

# Package ‘signnet’

January 8, 2024

**Title** Methods to Analyse Signed Networks

**Version** 1.0.4

**Description** Methods for the analysis of signed networks. This includes several measures for structural balance as introduced by Cartwright and Harary (1956) <[doi:10.1037/h0046049](https://doi.org/10.1037/h0046049)>, blockmodeling algorithms from Doreian (2008) <[doi:10.1016/j.socnet.2008.03.005](https://doi.org/10.1016/j.socnet.2008.03.005)>, various centrality indices, and projections of signed two-mode networks introduced by Schoch (2020) <[doi:10.1080/0022250X.2019.1711376](https://doi.org/10.1080/0022250X.2019.1711376)>.

**License** MIT + file LICENSE

**URL** <https://github.com/schochastics/signnet>,  
<https://schochastics.github.io/signnet/>

**BugReports** <https://github.com/schochastics/signnet/issues>

**Depends** R (>= 3.2.0)

**Imports** igraph, Matrix, Rcpp

**Suggests** covr, ggplot2, ggraph, knitr, ompr, ompr.roi, rmarkdown, ROI, ROI.plugin.glpk, testthat (>= 2.1.0)

**LinkingTo** Rcpp, RcppArmadillo

**VignetteBuilder** knitr

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.2.3

**NeedsCompilation** yes

**Author** David Schoch [aut, cre] (<<https://orcid.org/0000-0003-2952-4812>>)

**Maintainer** David Schoch <david@schochastics.net>

**Repository** CRAN

**Date/Publication** 2024-01-08 22:00:02 UTC

**R topics documented:**

as_adj_complex	2
as_adj_signed	3
as_complex_edges	4
as_incidence_complex	4
as_incidence_signed	5
as_signed_proj	6
as_unsigned_2mode	7
avatar	8
balance_score	8
complex_walks	9
count_complex_triangles	10
count_signed_triangles	11
cowList	11
degree_signed	12
eigen_centrality_signed	13
frustration_exact	14
ggblock	15
ggsigned	16
graph_circular_signed	16
graph_from_adjacency_matrix_signed	17
graph_from_edgelist_signed	18
is_signed	18
laplacian_matrix_complex	19
laplacian_matrix_signed	20
pn_index	21
sample_bipartite_signed	22
sample_gnp_signed	23
sample_islands_signed	23
signed_blockmodel	24
signed_blockmodel_general	25
signed_triangles	26
triad_census_signed	27
tribes	28
<b>Index</b>	<b>29</b>

---

as_adj_complex	<i>Convert a signed graph to a complex adjacency matrix</i>
----------------	---

---

**Description**

This function returns the adjacency matrix for a signed graph that contains ambivalent ties

**Usage**

```
as_adj_complex(g, attr)
```

**Arguments**

g	igraph object
attr	edge attribute name that encodes positive ("P"), negative ("N") and ambivalent ("A") ties.

**Value**

complex adjacency matrix

**See Also**

[as\\_adj\\_signed](#)

---

as_adj_signed	<i>Convert a signed graph to a signed adjacency matrix</i>
---------------	--

---

**Description**

This function returns the adjacency matrix for a signed graph

**Usage**

```
as_adj_signed(g, sparse = FALSE)
```

**Arguments**

g	igraph object. Must have a "sign" edge attribute.
sparse	Logical scalar, whether to return the result as a sparse matrix. The Matrix package is required for sparse matrices.

**Value**

signed adjacency matrix

**See Also**

[as\\_adj\\_complex](#)

---

as\_complex\_edges      *Convert Signed Network to Complex*

---

**Description**

Convert Signed Network to Complex

**Usage**

```
as_complex_edges(g, attr = "type")
```

**Arguments**

**g**                    igraph object. Must have a "sign" edge attribute.  
**attr**                new edge attribute name that encodes positive ("P"), negative ("N") and ambivalent ("A") ties.

