Package ‘parmigene’

October 14, 2022

Version 1.1.0

Date 2020-07-28

Title Parallel Mutual Information Estimation for Gene Network Reconstruction

Author Gabriele Sales <gabriele.sales@unipd.it>, Chiara Romualdi
      <chiara.romualdi@unipd.it>

Maintainer Gabriele Sales <gabriele.sales@unipd.it>

Description Parallel estimation of the mutual information based on entropy estimates from k-nearest neighbors distances and algorithms for the reconstruction of gene regulatory networks (Sales et al, 2011 <doi:10.1093/bioinformatics/btr274>).

License AGPL-3

URL https://github.com/sales-lab/parmigene

BugReports https://github.com/sales-lab/parmigene/issues

NeedsCompilation yes

Repository CRAN

Date/Publication 2020-08-03 09:50:03 UTC

R topics documented:

aracne.a ................................................................. 2
aracne.m ................................................................. 3
clr ....................................................................... 4
knnmi ................................................................. 5
knnmi.all ............................................................ 6
knnmi.cross .......................................................... 7
mrnet ................................................................. 8

Index 10
Algorithm for the Reconstruction of Accurate Cellular Networks

Description

A function that implements the ARACNE algorithm for the reconstruction of gene interaction networks (additive model).

Usage

aracne.a(mi, eps=0.05)

Arguments

mi matrix of the mutual information.
eps a positive numeric value used to remove the weakest edge of each triple of nodes.

Details

This algorithm considers each triple of edges independently and removes the weakest one if

\[ MI(i; j) < MI(j; k) - \varepsilon \]

and

\[ MI(i; j) < MI(i; k) - \varepsilon \]

By default, the function uses all the available cores. You can set the actual number of threads used to N by exporting the environment variable OMP_NUM_THREADS=N.

Value

A square weighted adjacency matrix of the inferred network.

References


See Also

aracne.m
clr
mrnet
Examples

mat <- matrix(rnorm(1000), nrow=10)
mi <- knnmi.all(mat)
grn <- aracne.a(mi, 0.05)

Description

A function that implements the ARACNE algorithm for the reconstruction of gene interaction networks (multiplicative model).

Usage

aracne.m(mi, tau=0.15)

Arguments

mi matrix of the mutual information.
tau a positive numeric value used to remove the weakest edge of each triple of nodes.

Details

This algorithm considers each triple of edges independently and removes the weakest one if

\[ MI(i; j) < MI(j; k) \times (1 - \tau) \]

and

\[ MI(i; j) < MI(i; k) \times (1 - \tau) \]

By default, the function uses all the available cores. You can set the actual number of threads used to N by exporting the environment variable OMP_NUM_THREADS=N.

Value

A square weighted adjacency matrix of the inferred network.

References

Description

A function that infers the interaction network using the CLR algorithm.

Usage

clr(mi)

Arguments

mi matrix of the mutual information.

Details

CLR computes the score

\[ \sqrt{z_i^2 + z_j^2} \]

for each pair of variables \( i, j \), where

\[ z_i = \max(0, (I(X_i; X_j) - mean(X_i))/sd(X_i)) \]

and \( mean(X_i) \) and \( sd(X_i) \) are the mean and the standard deviation of the mutual information values \( I(X_i; X_k) \) for all \( k = 1, \ldots, n \).

By default, the function uses all the available cores. You can set the actual number of threads used to \( N \) by exporting the environment variable \( \text{OMP_NUM_THREADS}=N \).

Value

A square weighted adjacency matrix of the inferred network.
knnmi

References

See Also
aracne.a
aracne.m
mrnet

Examples
mat <- matrix(rnorm(1000), nrow=10)
mi <- knnmi.all(mat)
grn <- clr(mi)

---

knnmi
Parallel Mutual Information Estimation

Description
A function to perform a parallel estimation of the mutual information of vectors x and y using entropy estimates from K-nearest neighbor distances.

Usage
knnmi(x, y, k=3, noise=1e-12)

Arguments

- `x` a numeric vector.
- `y` a numeric vector with the same length of `x`.
- `k` the number of nearest neighbors to be considered to estimate the mutual information. Must be less than the number of elements of `x`.
- `noise` the magnitude of the random noise added to break ties.

