Package ‘motifcluster’

May 9, 2026

Title Motif-Based Spectral Clustering of Weighted Directed Networks

Version 0.2.3

Description Tools for spectral clustering of weighted directed networks using motif adjacency matrices. Methods perform well on large and sparse networks, and random sampling methods for generating weighted directed networks are also provided. Based on methodology detailed in Underwood, Elliott and Cucuringu (2020) <doi:10.48550/arXiv.2004.01293>.

URL `https://github.com/wgunderwood/motifcluster`

Language en-US

BugReports `https://github.com/wgunderwood/motifcluster/issues`

License GPL-3

Encoding UTF-8

RoxygenNote 7.2.1

Depends R (>= 3.6.0)

Imports igraph (>= 1.2.5), Matrix (>= 1.2), RSpectra (>= 0.16.0)

Suggests covr (>= 3.5.0), knitr (>= 1.28), mclust (>= 5.4.6), rmarkdown (>= 2.1), testthat (>= 2.3.2)

VignetteBuilder knitr

NeedsCompilation no

Author William George Underwood [aut, cre]

Maintainer William George Underwood <`wgu2@princeton.edu`>

Repository CRAN

Date/Publication 2022-11-18 08:10:02 UTC

Contents

`build_laplacian` *Build a Laplacian matrix*

Description

Build a Laplacian matrix (combinatorial Laplacian or random-walk Laplacian) from a symmetric (weighted) graph adjacency matrix.

Usage

```
build_laplacian(adj_mat, type_lap = c("comb", "rw"))
```

Arguments

`adj_mat`	Symmetric adjacency matrix from which to build the Laplacian.
`type_lap`	Type of Laplacian to build. One of `"comb"` (combinatorial) or `"rw"` (random-walk).

Value

The specified Laplacian matrix.

Examples

```
adj_mat <- matrix(c(1:9), nrow = 3)
build_laplacian(adj_mat, "rw")
```

`build_motif_adjacency_matrix` *Build a motif adjacency matrix*

Description

Build a motif adjacency matrix from an adjacency matrix.

Usage

```
build_motif_adjacency_matrix(
  adj_mat,
  motif_name,
  motif_type = c("struc", "func"),
  mam_weight_type = c("unweighted", "mean", "poisson"),
  mam_method = c("sparse", "dense")
)
```

Arguments

`adj_mat`	Adjacency matrix from which to build the motif adjacency matrix.
`motif_name`	Motif used for the motif adjacency matrix.
`motif_type`	Type of motif adjacency matrix to build. One of `"func"` or `"struc"`.
`mam_weight_type`	The weighting scheme to use. One of `"unweighted"`, `"mean"` or `"product"`.
`mam_method`	Which formulation to use. One of `"dense"` or `"sparse"`. The sparse formulation avoids generating large dense matrices so tends to be faster for large sparse graphs.

Details

Entry (i, j) of a motif adjacency matrix is the sum of the weights of all motifs containing both nodes i and j. The motif is specified by name and the type of motif instance can be one of:

- Functional: motifs should appear as subgraphs.
- Structural: motifs should appear as induced subgraphs.

The weighting scheme can be one of:

- Unweighted: the weight of any motif instance is one.
- Mean: the weight of any motif instance is the mean of its edge weights.
- Product: the weight of any motif instance is the product of its edge weights.

Value

A motif adjacency matrix.

Examples

```
adj_mat <- matrix(c(1:9), nrow = 3)
build_motif_adjacency_matrix(adj_mat, "M1", "func", "mean")
```

`get_largest_component` *Get largest connected component*

Description

Get the indices of the vertices in the largest connected component of a graph from its adjacency matrix.

Usage

```
get_largest_component(adj_mat)
```

Arguments

`adj_mat`	An adjacency matrix of a graph.

Value

A vector of indices corresponding to the vertices in the largest connected component.

Examples

```
adj_mat <- matrix(c(0, 1, 0, 0, 0, 0, 0, 0, 0), nrow = 3)
get_largest_component(adj_mat)
```

`get_motif_names` *Get common motif names*

Description

Get the names of some common motifs as strings.

Usage

```
get_motif_names()
```

Value

A vector of names (strings) of common motifs.

`kmeanspp` *kmeans++ clustering*

Description

Use the kmeans++ algorithm to cluster points into `k` clusters, as implemented in the deprecated LICORS package, using the built-in function kmeans.

Usage

```
kmeanspp(data, k = 2, iter.max = 100, nstart = 10, ...)
```

Arguments

`data`	An $N \times d$ matrix, where there are N samples in dimension d.
`k`	The number of clusters.
`iter.max`	The maximum number of iterations.
`nstart`	The number of restarts.
`...`	Additional arguments passed to `kmeans`.