**Value**

igraph object

**Author(s)**

David Schoch

**Examples**

```
g <- sample_islands_signed(2, 10, 1, 10)
as_complex_edges(g)
```

---

as\_incidence\_complex      *Complex Incidence Matrix*

---

**Description**

The complex incidence matrix of a signed graph containing ambivalent ties.

**Usage**

```
as_incidence_complex(g, attr)
```

**Arguments**

**g**                    igraph object.  
**attr**                edge attribute name that encodes positive ("P"), negative ("N") and ambivalent ("A") ties.

**Details**

This function is slightly different than [as\\_incidence\\_matrix](#) since it is defined for bipartite graphs. The incidence matrix here is defined as a  $S \in \mathbb{C}^{n,m}$ , where  $n$  is the number of vertices and  $m$  the number of edges. Edges  $(i,j)$  are oriented such that  $i < j$  and entries are defined as

$$S_{i(i,j)} = \sqrt{A_{ij}}$$

$$S_{j(i,j)} = -\sqrt{A_{ji}} \text{ if } (i,j) \text{ is an ambivalent tie}$$

$$S_{j(i,j)} = -A_{ji} \sqrt{A_{ji}} \text{ else}$$

**Value**

a complex matrix

**Author(s)**

David Schoch

**See Also**

[laplacian\\_matrix\\_complex](#), [as\\_adj\\_complex](#)

---

as\_incidence\_signed    *Convert a signed two-mode network to a signed matrix*

---

**Description**

This function returns the incidence matrix for a signed two-mode network.

**Usage**

```
as_incidence_signed(g, sparse = FALSE)
```

**Arguments**

g	igraph object (bipartite). Must have a "sign" edge attribute.
sparse	Logical scalar, whether to return the result as a sparse matrix. The Matrix package is required for sparse matrices.

**Value**

signed incidence matrix

---

as\_signed\_proj      *convert unsigned projection to signed*

---

**Description**

convert unsigned projection to signed

**Usage**

```
as_signed_proj(g)
```

**Arguments**

g                    igraph object

**Value**

igraph object

**Author(s)**

David Schoch

**See Also**

[as\\_unsigned\\_2mode](#)

**Examples**

```
library(igraph)

# create a simple signed two mode network
e1 <- matrix(c(1, "a", 1, "b", 1, "c", 2, "a", 2, "b"), ncol = 2, byrow = TRUE)
g <- graph_from_edgelist(e1, directed = FALSE)
E(g)$sign <- c(1, 1, -1, 1, -1)
V(g)$type <- c(FALSE, TRUE, TRUE, TRUE, FALSE)

# convert to unsigned two-mode network and project
l <- as_unsigned_2mode(g, primary = TRUE)
p <- bipartite_projection(l, which = "true")

# turn the unsigned projection back to a signed network
as_signed_proj(p)
```

---

as\_unsigned\_2mode      *convert signed two-mode network to unsigned*

---

### Description

convert signed two-mode network to unsigned

### Usage

```
as_unsigned_2mode(g, primary = TRUE)
```

### Arguments

**g**                    igraph object. Two-mode network, must have a "sign" edge attribute.  
**primary**             logical. Which mode to transform

### Value

igraph object

### Author(s)

David Schoch

### See Also

[as\\_signed\\_proj](#)

### Examples

```
library(igraph)

# create a simple signed two mode network
e1 <- matrix(c(1, "a", 1, "b", 1, "c", 2, "a", 2, "b"), ncol = 2, byrow = TRUE)
g <- graph_from_edgelist(e1, directed = FALSE)
E(g)$sign <- c(1, 1, -1, 1, -1)
V(g)$type <- c(FALSE, TRUE, TRUE, TRUE, FALSE)

# convert to unsigned two-mode network and project
l <- as_unsigned_2mode(g, primary = TRUE)
p <- bipartite_projection(l, which = "true")

# turn the unsigned projection back to a signed network
as_signed_proj(p)
```

---

avatar	<i>Signed networks from Avatar: The Last Airbender</i>
--------	--

---

**Description**

Allies/Enemy relations from Avatar: The Last Airbender

**Usage**

```
avatar
```

**Format**

igraph object

**Source**

scraped from Avatar Wiki (<https://avatar.fandom.com/wiki/Category:Characters>)

---

balance_score	<i>balancedness of signed network</i>
---------------	---------------------------------------

---

**Description**

Implements several indices to assess the balancedness of a network.

