetric permutation matrix

---

sample_dcer	<i>Sample from a DCER</i>
-------------	---------------------------

---

**Description**

Sample an adjacency matrix from a degree-corrected Erdős–Rényi model (DCER).

**Usage**

```
sample_dcer(theta)
```

**Arguments**

theta            Node connectivity propensity vector ( $n * 1$ )

**Value**

An adjacency matrix following DCSBM

---

sample_dclvm	<i>Sample from a DCLVM</i>
--------------	----------------------------

---

**Description**

A DCLVM with  $K$  clusters has edges generated as

$$E[A_{ij} | x, \theta] \propto \theta_i \theta_j e^{-\|x_i - x_j\|^2}$$

where  $x_i = 2e_{z_i} + w_i$ ,  $e_k$  is the  $k$ th basis vector of  $R^d$ ,  $w_i \sim N(0, I_d)$ , and  $\{z_i\} \subset [K]^n$ . The proportionality constant is chosen such that the overall network has expected average degree  $\lambda$ . To calculate the scaling constant, we approximate  $E[e^{-\|x_i - x_j\|^2}]$  for  $i \neq j$  by generating random npairs  $\{z_i, z_j\}$  and average over them.

**Usage**

```
sample_dclvm(z, lambda, theta, npairs = NULL)
```

**Arguments**

z                    a vector of cluster labels  
lambda                desired average degree of the network  
theta                 degree parameter  
npairs                number of pairs of  $\{z_i, z_j\}$

**Details**

Sample from a degree-corrected latent variable model with Gaussian kernel

**Value**

Adjacency matrix of DCLVM

---

sample_dcpp	<i>Sample from a DCPP</i>
-------------	---------------------------

---

**Description**

Sample from a degree-corrected planted partition model

**Usage**

```
sample_dcpp(
  n,
  lambda,
  K,
  oir,
  theta = NULL,
  pri = rep(1, K)/K,
  normalize_theta = FALSE
)
```

**Arguments**

n	number of nodes
lambda	average degree
K	number of communities
oir	out-in ratio
theta	propensity parameter, if not given will be samples from a Pareto distribution with scale parameter $2/3$ and shape parameter 3
pri	prior distribution of node labels
normalize_theta	whether to normalize theta so that $\max(\theta) == 1$

**Value**

an adjacency matrix following a degree-corrected planted partition model

**See Also**

[sample\\_dcsbm](#), [sample\\_tdcslbm](#)

---

sample_dcsbm	<i>Sample from a DCSBM</i>
--------------	----------------------------

---

**Description**

Sample an adjacency matrix from a degree-corrected block model (DCSBM)

**Usage**

```
sample_dcsbm(z, B, theta = 1)
```

**Arguments**

z	Node labels ( $n * 1$ )
B	Connectivity matrix ( $K * K$ )
theta	Node connectivity propensity vector ( $n * 1$ )

**Value**

An adjacency matrix following DCSBM

**See Also**

[sample\\_dcpp](#), [fast\\_sbm](#), [sample\\_tdcslbm](#)

**Examples**

```
B = pp_conn(n = 10^3, oir = 0.1, lambda = 7, pri = rep(1,3))$B
head(sample_dcsbm(sample(1:3, 10^3, replace = TRUE), B, theta = rexp(10^3)))
```

---

sample_tdcslbm	<i>Sample truncated DCSBM (fast)</i>
----------------	--------------------------------------

---

**Description**

Sample an adjacency matrix from a truncated degree-corrected block model (DCSBM) using a fast algorithm.

**Usage**

```
sample_tdcslbm(z, B, theta = 1)
```

**Arguments**

z	Node labels ( $n * 1$ )
B	Connectivity matrix ( $K * K$ )
theta	Node connectivity propensity vector ( $n * 1$ )

**Details**

The function samples an adjacency matrix from a truncated DCSBM, with entries having Bernoulli distributions with mean

$$E[A_{ij}|z] = B_{z_i, z_j} \min(1, \theta_i \theta_j).$$

The approach uses the masking idea of Aiyou Chen, leading to fast sampling for sparse networks. The masking, however, truncates  $\theta_i \theta_j$  to at most 1, hence we refer to it as the truncated DCSBM.

**Value**

An adjacency matrix following DCSBM

**Examples**

```
B = pp_conn(n = 10^4, oir = 0.1, lambda = 7, pri = rep(1,3))$B
head(sample_tdcsgm(sample(1:3, 10^4, replace = TRUE), B, theta = rexp(10^4)))
```

---

simulate\_roc

*Simulate data to estimate ROC curves*

---

**Description**

Simulate data from the null and alternative distributions to estimate ROC curves for a collection of methods.

