Package ‘networkABC’
March 20, 2021

Type Package
Title Network Reverse Engineering with Approximate Bayesian Computation
Version 0.8-0
Date 2021-03-20
Depends R (>= 3.0.0)
Imports RColorBrewer, network, sna
Suggests ggplot2, knitr, rmarkdown
Author Frederic Bertrand [cre, aut] (<https://orcid.org/0000-0002-0837-8281>),
    Myriam Maumy-Bertrand [aut] (<https://orcid.org/0000-0002-4615-1512>),
    Khadija Musayeva [ctb],
    Nicolas Jung [ctb],
    Université de Strasbourg [cph],
    CNRS [cph]
Maintainer Frederic Bertrand <frederic.bertrand@math.unistra.fr>
Description We developed an inference tool based on approximate Bayesian computation to decipher network data and assess the strength of the inferred links between network's actors. It is a new multi-level approximate Bayesian computation (ABC) approach. At the first level, the method captures the global properties of the network, such as scale-freeness and clustering coefficients, whereas the second level is targeted to capture local properties, including the probability of each couple of genes being linked. Up to now, Approximate Bayesian Computation (ABC) algorithms have been scarcely used in that setting and, due to the computational overhead, their application was limited to a small number of genes. On the contrary, our algorithm was made to cope with that issue and has low computational cost. It can be used, for instance, for elucidating gene regulatory network, which is an important step towards understanding the normal cell physiology and complex pathological phenotype. Reverse-engineering consists in using gene expressions over time or over different experimental conditions to discover the structure of the gene network in a targeted cellular process. The fact that gene expression data are usually noisy, highly correlated, and have high dimensionality explains the need for specific statistical methods to reverse engineer the underlying network.
NeedsCompilation yes
License GPL-3
Description

ABC algorithm for network reverse-engineering

Usage

```r
abc(
  data,
  clust_coeffs = c(0.33, 0.66, 1),
  tolerance = NA,
  number_hubs = NA,
  iterations = 10,
  number_networks = 1000,
  hub_probs = NA,
  neighbour_probs = NA,
)```
is_probs = 1
)

Arguments

- **data**: Any microarray data in the form of a matrix (rows are genes and columns are time points)
- **clust_coeffs**: one dimensional array of size clust_size of clustering coefficients (these clustering coefficient are tested in the ABC algorithm).
- **tolerance**: a real value based for the tolerance between the generated networks and the reference network
- **number_hubs**: number of hubs in the network
- **iterations**: number of times to repeat ABC algorithm
- **number_networks**: number of generated networks in each iteration of the ABC algorithm
- **hub_probs**: one-dimensional array of size number_genes for the each label to be in the role of a hub
- **neighbour_probs**: this is the matrix of neighbour probabilities of size number_nodes*number_nodes
- **is_probs**: this needs to be set either to one (if you specify hub_probs and neighbour_probs) or to zero (if neither probabilities are specified). Attention: you should specify both hub_probs and neighbour_probs if is_probs is one. If is_prob is zero these arrays should simply indicate an array of a specified size..

Examples

```r
M<-matrix(rnorm(30),10,3)
result<-abc(data=M)
```

clusteringCoefficient  

*Calculate the clustering coefficient*

Description

Calculate the clustering coefficient for an adjacency matrix. By default, the local clustering coefficient is calculated. From the PCIT package after it was archived on the CRAN.

Usage

`clusteringCoefficient(adj, FUN = "localClusteringCoefficient", ...)`

Arguments

- **adj**: An adjacency matrix. Calculating the clustering coefficient only makes sense if some connections are zero i.e. no connection.
- **FUN**: The function for calculating the clustering coefficient.
- **...**: Arguments to pass to FUN
localClusteringCoefficient

Value
The clustering coefficient(s) for the adjacency matrix.

Author(s)
Nathan S. Watson-Haigh

See Also
localClusteringCoefficient

Examples
clusteringCoefficient(network_gen(50,.33)$network)

Description
Calculate the local clustering coefficient for each node in an adjacency matrix. The clustering coefficient is defined as the proportion of existing connections from the total possible (Watts and Strogatz, 1998).

Usage
localClusteringCoefficient(adj)

Arguments
adj An adjacency matrix. Calculating the clustering coefficient only makes sense if some connections are zero i.e. no connection.

Value
A vector of local clustering coefficients for each node/gene of the adjacency matrix.

Author(s)
Nathan S. Watson-Haigh

References
netsimul

See Also
  clusteringCoefficient

Examples

  localClusteringCoefficient(network_gen(50,.33)$network)

<table>
<thead>
<tr>
<th>netsimul</th>
<th>Simulated network</th>
</tr>
</thead>
</table>

Description

Result of the use of the network_gen function.

Usage

  netsimul

Format

A list of three objects:

  number_genes  The number of genes in the network
  clust_coef    The clustering coefficient
  network       The simulated network

<table>
<thead>
<tr>
<th>networkABC</th>
<th>networkABC</th>
</tr>
</thead>
</table>

Description

An inference tool based on approximate Bayesian computations to decipher network data and assess the strength of their inferred links.

References

  networkABC: An inference tool for networks based on approximate Bayesian computation, Myriam Maumy-Bertrand, Frédéric Bertrand, preprint.
network_gen

Random scale-free network generation. This function is used intensively in the abc function.

Description
Generate random network topology

Usage

network_gen(number_genes, clust_coef)

Arguments

number_genes A number
clust_coef A number

Value

A list with the number of genes, the targeted clustering coefficient and the resulting network

Examples

network_gen(10, 1)

resabc

Result of an ABC inference

Description
Result for the reverse engineering of a simulated Cascade network

Usage

resabc

Format

A list of 14 objects:

data : The microarray data used, rows are genes and columns are time points.
ngenues : The number of genes.
ntimes : The number of timepoints
clust_size : the size of clusters
clust_coeffs : the clustering coefficient
tolerance : the tolerance between the generated networks and the reference network
number_hubs : number of hubs in the network
iterations : number of times to repeat ABC algorithm
number_networks : number of generated networks in each iteration of the ABC algorithm
hub_probs : one-dimensional array of size number_genes for each label to be in the role of a hub
neighbour_probs : matrix of neighbour probabilities of size number_nodes*number_nodes
is_probs : is equal to 1 since hub_probs and neighbour_probs were specified

---

**showHp**

*Plot for the hub probabilities*

**Description**

Plot for the hub probabilities; there is one probability for each node in the network.

**Usage**

showHp(result)

**Arguments**

result : The result of the abc algorithm.

**Examples**

data(resabc)
showHp(resabc)

---

**showNetwork**

*Plot the final network.*

**Description**

Plot the final network.

**Usage**

showNetwork(res, min_prob)

**Arguments**

res : The result of the abc algorithm.
min_prob : numeric; under this probabilitie value, the link between two genes is set to 0.
Examples

```r
data(resabc)
showNetwork(resabc,.2)
```

---

**showNp**

*Plot for the neighbourhood probabilities*

---

**Description**

Plot for the neighbourhood probabilities; there is one probability for each pair of nodes in the network.

**Usage**

```r
showNp(result)
```

**Arguments**

- `result`: The result of the abc algorithm.

**Examples**

```r
data(resabc)
showNp(resabc)
```
Index

* datasets
  netsimul, 5
  resabc, 6

abc, 2

clusteringCoefficient, 3, 5

localClusteringCoefficient, 4, 4

netsimul, 5
network_gen, 6
networkABC, 5

resabc, 6

showHp, 7
showNetwork, 7
showNp, 8