**Usage**

```
balance_score(g, method = "triangles")
```

**Arguments**

g	igraph object with a sign edge attribute.
method	string indicating the method to be used. See details for options

**Details**

The method parameter can be one of

**triangles** Fraction of balanced triangles. Maximal (=1) if all triangles are balanced.

**walk**  $\sum \exp(\lambda_i) / \sum \exp(\mu_i)$  where  $\lambda_i$  are the eigenvalues of the signed adjacency matrix and  $\mu_i$  of the unsigned adjacency matrix. Maximal (=1) if all walks are balanced.

**frustration** The frustration index assumes that the network can be partitioned into two groups, where intra group edges are positive and inter group edges are negative. The index is defined as the sum of intra group negative and inter group positive edges. Note that the problem is NP complete and only an upper bound is returned (based on simulated annealing). Exact methods can be found in the work of Aref. The index is normalized such that it is maximal (=1) if the network is balanced.



**Value**

numeric balancedness score between 0 and 1

**Author(s)**

David Schoch

**References**

Estrada, E. (2019). Rethinking structural balance in signed social networks. *Discrete Applied Mathematics*.

Samin Aref, Mark C Wilson (2018). Measuring partial balance in signed networks. *Journal of Complex Networks*, 6(4): 566–595, <https://doi.org/10.1093/comnet/cnx044>

**Examples**

```
library(igraph)
g <- graph.full(4)
E(g)$sign <- c(-1, 1, 1, -1, -1, 1)

balance_score(g, method = "triangles")
balance_score(g, method = "walk")
```

---

complex\_walks

*Count Walks in complex signed network*

---

**Description**

Count Walks in complex signed network

**Usage**

```
complex_walks(g, attr, k)
```

**Arguments**

g	igraph object.
attr	edge attribute that encodes positive ("P"), negative ("N") and ambivalent ("A") ties.
k	integer. length of walks

**Value**

igraph object

**Author(s)**

David Schoch

**Examples**

```
g <- sample_islands_signed(2, 10, 1, 10)
g <- as_complex_edges(g, attr = "type")
complex_walks(g, attr = "type", k = 3)
```

---

count\_complex\_triangles

*count complex triangles*

---

**Description**

Counts the number of all possible signed triangles (+++),(++-), (+-) and (—)

**Usage**

```
count_complex_triangles(g, attr)
```

**Arguments**

<code>g</code>	igraph object.
<code>attr</code>	edge attribute name that encodes positive ("P"), negative ("N") and ambivalent ("A") ties.

**Value**

counts for all complex triangle types

**Author(s)**

David Schoch

**See Also**

[signed\\_triangles](#)

**Examples**

```
library(igraph)
g <- graph.full(4)
E(g)$type <- c("P", "N", "A", "A", "P", "N")
count_complex_triangles(g, attr = "type")
```

---

count\_signed\_triangles  
*count signed triangles*

---

**Description**

Counts the number of all possible signed triangles (+++),(++-), (+-) and (—)

**Usage**

```
count_signed_triangles(g)
```

**Arguments**

g                   igraph object with a sign edge attribute.

