Package 'sparsecommunity'

April 8, 2026

Title Spectral Community Detection for Sparse Networks

Version 0.1.1

Description Implements spectral clustering algorithms for community detection in sparse networks under the stochastic block model ('SBM') and degree-corrected stochastic block model ('DCSBM'), following the methods of Lei and Rinaldo (2015) <doi:10.1214/14-AOS1274>. Provides a regularized normalized Laplacian embedding, spherical k-median clustering for 'DCSBM', standard k-means for 'SBM', simulation utilities for both models, and a misclustering rate evaluation metric. Also includes the 'NCAA' college football network of Girvan and Newman (2002) <doi:10.1073/pnas.122653799> as a benchmark dataset, and the Bethe-Hessian community number estimator of Hwang (2023) <doi:10.1080/01621459.2023.2223793>.

License GPL-3

Encoding UTF-8

Language en-US

LazyData true

RoxygenNote 7.2.3

Depends R (>= 4.0.0)

Imports graphics, Matrix, methods, RSpectra, stats

Suggests irlba, clue, igraph, igraphdata, testthat (>= 3.0.0), knitr, rmarkdown

Config/testthat/edition 3

VignetteBuilder knitr

NeedsCompilation no

Author Neil Hwang [aut, cre] (ORCID: <https://orcid.org/0000-0002-5175-9397>)

Maintainer Neil Hwang <neil.hwang@bcc.cuny.edu>

Repository CRAN

Date/Publication 2026-04-08 14:10:02 UTC

Contents

`community_detect` *Spectral community detection for sparse networks*

Description

Detects community structure in a sparse network using the regularized normalized Laplacian spectral embedding of Lei and Rinaldo (2015). Two model variants are supported:

Usage

```
community_detect(
  A,
  K,
  model = c("dcsbm", "sbm"),
  n_init = 20L,
  max_iter = 100L,
  reg = TRUE,
  seed = NULL
)
```

Arguments

`A`	An `n x n` symmetric adjacency matrix. Can be a sparse `Matrix::sparseMatrix` or a dense numeric matrix. Self-loops are ignored.
`K`	Positive integer. Number of communities to detect.
`model`	Character. One of `"dcsbm"` (default) or `"sbm"`.
`n_init`	Positive integer. Number of random restarts for the clustering step. Default is 20.
`max_iter`	Positive integer. Maximum iterations per restart. Default is 100.
`reg`	Logical. If `TRUE` (default), apply degree regularization in the Laplacian embedding (recommended for sparse networks).
`seed`	Integer or `NULL`. Random seed for reproducibility.

Details

- "sbm": Standard stochastic block model. Applies k-means to the spectral embedding (Theorem 3.1 of Lei & Rinaldo).
- "dcsbm": Degree-corrected stochastic block model. Row-normalizes the spectral embedding to the unit sphere and applies spherical k-median clustering (Theorem 4.2 of Lei & Rinaldo).

Value

An object of class "sparsecommunity", a list with components:

`labels` Integer vector of length `n`. Community assignments (integers 1 to `K`).

`embedding` `n x K` numeric matrix. The spectral embedding used for clustering. Row-normalized for "dcsbm", unnormalized for "sbm".

`eigenvalues` Numeric vector of length `K`. Top-`K` eigenvalues of the regularized normalized Laplacian.

`centers` `K x K` numeric matrix. Cluster centers in the embedding space.

`objective` Scalar. Total within-cluster sum of distances at convergence (Euclidean for both models).

`model` Character. The model used ("sbm" or "dcsbm").

`K` Integer. Number of communities.

`n` Integer. Number of nodes.

`n_init` Integer. Number of restarts used.

`regularized` Logical. Whether degree regularization was applied.

`call` The matched call.

