

# Package ‘motifcluster’

April 30, 2020

**Title** Motif-Based Spectral Clustering of Weighted Directed Networks

**Version** 0.0.1

**Description** Construct motif adjacency matrices for (weighted directed) networks, and use them for spectral clustering. Also provides random sampling methods for weighted directed networks.

**URL** <https://github.com/wgunderwood/motif-based-clustering>

**BugReports** <https://github.com/wgunderwood/motif-based-clustering/issues>

**License** GPL-3 + file LICENSE

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.1.0

**Depends** R (>= 3.6.0)

**Imports** igraph, LICORS, Matrix, RSpectra

**Suggests** covr, knitr, mclust, rmarkdown, testthat

**VignetteBuilder** knitr

## R topics documented:

build_laplacian . . . . .	2
build_motif_adjacency_matrix . . . . .	2
get_largest_component . . . . .	3
get_motif_names . . . . .	4
run_laplace_embedding . . . . .	4
run_motif_clustering . . . . .	5
run_motif_embedding . . . . .	6
sample_bsbm . . . . .	7
sample_dsbm . . . . .	8

<b>Index</b>	<b>10</b>
--------------	-----------

---

build_laplacian	<i>Build a Laplacian matrix</i>
-----------------	---------------------------------

---

### 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	<i>Build a motif adjacency matrix</i>
------------------------------	---------------------------------------

---

### 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	<i>Get common motif names</i>
-----------------	-------------------------------

---

**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	<i>Run Laplace embedding</i>
-----------------------	------------------------------

---

**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(1:9), nrow = 3)
run_motif_clustering(adj_mat, "M1", "func")
```

---

run_motif_embedding	<i>Run motif embedding</i>
---------------------	----------------------------

---

### 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

<code>adj_mat</code>	Adjacency matrix to be embedded.
<code>motif_name</code>	Motif used for the motif adjacency matrix.
<code>motif_type</code>	Type of motif adjacency matrix to use. One of "func" or "struc".
<code>mam_weight_type</code>	Weighting scheme for the motif adjacency matrix. One of "unweighted", "mean" or "product".
<code>mam_method</code>	The method to use for building the motif adjacency matrix. One of "sparse" or "dense".
<code>num_eigs</code>	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

`build_laplacian`, [2](#)  
`build_motif_adjacency_matrix`, [2](#)  
  
`get_largest_component`, [3](#)  
`get_motif_names`, [4](#)  
  
`run_laplace_embedding`, [4](#)  
`run_motif_clustering`, [5](#)  
`run_motif_embedding`, [6](#)  
  
`sample_bsbm`, [7](#)  
`sample_dsbm`, [8](#)