**Value**

counts for all 4 signed triangle types

**Author(s)**

David Schoch

**See Also**

[signed\\_triangles](#)

**Examples**

```
library(igraph)
g <- graph.full(4)
E(g)$sign <- c(-1, 1, 1, -1, -1, 1)
count_signed_triangles(g)
```

---

cowList                   *Signed networks from Correlates of War*

---

**Description**

51 signed networks of inter state relations

**Usage**

```
cowList
```

**Format**

List of igraph objects

**Source**

<http://mrvar.fdv.uni-lj.si/pajek/SVG/CoW/default.htm>

**References**

Doreian, P. and Mrvar, A. (2015). "Structural Balance and Signed International Relations". *Journal of Social Structure*, 16(2)

---

degree\_signed

*Signed Degree*

---

**Description**

several options to calculate the signed degree of vertices

**Usage**

```
degree_signed(
  g,
  mode = c("all", "in", "out"),
  type = c("pos", "neg", "ratio", "net")
)
```

**Arguments**

g	igraph object with a sign edge attribute.
mode	character string, "out" for out-degree, "in" for in-degree or "all" for undirected networks.
type	character string, "pos" or "neg" for counting positive or negative neighbors only, "ratio" for pos/(pos+neg), or "net" for pos-neg.

**Value**

centrality scores as numeric vector.

**Author(s)**

David Schoch

---

eigen\_centrality\_signed  
*Signed Eigenvector centrality*

---

### Description

returns the eigenvector associated with the dominant eigenvalue from the adjacency matrix.

### Usage

```
eigen_centrality_signed(g, scale = TRUE)
```

### Arguments

g	igraph object with a sign edge attribute.
scale	Logical scalar, whether to scale the result to have a maximum score of one. If no scaling is used then the result vector is the same as returned by <code>eigen()</code> .

### Details

Note that, with negative values, the adjacency matrix may not have a dominant eigenvalue. This means it is not clear which eigenvector should be used. In addition it is possible for the adjacency matrix to have repeated eigenvalues and hence multiple linearly independent eigenvectors. In this case certain centralities can be arbitrarily assigned. The function returns an error if this is the case.

### Value

centrality scores as numeric vector.

### Author(s)

David Schoch

### References

Bonacich, P. and Lloyd, P. (2004). "Calculating Status with Negative Relations." *Social Networks* 26 (4): 331–38.

Everett, M. and Borgatti, S.P. (2014). "Networks Containing Negative Ties." *Social Networks* 38: 111–20.

### Examples

```
library(igraph)
data("tribes")
eigen_centrality_signed(tribes)
```

---

frustration_exact	<i>Exact frustration index of a signed network</i>
-------------------	--

---

**Description**

Computes the exact frustration index of a signed network using linear programming

**Usage**

```
frustration_exact(g, ...)
```

**Arguments**

<code>g</code>	signed network
<code>...</code>	additional parameters for the ompr solver

**Details**

The frustration index indicates the minimum number of edges whose removal results in a balance network. The function needs the following packages to be installed: `ompr`, `ompr.roi`, `ROI`, and `ROI.plugin.glpk`. The function implements the AND model in Aref et al., 2020

**Value**

list containing the frustration index and the bipartition of nodes

**Author(s)**

David Schoch

**References**

Aref, Samin, Andrew J. Mason, and Mark C. Wilson. "Computing the line index of balance using linear programming optimisation." *Optimization problems in graph theory*. Springer, Cham, 2018. 65-84.

Aref, Samin, Andrew J. Mason, and Mark C. Wilson. "A modeling and computational study of the frustration index in signed networks." *Networks* 75.1 (2020): 95-110.

---

`ggblock`*Plot Blockmodel matrix*

---

**Description**

Plot Blockmodel matrix

**Usage**

```
ggblock(  
  g,  
  blocks = NULL,  
  cols = NULL,  
  show_blocks = FALSE,  
  show_labels = FALSE  
)
```

**Arguments**

<code>g</code>	igraph object with a sign edge attribute.
<code>blocks</code>	vector of block membership as obtained, e.g. from <a href="#">signed_blockmodel</a>
<code>cols</code>	colors used for negative and positive ties
<code>show_blocks</code>	logical. Should block borders be displayed? (Default: FALSE)
<code>show_labels</code>	logical. Should node labels be displayed? (Default: FALSE)

