Package ‘incidentally’

April 8, 2022

Title Generates Incidence Matrices and Bipartite Graphs

Version 0.9.0

Description Functions to generate incidence matrices and bipartite graphs that have (1) a fixed fill rate, (2) given marginal sums, (3) marginal sums that follow given distributions, or (4) are generated by a social process mirroring team, group, or organization formation.

License GPL-3

Encoding UTF-8

RoxygenNote 7.1.2

Depends R (>= 2.10)

Imports igraph, methods, stats

Suggests rmarkdown, knitr

VignetteBuilder knitr

URL `https://www.zacharyneal.com/backbone`,
`https://github.com/zpneal/incidentally`

BugReports `https://github.com/zpneal/incidentally/issues`

NeedsCompilation no

Author Zachary Neal [aut, cre] (<`https://orcid.org/0000-0003-3076-4995`>)

Maintainer Zachary Neal <`zpneal@msu.edu`>

Repository CRAN

Date/Publication 2022-04-08 08:22:30 UTC

R topics documented:

add.blocks *Adds a block structure to an incidence matrix*

Description

add.blocks shuffles an incidence matrix to have a block structure or planted partition while preserving the row and column sums

Usage

```
add.blocks(I, blocks = 2, density = 0.5, max.tries = 1e+05)
```

Arguments

I	matrix: An incidence matrix
blocks	integer: number of blocks to add (between 2 and 26)
density	numeric: desired within-block density
max.tries	numeric: number of ineligible re-wiring attempts before giving up

Details

Stochastic block and planted partition models generate graphs in which the probability that two nodes are connected depends on whether they are members of the same or different blocks/partitions. Functions such as sample_sbm can randomly sample from stochastic block models with given probabilities. In contrast add.blocks attempts to generate a block model that preserves the degree sequences (i.e., a matrix with preserved row and column sums).

Each row and each column node are randomly assigned to one of blocks number of groups. Then checkerboard swaps are performed that increase the within-block density, until density is achieved. Eligible swaps are identified randomly, so the re-wiring can be slow when I is large. The process can get stuck when no eligible swaps remain but the target density has not been achieved; if this happens, increase max.tries to keep looking for eligible swaps or reduce the target density.

Value

matrix: An incidence matrix, row and column names begin with a letter indicating their block membership

Examples

```
I <- incidence.from.probability(R = 20, C = 20, P = .5)
I <- add.blocks(I, blocks = 2, density = .7)
```

`curveball` *Randomize a binary matrix using the curveball algorithm*

Description

`curveball` randomizes a binary matrix, preserving the row and column sums

Usage

```
curveball(M, trades = 5 * nrow(M))
```

Arguments

`M`	a binary matrix
`trades`	integer: number of trades; the default is 5 * nrow(M) (approx. mixing time)

Details

Strona et al. (2014) provided an initial implementation of the Curveball algorithm in R. `curveball()` is a modified R implementation that is slightly more efficient. For an even more efficient algorithm, see `backbone::fastball()`.

Value

A random binary matrix with same row sums and column sums as M

References

Strona, Giovanni, Domenico Nappo, Francesco Boccacci, Simone Fattorini, and Jesus San-Miguel-Ayanz. 2014. A Fast and Unbiased Procedure to Randomize Ecological Binary Matrices with Fixed Row and Column Totals. *Nature Communications, 5*, 4114. doi: 10.1038/ncomms5114

Godard, Karl and Neal, Zachary P. 2022. fastball: A fast algorithm to sample bipartite graphs with fixed degree sequences. *arXiv:2112.04017*

Examples

```
M <- incidence.from.probability(5,5,.5)  #A matrix
Mrand <- curveball(M)  #Random matrix with same row/col sums
all.equal(rowSums(M), rowSums(curveball(M)))
all.equal(colSums(M), colSums(curveball(M)))
```

incidence.from.adjacency

Generates an incidence matrix from an adjacency matrix

Description

`incidence.from.adjacency` generates an incidence matrix from an adjacency matrix or network using a given generative model

Usage

```
incidence.from.adjacency(G, k = 1, p = 1, d = 2, model = "team", class = NULL)
```

Arguments

`G`	A symmetric, binary adjacency matrix of class `matrix` or `Matrix`, a `data.frame` containing a symbolic edge list in the first two columns, or an undirected, unweighted unipartite graph of class igraph.
`k`	integer: Number of artifacts to generate
`p`	numeric: Tuning parameter for artifacts, 0 <= p <= 1
`d`	numeric: Number of dimensions in Blau space, d >= 2
`model`	string: Generative model, one of c("team", "group", "blau") (see details)
`class`	string: Return object as `matrix`, `igraph`, or `edgelist`. If `NULL`, object is returned in the same class as `G`.

Details

Given a unipartite network composed of *i agents* (i.e. nodes) that can be represented by an *i x i* adjacency matrix, `incidence.from.adjacency` generates a random *i x k* incidence matrix that indicates whether agent *i* is associated with *artifact k*. Generative models differ in how they conceptualize artifacts and how they associate agents with these artifacts.

