ergm-terms {ergm} R Documentation

Terms used in Exponential Family Random Graph Models

Description

The function ergm is used to fit exponential random graph models, in which the probability of a given network, *y*, on a set of nodes is $h(y) \exp\{\eta(\theta) \cdot g(y)\}/c(\theta)$, where *h(y)* is the reference measure (for valued network models), *g(y)* is a vector of network statistics for *y*, *η(θ)* is a natural parameter vector of the same length (with *η(θ)=θ* for most terms), and *c(θ)* is the normalizing constant for the distribution.

The network statistics *g(y)* are entered as terms in the function call to `ergm`.

This page describes the possible terms (and hence network statistics) included in `ergm` package. Other packages may add their own terms, and package `ergm.userterms` provides tools for implementing them.

The current recommendation for any package implementing additional terms is to create a help file with a name or alias `ergm-terms`, so that `help("ergm-terms")` will list ERGM terms available from all loaded packages.

Specifying models

Terms to `ergm` are specified by a formula to represent the network and network statistics. This is done via a `formula`, that is, an R formula object, of the form `y ~ <term 1> + <term 2> ...`, where `y` is a network object or a matrix that can be coerced to a network object, and `<term 1>`, `<term 2>`, etc, are each terms chosen from the list given below. To create a network object in R, use the `network` function, then add nodal attributes to it using the `%v%` operator if necessary.

Binary and valued ERGM terms

`ergm` functions such as `ergm` and `simulate` (for ERGMs) may operate in two modes: binary and weighted/valued, with the latter activated by passing a non-NULL value as the `response` argument, giving the edge attribute name to be modeled/simulated.

Binary ERGM statistics cannot be used in valued mode and vice versa. However, a substantial number of binary ERGM statistics — particularly the ones with dyadic indepenence — have simple generalizations to valued ERGMs, and have been adapted in `ergm`. They have the same form as their binary ERGM counterparts, with an additional argument: `form`, which, at this time, has two possible values: `"sum"` (the default) and `"nonzero"`. The former creates a statistic of the form $\sum_{i,j} x_{i,j} y_{i,j}$, where $y_{i,j}$ is the value of dyad *(i,j)* and $x_{i,j}$ is the term's covariate associated with it. The latter computes the binary version, with the edge considered to be present if its value is not 0.

Valued version of some binary ERGM terms have an argument `threshold`, which sets the value above which a dyad is conidered to have a tie. (Value less than or equal to `threshold` is considered a nontie.)

Covariate transformations

Some terms taking nodal or dyadic covariates take optional `transform` and `transformname` arguments. `transform` should be a function with one argument, taking a data structure of the same mode as the covariate and returning a similarly structured data structure, transforming the covariate as needed.

For example, `nodecov("a", transform=function(x) x^2)` will add a nodal covariate having the square of the value of the nodal attribute `"a"`.

`transformname`, if given, will be added to the term's name to help identify it.

Terms to represent network statistics included in the `ergm` package

A cross-referenced html version of the term documentation is is available via `vignette('ergm-term-crossRef')` and terms can also be searched via `search.ergmTerms`.

`absdiff(attrname, pow=1)` (binary) (dyad-independent) (frequently-used) (directed) (undirected) (quantitative nodal attribute), `absdiff(attrname, pow=1, form ="sum")` (valued) (dyad-independent) (directed) (undirected) (quantitative nodal attribute)

> *Absolute difference:* The `attrname` argument is a character string giving the name of a quantitative attribute in the network's vertex attribute list. This term adds one network statistic to the model equaling the sum of `abs(attrname[i]-attrname[j])^pow` for all edges (i,j) in the network.

`absdiffcat(attrname, base=NULL)` (binary) (dyad-independent) (directed) (undirected) (categorical nodal attribute), `absdiffcat(attrname, base=NULL, form="sum")` (valued) (dyad-independent) (directed) (undirected) (categorical nodal attribute)

> *Categorical absolute difference:* The `attrname` argument is a character string giving the name of a quantitative attribute in the network's vertex attribute list. This term adds one statistic for every possible nonzero distinct value of `abs(attrname[i]-attrname[j])` in the network; the value of each such statistic is the number of edges in the network with the corresponding absolute difference. The optional base argument is a vector indicating which nonzero differences, in order from smallest to largest, should be omitted from the model (i.e., treated like the zero-difference category). The base argument, if used, should contain indices, not differences themselves. For instance, if the possible values of `abs(attrname[i]-attrname[j])` are 0, 0.5, 3, 3.5, and 10, then to omit 0.5 and 10 one should set base=`c(1, 4)`. Note that this term should generally be used only when the quantitative attribute has a limited number of possible values; an example is the `"Grade"` attribute of the `faux.mesa.high` or `faux.magnolia.high` datasets.

`altkstar(lambda, fixed=FALSE)` (binary) (undirected) (curved) (categorical nodal attribute)

> *Alternating k-star:* This term adds one network statistic to the model equal to a weighted alternating sequence of k-star statistics with weight parameter `lambda`. This is the version given in Snijders et al. (2006). The `gwdegree` and `altkstar` produce mathematically equivalent models, as long as they are used together with the edges (or `kstar(1)`) term, yet the interpretation of the `gwdegree` parameters is slightly more straightforward than the interpretation of the `altkstar` parameters. For this reason, we recommend the use of the `gwdegree` instead of `altkstar`. See Section 3 and especially equation (13) of Hunter (2007) for details. The optional argument `fixed` indicates whether the decay parameter is fixed at the given value, or is to be fit as a curved exponential family model (see Hunter and Handcock, 2006). The default is `FALSE`, which means the scale parameter is not fixed and thus the model is a CEF model. This term can only be used with undirected networks.

