# Package: levelnet (via r-universe)

June 24, 2023

**Type** Package  
**Title** Methods to Analyze One-mode Projections of Two-mode Networks  
**Version** 0.5.0  
**Maintainer** David Schoch <david@schochastics.net>  
**Description** Methods to analyze one-mode projections of two-mode networks. Focus lies on methods to determine significant edges.

**URL** [https://github.com/schochastics/levelnet](https://github.com/schochastics/levelnet)  
**BugReports** [https://github.com/schochastics/levelnet/issues](https://github.com/schochastics/levelnet/issues)  
**Depends** R (>= 3.0.1)  
**License** MIT + file LICENSE  
**Encoding** UTF-8  
**LazyData** true  
**RoxygenNote** 7.1.1  
**Imports** igraph, Matrix, Rcpp  
**LinkingTo** Rcpp  
**Repository** [https://schochastics.r-universe.dev](https://schochastics.r-universe.dev)  
**RemoteUrl** [https://github.com/schochastics/levelnet](https://github.com/schochastics/levelnet)  
**RemoteRef** HEAD  
**RemoteSha** 775cf5e91b83cb73fe35e378ed1d7facb1d741eb

### R topics documented:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>bipartite_from_data_frame</td>
<td>2</td>
</tr>
<tr>
<td>cosponsor_senate_15</td>
<td>3</td>
</tr>
<tr>
<td>disparity_filter</td>
<td>3</td>
</tr>
<tr>
<td>fiedler_order</td>
<td>4</td>
</tr>
<tr>
<td>graph_indifference</td>
<td>4</td>
</tr>
</tbody>
</table>
bipartite_from_data_frame

Create a two-mode network from a data.frame

Usage

bipartite_from_data_frame(df, type1, type2)

Arguments

df data.frame
type1 column name of mode 1
type2 column name of mode 2

Value
two mode network as igraph object

Author(s)
David Schoch
**Description**

Bill cosponsorship data for the 115th Senate

**Usage**

```r
cosponsor_senate_15
```

**Format**

a data frame of bill cosponsorships

**References**

govtrack.us

---

**disparsity_filter**

*Disparity Filter*

**Description**

Extract significant edges of a weighted network or one-mode projection with the disparsity filter.

**Usage**

```r
disparsity_filter(g, proj = "true", alpha = 0.05, cut_mode = "or")
```

**Arguments**

- `g`: igraph object. either two-mode or weighted network
- `proj`: string. Which mode to project on ("true"/"false")
- `alpha`: significant level
- `cut_mode`: ‘and’ (retain edge if both directions are significant) or ’or’ (retain edge if one direction is significant)

**Value**

backbone of weighted network as igraph object

**Author(s)**

David Schoch
References
Serrano et al. (2009). Extracting the multiscale backbone of complex weighted networks

---

### fiedler_order

*Permutation induced by Fiedler vector*

**Description**
Returns the permutation induced by sorting the Fiedler vector of the Laplacian matrix of a graph

**Usage**

```r
fiedler_order(g, mode = "cols")
```

**Arguments**

- `g` an igraph object or a (0,1)-matrix
- `mode` one of "mcl" (clique vertex matrix), "cols" (Lazarus count of columns) "rows" (Lazarus count of rows) or "sym" (Lazarus count of both columns and rows).

**Value**
numeric vector

**Author(s)**
David Schoch

---

### graph_indifference

*Random Indifference Graph*

**Description**
Create a random indifference graph. An indifference graph is an interval graph where intervals have length 1.

**Usage**

```r
graph_indifference(n, r = 2)
```

**Arguments**

- `n` number of nodes
- `r` radius
Details
‘n’ points (x) are sampled uniformly at random between 0 and ‘r’. The interval is then given by (x,x+1).

Value
indifference graph as igraph object and interval representation (a,b)

Author(s)
David Schoch

See Also
[graph_interval,graph_tolerance]

Examples

graph_indifference(n = 10)

description
Create a random interval graph. In an interval graph, each node is characterized by an interval on the real line. Two nodes are connected, if their intervals overlap.

Usage

graph_interval(n, r = 2, sd = 0.5)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
<td>number of nodes</td>
</tr>
<tr>
<td>r</td>
<td>radius (see details)</td>
</tr>
<tr>
<td>sd</td>
<td>standard deviation (see details)</td>
</tr>
</tbody>
</table>

Details
Interval graphs are created as follows. First, n random points x are created uniformly at random between 0 and ‘r’. For each point, a value Y is created from a normal distribution with mean X and standard deviation is ‘sd’. In this way, it is possible to control the density of the network. The larger ‘r’ and the larger ‘sd’ the more likely do intervals overlap.

