BiRewire

Andrea Gobbi, Francesco Iorio

Contents

1 Overview

BiRewire is an R package implementing high-performing routines for the randomisation of bipartite graphs preserving their node degrees (i.e. Network Rewiring), through the Switching Algorithm (SA) [4].
This package is particularly useful for the randomisation of '0-1' tables (or presence-absence matrices) in which the distributions of non-null entries (i.e. presence distributions) must be preserved both across rows and columns. By considering these tables as incidence matrices of bipartite graphs then this problem reduces to bipartite network rewiring.
For example, by modeling a genomic dataset as a binary event matrix (BEM), in which rows correspond to samples, columns correspond to genes and the (i, j) entry is non-null if the i-th sample harbours a mutation in the j-th gene, then with BiRewire is possible to randomise the dataset preserving its mutation rates both across samples and genes. This is crucial to preserve tumour specific alterations, dependencies between gene-mutations and heterogeneity in mutation/copy-number-alteration rates across patients.
Large collections of such randomised tables can be then used to approximate samples from the uniform distribution of all the possible genomic datasets with the same mutation-rates of the initial one. Finally this data can be used as null model to test the statistical significance of several combinatorial properties of the original dataset: for example the tendency of a group of genes to be co- or

mutually-mutated [6].

Specifically, with *BiRewire* users can:

1. create bipartite graphs from genomic BEMs (or, generally, from any kind of ṕresence-absencemátrix);
2. perform an analysis, which consists of studying the trend of Jaccard Similarity between the original network and its rewired versions across the switching steps (by using a user-defined sampling time), and analytically estimating the number of steps at which this similarity reaches a plateau (i.e. the maximal level of randomness is achieved) according to the lower bound derived in [1];
3. generate rewired versions of a bipartite graph with the analytically derived bound as number of switching steps or a user-defined one;
4. derive projections of the starting network and its rewired version and perform different graph-theory analysis on them.

All the functions of the package are written in C-code and R-wrapped.

2 Installation

It is possible to download the package from `http://www.ebi.ac.uk/~iorio/BiRewire` and install it with the shell-command:

```
R CMD INSTALL BiRewire_xx.yy.zz.tar.gz
```

or with biocLite() directly in R:

```
source("http://bioconductor.org/biocLite.R")
biocLite("BiRewire")
```

To load BiRewire use the following commands:

```
> library(BiRewire)
```

3 Package Dependencies

BiRewire requires the R package **igraph** (see [5]) available at the CRAN repository, or downloadable at `http://cran.r-project.org/web/packages/igraph/index.html`.

4 Notation

Let $\mathcal{G}$ be a bipartite graph, i.e. a graph containing two classes of nodes V_r and V_c such that every edge $e \in E$ connects one node in the first class to a node in the second class.
Let $\mathcal{B}$ be the incidence matrix of $\mathcal{G}$, i.e. the $|V_r| \times |V_c|$ binary matrix whose generic entry $m_{i,j}$ is not null if an only if $(i, j) \in E$.

The number of edges is indicated with $e = |E|$ and the edge density with $d = \frac{e}{|V_r||V_c|}$.

The SA performs N Switching Steps (SSs), in which:

1. two edges (a, b) and (c, d) both $\in E$ are randomly selected,
2. if $a \neq c$, $b \neq d$, $(a, d) \notin E$ and $(b, d) \notin E$ then:
   (a) the edges (a, d) and (b, d) are added to E and
   (b) the edges (a, b) and (c, d) are removed from E.

Notice that we count a SS only if it is successfully performed.

The *Jaccard Index* (JI, [9]) is used to quantify the similarity between the original graph and its rewired version at the k-th SS. Since the SA preserves the degree distribution and does not alter the number of nodes, the JI, indicated with $s^{(k)}$, can be computed as

$$s^{(k)} = \frac{x^{(k)}}{2e - x^{(k)}},$$

where $x^{(k)}$ is the number of edges in common between the two graphs. The number N of SSs providing the rewired version of a network with the maximally achievable level of randomness (in terms of average dissimilarity from the original network) is asymptotically equal to

$$\frac{e(1-d)}{2} \ln e(1-d).$$

This bound is much lower than the empirical one proposed in [4] (see Reference for details).