`asymmetric(attrname=NULL, diff=FALSE, keep=NULL)` (binary) (directed) (dyad-independent) (triad-related)

> *Asymmetric dyads:* This term adds one network statistic to the model equal to the number of pairs of actors for which exactly one of *(i,j)*

or *(j,i)* exists. This term can only be used with directed networks. If the optional `attrname` argument is used, only asymmetric pairs that match on the named vertex attribute are counted. The optional modifiers `diff` and `keep` are used in the same way as for the `nodematch` term; refer to this term for details and an example.

`atleast(threshold=0)` (valued) (directed) (undirected) (dyad-independent)

Number of dyads with values greater than or equal to a threshold Adds one statistic equaling to the number of dyads whose values equal or exceed `threshold`.

`atmost(threshold=0)` (valued) (directed) (undirected) (dyad-independent)

Number of dyads with values less than or equal to a threshold Adds one statistic equaling to the number of dyads whose values equal or are exceeded by `threshold`.

`b1concurrent(by=NULL)` (binary) (bipartite) (undirected) (categorical nodal attribute)

Concurrent node count for the first mode in a bipartite (aka two-mode) network: This term adds one network statistic to the model, equal to the number of nodes in the first mode of the network with degree 2 or higher. The first mode of a bipartite network object is sometimes known as the "actor" mode. The optional argument by is a character string giving the name of an attribute in the network's vertex attribute list; it functions just like the by argument of the `b1degree` term. Without the optional argument, this statistic is equivalent to `b1mindegree(2)`. This term can only be used with undirected bipartite networks.

`b1cov(attrname, transform, transformname)` (binary) (undirected) (bipartite) (dyad-independent) (quantitative nodalattribute) (frequently-used), `b1cov(attrname, transform, transformname, form="sum")` (valued) (undirected) (bipartite) (dyad-independent) (quantitative nodal attribute) (frequently-used)

Main effect of a covariate for the first mode in a bipartite (aka two-mode) network: The `attrname` argument is a character string giving the name of a numeric (not categorical) attribute in the network's vertex attribute list. This term adds a single network statistic to the model equaling the total value of `attrname(i)` for all edges *(i,j)* in the network. This term may only be used with bipartite networks. For categorical attributes, see `b1factor`.

`b1degrange(from, to=+Inf, by=NULL, homophily=FALSE)` (binary) (bipartite) (undirected)

Degree range for the first mode in a bipartite (a.k.a. two-mode) network: The `from` and `to` arguments are vectors of distinct integers (or `+Inf`, for `to` (its default)). If one of the vectors has length 1, it is recycled to the length of the other. Otherwise, they must have the same length. This term adds one network statistic to the model for each element of `from` (or `to`); the *i*th such statistic equals the number of nodes of the first mode ("actors") in the network of degree greater than or equal to `from[i]` but strictly less than `to[i]`, i.e. with edge count in semiopen interval `[from,to)`. The optional argument by is a character string giving the name of an attribute in the network's vertex attribute list. If this is specified and `homophily` is TRUE, then degrees are calculated using the subnetwork consisting of only edges whose endpoints have the same value of the by attribute. If by is specified and `homophily` is FALSE (the default), then separate degree range statistics are calculated for nodes having each separate value of the attribute.

This term can only be used with bipartite networks; for directed networks see `idegrange` and `odegrange`. For undirected networks, see `degrange`, and see `b2degrange` for degrees of the second mode ("events").

`b1degree(d, by=NULL)` (binary) (bipartite) (undirected) (categorical nodal attribute) (frequently-used)

Degree for the first mode in a bipartite (aka two-mode) network: The d argument is a vector of distinct integers. This term adds one network statistic to the model for each element in d; the *i*th such statistic equals the number of nodes of degree `d[i]` in the first mode of a bipartite network, i.e. with exactly `d[i]` edges. The first mode of a bipartite network object is sometimes known as the "actor" mode. The optional argument by is a character string giving the name of an attribute in the network's vertex attribute list. If this is specified then each node's degree is tabulated only with other nodes having the same value of the by attribute. This term can only be used with undirected bipartite networks.

`b1factor(attrname, base=1)` (binary) (bipartite) (undirected) (dyad-independent) (frequently-used) (categorical nodal attribute), `b1factor(attrname, base=1, form="sum")` (valued) (bipartite) (undirected) (dyad-independent) (frequently-used) (categorical nodal attribute)

Factor attribute effect for the first mode in a bipartite (aka two-mode) network: The `attrname` argument is a character string giving the name of a categorical attribute in the network's vertex attribute list. This term adds multiple network statistics to the model, one for each of (a subset of) the unique values of the `attrname` attribute. Each of these statistics gives the number of times a node with that attribute in the first mode of the network appears in an edge. The first mode of a bipartite network object is sometimes known as the "actor" mode. To include all attribute values is usually not a good idea, because the sum of all such statistics equals the number of edges and hence a linear dependency would arise in any model also including edges. Thus, the base argument tells which value(s) (numbered in order according to the `sort` function) should be omitted. The default value, base=1, means that the smallest (i.e., first in sorted order) attribute value is omitted. For example, if the “fruit” factor has levels “orange”, “apple”, “banana”, and “pear”, then to add just two terms, one for “apple” and one for “pear”, then set “banana” and “orange” to the base (remember to sort the values first) by using `nodefactor("fruit", base=2:3)`. This term can only be used with undirected bipartite networks.

