Package ‘BiRewire’

April 9, 2015

Version 1.8.0
Date 2014-09-16
Title High-performing routines for the randomization of a bipartite
  graph (or a binary event matrix) preserving degree distribution
  (or marginal totals).
Maintainer Andrea Gobbi <gobbi.andrea@mail.com>
Description Fast functions for bipartite network rewiring through N
  consecutive switching steps (See References) and for the
  computation of the minimal number of switching steps to be
  performed in order to maximise the dissimilarity with respect
  to the original network. Includes function for the analysis of
  the introduced randomness across the switching and several
  other routines to analyse the resulting networks and their
  natural projections. Extension to undirected networks (not
  bipartite) is also provided.
License GPL-3
Depends igraph
Suggests RUnit, BiocGenerics
Author Andrea Gobbi [aut], Davide Albanese [cbl], Francesco Iorio
  [cbl], Giuseppe Jurman [cbl], Julio Saez-Rodriguez [cbl].
URL http://www.ebi.ac.uk/~iorio/BiRewire
biocViews Network

R topics documented:

  BiRewire-package .................................................. 2
  birewire.analysis ................................................ 3
  birewire.analysis.undirected .................................. 5
  birewire.bipartite.from.incidence ............................. 8
  birewire.rewire .................................................. 9
  birewire.rewire.bipartite ..................................... 11
  birewire.rewire.bipartite.and.projections .................. 12
The BiRewire package

Description

R package for computationally-efficient rewiring of bipartite graphs (or randomisation of 0-1 tables with prescribed marginal totals). The package provides useful functions for the analysis and the randomisation of large biological datasets that can be encoded as 0-1 tables, hence modeled as bipartite graphs by considering a 0-1 table as an incidence matrix. Large collections of such randomised tables can be used to approximate null models, preserving event-rates both across rows and columns, for statistical significance tests of combinatorial properties of the original dataset. Routines for undirected graphs are also provided.

Details

Summary:

<table>
<thead>
<tr>
<th>Package</th>
<th>BiRewire</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>1.3.2</td>
</tr>
<tr>
<td>Date</td>
<td>2013-07-15</td>
</tr>
<tr>
<td>Require:</td>
<td>igraph, R&gt;=2.10</td>
</tr>
<tr>
<td>URL</td>
<td><a href="http://www.ebi.ac.uk/~iorio/BiRewire">http://www.ebi.ac.uk/~iorio/BiRewire</a></td>
</tr>
<tr>
<td>License</td>
<td>GPL-3</td>
</tr>
</tbody>
</table>

Author(s)

Andrea Gobbi [aut], Davide Albanese [cbt], Francesco Iorio [cbt], Giuseppe Jurman [cbt].

Maintainer: Andrea Gobbi <gobbi.andrea@mail.com>

References


---

**birewire.analysis**

*Analysis of Jaccard similarity trends across switching steps.*

**Description**

This function performs a sequence of *max.iter* switching steps on the input bipartite graph *g* and compute the Jaccard similarity between *g* (the initial network) and its rewired version each *step* switching steps.

**Usage**

```r
birewire.analysis(incidence, step=10, max.iter="n",accuracy=1, verbose=TRUE,MAXITER_MUL=10,exact=F)
```

**Arguments**

- **incidence**: Incidence matrix of the initial bipartite graph *g* (can be extracted from an igraph bipartite graph using the `get.incidence` function).
- **step**: 10 (default): the interval (in terms of switching steps) at which the Jaccard index between *g* and the its current rewired version is computed.
- **max.iter**: "n" (default) the number of switching steps to be performed (or if `exact==TRUE` the number of successful switching steps). If equal to "n" then this number is considered equal to the analytically derived lower bound presented in Gobbi et al. (see References): \(N = \frac{e}{2(1 - d) \ln (e - d e)}\) if exact is FALSE, \(N = \frac{e(1 - d)}{2 \ln (e - d e)}\) otherwise , where *e* is the number of edges of *g* and *d* its edge density . This bound is much lower than the empirical one proposed in Milo et al. 2003 (see References);
- **accuracy**: 1 (default) is the desired level of accuracy reflecting the average distance between the Jaccard index at the N-th step and its analytically derived fixed point;
- **verbose**: TRUE (default). When TRUE a progression bar is printed during computation;
- **MAXITER_MUL**: 10 (default). If `exact==TRUE` in order to prevent a possible infinite loop the program stops anyway after `MAXITER_MUL*max.iter` iterations;
- **exact**: FALSE (default). If TRUE the program performs *max.iter* successful switching steps, otherwise the program will count also the not-performed switching steps;
Details