Value
interval graph as igraph object and interval representation as node attribute (a,b)
**Author(s)**

David Schoch

**See Also**

[graph_indifference, graph_tolerance]

**Examples**

```r
graph_interval(n = 10)
```

---

**Description**

generate random roll-call votes based on ideology space

**Usage**

```r
graph_random_vote(
  M = 101,
  D = 1,
  p = 4,
  pd = 2,
  beta = 1,
  r = 9,
  noprob = 0.05,
  Nrand = 1000,
  N = 525
)
```

**Arguments**

- `M` number of members
- `D` distance between means
- `p` dimension of space
- `pd` dimensions where distributions are separated
- `beta` scaling parameter for probabilistic voting
- `r` radius of hypersphere for random generation
- `noprob` probability of non voting
- `Nrand` number of randomly generated votes
- `N` number of votes to sample from randomly generated votes
graph_rectangle

Value

list with random votes and ideologies

Author(s)

David Schoch

References


Description

Create a random graph with boxicity 2.

Usage

graph_rectangle(n, r = 2, sd = 0.5)

Arguments

\begin{itemize}
\item \textbf{n} \hspace{1cm} \text{number of nodes}
\item \textbf{r} \hspace{1cm} \text{radius}
\item \textbf{sd} \hspace{1cm} \text{standard deviation}
\end{itemize}

Value

Boxicity 2 graph as igraph object

Author(s)

David Schoch
**Description**

Create a random tolerance graph. A tolerance graph is an interval graph, where nodes are only connected if the overlap is larger than a node's tolerance level. These graphs are directed.

**Usage**

```r
graph_tolerance(n, r = 2, sd = 0.5, tol = 0.5)
```

**Arguments**

- `n`: number of nodes
- `r`: radius (see details)
- `sd`: standard deviation (see details)
- `tol`: tolerance

**Details**

Tolerance graphs are created as follows. First, `n` random points `x` are created uniformly at random between 0 and `r`. For each point, a value `Y` is created from a normal distribution with mean `X` and standard deviation is `sd`. In this way, it is possible to control the density of the network. The larger `r` and the larger `sd` the more likely do intervals overlap. When overlaps are calculated, it is checked whether the overlap is larger than the tolerance of the node. If so, the edge is included.

**Value**

tolerance graph as igraph object and interval representation and tolerance as node attributes

**Author(s)**

David Schoch

**See Also**

[graph_interval, graph_indifference]

**Examples**

```r
graph_tolerance(n = 10)
```
Description
small helper functions

Usage
clique_vertex_mat(g)
is_bipartite1(g)

Arguments
g         igraph object.

Value     igraph object

Author(s)
David Schoch

is_interval  Check whether graph is interval graph

Description
Check whether graph is interval graph.

Usage
is_interval(g)

Arguments
g         igraph object

Details
This function is not very efficient since it relies on the clique vertex matrix. More efficient linear time algorithms will be implemented in the future.
lazarus_count

Value

Logical scalar, whether graph is an interval graph

Author(s)

David Schoch

laplacian_vectors

Laplacian Vectors

Description

Returns Laplacian eigenvectors associated with the k smallest positive eigenvalues

Usage

laplacian_vectors(g, k = 2)

Arguments

g       igraph object
k       number of vectors to return

Value

data.frame of vectors

Author(s)

David Schoch

lazarus_count

Lazarus Count

Description

Calculates the Lazarus count of a matrix/graph.

Usage

lazarus_count(g, perm = NULL, mode = "cols")
Arguments

- **g**: either an igraph object or a (0,1)-matrix
- **perm**: permutation or NULL
- **mode**: one of "mcl" (clique vertex matrix), "cols" (Lazarus count of columns) "rows" (Lazarus count of rows) or "sym" (Lazarus count of both columns and rows).

Details

The Lazarus count of a matrix is the number of "holes" in each column. A hole is a number of zero entries surrounded by ones. For an interval graph, this count is zero for the `clique_vertex_mat`. If `perm` is NULL, a row permutation based on the Fiedler vector of the Laplacian is calculated.