`b1mindegree(d)` (binary) (bipartite) (undirected)

Minimum degree for the first mode in a bipartite (aka two-mode) network: The d argument is a vector of distinct integers. This term adds one network statistic to the model for each element in d; the *i*th such statistic equals the number of nodes in the first mode of a bipartite network with at least degree `d[i]`. The first mode of a bipartite network object is sometimes known as the "actor" mode. This term can only be used with undirected bipartite networks.

`b1nodematch(attrname, diff=FALSE, keep=NULL, by=NULL, alpha=1, beta=1, byb2attr=NULL)` (binary) (bipartite) (undirected) (dyad-independent) (categorical nodal attribute) (frequently-used)

Nodal attribute-based homophily effect for the first mode in a bipartite (aka two-mode) network: This term is introduced in Bomiriya et al (2014). The `attrname` argument is a character string giving the name of a categorical attribute in the network's vertex attribute list. Out of the two arguments (discount parameters) `alpha` and `beta`, both which takes values from [0,1], only one should be set at a time. If none is set to a value other than 1, this term will simply be a homophily based two-star statistic. This term adds one statistic to the model unless `diff` is set to TRUE, in which case the term adds multiple network statistics to the model, one for each of (a subset of) the unique values of the `attrname` attribute. To include only the attribute values you wish, use the keep argument. If an `alpha` discount parameter is used, each of these statistics gives the sum of the number of common second-mode nodes raised to the power `alpha` for each pair of first-mode nodes with that attribute. If a `beta` discount parameter is used, each of these statistics gives half the sum of the number of two-paths with two first-mode nodes with that attribute as the two ends of the two path raised to the power `beta` for each edge in the network. The `byb2attr` argument is a character string giving the name of a second mode categorical attribute in the network's attribute list. Setting this argument will separate the orginal statistics based on the values of the set second mode attribute— i.e. for example, if `diff` is FALSE, then the sum of all the statistics for each level of this second-mode attribute will be equal to the original `b1nodematch` statistic where `byb2attr` set to `NULL`. This term can only be used with undirected bipartite networks.

`b1star(k, attrname=NULL)` (binary) (bipartite) (undirected) (categorical nodal attribute)

> *k-Stars for the first mode in a bipartite (aka two-mode) network:* The `k` argument is a vector of distinct integers. This term adds one network statistic to the model for each element in `k`. The *i*th such statistic counts the number of distinct `k[i]`-stars whose center node is in the first mode of the network. The first mode of a bipartite network object is sometimes known as the "actor" mode. A *k*-star is defined to be a center node *N* and a set of *k* different nodes *\{O_1, ..., O_k\}* such that the ties *\{N, O_i\}* exist for *i=1, ..., k*. The optional argument `attrname` is a character string giving the name of an attribute in the network's vertex attribute list. If this is specified then the count is over the number of *k*-stars (with center node in the first mode) where all nodes have the same value of the attribute. This term can only be used for undirected bipartite networks. Note that `b1star(1)` is equal to `b2star(1)` and to `edges`.

`b1starmix(k, attrname, base=NULL, diff=TRUE)` (binary) (bipartite) (undirected) (categorical nodal attribute)

> *Mixing matrix for k-stars centered on the first mode of a bipartite network:* Only a single value of *k* is allowed. This term counts all k-stars in which the b2 nodes (called events in some contexts) are homophilous in the sense that they all share the same value of `attrname`. However, the b1 node (in some contexts, the actor) at the center of the k-star does NOT have to have the same value as the b2 nodes; indeed, the values taken by the b1 nodes may be completely distinct from those of the b2 nodes, which allows for the use of this term in cases where there are two separate nodal attributes, one for the b1 nodes and another for the b2 nodes (in this case, however, these two attributes should be combined to form a single nodal attribute called `attrname`. A different statistic is created for each value of `attrname` seen in a b1 node, even if no k-stars are observed with this value. Whether a different statistic is created for each value seen in a b2 node depends on the value of the `diff` argument: When `diff=TRUE`, the default, a different statistic is created for each value and thus the behavior of this term is reminiscent of the `nodemix` term, from which it takes its name; when `diff=FALSE`, all homophilous k-stars are counted together, though these k-stars are still categorized according to the value of the central b1 node. The `base` term may be used to control which of the possible terms are left out of the model: By default, all terms are included, but if `base` is set to a vector of indices then the corresponding terms (in the order they would be created when `base=NULL`) are left out.

`b1twostar(b1attrname, b2attrname, base=NULL)` (binary) (bipartite) (undirected) (categorical nodal attribute)

> *Two-star census for central nodes centered on the first mode of a bipartite network:* This term takes two nodal attribute names, one for b1 nodes (actors in some contexts) and one for b2 nodes (events in some contexts). Only `b1attrname` is required; if `b2attrname` is not passed, it is assumed to be the same as `b1attrname`. Assuming that there are *n_1* values of `b1attrname` among the b1 nodes and *n_2* values of `b2attrname` among the b2 nodes, then the total number of distinct categories of two stars according to these two attributes is *n_1(n_2)(n_2+1)/2*. This model term creates a distinct statistic counting each of these categories. The `base` term may be used to leave some of these categories out; when passed as a vector of integer indices (in the order the statistics would be created when `base=NULL`), the corresponding terms will be left out.