**Value**

ggplot2 object

**Author(s)**

David Schoch

**Examples**

```
## Not run:  
library(igraph)  
data("tribes")  
clu <- signed_blockmodel(tribes, k = 3, alpha = 0.5, annealing = TRUE)  
ggblock(tribes, clu$membership, show_blocks = TRUE, show_labels = TRUE)  
  
## End(Not run)
```

---

 ggsigned

*Plot a signed or complex network*


---

**Description**

Plot a signed or complex network

**Usage**

```
ggsigned(g, type = "signed", attr = NULL, edge_cols = NULL, weights = FALSE)
```

**Arguments**

<code>g</code>	igraph object. Must have a "sign" edge attribute or an attribute containing "P", "N", "A"
<code>type</code>	character string. either "signed" or "complex"
<code>attr</code>	character string. edge attribute that containing "P", "N", "A" if type="complex"
<code>edge_cols</code>	colors used for negative and positive (and ambivalent) ties
<code>weights</code>	logical. If TRUE, weights are computed based on sign. Defaults to FALSE

**Details**

This is a very rudimentary visualization of a signed network. If you are fluent in 'ggraph', you can probably cook up something more sophisticated. The function is thus mostly meant to give a quick overview of the network.

**Value**

ggplot2 object

**Author(s)**

David Schoch

---

 graph\_circular\_signed *circular signed graph*


---

**Description**

circular graph with positive and negative edges.

**Usage**

```
graph_circular_signed(n, r = 1, pos = 0.1, neg = 0.1)
```



**Arguments**

n	number of nodes
r	radius
pos	distance fraction between positive edges
neg	distance fraction between negative edges

**Value**

igraph graph

**Author(s)**

David Schoch

**Examples**

```
library(igraph)
graph_circular_signed(n = 50)
```

---

graph\_from\_adjacency\_matrix\_signed

*Create signed graphs from adjacency matrices*

---

**Description**

Create signed graphs from adjacency matrices

**Usage**

```
graph_from_adjacency_matrix_signed(A, mode = "undirected", ...)
```

**Arguments**

A	square adjacency matrix of a signed graph
mode	Character scalar, specifies how to interpret the supplied matrix. Possible values are: directed, undirected
...	additional parameters for from_adjacency()

**Value**

a signed network as igraph object

**Examples**

```
A <- matrix(c(0, 1, -1, 1, 0, 1, -1, 1, 0), 3, 3)
graph_from_adjacency_matrix_signed(A)
```

---

`graph_from_edgelist_signed`*Create a signed graph from an edgelist matrix*

---

**Description**

Create a signed graph from an edgelist matrix

**Usage**

```
graph_from_edgelist_signed(el, signs, directed = FALSE)
```

**Arguments**

<code>el</code>	The edgelist, a two column matrix, character or numeric.
<code>signs</code>	vector indicating the sign of edges. Entries must be 1 or -1.
<code>directed</code>	whether to create a directed graph.

**Value**

a signed network as igraph object

**Examples**

```
el <- matrix(c("foo", "bar", "bar", "foobar"), ncol = 2, byrow = TRUE)
signs <- c(-1, 1)
graph_from_edgelist_signed(el, signs)
```

---

`is_signed`*Check if network is a signed network*

---

**Description**

Check if network is a signed network

**Usage**

```
is_signed(g)
```

**Arguments**

<code>g</code>	igraph object
----------------	---------------

**Value**

logical scalar

**Examples**

```
g <- sample_islands_signed(2, 5, 1, 5)
is_signed(g)
```

---

laplacian\_matrix\_complex  
*Complex Graph Laplacian*

---

**Description**

The Laplacian of a signed graph containing ambivalent ties.

**Usage**

```
laplacian_matrix_complex(g, attr, norm = FALSE)
```

**Arguments**

g	igraph object.
attr	edge attribute name that encodes positive ("P"), negative ("N") and ambivalent ("A") ties.
norm	Whether to calculate the normalized Laplacian. See definitions below.