The **Team Model** (`model == "team"`) mirrors a team formation process, where each artifact represents a new team formed from the incumbants of a prior team (with probability `p`) and newcomers (with probability 1-`p`).

The **Group Model** (`model == "group"`) mirrors a social group formation process, where each artifact represents a social group. Group members attempt to recruit non-member friends, who join the group if it would have a density of at least `p`.

The **Blau Space Model** (`model == "blau"`) mirrors an organization (the artifact) recruiting members from social space, where those within the organization's niche join with probability `p`, and those outside the niche join with probability 1-`p`.

Value

An incidence matrix of class `matrix`, or a bipartite graph as an edgelist of igraph object.

Examples

```
G <- igraph::erdos.renyi.game(10, .4)
I <- incidence.from.adjacency(G, k = 1000, p = .95,
                              model = "team")
```

---

incidence.from.distribution

Generates an incidence matrix with row and column sums that follow given distributions

---

Description

`incidence.from.distribution` generates a random incidence matrix with row and column sums that approximately follow beta distributions with given parameters.

Usage

```
incidence.from.distribution(
  R,
  C,
  P,
  rowdist = c(1, 1),
  coldist = c(1, 1),
  class = "matrix"
)
```

Arguments

R	integer: number of rows
C	integer: number of columns
P	numeric: probability that a cell contains a 1
rowdist	vector length 2: Row marginals will approximately follow a Beta(a,b) distribution
coldist	vector length 2: Column marginals will approximately follow a Beta(a,b) distribution
class	string: the class of the returned backbone graph, one of c("matrix", "igraph")

Value

An incidence matrix of class `matrix` or a bipartite graph of class igraph.

Examples

```
I <- incidence.from.distribution(R = 100, C = 100, P = 0.1,
  rowdist = c(1,1), coldist = c(1,1))  #Uniform
I <- incidence.from.distribution(R = 100, C = 100, P = 0.1,
  rowdist = c(1,10), coldist = c(1,10))  #Right-tailed
I <- incidence.from.distribution(R = 100, C = 100, P = 0.1,
  rowdist = c(10,1), coldist = c(10,1))  #Left-tailed
I <- incidence.from.distribution(R = 100, C = 100, P = 0.1,
  rowdist = c(10,10), coldist = c(10,10))  #Normal
I <- incidence.from.distribution(R = 100, C = 100, P = 0.1,
  rowdist = c(10000,10000), coldist = c(10000,10000))  #Constant
```

`incidence.from.probability` *Generates an incidence matrix with a given cell-filling probability*

Description

`incidence.from.probability` generates a random incidence matrix in which each cell is filled with a 1 with a given probability.

Usage

```
incidence.from.probability(R, C, P = 0, constrain = TRUE, class = "matrix")
```

Arguments

`R` integer: number of rows

`C` integer: number of columns

`P` numeric: probability that a cell contains a 1; if P = 0 a probability will be chosen randomly

`constrain` boolean: ensure that no rows or columns sum to 0 (i.e., contain all 0s) or to 1 (i.e., contain all 1s)

`class` string: the class of the returned backbone graph, one of c("matrix", igraph").

Value

An incidence matrix of class `matrix` or a bipartite graph of class igraph.

Examples

```
I <- incidence.from.probability(R = 10, C = 10)
I <- incidence.from.probability(R = 10, C = 10, P = .5)
I <- incidence.from.probability(R = 10, C = 10, P = .5, class = "igraph")
```

`incidence.from.vector` *Generates an incidence matrix with given row and column marginal sums*

Description

`incidence.from.vector` generates a random incidence matrix with given row and column sums

Usage

```
incidence.from.vector(R, C, class = "matrix")
```

Arguments

`R`	numeric vector: row marginal sums
`C`	numeric vector: column marginal sums
`class`	string: the class of the returned backbone graph, one of c("matrix", "igraph")

Value

An incidence matrix of class `matrix` or a bipartite graph of class igraph.

Examples

```
I <- incidence.from.vector(R = c(1,1,2), C = c(1,1,2))
I <- incidence.from.vector(R = c(1,1,2), C = c(1,1,2), class = "igraph")
```

`incidentally` *incidentally: Generates incidence matrices and bipartite graphs*

Description

Functions to generate incidence matrices and bipartite graphs that have (1) a fixed fill rate, (2) given marginal sums, (3) marginal sums that follow given distributions, or (4) are generated by a social process mirroring team, group, or organization formation.

Incidence matrices can be generated:

- ...with a fixed fill rate: `incidence.from.probability()`.
- ...with given marginals: `incidence.from.vector()`.
- ...with marginals that follow given distributions: `incidence.from.distribution()`.
- ...from a network, by a social process mirroring team, group, or organization formation `incidence.from.adjacency()`
- ...with a block structure or planted partition: `add.blocks()`.

Index