`b2concurrent(by=NULL)` (binary) (bipartite) (undirected) (frequently-used)

> *Concurrent node count for the second mode in a bipartite (aka two-mode) network:* This term adds one network statistic to the model, equal to the number of nodes in the second mode of the network with degree 2 or higher. The second mode of a bipartite network object is sometimes known as the "event" mode. The optional argument `by` is a character string giving the name of an attribute in the network's vertex attribute list; it functions just like the `by` argument of the `b2degree` term. Without the optional argument, this statistic is equivalent to `b2mindegree(2)`. This term can only be used with undirected bipartite networks.

`b2cov(attrname, transform, transformname)` (binary) (undirected) (bipartite) (dyad-independent) (quantitative nodal attribute) (frequently-used), `b2cov(attrname, transform, transformname, form="sum")` (valued) (undirected) (bipartite) (dyad-independent) (quantitative nodal attribute) (frequently-used)

> *Main effect of a covariate for the second mode in a bipartite (aka two-mode) network:* The `attrname` argument is a character string giving the name of a numeric (not categorical) attribute in the network's vertex attribute list. This term adds a single network statistic to the model equaling the total value of `attrname(j)` for all edges *(i,j)* in the network. This term may only be used with bipartite networks. For categorical attributes, see `b2factor`.

`b2degrange(from, to=+Inf, by=NULL, homophily=FALSE)` (binary) (bipartite) (undirected)

> *Degree range for the second mode in a bipartite (a.k.a. two-mode) network:* The `from` and `to` arguments are vectors of distinct integers (or `+Inf`, for `to` (its default)). If one of the vectors has length 1, it is recycled to the length of the other. Otherwise, they must have the same length. This term adds one network statistic to the model for each element of `from` (or `to`); the *i*th such statistic equals the number of nodes of the second mode ("events") in the network of degree greater than or equal to `from[i]` but strictly less than `to[i]`, i.e. with edge count in semiopen interval `[from,to)`. The optional argument `by` is a character string giving the name of an attribute in the network's vertex attribute list. If this is specified and `homophily` is TRUE, then degrees are calculated using the subnetwork consisting of only edges whose endpoints have the same value of the `by` attribute. If `by` is specified and `homophily` is FALSE (the default), then separate degree range statistics are calculated for nodes having each separate value of the attribute.
>
> This term can only be used with bipartite networks; for directed networks see `idegrange` and `odegrange`. For undirected networks, see `degrange`, and see `b1degrange` for degrees of the first mode ("actors").

`b2degree(d, by=NULL)` (binary) (bipartite) (undirected) (categorical nodal attribute) (frequently-used)

> *Degree for the second mode in a bipartite (aka two-mode) network:* The `d` argument is a vector of distinct integers. This term adds one network statistic to the model for each element in `d`; the *i*th such statistic equals the number of nodes of degree `d[i]` in the second mode of a bipartite network, i.e. with exactly `d[i]` edges. The second mode of a bipartite network object is sometimes known as the "event" mode. The optional term `by` is a character string giving the name of an attribute in the network's vertex attribute list. If this is specified then each node's degree is tabulated only with other nodes having the same value of the `by` attribute. This term can only be used with undirected bipartite networks.

`b2factor(attrname, base=1)` (binary) (bipartite) (undirected) (dyad-independent) (categorical nodal attribute) (frequently-used), `b2factor(attrname, base=1, form="sum")` (valued) (bipartite) (undirected) (dyad-independent) (categorical nodal attribute) (frequently-used)

> *Factor attribute effect for the second mode in a bipartite (aka two-mode) network :* The `attrname` argument is a character string giving the name of a categorical attribute in the network's vertex attribute list. This term adds multiple network statistics to the model, one for each of (a subset of) the unique values of the `attrname` attribute. Each of these statistics gives the number of times a node with that attribute in the second mode of the network appears in an edge. The second mode of a bipartite network object is sometimes known as the "event" mode. To include all attribute values is usually not a good idea, because the sum of all such statistics equals the number of edges and hence a linear dependency would arise in any model also including edges. Thus, the `base` argument tells which value(s) (numbered in order according to the `sort` function) should be omitted. The default value, `base=1`, means that the smallest (i.e., first in sorted order) attribute value is omitted. For example, if the “fruit” factor has levels “orange”, “apple”, “banana”, and “pear”, then to add just two terms, one for “apple” and one for “pear”, then set “banana” and “orange” to the base (remember to sort the values first) by using `nodefactor("fruit", base=2:3)`. This term can only be used with undirected bipartite networks.

`b2mindegree(d)` (binary) (bipartite) (undirected)

> *Minimum degree for the second mode in a bipartite (aka two-mode) network:* The `d` argument is a vector of distinct integers. This term

adds one network statistic to the model for each element in `d`; the *i*th such statistic equals the number of nodes in the second mode of a bipartite network with at least degree `d[i]`. The second mode of a bipartite network object is sometimes known as the "event" mode. This term can only be used with undirected bipartite networks.