**Details**

See [laplacian\\_matrix](#) of igraph for more details. In the complex case, D is a diagonal matrix containing the absolute values of row sums of the complex adjacency matrix.

**Value**

a complex matrix

**Author(s)**

David Schoch

**See Also**

[laplacian\\_matrix\\_signed](#)

---

`laplacian_matrix_signed`*Signed Graph Laplacian*

---

**Description**

The Laplacian of a signed graph.

**Usage**

```
laplacian_matrix_signed(g, norm = FALSE, sparse = FALSE)
```

**Arguments**

<code>g</code>	igraph object with a sign edge attribute.
<code>norm</code>	Whether to calculate the normalized Laplacian. See definitions below.
<code>sparse</code>	Logical scalar, whether to return the result as a sparse matrix. The Matrix package is required for sparse matrices.

**Details**

See [laplacian\\_matrix](#) of igraph for more details. In the signed case, D is a diagonal matrix containing the absolute values of row sums of the signed adjacency matrix.

**Value**

a numeric matrix

**Author(s)**

David Schoch

**Examples**

```
library(igraph)
g <- sample_islands_signed(3, 10, 5 / 10, 1)
laplacian_matrix_signed(g)
laplacian_matrix_signed(g, norm = TRUE)
```

pn\_index

*PN Centrality Index***Description**

centrality index for signed networks by Everett and Borgatti

**Usage**

```
pn_index(g, mode = c("all", "in", "out"))
```

**Arguments**

**g** igraph object with a sign edge attribute.  
**mode** character string, "out" for out-pn, "in" for in-pn or "all" for undirected networks.

**Value**

centrality scores as numeric vector.

**Author(s)**

David Schoch

**References**

Everett, M. and Borgatti, S. (2014) Networks containing negative ties. *Social Networks* 38 111-120

**Examples**

```
A <- matrix(c(
  0, 1, 0, 1, 0, 0, 0, -1, -1, 0,
  1, 0, 1, -1, 1, -1, -1, 0, 0, 0,
  0, 1, 0, 1, -1, 0, 0, 0, -1, 0,
  1, -1, 1, 0, 1, -1, -1, 0, 0, 0,
  0, 1, -1, 1, 0, 1, 0, -1, 0, -1,
  0, -1, 0, -1, 1, 0, 1, 0, 1, -1,
  0, -1, 0, -1, 0, 1, 0, 1, -1, 1,
  -1, 0, 0, 0, -1, 0, 1, 0, 1, 0,
  -1, 0, -1, 0, 0, 1, -1, 1, 0, 1,
  0, 0, 0, 0, -1, -1, 1, 0, 1, 0
), 10, 10)
g <- graph_from_adjacency_matrix_signed(A, "undirected")
pn_index(g)
```

---

`sample_bipartite_signed`*Bipartite random signed graphs*

---

**Description**

Bipartite random signed graphs

**Usage**

```
sample_bipartite_signed(  
  n1,  
  n2,  
  p,  
  p_neg,  
  directed = FALSE,  
  mode = c("out", "in", "all")  
)
```

**Arguments**

<code>n1</code>	Integer scalar, the number of bottom vertices.
<code>n2</code>	Integer scalar, the number of top vertices.
<code>p</code>	The probability for drawing an edge between two arbitrary vertices.
<code>p_neg</code>	The probability of a drawn edge to be a negative tie
<code>directed</code>	logical, whether the graph will be directed. defaults to FALSE.
<code>mode</code>	Character scalar, specifies how to direct the edges in directed graphs. If it is 'out', then directed edges point from bottom vertices to top vertices. If it is 'in', edges point from top vertices to bottom vertices. 'out' and 'in' do not generate mutual edges. If this argument is 'all', then each edge direction is considered independently and mutual edges might be generated. This argument is ignored for undirected graphs.

**Value**

A signed bipartite igraph graph.