This function performs `max.iter` switching steps (see references). In particular, at each step two edges are randomly selected from the current version of `g`. Let these two edges be `(a, b)` and `(c, d)` (where `a` and `c` belong to the first class of nodes whereas `b` and `d` belong to the second one), with `a ≠ c` and `b ≠ d`. If the `(a, d)` and `(c, b)` edges are not already present in the current version of `g` then `(a, d)` and `(c, b)` replace `(a, b)` and `(c, d)

At each step number of switching steps the function computes the Jaccard index between the original graph `g` and its current version.

Value

A list containing a vector of Jaccard index values computed each (scores) switching steps, whose length is equal to `max.iter/step`, and the analytically derived lower bound (`N`) of switching steps to be performed by the switching algorithm in order to provide the revised version of `g` with the maximal level of achievable randomness (in terms of dissimilarity from the initial `g`).

Author(s)

Andrea Gobbi
Maintainer: Andrea Gobbi <gobbi.andrea@mail.com>
Special thanks to:
Davide Albanese

References


Examples

library(igraph)
library(BiRewire)
g <- simplify(graph.bipartite( rep(0:1,length=100),
c(c(1:100),seq(1,100,3),seq(1,100,7),100,seq(1,100,13),
seq(1,100,17),seq(1,100,19),seq(1,100,23),100
)))

##get the incidence matrix of g
m<-as.matrix(get.incidence(graph=g))

## set parameters
step=1
max=100*length(E(g))

## perform two different analysis using two different maximal number of switching steps
scores<-birewire.analysis(m,step,max)
scores2<-birewire.analysis(m,step,"n")

## plot the Jaccard index scores across intervals of switching steps
plot(x=step*seq(1:length(scores$similarity_scores)),y= scores$similarity_scores,
type="l",xlab="Number of rewiring",ylab="Index value",ylim=c(0,1))
lines(x=step*seq(1:length(scores2$similarity_scores)),y= scores2$similarity_scores,
col="red")
legend(max*0.5,1, c("Jaccard index","Jaccard index with lower-bound N"), cex=0.9,
col=c("black","red"), lty=1:1,lwd=3)
Arguments

adjacency  adjacency matrix of the undirected graph \( g \) (can be extracted from a \texttt{igraph} graph using the \texttt{get.adjacency} function);

step       10 (default): the interval (in terms of switching steps) at which the Jaccard index between \( g \) and its current rewired version is computed;

\( \text{max.iter} \)  "n" (default) the number of switching steps to be performed (or if \( \text{exact} = \text{TRUE} \) the number of successful switching steps). If equal to "n" then this number is considered equal to the analytically derived lower bound presented in Gobbi \textit{et al.} (see References): \[ N = \frac{e}{(2d^3 - 6d^2 + 2d + 2)} \ln (e - de) \] if \( \text{exact} \) is \( \text{FALSE} \), \[ N = \frac{e(1 - d)}{2} \ln (e - de) \] otherwise , where \( e \) is the number of edges of \( g \) and \( d \) its edge density . This bound is much lower than the empirical one proposed in Milo \textit{et al.} 2003 (see References);

accuracy  1 (default) is the desired level of accuracy reflecting the average distance between the Jaccard index at the N-th step and its analytically derived fixed point.

verbose    TRUE (default). When TRUE a progression bar is printed during computation.

\( \text{MAXITER_MUL} \)  10 (default). If \( \text{exact} = \text{TRUE} \) in order to prevent a possible infinite loop the program stops anyway after \( \text{MAXITER_MUL} \times \text{max.iter} \) iterations;

\( \text{exact} \)  \( \text{FALSE} \) (default). If \( \text{TRUE} \) the program performs \( \text{max.iter} \) successful switching steps, otherwise the program will count also the not-performed switching steps;

Details

This function performs \( \text{max.iter} \) switching steps (see references). In particular, at each step two edges are randomly selected from the current version of \( g \). Let these two edges be \((a, b)\) and \((c, d)\), with \( a \neq c \), \( b \neq d \), \( a \neq d \), \( b \neq c \).