`b2nodematch(attrname, diff=FALSE, keep=NULL, by=NULL, alpha=1, beta=1, byb1attr=NULL)` (binary) (bipartite) (undirected) (dyad-independent) (categorical nodal attribute) (frequently-used)

Nodal attribute-based homophily effect for the second mode in a bipartite (aka two-mode) network: This term is introduced in Bomiriya et al (2014). The `attrname` argument is a character string giving the name of a categorical attribute in the network's vertex attribute list. Out of the two arguments (discount parameters) `alpha` and `beta`, both which takes values from [0,1], only one should be set at a time. If none is set to a value other than 1, this term will simply be a homophily based two-star statistic. This term adds one statistic to the model unless `diff` is set to TRUE, in which case the term adds multiple network statistics to the model, one for each of (a subset of) the unique values of the `attrname` attribute. To include only the attribute values you wish, use the keep argument. If an `alpha` discount parameter is used, each of these statistics gives the sum of the number of common first-mode nodes raised to the power `alpha` for each pair of second-mode nodes with that attribute. If a `beta` discount parameter is used, each of these statistics gives half the sum of the number of two-paths with two second-mode nodes with that attribute as the two ends of the two path raised to the power `beta` for each edge in the network. The `byb1attr` argument is a character string giving the name of a first mode categorical attribute in the network's attribute list. Setting this argument will separate the orginal statistics based on the values of the set first mode attribute— i.e. for example, if `diff` is `FALSE`, then the sum of all the statistics for each level of this first-mode attribute will be equal to the original `b2nodematch` statistic where `byb1attr` set to `NULL`. This term can only be used with undirected bipartite networks.

`b2star(k, attrname=NULL)` (binary) (bipartite) (undirected) (categorical nodal attribute)

k-Stars for the second mode in a bipartite (aka two-mode) network: The `k` argument is a vector of distinct integers. This term adds one network statistic to the model for each element in `k`. The *i*th such statistic counts the number of distinct `k[i]`-stars whose center node is in the second mode of the network. The second mode of a bipartite network object is sometimes known as the "event" mode. A *k*-star is defined to be a center node *N* and a set of *k* different nodes *\{O_1, ..., O_k\}* such that the ties *\{N, O_i\}* exist for *i=1, ..., k*. The optional argument `attrname` is a character string giving the name of an attribute in the network's vertex attribute list. If this is specified then the count is over the number of *k*-stars (with center node in the second mode) where all nodes have the same value of the attribute. This term can only be used for undirected bipartite networks. Note that `b2star(1)` is equal to `b1star(1)` and to `edges`.

`b2starmix(k, attrname, base=NULL, diff=TRUE)` (binary) (bipartite) (undirected) (categorical nodal attribute)

Mixing matrix for k-stars centered on the second mode of a bipartite network: This term is exactly the same as `b1starmix` except that the roles of b1 and b2 are reversed.

`b2twostar(b1attrname, b2attrname, base=NULL)` (binary) (bipartite) (undirected) (categorical nodal attribute)

Two-star census for central nodes centered on the second mode of a bipartite network: This term is exactly the same as `b1twostar` except that the roles of b1 and b2 are reversed.

`balance` (binary) (triad-related) (directed) (undirected)

Balanced triads: This term adds one network statistic to the model equal to the number of triads in the network that are balanced. The balanced triads are those of type `102` or `300` in the categorization of Davis and Leinhardt (1972). For details on the 16 possible triad types, see `?triad.classify` in the `{sna}` package. For an undirected network, the balanced triads are those with an even number of ties (i.e., 0 and 2).

`coincidence(d=NULL,active=0)` (binary) (bipartite) (undirected)

Coincident node count for the second mode in a bipartite (aka two-mode) network: By default this term adds one network statistic to the model for each pair of nodes of mode two. It is equal to the number of (first mode) mutual partners of that pair. The first mode of a bipartite network object is sometimes known as the "actor" mode and the seconds as the "event" mode. So this is the number of actors going to both events in the pair. The optional argument `d` is a two-column matrix of (row-wise) pairs indices where the first row is less than the second row. The second optional argument, `active`, selects pairs for which the observed count is at least `active`. This term can only be used with undirected bipartite networks.

`concurrent(by=NULL)` (binary) (undirected) (categorical nodal attribute)

Concurrent node count: This term adds one network statistic to the model, equal to the number of nodes in the network with degree 2 or higher. The optional argument by is a character string giving the name of an attribute in the network's vertex attribute list; it functions just like the by argument of the `degree` term. This term can only be used with undirected networks.

`concurrentties(by=NULL)` (binary) (undirected) (categorical nodal attribute)

Concurrent tie count: This term adds one network statistic to the model, equal to the number of ties incident on each actor beyond the first. The optional argument by is a character string giving the name of an attribute in the network's vertex attribute list; it functions just like the by argument of the `degree` term. This term can only be used with undirected networks.

`ctriple(attrname=NULL)` (binary) (directed) (triad-related) (categorical nodal attribute) , a.k.a. `ctriad` (binary) (directed) (triad-related) (categorical nodal attribute)

Cyclic triples: This term adds one statistic to the model, equal to the number of cyclic triples in the network, defined as a set of edges of the form *{(i,j), (j,k), (k,i)}*. Note that for all directed networks, `triangle` is equal to `ttriple+ctriple`, so at most two of these three terms can be in a model. The optional argument `attrname` is a character string giving the name of an attribute in the network's vertex attribute list. If this is specified then the count is over the number of cyclic triples where all three nodes have the same value of the attribute. This term can only be used with directed networks.

`cycle(k)` (binary) (directed) (undirected)

Cycles: The `k` argument is a vector of distinct integers. This term adds one network statistic to the model for each element in `k`; the *i*th such statistic equals the number of cycles in the network with length exactly `k[i]`. The cycle statistic applies to both directed and undirected networks. For directed networks, it counts directed cycles of length *k*, as opposed to undirected cycles in the undirected case. The directed cycle terms of lengths 2 and 3 are equivalent to `mutual` and `ctriple` (respectively). The undirected cycle term of length 3 is equivalent to `triangle`, and there is no undirected cycle term of length 2.