**Examples**

```
sample_bipartite_signed(10, 10, 0.5, 0.5)
```

---

sample_gnp_signed	<i>Generate random signed graphs according to the G(n,p) Erdos-Renyi model</i>
-------------------	--

---

**Description**

Generate random signed graphs according to the G(n,p) Erdos-Renyi model

**Usage**

```
sample_gnp_signed(n, p, p_neg, directed = FALSE, loops = FALSE)
```

**Arguments**

n	The number of vertices in the graph.
p	The probability for drawing an edge between two arbitrary vertices.
p_neg	The probability of a drawn edge to be a negative tie
directed	logical, whether the graph will be directed. defaults to FALSE.
loops	logical, whether to add loop edges, defaults to FALSE.

**Value**

a signed igraph graph object

**References**

Erdos, P. and Renyi, A., On random graphs, *Publicationes Mathematicae* 6, 290–297 (1959).

**Examples**

```
sample_gnp_signed(10, 0.4, 0.5)
```

---

sample_islands_signed	<i>A graph with random subgraphs connected by negative edges</i>
-----------------------	--

---

**Description**

Create a number of Erdos-Renyi random graphs with identical parameters, and connect them with the specified number of negative ties.

**Usage**

```
sample_islands_signed(islands.n, islands.size, islands.pin, n.inter)
```

**Arguments**

islands.n	The number of islands in the graph.
islands.size	The size of the islands in the graph.
islands.pin	The probability of intra-island edges.
n.inter	number of negative edges between two islands.

**Value**

a signed igraph graph

**Author(s)**

David Schoch

**Examples**

```
library(igraph)
sample_islands_signed(3, 10, 0.5, 1)
```

---

signed\_blockmodel      *Blockmodeling for signed networks*

---

**Description**

Finds blocks of nodes with intra-positive and inter-negative edges

**Usage**

```
signed_blockmodel(g, k, alpha = 0.5, annealing = FALSE)
```

**Arguments**

g	igraph object with a sign edge attribute.
k	number of blocks
alpha	see details
annealing	logical. if TRUE, use simulated annealing (Default: FALSE)

**Details**

The function minimizes  $P(C) = \alpha N + (1 - \alpha)P$ , where  $N$  is the total number of negative ties within plus-sets and  $P$  be the total number of positive ties between plus-sets. This function implements the structural balance model. That is, all diagonal blocks are positive and off-diagonal blocks negative. For the generalized version see [signed\\_blockmodel\\_general](#).

**Value**

numeric vector of block assignments and the associated criterion value



**Author(s)**

David Schoch

**References**

Doreian, Patrick and Andrej Mrvar (2009). Partitioning signed social networks. *Social Networks* 31(1) 1-11

**Examples**

```
library(igraph)

g <- sample_islands_signed(10, 10, 1, 20)
clu <- signed_blockmodel(g, k = 10, alpha = 0.5)
table(clu$membership)
clu$criterion

# Using simulated annealing (less change of getting trapped in local optima)
data("tribes")
clu <- signed_blockmodel(tribes, k = 3, alpha = 0.5, annealing = TRUE)
table(clu$membership)
clu$criterion
```

---

signed\_blockmodel\_general

*Generalized blockmodeling for signed networks*

---

**Description**

Finds blocks of nodes with specified inter/intra group ties

**Usage**

```
signed_blockmodel_general(g, blockmat, alpha = 0.5)
```

**Arguments**

g	igraph object with a sign edge attribute.
blockmat	Integer Matrix. Specifies the inter/intra group patterns of ties
alpha	see details

**Details**

The function minimizes  $P(C) = \alpha N + (1 - \alpha)P$ , where  $N$  is the total number of negative ties within plus-sets and  $P$  be the total number of positive ties between plus-sets. This function implements the generalized model. For the structural balance version see [signed\\_blockmodel](#).

