## Package ‘rSpectral’

January 18, 2023

<table>
<thead>
<tr>
<th><strong>Type</strong></th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Title</strong></td>
<td>Spectral Modularity Clustering</td>
</tr>
<tr>
<td><strong>Version</strong></td>
<td>1.0.0.10</td>
</tr>
</tbody>
</table>

**Description** Implements the network clustering algorithm described in Newman (2006) [doi:10.1103/PhysRevE.74.036104]. The complete iterative algorithm comprises of two steps. In the first step, the network is expressed in terms of its leading eigenvalue and eigenvector and recursively partition into two communities. Partitioning occurs if the maximum positive eigenvalue is greater than the tolerance (10e-5) for the current partition, and if it results in a positive contribution to the Modularity.

Given an initial separation using the leading eigen step, 'rSpectral' then continues to maximise for the change in Modularity using a fine-tuning step - or variate thereof. The first stage here is to find the node which, when moved from one community to another, gives the maximum change in Modularity. This node’s community is then fixed and we repeat the process until all nodes have been moved. The whole process is repeated from this new state until the change in the Modularity, between the new and old state, is less than the predefined tolerance.

A slight variant of the fine-tuning step, which can improve speed of the calculation, is also provided. Instead of moving each node into each community in turn, we only consider moves of neighbouring nodes, found in different communities, to the community of the current node of interest. The two steps process is repeatedly applied to each new community found, subdivided each community into two new communities, until we are unable to find any division that results in a positive change in Modularity.

**URL** [https://github.com/cmclean5/rSpectral](https://github.com/cmclean5/rSpectral)

**BugReports** [https://github.com/cmclean5/rSpectral/issues/](https://github.com/cmclean5/rSpectral/issues/)

**License** GPL-2

**Encoding** UTF-8
**R topics documented:**

- rSpectral ........................................... 2
- spectral_graphNEL .................................. 3
- spectral_igraph_communities .................. 4
- spectral_igraph_membership ................... 5

Index 7

---

**Description**

This package implements the Spectral Modularity clustering algorithm for `igraph` and `graphNEL` graphs. The algorithm was proposed in (Newman 2006) and an example of its application to the real biological network could be found in (Roy et al. 2018).

**Author(s)**

Colin Mclean <Colin.D.Mclean@ed.ac.uk>

**References**


spectral_graphNEL

See Also

Useful links:

- https://github.com/cmclean5/rSpectral
- Report bugs at https://github.com/cmclean5/rSpectral/issues/

---

spectral_graphNEL  Spectral clustering for graphNEL objects

Description

Spectral clustering for graphNEL objects

Usage

spectral_graphNEL(g, Cn_min = 1L, tol = 1e-05, names = 1L, fix_neig = 0L)

Arguments

- g: graphNEL object
- Cn_min: minimum cluster size
- tol: tolerance
- names: are we dealing with alphaNumeric (1) or numeric (!1) ids
- fix_neig: whether to fix neighbouring nodes found in same community

Value

data.frame with node names and membership information

See Also

spectral_igraph_membership

Examples

library(graph)
V = letters[1:12]
g2 = randomEGraph(V, edges=20)
mem.df = spectral_graphNEL(g2)
head(mem.df)
Spectral clustering for igraph objects

Description

This function invokes `spectral_igraph_membership` to calculate clustering and convert it into a `communities` object for seamless work with native igraph clustering functions.

Usage

```r
spectral_igraph_communities(
  g,
  Cn_min = 1L,
  tol = 1e-05,
  names = 1L,
  fix_neig = 0L
)
```

Arguments

- `g` igraph object
- `Cn_min` minimum cluster size
- `tol` tolerance
- `names` whether we are dealing with alphaNumeric (1) or numeric (!1) ids
- `fix_neig` whether to fix neighbouring nodes found in same community

Value

`communities` object

Examples

```r
data(karate, package="igraphdata")
c <- spectral_igraph_communities(karate)
```
**spectral_igraph_membership**

*Spectral clustering for igraph objects*

**Description**

This function implements the network clustering algorithm described in (M. E. J. Newman, 2006).

**Usage**

```r
spectral_igraph_membership(
  g,
  Cn_min = 1L,
  tol = 1e-05,
  names = 1L,
  fix_neig = 0L
)
```

**Arguments**

- `g` igraph object
- `Cn_min` minimum cluster size
- `tol` tolerance
- `names` are we dealing with alphaNumeric (1) or numeric (!1) ids
- `fix_neig` whether to fix neighbouring nodes found in same community

**Details**

The complete iterative algorithm comprises of two steps. In the first step, the network is expressed in terms of its leading eigenvalue and eigenvector and recursively partition into two communities. Partitioning occurs if the maximum positive eigenvalue is greater than the tolerance (\( tol=10^{-5} \)) for the current partition, and if it results in a positive contribution to the Modularity.

Given an initial separation using the leading eigen step, the function then continues to maximise for the change in Modularity using a fine-tuning step - or variate thereof. The first stage here is to find the node which, when moved from one community to another, gives the maximum change in Modularity. This node’s community is then fixed and we repeat the process until all nodes have been moved. The whole process is repeated from this new state until the change in the Modularity, between the new and old state, is less than the predefined tolerance (\( tol \)).

A slight variant of the fine-tuning step, which can reduce execution time by factor 2 to 5, is also provided. Instead of moving each node into each community in turn, we only consider moves of neighbouring nodes, found in different communities, to the community of the current node of interest. This variant of the node-moving algorithm effectively ‘fixes’ neighbouring nodes \( fix\_neig \) in the community being considered.

The two steps process is repeatedly applied to each new community found, subdivided each community into two new communities, until we are unable to find any division that results in a positive
change in Modularity. An additional stopping criteria, based on the minimum cluster size \( C_{n_{\text{min}}} \), is also provided.

**Value**

data.frame with node names and membership information

**Examples**

data(karate, package='igraphdata')
df.mem<-spectral_igraph_membership(karate)
Index

communities, 4

graphNEL, 2, 3

igraph, 2, 4

rSpectral, 2
rSpectral-package (rSpectral), 2

spectral_graphNEL, 3
spectral_igraph_communities, 4
spectral_igraph_membership, 3, 4, 5