`cyclicalties(attrname=NULL)` (binary) (directed), `cyclicalties(threshold=0)` (valued) (directed) (undirected)

Cyclical ties: This term adds one statistic, equal to the number of ties *i-->j* such that there exists a two-path from *i* to *j*. (Related to the `ttriple` term.) The binary version takes a nodal attribute `attrname`, and, if given, all three nodes involved (*i*, *j*, and the node on the two-path) must match on this attribute in order for *i-->j* to be counted. The binary version of this term can only be used with directed networks. The valued version can be used with both directed and undirected.

`cyclicalweights(twopath="min",combine="max",affect="min")` (valued) (directed) (undirected)

Cyclical weights: This statistic implements the cyclical weights statistic, like that defined by Krivitsky (2012), Equation 13, but with the focus dyad being *y_{j,i}* rather than *y_{i,j}*. The currently implemented options for `twopath` is the minimum of the constituent dyads (`"min"`) or their geometric mean (`"geomean"`); for `combine`, the maximum of the 2-path strengths (`"max"`) or their sum (`"sum"`); and for `affect`, the minimum of the focus dyad and the combined strength of the two paths (`"min"`) or their geometric mean (`"geomean"`). For each of these options, the first (and the default) is more stable but also more conservative, while the second is more sensitive but more likely to induce a multimodal distribution of networks.

`ddsp(d, type="OTP")` (binary) (directed)

Directed dyadwise shared partners: This term adds one network statistic to the model for each element in `d` where the *i*th such statistic equals the number of *dyads* in the network with exactly `d[i]` shared partners. This term can only be used with directed networks. Multiple shared partner definitions are possible; the `type` argument may be used to select the type of shared partner to be counted (see below for type codes). By default, outgoing two-paths are employed.

While there is only one shared partner configuration in the undirected case, nine distinct configurations are possible for directed graphs. Currently, edgewise shared partner terms may be defined with respect to five of these configurations; they are defined here as follows (using terminology from Butts (2008) and the `relevent` package):

Outgoing Two-path (OTP)

vertex *k* is an OTP shared partner of ordered pair *(i,j)* iff *i->k->j*. Also known as "transitive shared partner".

Incoming Two-path (ITP)

vertex *k* is an ITP shared partner of ordered pair *(i,j)* iff *j->k->i*. Also known as "cyclical shared partner"

Outgoing Shared Partner (OSP)

vertex *k* is an OSP shared partner of ordered pair *(i,j)* iff *i->k, j->k*.

Incoming Shared Partner (ISP)

vertex *k* is an ISP shared partner of ordered pair *(i,j)* iff *k->i, k->j*.

Note that Robins et al. (2009) define closely related statistics to several of the above, using slightly different terminology.

`degrange(from, to=+Inf, by=NULL, homophily=FALSE)` (binary) (undirected) (categorical nodal attribute)

Degree range: The `from` and `to` arguments are vectors of distinct integers (or `+Inf`, for `to` (its default)). If one of the vectors has length 1, it is recycled to the length of the other. Otherwise, they must have the same length. This term adds one network statistic to the model for each element of `from` (or `to`); the *i*th such statistic equals the number of nodes in the network of degree greater than or equal to `from[i]` but strictly less than `to[i]`, i.e. with edges in semiopen interval `[from,to)`. The optional argument `by` is a character string giving the name of an attribute in the network's vertex attribute list. If this is specified and `homophily` is TRUE, then degrees are calculated using the subnetwork consisting of only edges whose endpoints have the same value of the `by` attribute. If `by` is specified and `homophily` is FALSE (the default), then separate degree range statistics are calculated for nodes having each separate value of the attribute.

This term can only be used with undirected networks; for directed networks see `idegrange` and `odegrange`. This term can be used with bipartite networks, and will count nodes of both first and second mode in the specified degree range. To count only nodes of the first mode ("actors"), use `b1degrange` and to count only those fo the second mode ("events"), use `b2degrange`.

`degree(d, by=NULL, homophily=FALSE)` (binary) (undirected) (categorical nodal attribute) (frequently-used)

Degree: The `d` argument is a vector of distinct integers. This term adds one network statistic to the model for each element in `d`; the *i*th such statistic equals the number of nodes in the network of degree `d[i]`, i.e. with exactly `d[i]` edges. The optional argument `by` is a character string giving the name of an attribute in the network's vertex attribute list. If this is specified and `homophily` is TRUE, then degrees are calculated using the subnetwork consisting of only edges whose endpoints have the same value of the `by` attribute. If `by` is specified and `homophily` is FALSE (the default), then separate degree statistics are calculated for nodes having each separate value of the attribute. This term can only be used with undirected networks; for directed networks see `idegree` and `odegree`.

`degree1.5` (binary) (undirected)

Degree to the 3/2 power: This term adds one network statistic to the model equaling the sum over the actors of each actor's degree taken to the 3/2 power (or, equivalently, multiplied by its square root). This term is an undirected analog to the terms of Snijders et al. (2010), equations (11) and (12). This term can only be used with undirected networks.

`degreepopularity` (binary) (undirected) (deprecated)

Degree popularity (deprecated): see `degree1.5`.

`degcrossprod` (binary) (undirected)

Degree Cross-Product: This term adds one network statistic equal to the mean of the cross-products of the degrees of all pairs of nodes in the network which are tied. Only coded for undirected networks.

`degcor` (binary) (undirected)

Degree Correlation: This term adds one network statistic equal to the correlation of the degrees of all pairs of nodes in the network which are tied. Only coded for undirected networks.