Details
The function adds a small random noise to the data in order to break ties due to limited numerical precision.

By default, the function uses all available cores. You can set the actual number of threads used to `N` by exporting the environment variable `OMP_NUM_THREADS=N`. 
References

See Also
knnmi.cross
knnmi.all

Examples
```r
x <- rnorm(100)
y <- rnorm(100)
knnmi(x, y, 5)
```

---

### knnmi.all

**Parallel Mutual Information Estimation Between All Matrix Rows**

**Description**
A function that computes the mutual information between all pairs of rows of matrix `mat` using entropy estimates from K-nearest neighbor distances.

**Usage**
```r
knnmi.all(mat, k=3, noise=1e-12)
```

**Arguments**
- `mat` a numeric matrix (for the reconstruction of gene regulatory networks, genes on rows and samples on columns).
- `k` the number of nearest neighbors to consider to estimate the mutual information. Must be less than the number of columns of `mat`.
- `noise` the magnitude of the random noise added to break ties.

**Details**
The function adds a small random noise to the data in order to break ties due to limited numerical precision.

By default, the function uses all the available cores. You can set the actual number of threads used to `N` by exporting the environment variable `OMP_NUM_THREADS=N`.

**References**
**knnmi.cross**

**See Also**

- knnmi
- knnmi.cross

**Examples**

```r
mat <- matrix(rnorm(1000), nrow=10)
knnmi.all(mat, 5)
```

---

**knnmi.cross**  
*Parallel Mutual Information Estimation Between the Rows of Two Matrices*

**Description**

A function that estimates the mutual information between all pairs of rows of matrices `mat1` and `mat2` using entropy estimates from K-nearest neighbor distances.

**Usage**

```r
knnmi.cross(mat1, mat2, k=3, noise=1e-12)
```

**Arguments**

- **mat1**: a numeric matrix (for the reconstruction of gene regulatory networks, genes on rows and samples on columns).
- **mat2**: a numeric matrix with the same number of columns as `mat1`.
- **k**: the number of nearest neighbors to consider to estimate the mutual information. Must be less than the number of columns of `mat1`.
- **noise**: the magnitude of the random noise added to break ties.

**Details**

The function adds a small random noise to the data in order to break ties due to limited numerical precision.

By default, the function uses all the available cores. You can set the actual number of threads used to `N` by exporting the environment variable `OMP_NUM_THREADS=N`.

**References**


**See Also**

- knnmi
- knnmi.all
Examples

```r
mat1 <- matrix(rnorm(1000), nrow=10)
mat2 <- matrix(rnorm(1000), nrow=10)
knnmi.cross(mat1, mat2, 5)
```

---

**mrnet**  
*Maximum Relevance Minimum Redundancy*

**Description**

A function that infers the interaction network using the MRNET algorithm.

**Usage**

```r
mrnet(mi)
```

**Arguments**

- `mi`  
  matrix of the mutual information.

**Details**

The MRNET approach starts by selecting the variable $X_i$ having the highest mutual information with the target $Y$.

Then, it repeatedly enlarges the set of selected variables $S$ by taking the $X_k$ that maximizes

$$I(X_k;Y) - \text{mean}(I(X_k;X_i))$$

for all $X_i$ already in $S$.

The procedure stops when the score becomes negative.

By default, the function uses all the available cores. You can set the actual number of threads used to $N$ by exporting the environment variable `OMP_NUM_THREADS=N`.

**Value**

A square weighted adjacency matrix of the inferred network.

**References**

mrnet

See Also

aracne.a
aracne.m
clr

Examples

mat <- matrix(rnorm(1000), nrow=10)
mi  <- knnmi.all(mat)
grn <- mrmnet(mi)
Index

aracne.a, 2, 4, 5, 9
aracne.m, 2, 3, 5, 9
clr, 2, 4, 4, 9
knnmi, 5, 7
knnmi.all, 6, 6, 7
knnmi.cross, 6, 7, 7
mrnet, 2, 4, 5, 8