References

Lei, J. and Rinaldo, A. (2015). Consistency of spectral clustering in stochastic block models. *Annals of Statistics*, **43**(1), 215–237. doi:10.1214/14AOS1274

Examples

```
# Simulate a balanced 2-community SBM
sim <- simulate_sbm(n = 200, K = 2,
                    B = matrix(c(0.3, 0.05, 0.05, 0.3), 2, 2),
                    seed = 1)
fit <- community_detect(sim$A, K = 2, model = "sbm", seed = 1)
print(fit)
misclustering_rate(sim$labels, fit$labels)

# Simulate a degree-corrected SBM
sim2 <- simulate_dcsbm(n = 300, K = 3,
                       B = matrix(c(0.4, 0.05, 0.05,
                                    0.05, 0.4, 0.05,
                                    0.05, 0.05, 0.4), 3, 3),
                       seed = 2)
fit2 <- community_detect(sim2$A, K = 3, model = "dcsbm", seed = 2)
misclustering_rate(sim2$labels, fit2$labels)
```

`estimate_K` *Estimate the number of communities in a sparse network*

Description

Estimates the number of communities K using the Bethe–Hessian spectral method of Hwang (2023). The estimator constructs a data-adaptive Bethe–Hessian matrix $H(\hat{\zeta})$ from the adjacency matrix and counts the number of negative eigenvalues, which equals the number of communities under the stochastic block model in the sparse regime.

Usage

```
estimate_K(A, K_max = 10L, dmin.est.c = 0.5)
```

Arguments

`A`	An `n x n` symmetric adjacency matrix (sparse or dense).
`K_max`	Positive integer. Maximum number of communities to consider. Default is 10.
`dmin.est.c`	Numeric in `(0, 1)`. Quantile threshold used to estimate the minimum degree in the Bethe–Hessian radius estimator. Smaller values use a lower quantile of the degree distribution. Default is 0.5.

Value

An integer: the estimated number of communities $\hat{K}$.

References

Hwang, N. (2023). On the estimation of the number of communities for sparse networks. *Journal of the American Statistical Association*.

Examples

```
B   <- matrix(c(0.3, 0.05, 0.05, 0.3), 2, 2)
sim <- simulate_sbm(n = 300, K = 2, B = B, seed = 1)
estimate_K(sim$A, K_max = 8)             # default dmin.est.c = 0.5
estimate_K(sim$A, K_max = 8, dmin.est.c = 0.3)  # more conservative
```

`football_A` *NCAA college football network*

Description

A network of 115 NCAA Division I-A college football teams and the regular-season games played between them during the Fall 2000 season. Edges connect teams that played each other. Teams are divided into 12 athletic conferences (communities), and teams within the same conference play each other more often than teams across conferences.

Usage

```
football_A

football_labels

football_teams
```

Format

Three objects are loaded by `data("football")`:

`football_A` A 115 $\times$ 115 sparse symmetric adjacency matrix (class `dgCMatrix` from package **Matrix**).

`football_labels` Integer vector of length 115: conference membership (1 to 12) for each team.

`football_teams` Character vector of length 115: team names.

An object of class `dgCMatrix` with 115 rows and 115 columns.

An object of class `integer` of length 115.

An object of class `character` of length 115.

Details

The network is sparse: mean degree 10.66, compared to $\log(115) = 4.74$. It has been widely used as a benchmark for community detection algorithms.

Source

M. Girvan and M. E. J. Newman (2002). Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, **99**(12), 7821–7826. doi:10.1073/pnas.122653799

Network data downloaded from M. E. J. Newman's network data repository.

Examples

```
data("football")
dim(football_A)        # 115 x 115
table(football_labels) # 12 conferences

fit <- community_detect(football_A, K = 12, model = "sbm", seed = 1)
misclustering_rate(football_labels, fit$labels)
```

`misclustering_rate` *Misclustering rate*

Description

Computes the best-permutation misclustering rate between a vector of true community labels and a vector of estimated labels. Because community labels are only identified up to permutation, this finds the label alignment that minimizes the fraction of misclassified nodes.

Usage

```
misclustering_rate(true_labels, est_labels)
```

Arguments

`true_labels`	Integer vector of length `n`. Ground-truth community assignments (values in `1:K`).
`est_labels`	Integer vector of length `n`. Estimated community assignments (values in `1:K`).

Details

If the **clue** package is available, the Hungarian algorithm is used for optimal alignment. Otherwise a greedy column-matching fallback is used.

Value

A scalar in `[0, 1]`: the fraction of nodes assigned to the wrong community under the best label permutation.

Examples

```
true <- c(1, 1, 2, 2, 3, 3)
est  <- c(2, 2, 1, 1, 3, 3)   # same partition, different labels
misclustering_rate(true, est)  # should be 0
```

plot.sparsecommunity *Plot the spectral embedding of a community detection fit*

Description

Produces a scatter plot of the first two dimensions of the spectral embedding, with points colored by detected community.