Value

Lazarus count of g

Author(s)

David Schoch

Examples

```r
set.seed(10)
#the lazarus count of an interval graph is zero
g <- graph_interval(n = 10, r = 1)
lazarus_count(g, mode = "mcl")
```

---

**multiLexBFS**  
*Multisweep Lex-BFS*

Description

Multisweep lexicographical BFS

Usage

`multiLexBFS(g, k = 4)`

Arguments

- **g**: igraph object
- **k**: number of sweeps

Details

LexBFS is used to recognize interval graphs. Not fully implemented yet.
Value

permutation

Author(s)

David Schoch

Description

Create a box representation from permutations

Usage

perm2box(g, perm, dim)

Arguments

g | igraph object.
perm | integer vector of length n times dim
dim | integer. dimensionality of boxes

Value

coordinates

References

Flexible Stochastic Degree Sequence Model

**Usage**

```r
fsdsm(
  g,
  row_constr,
  proj = "true",
  model = "logit",
  max_iter = 1000,
  alpha = 0.05,
  params = list(b0 = 0.1, b1 = 5e-05, b2 = 5e-05, b3 = 5e-05, a = 0.01),
  verbose = FALSE
)
```

```r
sdsm_prob(
  g,
  proj = "true",
  model = "logit",
  max_iter = 1000,
  params = list(b0 = 0.1, b1 = 5e-05, b2 = 5e-05, b3 = 5e-05, a = 0.01),
  verbose = FALSE
)
```

**Arguments**

- `g`: igraph object. The two-mode network
- `row_constr`: constraint matrix
- `proj`: string. Which mode to project on ("true"/"false")
- `model`: string. which link to be used ("logit","probit","cloglog" or "scobit")
- `max_iter`: number of randomly sampled networks
- `alpha`: significance level
- `params`: named parameter list for scobit model
- `verbose`: print status during execution

**Details**

A flexible implementation of the stochastic degree sequence model, allowing for the addition of constraints (use `sdsm` from the backbone package for the regular model)
**Value**

backbone of one-mode projection

**Author(s)**

David Schoch

**References**

Neal, Zachary (2014). The backbone of bipartite projections: Inferring relationships from co-authorship, co-sponsorship, co-attendance and other co-behaviors

---

**sdsm_diagnostic**  
**sdsm model diagnostics**

**Description**

check which binary outcome model fits the data best

**Usage**

```r
sdsm_diagnostic(
  g,  
  proj = "true",  
  iter = 10,  
  verbose = FALSE,  
  params = list(b0 = 0.1, b1 = 5e-05, b2 = 5e-05, b3 = 5e-05, a = 0.01)
)
```

**Arguments**

- **g**  
  igraph object. The two-mode network
- **proj**  
  string. Which mode to project on
- **iter**  
  number of fits per model
- **verbose**  
  logical. print additional information (default: FALSE)
- **params**  
  named parameter list for scobit model

**Value**

rmse and runtime of various models

**Author(s)**

David Schoch
superbox_graph

Supergraph with given boxicity

Description

Create a supergraph with given boxicity using simulated annealing (SA)

Usage

```r
superbox_graph(
  l,
  dim = 1,
  perm = NULL,
  iter = 15000,
  temp = 10,
  tmax = 5,
  verbose = FALSE
)
```

Arguments

- `dim`: integer. target boxicity
- `perm`: starting permutation for SA. If NULL, a random permutation is created
- `iter`: integer. number of iterations for SA
- `temp`: integer. starting temperature for SA
- `tmax`: integer. number of function evaluations at each temperature for SA
- `verbose`: logical. print report during SA (defaults to FALSE)
- `g`: igraph object

Value

A list with entries

- `perm`: permutation vector. All permutations are concatenated to one long vector
- `ged`: graph edit distance from original graph
- `A`: adjacency matrix of supergraph with given boxicity

References

Index

* datasets
  cosponsor_senate_15, 3
bipartite_from_data_frame, 2
clique_vertex_mat (helpers), 9
cosponsor_senate_15, 3
dispersity_filter, 3
fiedler_order, 4
fsdm (sdsm), 13
graph_indifference, 4
graph_interval, 5
graph_random_vote, 6
graph_rectangle, 7
graph_tolerance, 8
helpers, 9
is_bipartite (helpers), 9
is_interval, 9
laplacian_vectors, 10
lazarus_count, 10
multiLexBFS, 11
perm2box, 12
sdsm, 13
sdsm_diagnostic, 14
sdsm_prob (sdsm), 13
superbox_graph, 15