5 Function Description

In this section all the functions implemented in BiRewire are described with a simple practical example in which a real breast cancer dataset is modeled as a bipartite network, and randomised preserving the mutation-rate both across samples and genes (i.e. the corresponding bipartite network is rewired). In each of the following functions it is possible to perform N **successful** switching steps (see [1] for more details about this more general bound) using the flag exact=TRUE. To prevent a possible indinite loop, the program performs at maximum MAXITER_MUL*max.iter iterations.

5.1 birewire.analysis andbirewire.analysis.undirected

First of all, we create a bipartite network modeling a genomic breast cancer dataset downloaded from the Cancer Genome Atlas (TCGA) projects data portal `http://tcga.cancer.gov/dataportal/`, used in [1] From this dataset germline mutations were filtered out with state-of-the-art softwares; synonymous mutations and mutations identified as benign and tolerated were also removed. The resulting bipartite graph has $n_r = 757$ nodes (corresponding to samples),

$n_c = 9,757$ nodes (corresponding to genes), and $e = 19,758$ edges connecting a node in n_r to a node in n_c if the gene corresponding to the node in n_r is mutated to the samples corresponding to the node in n_C. The edge density of this network is 0.27%.

The genomic dataset (in the form of a binary matrix in which rows correspond to samples, columns correspond to genes and the (i, j) entry is non null if the i-th sample harbours a mutation in the j-th gene) can be loaded and modeled as a bipartite graph, with the following commands:

```
> data(BRCA_binary_matrix)##loads an binary genomic event matrix for the
>
> g=birewire.bipartite.from.incidence(BRCA_binary_matrix)##models the dataset
>
```

Once the bipartite graph is created it is possible to conduct the analysis by calling the **birewire.analysis** function, using the following commands:

```
> step=5000
> max=100*sum(BRCA_binary_matrix)
> scores<-birewire.analysis(BRCA_binary_matrix,step,verbose=FALSE,max.iter=max)
> plot(x=step*seq(1:length(scores$similarity_scores)),y= scores$similarity_scores,
+                 type='l', xlab="Number of switching steps",
+                 ylab="Jaccard Similarity Score",ylim=c(0,1))
> legend(max*0.8,1, c("Jaccard Similarity","N"),
+        cex=0.9, col=c("black","red"), lty=1:1,lwd=3)
> abline(v=scores$N,col='red')
> plot(x=step*seq(1:length(scores$similarity_scores)),y= scores$similarity_scores,
+                         type='l',xlab="Number of switching steps",
+                         ylab="Jaccard Similarity Score",log="xy",main="Log-Log plot")
> legend("topright", c("Jaccard Similarity","N"),
+        cex=0.9, col=c("black","red"), lty=1:1,lwd=3)
> abline(v=scores$N,col='red')
```

The function **birewire.analysis** returns the Jaccard similarity sampled every 5000 SSs. In the resulting plots the value of the analyitically derived lower bound to the number of switching steps is also visualised **$N** . For more details see the the documentation.

The same analysis can be performed on general undirected networks (not bipartite).

```
> g.und<-erdos.renyi.game(directed=F,loops=F,n=1000,p.or.m=0.01)
> m.und<-get.adjacency(g.und,sparse=FALSE)
> step=100
> max=100*length(E(g.und))
> scores.und<-birewire.analysis.undirected(m.und,step=step,verbose=FALSE,max.iter=max)
> plot(x=step*seq(1:length(scores.und$similarity_scores)),y= scores.und$similarity_scores,
+                 type='l', xlab="Number of switching steps",
+                 ylab="Jaccard Similarity Score",ylim=c(0,1))
> legend(max*0.8,1, c("Jaccard Similarity","N"),
```

```
+        cex=0.9, col=c("black","red"), lty=1:1,lwd=3)
> abline(v=scores.und$N,col='red')
> plot(x=step*seq(1:length(scores.und$similarity_scores)),y= scores.und$similarity_scores,
+                          type='l',xlab="Number of switching steps",
+                          ylab="Jaccard Similarity Score",log="xy",main="Log-Log plot")
> legend("topright", c("Jaccard Similarity","N"),
+        cex=0.9, col=c("black","red"), lty=1:1,lwd=3)
> abline(v=scores.und$N,col='red')
```

