Package ‘backbone’

August 27, 2020

Type Package

Title Extracts the Backbone from Weighted Graphs

Version 1.2.2


License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.1.0

Depends R (>= 2.10)

Imports Matrix, methods, stats, utils, CVXR, igraph, network, Rcpp

Suggests knitr, rmarkdown, speedglm, testthat

VignetteBuilder knitr

URL https://github.com/domagal9/backbone,

https://www.zacharyneal.com/backbone

BugReports https://github.com/domagal9/backbone/issues

LinkingTo Rcpp

NeedsCompilation yes

Author Rachel Domagalski [aut, cre],
Zachary Neal [aut],
Bruce Sagan [aut]

Maintainer Rachel Domagalski <domagal9@msu.edu>
Description

Provides methods for extracting from a weighted graph a binary or signed backbone that retains only the significant edges. The user may input a weighted graph, or a bipartite graph from which a weighted graph is first constructed via projection. Backbone extraction methods include:

- as well as a universal threshold method.

Details

Some features of the package are:

- 'universal': returns a backbone graph in which edge weights are set to 1 if above the given upper parameter threshold, and set to -1 if below the given lower parameter threshold, and are 0 otherwise.
- 'sdsm': computes the probability of edge weights being above or below the observed edge weights in a bipartite projection using the stochastic degree sequence model. Once computed, use `backbone.extract` to return the backbone matrix for a given alpha value.
• 'hyperg': computes the probability of edge weights being above or below the observed edge weights in a bipartite projection using the hypergeometric model. Once computed, use `backbone.extract` to return the backbone matrix for a given alpha value.
• 'fdsm': computes the proportion of edge weights above or below the observed edge weights in a bipartite projection using the fixed degree sequence model. Once computed, use `backbone.extract` to return the backbone matrix for a given alpha value.
• 'backbone.extract': returns a backbone graph object that retains only the significant edges.

Additional functions that aid in the use of the above models are exported:

• 'polytope': finds a matrix that maximizes the entropy function, used in `sdsm`.
• 'curveball': generates a random 0/1 matrix with the same row and column sums as the input, used in `sdsm` and `fdsm`.

For additional documentation and background on the package functions, see `vignette("backbone", package = "backbone")`.

---

**backbone.extract**

Extracts the backbone of a weighted network using results from a null model

---

**Description**

'backbone.extract' returns a binary or signed adjacency matrix containing the backbone that retains only the significant edges.

**Usage**

```r
backbone.extract(
  backbone,  
signed = TRUE,  
alpha = 0.05,  
fwer = "none",  
class = "original",  
narrative = FALSE
)
```

**Arguments**

- **backbone**: backbone S3 class object, as returned by `sdsm`, `fdsm`, or `hyperg`.
- **signed**: Boolean: TRUE if signed backbone is to be returned, FALSE if binary backbone is to be returned.
- **alpha**: Real: Precision of significance test.
- **fwer**: string: type of familywise error rate correction to be applied; c("none","bonferroni",holm"). If "holm", Holm Bonferroni Family-wise Error Rate test is used, if "bonferroni", Bonferroni Family-wise Error Rate test should be used. By default, the given 'alpha' value is used for all tests with no correction for family-wise error rates.
class string: the class of the returned backbone graph, one of c("original", "matrix", "sparseMatrix", "igraph", "network", "edgelist"), converted via class.convert. If "original", the backbone graph returned is of the same class as the data inputted in one of sdsm, fdsm, or hyperg.

narrative Boolean: TRUE if suggested text for a manuscript is to be returned

Details
The "backbone" S3 class object is composed of two matrices, a summary dataframe and (optionally, if generated by using fdsm) a ‘dyad_values’ vector.
The Holm Bonferroni correction was originally a port from python code written by Dr. Samin Aref. The authors thank Dr. Aref greatly for his contribution to this package!

Value
backbone graph: Binary or signed backbone graph of class given in parameter ‘class’.

Examples
```r
probs <- sdsm(davis)
bb <- backbone.extract(probs, alpha = .2, signed = TRUE, fwer = "none")
```

**curveball**

*curveball algorithm*

Description
curveball algorithm

Usage
curveball(M)

Arguments

M matrix

Value
rm, a matrix with same row sums and column sums as M, but randomized 0/1 entries.

References
Examples

curveball(davis)

---

davis | Davis Southern Women Data Set

Description

A two mode matrix of 18 women and attendance of 14 social events.