If the \((a, d)\) and \((c, b)\) (or \((a, d)\) and \((b, d)\)) edges are not already present in the current version of \( g \) then \((a, d)\) and \((c, b)\) replace \((a, b)\) and \((c, d)\) (or \((a, b)\) and \((c, d)\) replace \((a, c)\) and \((b, d)\)). If both of the configurations are allowed, then one of them is randomly selected.

At each \( \text{step} \) switching steps the function computes the Jaccard index between the original graph \( g \) and its current rewired version.

Value

A list containing a vector of Jaccard index values computed each \( \text{(scores)} \) switching steps whose length is \( \text{max.iter}/\text{step} \) and the analytically derived lower bound \( (N) \) of switching steps to be performed by the switching algorithm in order to provide the rewired version of \( g \) with maximal achievable level of randomness (in terms of dissimilarity from the initial \( g \)).

Author(s)

Andrea Gobbi
Maintainer: Andrea Gobbi <gobbi.andrea@mail.com>
Special thanks to:
Davide Albanese
References


Examples

library(igraph)
library(BiRewire)
g <- erdos.renyi.game(1000,0.1)
##get the incidence matrix of g
m<-as.matrix(get.adjacency(graph=g,sparse=FALSE))

## set parameters
step=1000
max=100*length(E(g))

## perform two different analysis using two different numbers of switching steps
scores<-birewire.analysis.undirected(m,step,max)
scores2<-birewire.analysis.undirected(m,step,"n")

## plot the Jaccard index scores across intervals of switching steps
plot(x=step*seq(1:length(scores$similarity_scores)),y= scores$similarity_scores,
    type=l,xlab="Number of rewiring",ylab="Index value",ylim=c(0,1))
lines(x=step*seq(1:length(scores2$similarity_scores)),y= scores2$similarity_scores,
     col="red")
legend(max*0.5,1, c("Jaccard index","Jaccard index with lower-bound N"), cex=0.9,
       col=c("black","red"), lty=1:1,lwd=3)
birewire.bipartite.from.incidence

Converts an incidence matrix into a bipartite graph.

Description

This function creates an igraph bipartite graph from an incidence matrix.

Usage

birewire.bipartite.from.incidence(matrix, directed=FALSE, reverse=FALSE)

Arguments

matrix incidence matrix: an (n-by-m) binary matrix where rows correspond to vertices in the first class while columns correspond to vertices in the second one;
directed Logical, if TRUE a directed graph is created.
reverse Logical, if TRUE the edges will be directed from the nodes in the second class to those in the first one.

Details

The function calls graph.bipartite of package igraph. See igraph documentation for more details.

Value

Bipartite igraph graph.

Author(s)

Andrea Gobbi
Maintainer: Andrea Gobbi <gobbi.andrea@mail.com>

References


Examples

library(igraph)
library(BiRewire)
g <- simplify(graph.bipartite( rep(0:1,length=100),
c(c(1:100), seq(1,100,3), seq(1,100,7), 100, seq(1,100,13),
   seq(1,100,17), seq(1,100,19), seq(1,100,23), 100
 ))
##gets the incidence matrix of g
m<-as.matrix(get.incidence(graph=g))

##rewire the current graph
m2=birewire.rewire.bipartite(m,1/zero.noslash/zero.noslash)

#create the rewired bipartite graph
g2<-birewire.bipartite.from.incidence(m2,TRUE,FALSE)

---

**birewire.rewire**  
*Efficient rewiring of undirected graphs*

###Description
Optimal implementation of the switching algorithm. It returns the rewired version of the initial undirected graph or its adjacency matrix.

####Usage

```r
birewire.rewire(adjacency, max.iter="n",accuracy=1,
verbose=TRUE,MAXITER_MUL=10,exact=F)
```