`density` (binary) (dyad-independent) (directed) (undirected)

Density: This term adds one network statistic equal to the density of the network. For undirected networks, `density` equals `kstar(1)` or `edges` divided by *n(n-1)/2*; for directed networks, `density` equals `edges` or `istar(1)` or `ostar(1)` divided by *n(n-1)*.

`diff(attrname, pow=1, dir="t-h", sign.action="identity")` (binary) (dyad-independent) (frequently-used) (directed) (undirected) (quantitative nodal attribute), `diff(attrname, pow=1, dir="t-h", sign.action="identity", form ="sum")` (valued) (dyad-independent) (directed) (undirected) (quantitative nodal attribute)

Difference: The `attrname` argument is a character string giving the name of a quantitative attribute in the network's vertex attribute list. For values of `pow` other than `0`, this term adds one network statistic to the model, equaling the sum, over directed edges *(i,j)*, of `sign.action(attrname[i]-attrname[j])^pow` if `dir` is `"t-h"` (the default), `"tail-head"`, or `"b1-b2"` and of `sign.action(attrname[j]-attrname[i])^pow` if `"t-h"`, `"tail-head"`, or `"b2-b1"`. That is, the argument `dir` determines which vertex's attribute is subtracted from which, with tail being the origin of a directed edge and head being its destination, and bipartite networks' edges being treated as going from the first part (b1) to the second (b2).

If `pow==0`, the exponentiation is replaced by the signum function: `+1` if the difference is positive, `0` if there is no difference, and `-1` if the difference is negative. Note that this function is applied *after* the `sign.action`. The comparison is exact, so when using calculated values of `attrname`, ensure that values that you want to be considered equal are, in fact, equal.

The following `sign.actions` are possible:

`"identity"` (the default)

no transformation of the difference regardless of sign

`"abs"`

absolute value of the difference: equivalent to the `absdiff` term

`"posonly"`

positive differences are kept, negative differences are replaced by 0

`"negonly"`

negative differences are kept, positive differences are replaced by 0

Note that this term may not be meaningful for unipartite undirected networks unless `sign.action=="abs"`. When used on such a network, it behaves as if all edges were directed, going from the lower-indexed vertex to the higher-indexed vertex.

`desp(d, type="OTP")` (binary) (directed)

Directed edgewise shared partners: This term adds one network statistic to the model for each element in `d` where the *i*th such statistic equals the number of *edges* in the network with exactly `d[i]` shared partners. This term can only be used with directed networks. Multiple shared partner definitions are possible; the `type` argument may be used to select the type of shared partner to be counted (see `ddsp` for type codes). By default, outgoing two-paths are employed.

`dgwdsp(decay=0, fixed=FALSE, cutoff=30, type="OTP")` (binary) (directed)

Geometrically weighted dyadwise shared partner distribution: This term adds one network statistic to the model equal to the geometrically weighted dyadwise shared partner distribution with decay parameter `decay` parameter, which should be non-negative. (this parameter was called `alpha` prior to `ergm 3.7`). The value supplied for this parameter may be fixed (if `fixed=TRUE`), or it may be used instead as the starting value for the estimation of `decay` in a curved exponential family model (when `fixed=FALSE`, the default) (see Hunter and Handcock, 2006). Note that the GWDSP statistic is equal to the sum of GWNSP plus GWESP. For a directed network, multiple shared partner definitions are possible; the `type` argument may be used to select the type of shared partner to employ (see `ddsp` for definitions). By default, outgoing two-paths are employed. The optional argument `cutoff` sets the number of underlying DSP terms to use in computing the statistics to reduce the computational burden.

`dgwesp(decay=0, fixed=FALSE, cutoff=30, type="OTP")` (binary) (directed)

Geometrically weighted edgewise shared partner distribution: This term adds a statistic equal to the geometrically weighted *edgewise* (not dyadwise) shared partner distribution with decay parameter `decay` parameter, which should be non-negative. (this parameter was called `alpha` prior to `ergm 3.7`). The value supplied for this parameter may be fixed (if `fixed=TRUE`), or it may be used instead as the starting value for the estimation of `decay` in a curved exponential family model (when `fixed=FALSE`, the default) (see Hunter and Handcock, 2006). For a directed network, multiple shared partner definitions are possible; the `type` argument may be used to select the type of shared partner to employ (see `ddsp` for definitions). By default, outgoing two-paths are employed. The optional argument `cutoff` sets the number of underlying ESP terms to use in computing the statistics to reduce the computational burden.

`dgwnsp(decay=0, fixed=FALSE, cutoff=30, type="OTP")` (binary) (directed)

Geometrically weighted non-edgewise shared partner distribution: This term is just like gwesp and gwdsp except it adds a statistic equal to the geometrically weighted nonedgewise (that is, over dyads that do not have an edge) shared partner distribution with decay parameter `decay` parameter, which should be non-negative. (this parameter was called `alpha` prior to `ergm 3.7`). The value supplied for this parameter may be fixed (if `fixed=TRUE`), or it may be used instead as the starting value for the estimation of `decay` in a curved exponential family model (when `fixed=FALSE`, the default) (see Hunter and Handcock, 2006). For a directed network, multiple shared partner definitions are possible; the `type` argument may be used to select the type of shared partner to employ (see `ddsp` for definitions). By default, outgoing two-paths are employed. The optional argument `cutoff` sets the number of underlying NSP terms to use in computing the statistics to reduce the computational burden.