**Value**

numeric vector of block assignments and the associated criterion value

**Author(s)**

David Schoch

**References**

Doreian, Patrick and Andrej Mrvar (2009). Partitioning signed social networks. *Social Networks* 31(1) 1-11

**Examples**

```
library(igraph)
# create a signed network with three groups and different inter/intra group ties
g1 <- g2 <- g3 <- graph.full(5)

V(g1)$name <- as.character(1:5)
V(g2)$name <- as.character(6:10)
V(g3)$name <- as.character(11:15)

g <- Reduce("%u%", list(g1, g2, g3))
E(g)$sign <- 1
E(g)$sign[1:10] <- -1
g <- add.edges(g, c(rbind(1:5, 6:10)), attr = list(sign = -1))
g <- add.edges(g, c(rbind(1:5, 11:15)), attr = list(sign = -1))
g <- add.edges(g, c(rbind(11:15, 6:10)), attr = list(sign = 1))

# specify the link patterns between groups
blockmat <- matrix(c(1, -1, -1, -1, 1, 1, -1, 1, -1), 3, 3, byrow = TRUE)
signed_blockmodel_general(g, blockmat, 0.5)
```

---

signed\_triangles      *list signed triangles*

---

**Description**

lists all possible signed triangles

**Usage**

```
signed_triangles(g)
```

**Arguments**

**g**                    igraph object with a sign edge attribute.

**Value**

matrix of vertex ids and the number of positive ties per triangle

**Author(s)**

David Schoch

**See Also**

[count\\_signed\\_triangles](#)

**Examples**

```
library(igraph)
g <- graph.full(4)
E(g)$sign <- c(-1, 1, 1, -1, -1, 1)
signed_triangles(g)
```

---

triad\_census\_signed    *signed triad census*

---

**Description**

triad census for signed graphs

**Usage**

```
triad_census_signed(g)
```

**Arguments**

**g**                    igraph object with a sign edge attribute.

**Value**

counts for all 139 signed directed triangle types

**Author(s)**

David Schoch

**Examples**

```
library(igraph)
g <- graph.full(4, directed = TRUE)
E(g)$sign <- rep(c(-1, 1, 1, -1, -1, 1), 2)
triad_census_signed(g)
```

---

tribes

*Signed network of New Guinean highland tribes*

---

**Description**

Signed social network of tribes of the Gahuku–Gama alliance structure of the Eastern Central Highlands of New Guinea, from Kenneth Read. The network contains sixteen tribes connected by friendship ("rova") and enmity ("hina").

**Usage**

tribes

**Format**

An igraph object

**Source**

<http://vlado.fmf.uni-lj.si/pub/networks/data/ucinet/gama.dat>

**References**

Read, K. E. (1954) Cultures of the central highlands, New Guinea. *Southwestern Journal of Anthropology*, 1–43.

# Index

## \* datasets

avatar, 8  
cowList, 11  
tribes, 28

as\_adj\_complex, 2, 3, 5  
as\_adj\_signed, 3, 3  
as\_complex\_edges, 4  
as\_incidence\_complex, 4  
as\_incidence\_matrix, 5  
as\_incidence\_signed, 5  
as\_signed\_proj, 6, 7  
as\_unsigned\_2mode, 6, 7  
avatar, 8

balance\_score, 8

complex\_walks, 9  
count\_complex\_triangles, 10  
count\_signed\_triangles, 11, 27  
cowList, 11

degree\_signed, 12

eigen\_centrality\_signed, 13

frustration\_exact, 14

ggblock, 15  
ggsigned, 16  
graph\_circular\_signed, 16  
graph\_from\_adjacency\_matrix\_signed, 17  
graph\_from\_edgelist\_signed, 18

is\_signed, 18

laplacian\_matrix, 19, 20  
laplacian\_matrix\_complex, 5, 19  
laplacian\_matrix\_signed, 19, 20

pn\_index, 21

sample\_bipartite\_signed, 22  
sample\_gnp\_signed, 23  
sample\_islands\_signed, 23  
signed\_blockmodel, 15, 24, 25  
signed\_blockmodel\_general, 24, 25  
signed\_triangles, 10, 11, 26

triad\_census\_signed, 27  
tribes, 28