Usage

data(davis)

Format

An object of class matrix (inherits from array) with 18 rows and 14 columns.

Source

UCI Network Data Repository

References


---

fdsm | The fixed degree sequence model (fdsm) for backbone probabilities

Description

‘fdsm‘ computes the proportion of generated edges above or below the observed value using the fixed degree sequence model. Once computed, use backbone.extract to return the backbone matrix for a given alpha value.

Usage

fdsm(B, trials = 1000, dyad = NULL, progress = FALSE)
Arguments

- **graph**: Bipartite graph object of class matrix, sparse matrix, igraph, edgelist, or network object.
- **trials**: Integer: Number of random bipartite graphs generated
- **dyad**: Vector length 2: two row entries i,j. Saves each value of the i-th row and j-th column in each projected B* matrix. This is useful for visualizing an example of the empirical null edge weight distribution generated by the model. These correspond to the row and column indices of a cell in the projected matrix, and can be written as their string row names or as numeric values.
- **progress**: Boolean: If `txtProgressBar` should be used to measure progress

Details

During each iteration, fdsm computes a new B* matrix using the curveball algorithm. This is a random bipartite matrix with the same row and column sums as the original matrix B. If a value is supplied for the dyad parameter, when the B* matrix is projected (multiplied by its transpose), the value in the corresponding row and column will be saved. This allows the user to see the distribution of the edge weights for desired row and column.

The "backbone" S3 class object returned is composed of two matrices, a summary dataframe and (optionally, if generated by using `fdsm`) a 'dyad_values' vector.

Value

backbone, a list(positive, negative, dyad_values, summary). Here 'positive' is a matrix of proportion of times each entry of the projected matrix B is above the corresponding entry in the generated projection, 'negative' is a matrix of proportion of times each entry of the projected matrix B is below the corresponding entry in the generated projection, 'dyad_values' is a list of edge weight for i,j in each generated projection, and 'summary' is a data frame summary of the inputted matrix and the model used including: model name, number of rows, skew of row sums, number of columns, skew of column sums, and running time.

References


Examples

```r
fdsm_props <- fdsm(davis, trials = 100, dyad=c(3,6))
```
`hyperg`  
*Compute hypergeometric backbone probabilities*

**Description**

`hyperg` computes the probability of observing a higher or lower edge weight using the hypergeometric distribution. Once computed, use `backbone.extract` to return the backbone matrix for a given alpha value.

**Usage**

`hyperg(B)`

**Arguments**

- `B`  
  graph: Bipartite graph object of class matrix, sparse matrix, igraph, edgelist, or network object.

**Details**

Specifically, this function compares an edge’s observed weight in the projection $B * t(B)$ to the distribution of weights expected in a projection obtained from a random bipartite graph where the row vertex degrees are fixed but the column vertex degrees are allowed to vary.

The "backbone" S3 class object returned is composed of two matrices, a summary dataframe and (optionally, if generated by using `fdsm`) a 'dyad_values' vector.

**Value**

backbone, a list(positive, negative, summary). Here 'positive' is a matrix of probabilities of edge weights being equal to or above the observed value in the projection, 'negative' is a matrix of probabilities of edge weights being equal to or below the observed value in the projection, and 'summary' is a data frame summary of the inputted matrix and the model used including: model name, number of rows, skew of row sums, number of columns, skew of column sums, and running time.

**References**


**Examples**

`hyperg.probs <- hyperg(davis)`
**polytope**  
*Polytope method for finding a matrix that maximizes entropy function*

**Description**
Polytope method for finding a matrix that maximizes entropy function

**Usage**
polytope(G)

**Arguments**
G  
matrix, an adjacency matrix representing a graph

**Details**
Uses convex optimization via the **CVXR-package** to find a matrix $M$ that maximizes the entropy function where $M$ satisfies the following constraints: (1) the values of $M$ are between 0 & 1, (2) the row sums of the matrix are equal to the row sums of the original matrix, (3) the column sums of the matrix are equal to the column sums of the original matrix.

This method is utilized in the function **sdsm** to compute probabilities of an edge existing in a graph. Method is called polytope as it is optimizing over the convex hull of the set of matrices (thought of as vectors) with the same row and column sums as the input.

