Package ‘motifclustr’

April 9, 2020

Title Motif-Based Spectral Clustering of Weighted Directed Networks

Version 0.0.0.9000

Description Construct motif adjacency matrices for (weighted directed)
networks, and use them for spectral clustering.

URL `https://github.com/wgunderwood/motif-based-clustering`

BugReports `https://github.com/wgunderwood/motif-based-clustering/issues`

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.0.2

Depends R (>= 3.6.0)

Imports igraph, Matrix, RSpectra

Suggests testthat

R topics documented:

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.

build_motif_adjacency_matrix *Build a motif adjacency matrix*

Description

Build a motif adjacency matrix from an adjacency matrix. 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.

Usage

```
build_motif_adjacency_matrix(
  adj_mat,
  motif_name,
  motif_type = c("func", "struc"),
  weight_type = c("unweighted", "mean", "product"),
  method = c("dense", "sparse")
)
```

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.
`weight_type`	The weighting scheme to use. One of `"unweighted"`, `"mean"` or `"product"`.
`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.

Value

A motif adjacency matrix.

`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.

`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.

`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.

`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("func", "struc"),
  num_eigs,
  type_lap
)
```

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"`.
`num_eigs`	Number of eigenvalues and eigenvectors for the embedding.
`type_lap`	Type of Laplacian for the embedding. One of `"comb"` or `"rw"`.

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.
- `adj_mat_comps`: the original adjacency matrix restricted to the largest connected component of the motif adjacency matrix.
- `motif_adj_mat_comps`: the motif adjacency matrix restricted to its largest connected component.
- `vals`: the eigenvalues associated with the Laplace embedding of the motif adjacency matrix.
- `vects`: the eigenvectors associated with the Laplace embedding of the motif adjacency matrix.

`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,
  weight_type = c("unweighted", "deterministic", "poisson"),
  bipartite_weight_matrix = NULL
)
```

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.
`weight_type`	The type of weighting scheme. One of `"unweighted"`, `"deterministic"` or `"poisson"`.
`bipartite_weight_matrix`	A matrix containing the sourece block to destination block weight parameters. Unused for `weight_type = "deterministic"`. Defaults to `NULL`.

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)
weight_type = "poisson"
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, weight_type, bipartite_weight_matrix)
```

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_type = c("unweighted", "deterministic", "poisson"),
  weight_matrix = NULL
)
```

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_type`	The type of weighting scheme. One of `"unweighted"`, `"deterministic"` or `"poisson"`.
`weight_matrix`	A matrix containing the block-to-block weight parameters. Unused for `weight_type = "deterministic"`. Defaults to `NULL`.

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_type = "poisson"
weight_matrix = matrix(c(10, 3, 3, 10), nrow = 2, byrow = TRUE)
sample_dsbm(block_sizes, connection_matrix, weight_type, weight_matrix)
```

Index