**Usage**

```
simulate_roc(
  apply_methods,
  gen_null_data,
  gen_alt_data,
  nruns = 100,
  core_count = parallel::detectCores() - 1,
  seed = NULL
)
```

**Arguments**

apply_methods	a function that returns a data.frame with columns "method", "tstat" and "twosided"
gen_null_data	a function that generate data under the null model
gen_alt_data	a function that generate data under the alternative model
nruns	number of simulated data from the null/alternative model
core_count	number of cores used in parallel computing
seed	seed for random simulation

**Value**

a list of result	
roc	A data frame used to plot ROC curves with columns: method, whether a two sided test, false positive rate (FPR), and true positive rate (TPR)
raw	A data frame containing raw output from null and alternative models with columns: method, statistics value, whether a two sided test, and the type of hypothesis
elapsed_time	system elapsed time for generating ROC data

---

sinkhorn_knopp	<i>Sinkhorn–Knopp matrix scaling</i>
----------------	--------------------------------------

---

**Description**

Implements the Sinkhorn–Knopp algorithm for transforming a square matrix with positive entries to a stochastic matrix with given common row and column sums (e.g., a doubly stochastic matrix).

**Usage**

```
sinkhorn_knopp(
  A,
  sums = rep(1, nrow(A)),
  niter = 100,
  tol = 1e-08,
  sym = FALSE,
  verb = FALSE
)
```

**Arguments**

A	input matrix
sums	desired row/column sums
niter	number of iterations
tol	convergence tolerance
sym	whether to compute symmetric scaling $D A D$
verb	whether to print the current change

**Details**

Computes diagonal matrices  $D1$  and  $D2$  to make  $D1AD2$  into a matrix with given row/column sums. For a symmetric matrix  $A$ , one can set `sym = TRUE` to compute a symmetric scaling  $DAD$ .

**Value**

Diagonal matrices  $D1$  and  $D2$  to make  $D1AD2$  into a matrix with given row/column sums.

---

snac_resample	<i>Resampled SNAC+</i>
---------------	------------------------

---

### Description

Compute SNAC+ with resampling

### Usage

```
snac_resample(
  A,
  nrep = 20,
  Kmin = 1,
  Kmax = 13,
  ncores = parallel::detectCores() - 1,
  seed = 1234
)
```

### Arguments

A	adjacency matrix
nrep	number of times SNAC+ is computed
Kmin	minimum community number to use in SNAC+
Kmax	maximum community number to use in SNAC+
ncores	number of cores to use in the parallel computing
seed	seed for random sampling

### Value

A data frame with columns specifying repetition cycles, number of community numbers and the value of SNAC+ statistics

---

snac_select	<i>Estimate community number with SNAC+</i>
-------------	---

---

### Description

Applying SNAC+ test sequentially to estimate community number of a network fit to DCSBM

**Usage**

```
snac_select(
  A,
  Kmin = 1,
  Kmax,
  alpha = 1e-05,
  labels = NULL,
  cluster_fct = spec_clust,
  ...
)
```

**Arguments**

A	adjacency matrix.
Kmin	minimum candidate community number.
Kmax	maximum candidate community number.
alpha	significance level for rejecting the null hypothesis.
labels	a matrix with each column being a row label vector for a candidate community number. If not provided, will be computed by <code>cluster_fct</code> .
cluster_fct	community detection function to get label vectors to compute SNAC+ statistics (in <code>snac_test</code> ), by default using <code>spec_clust</code> .
...	additional arguments for <code>cluster_fct</code> .

**Value**

estimated community number.

**See Also**

[snac\\_test](#)

**Examples**

```
A <- sample_dcpp(500, 10, 3, 0.1)$adj
snac_select(A, Kmax = 6)
```

---

snac\_test

*SNAC test*

---

**Description**

The SNAC test to measure the goodness-of-fit of the DCSBM to network data.

The function computes the SNAC+ or SNAC statistics in the paper below. The row label vector of the adjacency matrix could be given through `z` otherwise will be estimated by `cluster_fct`. One can specify the ratio of nodes used to estimate column label vector. If `plus = TRUE`, the column labels will be estimated by `spec_clust` with  $(K+1)$  clusters, i.e. performing SNAC+ test, otherwise with  $K$  clusters SNAC test. One can also get multiple test statistics with repeated random subsampling on nodes.

**Usage**

```
snac_test(
  A,
  K,
  z = NULL,
  ratio = 0.5,
  fromEachCommunity = TRUE,
  plus = TRUE,
  cluster_fct = spec_clust,
  nrep = 1,
  ...
)
```

**Arguments**

A	adjacency matrix.
K	desired number of communities.
z	label vector for rows of adjacency matrix. If not provided, will be estimated by <code>cluster_fct</code>
ratio	ratio of subsampled nodes from the network.
fromEachCommunity	whether subsample from each estimated community or the full network, default is TRUE
plus	whether or not use column label vector with (K+1) communities to compute the statistics, default is TRUE.
cluster_fct	community detection function to estimate label vectors, by default using <a href="#">spec_clust</a> . The first two arguments have to be A and K.
nrep	number of times the statistics are computed.
...	additional arguments for <code>cluster_fct</code> .