####Arguments

- **adjacency**: An igraph undirected graph g or its adjacency matrix (can be extracted from g using `get.adjacency(g)`);
- **max.iter**: "n" (default) the number of switching steps to be performed (or if `exact==TRUE` the number of **successful** switching steps). If equal to "n" then this number is considered equal to the analytically derived lower bound presented in *Gobbi et al.* (see References): \( N = \frac{e}{2d^3 - 6d^2 + 2d + 2} \ln (e - de) \) if exact is FALSE, \( N = \frac{e(1 - d)}{2} \ln (e - de) \) otherwise, where e is the number of edges of g and d its edge density. This bound is much lower than the empirical one proposed in *Milo et al. 2003* (see References);
- **accuracy**: 1 (default) is the desired level of accuracy reflecting the average distance between the Jaccard index at the N-th step and its analytically derived fixed point;
- **verbose**: TRUE (default) boolean value. If TRUE print a processing bar during the rewiring algorithm.
- **MAXITER_MUL**: 10 (default). If `exact==TRUE` in order to prevent a possible infinite loop the program stops anyway after `MAXITER_MUL*max.iter` iterations;
- **exact**: FALSE (default). If TRUE the program performs `max.iter` **successful** switching steps, otherwise the program will count also the not-performed switching steps;

###Details
Performs at most `max.iter` number of rewiring steps producing a rewired version of an initial undirected graph.
Value

Adjacency matrix of the rewired graph.

Author(s)

Andrea Gobbi

Special thanks to:
Maintainer: Andrea Gobbi <gobbi.andrea@mail.com>
Davide Albanese

References


Examples

```r
library(igraph)
library(BiRewire)
g <- erdos.renyi.game(1000,0.1)
##gets the incidence matrix of g
m<-as.matrix(get.adjacency(graph=g,sparse=FALSE))

## sets parameters
step=1000
max=100*length(E(g))

#rewiring
m2=birewire.rewire(m,100*length(E(g)))
#creates the corresponding bipartite graph
g2<-graph.adjacency(m2,mode="undirected")
```
Efficient rewiring of bipartite graphs

Description

Optimal implementation of the switching algorithm. It returns the rewired version of the initial bipartite graph or its incidence matrix.

Usage

birewire.rewire.bipartite(incidence, max.iter="n",accuracy=1,verbose=TRUE,
MAXITER_MUL=10,exact=F)

Arguments

incidence
Incidence matrix of the initial bipartite graph \textit{g} (can be extracted from an \texttt{igraph} bipartite graph using the \texttt{get.incidence} function; or the entire bipartite \texttt{igraph} graph

max.iter
"n" (default) the number of switching steps to be performed (or if \texttt{exact==TRUE} the number of \textbf{successful} switching steps). If equal to "n" then this number is considered equal to the analytically derived lower bound presented in \texttt{Gobbi et al.} (see References): 
\[ N = \frac{e}{2(1-d) \ln (e-de)} \] if exact is FALSE, 
\[ N = \frac{e(1-d)/2 \ln (e-de)}{} \] otherwise , where \( e \) is the number of edges of \textit{g} and \( d \) its edge density. This bound is much lower than the empirical one proposed in \texttt{Milo et al. 2003} (see References);

accuracy
1 (default) is the desired level of accuracy reflecting the average distance between the Jaccard index at the N-th step and its analytically derived fixed point;

verbose
TRUE (default). When TRUE a progression bar is printed during computation.

MAXITER_MUL
10 (default). If \texttt{exact==TRUE} in order to prevent a possible infinite loop the program stops anyway after \texttt{MAXITER_MUL*max.iter} iterations;

exact
FALSE (default). If TRUE the program performs \texttt{max.iter} \textbf{successful} switching steps, otherwise the program will count also the not-performed switching steps.

Details

Main function of the package. It performs at most \texttt{max.iter} switching steps producing a rewired version of an initial bipartite graph.

Value

Incidence matrix of the rewired graph.
birewire.rewire.bipartite.and.projections

Analysis and rewiring function processing a bipartite graphs and its two projections

Description

This function performs the same analysis of birewire.analysis but additionally it provides in output a rewired version of the two networks resulting from the natural projections of the initial graph, together with the corresponding Jaccard index trends.
birewire.rewire.bipartite.and.projections

Usage

birewire.rewire.bipartite.and.projections(graph, step=10, max.iter="n", accuracy=1, verbose=TRUE, MAXITER_MUL=10)

Arguments

graph A bipartite graph g;
max.iter "n" (default) the number of successful switching steps to be performed. If equal to "n" then this number is considered equal to the analytically derived lower bound \( N = e(1 - d)/2 \ln(e - de) \) presented in Gobbi et al. (see References);
step 10 (default): the interval (in terms of switching steps) at which the Jaccard index between g and the its current rewired version is computed;
accuracy 1 (default) is the desired level of accuracy reflecting the average distance between the Jaccard index at the N-th step and its analytically derived fixed point;
verbose TRUE (default) boolean value. If TRUE print a processing bar during the rewiring algorithm.
MAXITER_MUL 10 (default). Since \( N \) indicates the number of successful switching steps, in order to prevent a possible infinite loop the program stops anyway after MAXITER_MUL*max.iter iterations ;

Details

See birewire.analysis for details.

