Package 'rSpectral'

December 7, 2025

Type Package

Title Spectral Modularity Clustering

Version 1.0.0.14

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

BugReports https://github.com/cmclean5/rSpectral/issues/

License GPL-2

Encoding UTF-8

RoxygenNote 7.3.3

Depends R (>= 3.5.0)

Imports Rcpp (>= 1.0.8.3), Rdpack, igraph, graph

RdMacros Rdpack

LinkingTo Rcpp, RcppArmadillo(>= 0.11.2.0.0)

Suggests RColorBrewer, Rgraphviz, igraphdata, testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation yes

Author Colin Mclean [aut] (algorithm implementation in Rcpp functions), Anatoly Sorokin [aut, cre] (R functions, cranification, documentation, testing, maintenance)

Maintainer Anatoly Sorokin <lptolik@gmail.com>

Repository CRAN

Date/Publication 2025-12-07 06:40:02 UTC

Contents

`rSpectral` *rSpectral*

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

Newman MEJ (2006). "Finding community structure in networks using the eigenvectors of matrices." *Phys. Rev. E*, **74**(3), 036104. doi:10.1103/PhysRevE.74.036104, https://link.aps.org/doi/10.1103/PhysRevE.74.036104.

Roy M, Sorokina O, McLean C, Tapia-González S, DeFelipe J, Armstrong JD, Grant SGN (2018). "Regional Diversity in the Postsynaptic Proteome of the Mouse Brain." *Proteomes*, **6**(3), 31. ISSN 2227-7382, doi:10.3390/proteomes6030031, https://www.mdpi.com/2227-7382/6/3/31.

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_igraph_communities *Spectral clustering for* igraph *objects*

Description

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

Usage

```
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	are we dealing with alphaNumeric (1) or numeric (!1) ids
fix_neig	whether to fix neighbouring nodes found in same community

Value

communities object

Examples

```
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

```
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' neigbouring 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 `Cn_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