**Value**

A list of result	
stat	SNAC or SNAC+ test statistic.
z	row label vector.

**References**

[Adjusted chi-square test for degree-corrected block models](#), Linfan Zhang, Arash A. Amini, arXiv preprint arXiv:2012.15047, 2020.

**See Also**

[nac\\_test](#)

**Examples**

```
A <- sample_dcpp(500, 10, 4, 0.1)$adj
snac_test(A, K = 4, niter = 3)$stat
```

---

spec_clust	<i>Spectral clustering (fast)</i>
------------	-----------------------------------

---

**Description**

Perform spectral clustering (with regularization) to estimate communities

**Usage**

```
spec_clust(
  A,
  K,
  type = "lap",
  tau = 0.25,
  nstart = 20,
  niter = 10,
  ignore_first_col = FALSE
)
```

**Arguments**

A	Adjacency matrix (n x n)
K	Number of communities
type	("lap"   "adj"   "adj2") Whether to use Laplacian or adjacency-based spectral clustering
tau	Regularization parameter for the Laplacian
nstart	argument from function 'kmeans'
niter	argument from function 'kmeans'
ignore_first_col	whether to ignore the first eigen vector when doing spectral clustering

**Value**

A label vector of size n x 1 with elements in 1,2,...,K

---

spec_repr	<i>Spectral Representation</i>
-----------	--------------------------------

---

**Description**

Provides a spectral representation of the network (with regularization) based on the adjacency or Laplacian matrices

**Usage**

```
spec_repr(A, K, type = "lap", tau = 0.25, ignore_first_col = FALSE)
```

**Arguments**

A	Adjacency matrix (n x n)
K	Number of communities
type	("lap"   "adj"   "adj2") Whether to use Laplacian or adjacency-based spectral clustering
tau	Regularization parameter for the Laplacian
ignore_first_col	whether to ignore the first eigen vector

**Value**

The n x K matrix resulting from a spectral embedding of the network into  $R^K$

# Index

- \* **comm\_detect**
  - fast\_cpl, 11
  - spec\_clust, 29
- \* **datasets**
  - polblogs, 18
- \* **estimation**
  - compute\_block\_sums, 5
  - estim\_dcsbm, 6
  - eval\_dcsbm\_like, 8
- \* **evaluation**
  - compute\_confusion\_matrix, 5
  - compute\_mutual\_info, 6
  - simulate\_roc, 24
- \* **mod\_sel**
  - bethe\_hessian\_select, 4
  - eval\_dcsbm\_bic, 7
  - eval\_dcsbm\_loglr, 9
  - snac\_select, 26
- \* **models**
  - fast\_sbm, 12
  - gen\_rand\_conn, 12
  - get\_dcsbm\_exav\_deg, 13
  - pp\_conn, 19
- \* **net\_repr**
  - spec\_repr, 30
- \* **plotting**
  - plot\_roc, 17
- \* **utils**
  - extract\_largest\_cc, 10
  - extract\_low\_deg\_comp, 10
  - label\_mat2vec, 14
  - label\_vec2mat, 14
  - printf, 20
  - rsymperm, 20
  - sinkhorn\_knopp, 25
- adj\_spec\_test, 3
- bethe\_hessian\_select, 4
- compute\_block\_sums, 5
- compute\_confusion\_matrix, 5
- compute\_mutual\_info, 6
- estim\_dcsbm, 3, 6, 8
- eval\_dcsbm\_bic, 7, 8, 9
- eval\_dcsbm\_like, 7, 8, 9
- eval\_dcsbm\_loglr, 7, 8, 9
- extract\_largest\_cc, 10
- extract\_low\_deg\_comp, 10
- fast\_cpl, 11
- fast\_sbm, 12, 23
- gen\_rand\_conn, 12
- get\_dcsbm\_exav\_deg, 13
- label\_mat2vec, 14
- label\_vec2mat, 14
- nac\_test, 15, 28
- plot\_deg\_dist, 16
- plot\_net, 16
- plot\_roc, 17
- plot\_smooth\_profile, 18
- polblogs, 18
- pp\_conn, 19
- printf, 20
- rsymperm, 20
- sample\_dcer, 21
- sample\_dc1vm, 21
- sample\_dcpp, 22, 23
- sample\_dcsbm, 22, 23
- sample\_tdcbsm, 22, 23, 23
- simulate\_roc, 17, 24
- sinkhorn\_knopp, 25
- snac\_resample, 26
- snac\_select, 26

snac\_test, [15](#), [27](#), [27](#)  
spec\_clust, [3](#), [11](#), [15](#), [27](#), [28](#), [29](#)  
spec\_repr, [30](#)