5.2 birewire.rewire.bipartite

To rewire a bipartite graph two modalities are available. Both of them can be used with the analytical bound N as number of switching steps or with a user defined value. The function takes in input an incidence matrix $\mathcal{B}$ or the an *igraph* bipartite graph.

```
> m2<-birewire.rewire.bipartite(BRCA_binary_matrix,verbose=FALSE)
> g2<-birewire.rewire.bipartite(g,verbose=FALSE)
```

The first function gives in output the incidence matrix of the rewired graph while the second one a rewired *igraph* graph. See documentation for further details.

5.3 birewire.rewire

To rewire a general undirected graph the following functions can be used:

```
> m2.und<-birewire.rewire(m.und,verbose=FALSE)
> g2.und<-birewire.rewire(g.und,verbose=FALSE)
```

5.4 birewire.similarity

This function computes the Jaccard index between two incidence matrices with same dimensions and node degrees.

```
> sc=birewire.similarity(BRCA_binary_matrix,m2)
> sc=birewire.similarity(BRCA_binary_matrix,t(m2))#also works
```

5.5 birewire.rewire.bipartite.and.projections

The following functions execute the Switching Algorithm and computes similarity trends across its switching steps for the two natural projections of the starting bipartite graph

```
> #use a smaller graph!
> gg <- 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
+ )))
> result=birewire.rewire.bipartite.and.projections(gg,step=10,
+         max.iter="n",accuracy=1,verbose=FALSE)
```

```
> plot(result$similarity_scores.proj2,type='l',col='red',ylim=c(0,1))
> lines(result$similarity_scores.proj1,type='l',col='blue')
> legend("top",1, c("Proj2","Proj1"), cex=0.9, col=c("blue","red"), lty=1:1,lwd=3)
```

References

[1] Gobbi, A. and Iorio, F. and Dawson, K. J. and Wedge, D. C. and Tamborero, D. and Alexandrov, L. B. and Lopez-Bigas, N. and Garnett, M. J. and Jurman, G. and Saez-Rodriguez, J. (2014) *Fast randomization of large genomic datasets while preserving alteration counts* Bioinformatics 2014 30 (17): i617-i623 doi: 10.1093/bioinformatics/btu474.

[2] Gobbi, A. and Jurman, G. *Number of required Switching Steps in the Switching Algorithm for undirected graphs*.in preparation, .

[3] Jaccard, P. (1901), *Etude comparative de la distribution florale dans une portion des Alpes et des Jura*, Bulletin de la Societe Vaudoise des Sciences Naturelles 37:547–579.

[4] R. Milo, N. Kashtan, S. Itzkovitz, M. E. J. Newman, U. Alon (2003), *On the uniform generation of random graphs with prescribed degree sequences*, eprint arXiv:cond-mat/0312028.

[5] Csardi, G. and Nepusz, T (2006) *The igraph software package for complex network research*, InterJournal, Complex Systems `http://igraph.sf.net`.

[6] Ciriello, G. and Cerami, E. and Sander, C. and Schultz, N.(2012) Mutual exclusivity analysis identifies oncogenic network modules, *Genome Research*, **22**, 398-406.

[7] Miklòs I, Podani J. Randomization of presence-absence matrices: comments and new algorithms. Ecology. Eco Soc America; 2004;85(1):86–92.

[8] Gotelli NJ. Null model analysis of species co-occurrence patterns. Ecology. 2000.

[9] Jaccard, Paul. Étude comparative de la distribution florale dans une portion des Alpes et des Jura. Bulletin del la Société Vaudoise des Sciences Naturelles; 1901; 37:547–579.