Usage

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

Arguments

x	A "sparsecommunity" object returned by community_detect().
...	Additional graphical parameters passed to plot().

Value

Invisibly returns x.

Examples

```
B   <- matrix(c(0.3, 0.05, 0.05, 0.3), 2, 2)
sim <- simulate_sbm(n = 200, K = 2, B = B, seed = 1)
fit <- community_detect(sim$A, K = 2, model = "sbm", seed = 1)
plot(fit)
```

plot_scree *Scree plot of regularized Laplacian eigenvalues*

Description

Computes the top-K_max eigenvalues of the regularized normalized Laplacian and plots them as a scree plot. The suggested number of communities is the index of the largest eigenvalue gap (drop between consecutive eigenvalues), marked with a dashed vertical line.

Usage

```
plot_scree(A, K_max = 10L, reg = TRUE, ...)
```

Arguments

A	An n x n symmetric adjacency matrix (sparse or dense).
K_max	Positive integer. Number of eigenvalues to compute and display. Default is 10.
reg	Logical. If TRUE (default), apply degree regularization.
...	Additional graphical parameters passed to plot().

Details

Use this plot alongside estimate_K() to guide the choice of K before calling community_detect().

Value

Invisibly returns a list with:

eigenvalues Numeric vector of length K_max: the top eigenvalues in decreasing order.

gaps Numeric vector of length K_max - 1: absolute differences between consecutive eigenvalues.

K_suggested Integer: the index of the largest gap, i.e., the suggested number of communities.

Examples

```
B   <- matrix(c(0.3, 0.05, 0.05, 0.3), 2, 2)
sim <- simulate_sbm(n = 300, K = 2, B = B, seed = 1)
result <- plot_scree(sim$A, K_max = 8)
result$K_suggested   # should be 2
```

---

simulate_dcsbm *Simulate from a degree-corrected stochastic block model*

Description

Generates a random graph under the degree-corrected stochastic block model (DCSBM). Edge probability between nodes i and j is theta[i] * theta[j] * B[z_i, z_j], clipped to [0, 1].

Usage

```
simulate_dcsbm(n, K, B, theta = NULL, pi = NULL, seed = NULL)
```

Arguments

n	Positive integer. Number of nodes.
K	Positive integer. Number of communities.
B	K x K numeric matrix of base edge probabilities. Must be symmetric.
theta	Numeric vector of length n. Degree heterogeneity parameters. All entries must be positive. Default: uniform (rep(1, n)).
pi	Numeric vector of length K with community size proportions. Default: balanced communities.
seed	Integer or NULL. Random seed for reproducibility.

Value

A list of class "dcsbm_sim" with components:

A n x n symmetric sparse adjacency matrix.

labels Integer vector of length n: true community assignments.

K Number of communities.

B The base block probability matrix.

theta The degree heterogeneity vector.

pi Community proportions used.

n Number of nodes.

Examples

```
set.seed(1)
B     <- matrix(c(0.4, 0.05, 0.05, 0.4), 2, 2)
theta <- runif(300, 0.5, 1.5)
sim   <- simulate_dcsbm(n = 300, K = 2, B = B, theta = theta, seed = 7)
table(sim$labels)
```

---

simulate_sbm *Simulate from a stochastic block model*

Description

Generates a random graph under the stochastic block model (SBM) with K communities. Each pair of nodes (i, j) is connected independently with probability B[z_i, z_j], where z_i is the community of node i.

Usage

```
simulate_sbm(n, K, B, pi = NULL, seed = NULL)
```

Arguments

n	Positive integer. Number of nodes.
K	Positive integer. Number of communities.
B	K x K numeric matrix of edge probabilities. Must be symmetric with entries in [0, 1].
pi	Numeric vector of length K with community size proportions. Default: balanced communities (each proportion 1/K).
seed	Integer or NULL. Random seed for reproducibility.

Value

A list of class "sbm_sim" with components:

A n x n symmetric sparse adjacency matrix (no self-loops).

labels Integer vector of length n: true community assignments.

K Number of communities.

B The block probability matrix used.

pi Community proportions used.

n Number of nodes.

Examples

```
B   <- matrix(c(0.3, 0.05, 0.05, 0.3), 2, 2)
sim <- simulate_sbm(n = 200, K = 2, B = B, seed = 42)
dim(sim$A)        # 200 x 200 sparse matrix
table(sim$labels) # community sizes
```

Index