**Value**
matrix containing optimal solution to entropy function under constraints

**Examples**
polytope(davis)

**sdsm**  
The stochastic degree sequence model (sdsm) for backbone probabilities

**Description**
`sdsm` computes the probability of edge weights being above or below the observed edge weights in a bipartite projection using the stochastic degree sequence model. Once computed, use **backbone.extract** to return the backbone matrix for a given alpha value.

**Usage**
sdsm(B, model = "polytope", trials = 1000)
Arguments

- **graph**: Bipartite graph object of class matrix, sparse matrix, igraph, edgelist, or network object.
- **trials**: Integer: If 'model' = 'curveball', number of random bipartite graphs generated using curveball to compute probabilities. Default is 1000.

Details

Specifically, the `sdsm` function compares an edge’s observed weight in the projection $B \times t(B)$ to the distribution of weights expected in a projection obtained from a random bipartite network where both the row vertex degrees and column vertex degrees are approximately fixed.

If the 'model' parameter is one of c('logit', 'probit', 'cauchit', 'log', 'cloglog','scobit'), then this model is used as a 'link' function for a binary outcome model conditioned on the row degrees and column degrees, as described by `glm` and `family`. If the 'model' parameter is 'oldlogit', then a logit link function is used but the model is conditioned on the row degrees, column degrees, and their product. If 'model = lpm', a linear probability model is used. If 'model = rcn', the probabilities are given by (row degree * column degree)/(total number of edges).

If 'model' = 'curveball' and 'trials' > 0, the probabilities are computed by using curveball function 'trials' times. The proportion of each cell being 1 is used as its probability. If 'model = polytope', the `polytope` function is used to find a matrix of probabilities that maximizes the entropy function, with same row and column sums.

The "backbone" S3 class object returned is composed of two matrices, a summary dataframe and (optionally, if generated by using `fdsm`) a 'dyad_values' vector.

Value

- **backbone**, a list(positive, negative, summary). Here ‘positive’ is a matrix of probabilities of edge weights being equal to or above the observed value in the projection, ‘negative’ is a matrix of probabilities of edge weights being equal to or below the observed value in the projection, and ‘summary’ is a data frame summary of the inputted matrix and the model used including: model name, number of rows, skew of row sums, number of columns, skew of column sums, and running time.

References


Examples

```r
sdsm_probs <- sdsm(davis)
## Not run: sdsm_probs2 <- sdsm(davis, model = "curveball", trials = 1000)
```
universal

Compute universal threshold backbone

Description

‘universal’ returns a backbone graph in which edge weights are set to 1 if above the given upper parameter threshold, set to -1 if below the given lower parameter threshold, and are 0 otherwise.

Usage

universal(M, upper = NULL, lower = NULL, bipartite = NULL, narrative = FALSE)

Arguments

M
  graph: Bipartite graph object of class matrix, sparse matrix, igraph, edgelist, or network object.

upper
  Real, FUN, or NULL: upper threshold value or function to be applied to the edge weights. Default is NULL.

lower
  Real, FUN, or NULL: lower threshold value or function to be applied to the edge weights. Default is NULL.

bipartite
  Boolean: TRUE if bipartite matrix, FALSE if weighted matrix. Default is NULL.

narrative
  Boolean: TRUE if suggested text for a manuscript is to be returned

Details

If both ‘upper’ and ‘lower’ are ‘NULL’, a weighted projection is returned.

If ‘bipartite’ is ‘NULL’, the function tries to guess at whether the data is bipartite or unipartite based on its shape.

Value

backbone, a list(backbone, summary). The ‘backbone’ object is a graph object of the same class as M. The ‘summary’ contains a data frame summary of the inputted matrix and the model used including: model name, number of rows, skew of row sums, number of columns, skew of column sums, and running time.

Examples

test <- universal(davis%*%t(davis), upper = function(x)mean(x)+sd(x), lower=function(x)mean(x))
test2 <- universal(davis, upper = function(x)mean(x)+2*sd(x), lower = 2, bipartite = TRUE)
test3 <- universal(davis, upper = 4, lower = 2, bipartite = TRUE)
Index

* datasets
  davis, 5

backbone, 2
backbone.extract, 2, 3, 3, 5, 7, 8

class.convert, 4
curveball, 3, 4, 6, 9
CVXR-package, 8

davis, 5

family, 9
fdsm, 3, 4, 5, 6, 7, 9

glm, 9

hyperg, 3, 4, 7

polytope, 3, 8, 9

sdsm, 2–4, 8, 8

txtProgressBar, 6

universal, 2, 10