Package 'netcom'

July 10, 2017

Type Package

Title Dynamic Network Alignment

Version 1.0.4

Date 2017-07-10

Description Functions to take two networks stored as matrices and return a node-level injection between them (bijection if the input networks are of the same size). The alignment is made by comparing diffusion kernels originating from each node in one network to those originating from each node in the other network. This creates a cost matrix where rows are nodes from one network and columns are nodes from the other network. Optimal node pairings are then found using the Hungarian algorithm.

URL https://github.com/langendorfr/netcom

Repository CRAN

License GPL-3

Encoding UTF-8

Depends R (>= 3.1.0)

Imports clue, expm, igraph, Matrix, pdist, pracma, vegan

LazyData true

RoxygenNote 6.0.1

NeedsCompilation no

Author Ryan Langendorf [aut, cre],
Debra Goldberg [ctb]

Maintainer Ryan Langendorf <ryan.langendorf@colorado.edu>

Date/Publication 2017-07-10 19:02:53 UTC

R topics documented:

`align` *Network Alignment*

Description

Network alignment by comparing the entropies of diffusion kernels simulated on two networks. `align` takes two networks stored as matrices and returns a node-level alignment between them.

Usage

```
align(network_1_input, network_2_input, base = 2, max_duration,
  characterization = "entropy", normalization = FALSE, unit_test = FALSE)
```

Arguments

`network_1_input`	The first network being aligned, which must be in matrix form. If the two networks are of different sizes, it will be easier to interpret the output if this is the smaller one.
`network_2_input`	The second network, which also must be a matrix.
`base`	Defaults to 1. The base in the series of time steps to sample the diffusion kernels at. If base = 1 every time step is sampled. If base = 2, only time steps that are powers of 2 are sampled, etc. Larger values place more emphasis on earlier time steps. This can be helpful if the diffusion kernel quickly converges to an equilibrium, and also runs faster.
`max_duration`	Defaults to twice the diameter of the larger network. Sets the number of time steps to allow the diffusion kernel to spread for, which is the smallest power of base that is at least as large as max_duration.
`characterization`	Defaults to "entropy". Determines how the diffusion kernels are characterized. Either "entropy" or "gini". "entropy" is a size-normalized version of Shannon's entropy with base e (Euler's number). This is also known as interaction or species evenness in ecology. "gini" is the Gini coefficient.
`normalization`	Defaults to FALSE. Determines if self-loops should be augmented such that edge weights are proportional to those in network_1_input and network_2_input. FALSE by default because this is inappropriate for unweighted binary/logical networks where edges indicate only the presence of an interaction.
`unit_test`	Defaults to FALSE. Saves the following intermediate steps to help with general troubleshooting: post-processing matrix representations of both networks, time steps at which the diffusion kernels were sampled, the diffusion kernels at those time steps, the characterizations of the diffusion kernels at those time steps, and the cost matrix fed into the Hungarian algorithm where the ij element is the difference between the characterization-over-time curves for node i in the first network and node j in the second network.

Details

Network alignment pairs nodes between two networks so as to maximize similarities in their edge structures. This allows information from well-studied systems to be used in poorly studied ones, such as to identify unknown protein functions or ecosystems that will respond similarly to a given disturbance. Most network alignment algorithms focus on measures of topological overlap between edges of the two networks. The method implemented here compares nodes using the predictability of dynamics originating from each node in each network. Consider network alignment as trying to compare two hypothetical cities of houses connected by roads. The approach implemented here is to pairwise compare each house with those in the other city by creating a house-specific signature. This is accomplished by quantifying the predictability of the location of a person at various times after they left their house, assuming they were moving randomly. This predictability across all houses captures much of the way each city is organized and functions. `align` uses this conceptual rationale to align two networks, with nodes as houses, edges as roads, and random diffusion representing people leaving their houses and walking around the city to other houses. The mechanics of this, which are conceptually akin to flow algorithms and Laplacian dynamics, can be analytically expressed as a Markov chain raised to successive powers which are the durations of diffusion.

Note that the novel part of `align` lies in creating a matrix where the ij entry is a measure of similarity between node i in the first network and node j in the second. The final alignment is found using `solve_LSAP` in the package `clue`, which uses the Hungarian algorithm to solve the assignment problem optimally.

Value

`score` Mean of all alignment scores between nodes in both original networks network_1_input and network_2_input.

`alignment` Data frame of the nodes in both networks, sorted numerically by the first network (why it helps to make the smaller network the first one), and the corresponding alignment score.

`score_with_padding`
Same as score but includes the padding nodes in the smaller network, which can be thought of as a size gap penalty for aligning differently sized networks. Only included if the input networks are different sizes.

`alignment_with_padding`
Same as alignment but includes the padding nodes in the smaller network. Only included if the input networks are different sizes.

References

Kuhn, H. W. (1955). The Hungarian method for the assignment problem. Naval Research Logistics (NRL), 2(1-2), 83-97.

C. Papadimitriou and K. Steiglitz (1982), Combinatorial Optimization: Algorithms and Complexity. Englewood Cliffs: Prentice Hall.

Examples

```
# The two networks to be aligned
net_one <- matrix(runif(25,0,1), nrow=5, ncol=5)
```

```
net_two <- matrix(runif(25,0,1), nrow=5, ncol=5)

align(net_one, net_two)
align(net_one, net_two, base = 1, characterization = "gini", normalization = TRUE)
```

---

gini *Gini coefficient*

Description

Takes a matrix and returns the Gini coefficient of each column.

Usage

```
gini(input, byrow = FALSE)
```

Arguments

input	A matrix where the Gini coefficient will be calculated on each column. Note that vector data must be converted to a single-column matrix.
byrow	Defaults to FALSE. Set to TRUE to calculate the Gini coefficient of each row.

Value

A vector of the Gini coefficients of each column.

References

Gini, C. (1912). Variabilita e mutabilita. Reprinted in Memorie di metodologica statistica (Ed. Pizetti E, Salvemini, T). Rome: Libreria Eredi Virgilio Veschi.

Examples

```
# Vectors are not supported. First convert to a single-column matrix.
sample_data <- runif(20, 0, 1)
gini(matrix(sample_data, ncol = 1))

# Multiple Gini coefficients can be calculated simultaneously
gini(matrix(sample_data, ncol = 2))
```

ics *Induced Conserved Structure (ICS)*

Description

Calculates the Induced Conserved Structure proposed by Patro and Kingsford (2012) of an alignment between two networks.

Usage

```
ics(network_1_input, network_2_input, alignment, flip = FALSE)
```

Arguments

network_1_input	The first network being aligned, which must be in matrix form. If the two networks are of different sizes, it will be easier to interpret the output if this is the smaller one.
network_2_input	The second network, which also must be a matrix.
alignment	A matrix, such as is output by the function NetCom, where the first two columns contain corresponding node IDs for the two networks that were aligned.
flip	Defaults to FALSE. Set to TRUE if the first network is larger than the second. This is necessary because ICS is not a symmetric measure of alignment quality.

Value

A number ranging between 0 and 1. If the Induced Conserved Structure is 1, the two networks are isomorphic (identical) under the given alignment.

References

Patro, R., & Kingsford, C. (2012). Global network alignment using multiscale spectral signatures. Bioinformatics, 28(23), 3105-3114.

Examples

```
# Note that ICS is only defined on unweighted networks.
net_one <- round(matrix(runif(25,0,1), nrow=5, ncol=5))
net_two <- round(matrix(runif(25,0,1), nrow=5, ncol=5))

ics(net_two, net_two, align(net_one, net_two)$alignment)
```

Index