Value

A list containing the three sequences of Jaccard index values (similarity_scores, similarity_scores.proj1, similarity_scores.proj2) for the three resulting graphs respectively (rewired, rewired.proj1, rewired.proj2). The first one is the rewired version of the initial graph g, while the second and the third one are rewired versions of its natural projections.

Author(s)

Andrea Gobbi
Maintainer: Andrea Gobbi <gobbi.andrea@mail.com>

References

Examples

```r
library(igraph)
library(BiRewire)
g <- simplify(graph.bipartite( rep(0:1,length=100),
c(c(1:100),seq(1,100,3),seq(1,100,7),100,seq(1,100,13),
seq(1,100,17),seq(1,100,19),seq(1,100,23),100 )))
##gets the incidence matrix of g
m<-as.matrix(get.incidence(graph=g))

## rewires g and its projections
result=birewire.rewire.bipartite.and.projections(g,step=10,max.iter="n",accuracy=1)
```

birewire.similarity

Compute the Jaccard similarity index between two binary matrices with the same number of non-null entries and the same row- and column-wise sums.

Description

Compute the Jaccard similarity index between two binary matrices with the same number of non-null entries and the same row- and column-wise sums.

Usage

`birewire.similarity( m1,m2)`

Arguments

- `m1`: First matrix;
- `m2`: Second matrix.

Details

The **Jaccard** index between two sets $M$ and $N$ is defined as:

$$\frac{|M \cup N|}{|M \cap N|}$$

With $M$ and $N$ binary matrices, the Jaccard index is computed as:

$$\frac{\sum N_{i,j} \land M_{i,j}}{\sum N_{i,j} \lor M_{i,j}}.$$

The Jaccard index ranges between 0 and 1.
BRCA_binary_matrix

Value

Returns the Jaccard similarity index between the two matrices

Author(s)

Andrea Gobbi
Maintainer: Andrea Gobbi <gobbi.andrea@mail.com>

Examples

```r
library(igraph)
library(BiRewire)
g <- simplify(graph.bipartite( rep(0:1,length=100),
c(c(1:100),seq(1,100,3),seq(1,100,7),100,seq(1,100,13),
seq(1,100,17),seq(1,100,19),seq(1,100,23),100 )))
g2=birewire.rewire.bipartite(g)

birewire.similarity(get.incidence(g,sparse=FALSE),get.incidence(g2,sparse=FALSE))
```

BRCA_binary_matrix  TCGA Brest Cancer data

Description

Breast cancer samples and their respective mutations downloaded from the Cancer Cancer Genome Atlas (TCGA), used in Gobbi et al. Germline mutations were filtered out of the list of reported mutations; synonymous mutations and mutations identified as benign and tolerated were also removed from the dataset. The bipartite graph resulting when considering this matrix as an incidence matrix has $n_r = 757, n_c = 9757, e = 19758$ for an edge density equal to 0.27%.

Usage

```r
data(BRCA_binary_matrix)
```

Source

http://tcga.cancer.gov/dataportal/

References

Index

*Topic bipartite graph, incidence matrix
   birewire.bipartite.from.incidence, 8

*Topic bipartite graph, projection, rewire
   birewire.rewire.bipartite.and.projections, 12

*Topic bipartite graph, rewire
   birewire.rewire.bipartite, 11

*Topic datasets
   BRCA_binary_matrix, 15

*Topic package
   BiRewire-package, 2

*Topic rewire, bipartite graph
   birewire.analysis, 3

*Topic rewire, undirected graph
   birewire.analysis.undirected, 5

*Topic similarity, jaccard
   birewire.similarity, 14

*Topic undirected graph, rewire
   birewire.rewire, 9

BiRewire (BiRewire-package), 2
BiRewire-package, 2
birewire.analysis, 3, 12, 13
birewire.analysis.undirected, 5
birewire.bipartite.from.incidence, 8
birewire.rewire, 9
birewire.rewire.bipartite, 11
birewire.rewire.bipartite.and.projections, 12
birewire.similarity, 14
BRCA_binary_matrix, 15