Value

A list with 9 entries:

- `cluster`: A vector of integers from 1:k indicating the cluster to which each point is allocated.
- `centers`: A matrix of cluster centers.
- `totss`: The total sum of squares.
- `withinss`: Vector of within-cluster sum of squares, one component per cluster.
- `tot.withinss`: Total within-cluster sum of squares, i.e.sum(withinss).
- `betweenss`: The between-cluster sum of squares, i.e.totss-tot.withinss.
- `size`: The number of points in each cluster.
- `iter`: The number of (outer) iterations.
- `ifault`: An integer indicator of a possible algorithm problem.
- `initial.centers`: The initial centers used.

References

Arthur, D. and S. Vassilvitskii (2007). “k-means++: The advantages of careful seeding.” In H. Gabow (Ed.), Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms [SODA07], Philadelphia, pp. 1027-1035. Society for Industrial and Applied Mathematics.

See Also

`kmeans`

Examples

```
set.seed(1984)
n <- 100
X = matrix(rnorm(n), ncol = 2)
Y = matrix(runif(length(X)*2, -1, 1), ncol = ncol(X))
Z = rbind(X, Y)
cluster_Z = kmeanspp(Z, k = 5)
```

`random_sparse_matrix` *Build a random sparse matrix*

Description

Build a sparse matrix of size `m * n` with non-zero probability `p`. Edge weights can be unweighted, constant-weighted or Poisson-weighted.

Usage

```
random_sparse_matrix(m, n, p, sample_weight_type = "constant", w = 1)
```

Arguments

`m, n`	Dimension of matrix to build is `(m, n)`.
`p`	Probability that each entry is non-zero (before weighting).
`sample_weight_type`	Type of weighting scheme.
`w`	Weight parameter.

Value

A random sparse matrix.

`run_laplace_embedding` *Run Laplace embedding*

Description

Run Laplace embedding on a symmetric (weighted) adjacency matrix with a specified number of eigenvalues and eigenvectors.

Usage

```
run_laplace_embedding(adj_mat, num_eigs, type_lap = c("comb", "rw"))
```

Arguments

`adj_mat`	Symmetric adjacency matrix to be embedded.
`num_eigs`	Number of eigenvalues and eigenvectors for the embedding.
`type_lap`	Type of Laplacian for the embedding. One of `"comb"` (combinatorial) or `"rw"` (random-walk).

Value

A list with two entries: `vals` contains the length-`num_eigs` vector of the first few eigenvalues of the Laplacian, and `vects` contains an `nrow(adj_mat)` by `num_eigs` matrix of the associated eigenvectors.

Examples

```
adj_mat <- matrix(c(1:9), nrow = 3)
run_laplace_embedding(adj_mat, 2, "rw")
```

`run_motif_clustering` *Run motif-based clustering*

Description

Run motif-based clustering on the adjacency matrix of a (weighted directed) network, using a specified motif, motif type, weighting scheme, embedding dimension, number of clusters and Laplacian type.

Usage

```
run_motif_clustering(
  adj_mat,
  motif_name,
  motif_type = c("struc", "func"),
  mam_weight_type = c("unweighted", "mean", "product"),
  mam_method = c("sparse", "dense"),
  num_eigs = 2,
  type_lap = c("comb", "rw"),
  restrict = TRUE,
  num_clusts = 2
)
```

Arguments

`adj_mat`	Adjacency matrix to be embedded.
`motif_name`	Motif used for the motif adjacency matrix.
`motif_type`	Type of motif adjacency matrix to use. One of `"func"` or `"struc"`.

`mam_weight_type`	Weighting scheme for the motif adjacency matrix. One of `"unweighted"`, `"mean"` or `"product"`.
`mam_method`	The method to use for building the motif adjacency matrix. One of `"sparse"` or `"dense"`.
`num_eigs`	Number of eigenvalues and eigenvectors for the embedding.
`type_lap`	Type of Laplacian for the embedding. One of `"comb"` or `"rw"`.
`restrict`	Whether or not to restrict the motif adjacency matrix to its largest connected component before embedding.
`num_clusts`	The number of clusters to find.

Value

A list with 8 entries:

- `adj_mat`: the original adjacency matrix.
- `motif_adj_mat`: the motif adjacency matrix.
- `comps`: the indices of the largest connected component of the motif adjacency matrix (if restrict = TRUE).
- `adj_mat_comps`: the original adjacency matrix restricted to the largest connected component of the motif adjacency matrix (if restrict = TRUE).
- `motif_adj_mat_comps`: the motif adjacency matrix restricted to its largest connected component (if restrict = TRUE).
- `vals`: a length-`num_eigs` vector containing the eigenvalues associated with the Laplace embedding of the (restricted) motif adjacency matrix.
- `vects`: a matrix containing the eigenvectors associated with the Laplace embedding of the (restricted) motif adjacency matrix.
- `clusts`: a vector containing integers representing the cluster assignment of each vertex in the (restricted) graph.

Examples

```
adj_mat <- matrix(c(1:16), nrow = 4)
run_motif_clustering(adj_mat, "M1", "func")
```

`run_motif_embedding` *Run motif embedding*

Description

Calculate a motif adjacency matrix for a given motif and motif type, restrict it to its largest connected component, and then run Laplace embedding with specified Laplacian type and number of eigenvalues and eigenvectors.

