Package: rnetcarto (via r-universe)

October 31, 2024

Type Package

Title Fast Network Modularity and Roles Computation by Simulated Annealing (Rgraph C Library Wrapper for R)

Version 0.2.6

Description Provides functions to compute the modularity and modularity-related roles in networks. It is a wrapper around the rgraph library (Guimera & Amaral, 2005, <doi:10.1038/nature03288>).

License GPL (>= 2)

Encoding UTF-8

LazyLoad no

SystemRequirements GNU GSL

NeedsCompilation yes

Suggests testthat, knitr, rmarkdown, igraph

VignetteBuilder knitr

RoxygenNote 7.2.1

Repository https://stouffer.r-universe.dev

RemoteUrl https://github.com/stouffer/rnetcarto

RemoteRef HEAD

RemoteSha 090169ef2e4064d7a4d8a86f329f1d10ad1ed357

Contents

rnetcarto *Computes modularity and modularity roles from a network.*

Description

Compute modularity and modularity roles for graphs using simulated annealing

Usage

```
netcarto(
  web,
  seed = as.integer(floor(runif(1, 1, 100000001))),
  iterfac = 1,
  coolingfac = 0.995,
  bipartite = FALSE
)
```

Arguments

`web` network either as a square adjacency matrix or a list describing E interactions a->b: the first (resp. second) element is the vector of the labels of a (resp. b), the third (optional) is the vector of interaction weights.

`seed` Seed for the random number generator: Must be a positive integer.

`iterfac` At each temperature of the simulated annealing (SA), the program performs fN^2 individual-node updates (involving the movement of a single node from one module to another) and fN collective updates (involving the merging of two modules and the split of a module). The number "f" is the iteration factor.

`coolingfac` Temperature cooling factor.

`bipartite` If True use the bipartite definition of modularity.

Value

A list. The first element is a dataframe with the name, module, z-score, and participation coefficient for each row of the input matrix. The second element is the modularity of this partition.

Examples

```
# Generate a simple random network
a = matrix(as.integer(runif(100)<.3), ncol=10)
a[lower.tri(a)] = 0
# Find an optimal partition for modularity using netcarto.
netcarto(a)
```

Index