Usage

```
run_motif_embedding(
  adj_mat,
  motif_name,
  motif_type = c("struc", "func"),
  mam_weight_type = c("unweighted", "mean", "product"),
  mam_method = c("sparse", "dense"),
  num_eigs = 2,
  type_lap = c("comb", "rw"),
  restrict = TRUE
)
```

Arguments

`adj_mat`	Adjacency matrix to be embedded.
`motif_name`	Motif used for the motif adjacency matrix.
`motif_type`	Type of motif adjacency matrix to use. One of `"func"` or `"struc"`.
`mam_weight_type`	Weighting scheme for the motif adjacency matrix. One of `"unweighted"`, `"mean"` or `"product"`.
`mam_method`	The method to use for building the motif adjacency matrix. One of `"sparse"` or `"dense"`.
`num_eigs`	Number of eigenvalues and eigenvectors for the embedding.
`type_lap`	Type of Laplacian for the embedding. One of `"comb"` or `"rw"`.
`restrict`	Whether or not to restrict the motif adjacency matrix to its largest connected component before embedding.

Value

A list with 7 entries:

- `adj_mat`: the original adjacency matrix.
- `motif_adj_mat`: the motif adjacency matrix.
- `comps`: the indices of the largest connected component of the motif adjacency matrix (if restrict = TRUE).
- `adj_mat_comps`: the original adjacency matrix restricted to the largest connected component of the motif adjacency matrix (if restrict = TRUE).
- `motif_adj_mat_comps`: the motif adjacency matrix restricted to its largest connected component (if restrict = TRUE).
- `vals`: a length-`num_eigs` vector containing the eigenvalues associated with the Laplace embedding of the (restricted) motif adjacency matrix.
- `vects`: a matrix containing the eigenvectors associated with the Laplace embedding of the (restricted) motif adjacency matrix.

Examples

```
adj_mat <- matrix(c(1:9), nrow = 3)
run_motif_embedding(adj_mat, "M1", "func")
```

`sample_bsbm` *Sample a bipartite stochastic block model (BSBM)*

Description

Sample the (weighted) adjacency matrix of a (weighted) bipartite stochastic block model (BSBM) with specified parameters.

Usage

```
sample_bsbm(
  source_block_sizes,
  dest_block_sizes,
  bipartite_connection_matrix,
  bipartite_weight_matrix = NULL,
  sample_weight_type = c("unweighted", "constant", "poisson")
)
```

Arguments

`source_block_sizes`	A vector containing the size of each block of source vertices.
`dest_block_sizes`	A vector containing the size of each block of destination vertices.
`bipartite_connection_matrix`	A matrix containing the source block to destination block connection probabilities.
`bipartite_weight_matrix`	A matrix containing the source block to destination block weight parameters. Unused for `sample_weight_type = "constant"`. Defaults to `NULL`.
`sample_weight_type`	The type of weighting scheme. One of `"unweighted"`, `"constant"` or `"poisson"`.

Value

A randomly sampled (weighted) adjacency matrix of a BSBM.

Examples

```
source_block_sizes <- c(10, 10)
dest_block_sizes <- c(10, 10, 10)
bipartite_connection_matrix <- matrix(c(0.8, 0.5, 0.1, 0.1, 0.5, 0.8),
      nrow = 2, byrow = TRUE)
bipartite_weight_matrix = matrix(c(20, 10, 2, 2, 10, 20),
      nrow = 2, byrow = TRUE)
sample_bsbm(source_block_sizes, dest_block_sizes,
      bipartite_connection_matrix, bipartite_weight_matrix, "poisson")
```

`sample_dsbm` *Sample a directed stochastic block model (DSBM)*

Description

Sample the (weighted) adjacency matrix of a (weighted) directed stochastic block model (DSBM) with specified parameters.

Usage

```
sample_dsbm(
  block_sizes,
  connection_matrix,
  weight_matrix = NULL,
  sample_weight_type = c("unweighted", "constant", "poisson")
)
```

Arguments

`block_sizes`	A vector containing the size of each block of vertices.
`connection_matrix`	A matrix containing the block-to-block connection probabilities.
`weight_matrix`	A matrix containing the block-to-block weight parameters. Unused for `sample_weight_type = "constant"`. Defaults to `NULL`.
`sample_weight_type`	The type of weighting scheme. One of `"unweighted"`, `"constant"` or `"poisson"`.

Value

A randomly sampled (weighted) adjacency matrix of a DSBM.

Examples

```
block_sizes <- c(10, 10)
connection_matrix <- matrix(c(0.8, 0.1, 0.1, 0.8), nrow = 2, byrow = TRUE)
weight_matrix <- matrix(c(10, 3, 3, 10), nrow = 2, byrow = TRUE)
sample_dsbm(block_sizes, connection_matrix, weight_matrix, "poisson")
```

Index