`dnsp(d, type="OTP")` (binary) (directed)

Directed non-edgewise shared partners: This term adds one network statistic to the model for each element in `d` where the *i*th such statistic equals the number of *non-edges* in the network with exactly `d[i]` shared partners. This term can only be used with directed networks. Multiple shared partner definitions are possible; the `type` argument may be used to select the type of shared partner to be counted (see `ddsp` for type codes). By default, outgoing two-paths are employed.

`dsp(d)` (binary) (directed) (undirected)

Dyadwise shared partners: The `d` argument is a vector of distinct integers. This term adds one network statistic to the model for each element in `d`; the *i*th such statistic equals the number of dyads in the network with exactly `d[i]` shared partners. This term can be used with directed and undirected networks. For directed networks the count is over homogeneous shared partners only (i.e., only partners on a directed two-path connecting the nodes in the dyad).

`dyadcov(x, attrname=NULL)` (binary) (dyad-independent) (directed) (undirected) (categorical nodal attribute)

Dyadic covariate: The `x` argument is either a square matrix of covariates, one for each possible edge in the network, the name of a network attribute of covariates, or a network; if the latter, optional argument `attrname` provides the name of the quantitative edge attribute to use for covariate values (in this case, missing edges in `x` are assigned a covariate value of zero). This term adds three statistics to the model, each equal to the sum of the covariate values for all dyads occupying one of the three possible non-empty dyad states (mutual, upper-triangular asymmetric, and lower-triangular asymmetric dyads, respectively), with the empty or null state serving as a reference category. If the network is undirected, `x` is either a matrix of edgewise covariates, or a network; if the latter, optional argument `attrname` provides the name of the edge attribute to use for edge values. This term adds one statistic to the model, equal to the sum of the covariate values for each edge appearing in the network. The `edgecov` and `dyadcov` terms are equivalent for undirected networks.

`edgecov(x, attrname=NULL)` (binary) (dyad-independent) (directed) (undirected) (frequently-used) , `edgecov(x, attrname=NULL,` `form="sum")` (valued) (directed) (undirected) (dyad-independent)

> *Edge covariate:* The x argument is either a square matrix of covariates, one for each possible edge in the network, the name of a network attribute of covariates, or a network; if the latter, optional argument `attrname` provides the name of the quantitative edge attribute to use for covariate values (in this case, missing edges in x are assigned a covariate value of zero). This term adds one statistic to the model, equal to the sum of the covariate values for each edge appearing in the network. The edgecov term applies to both directed and undirected networks. For undirected networks the covariates are also assumed to be undirected. The edgecov and dyadcov terms are equivalent for undirected networks.

`edges` (binary) (valued) (dyad-independent) (directed) (undirected) (frequently-used) , a.k.a `nonzero` (valued) (directed) (undirected) (dyad-independent)

> *Edges:* This term adds one network statistic equal to the number of edges (i.e. nonzero values) in the network. For undirected networks, `edges` is equal to `kstar(1)`; for directed networks, `edges` is equal to both `ostar(1)` and `istar(1)`.

`esp(d)` (binary) (directed) (undirected)

> *Edgewise shared partners:* This is just like the `dsp` term, except this term adds one network statistic to the model for each element in `d` where the *i*th such statistic equals the number of *edges* (rather than dyads) in the network with exactly `d[i]` shared partners. This term can be used with directed and undirected networks. For directed networks the count is over homogeneous shared partners only (i.e., only partners on a directed two-path connecting the nodes in the edge and in the same direction).

`equalto(value=0, tolerance=0)` (valued) (directed) (undirected) (dyad-independent)

> *Number of dyads with values equal to a specific value (within tolerance):* Adds one statistic equal to the number of dyads whose values are within `tolerance` of `value`, i.e., between `value-tolerance` and `value+tolerance`, inclusive.

`greaterthan(threshold=0)` (valued) (directed) (undirected) (dyad-independent)

> *Number of dyads with values strictly greater than a threshold:* Adds one statistic equal to the number of dyads whose values exceed `threshold`.

`gwb1degree(decay, fixed=FALSE, cutoff=30)` (binary) (bipartite) (undirected) (curved)

> *Geometrically weighted degree distribution for the first mode in a bipartite (aka two-mode) network:* This term adds one network statistic to the model equal to the weighted degree distribution with decay controlled by the decay parameter, which should be non-negative, for nodes in the first mode of a bipartite network. The first mode of a bipartite network object is sometimes known as the "actor" mode. The decay parameter is the same as theta_s in equation (14) in Hunter (2007). The value supplied for this parameter may be fixed (if `fixed=TRUE`), or it may be used as merely the starting value for the estimation in a curved exponential family model (the default). The optional argument `cutoff` is only relevant if `fixed=FALSE`. In that case it only uses this number of terms in computing the statistics to reduce the computational burden. This term can only be used with undirected bipartite networks.

`gwb2degree(decay, fixed=FALSE, cutoff=30)` (binary) (bipartite) (undirected) (curved)

> *Geometrically weighted degree distribution for the second mode in a bipartite (aka two-mode) network:* This term adds one network statistic to the model equal to the weighted degree distribution with decay controlled by the which should be non-negative, for nodes in the second mode of a bipartite network. The second mode of a bipartite network object is sometimes known as the "event" mode. The decay parameter is the same as theta_s in equation (14) in Hunter (2007). The value supplied for this parameter may be fixed (if `fixed=TRUE`), or it may be used as merely the starting value for the estimation in a curved exponential family model (the default). The optional argument `cutoff` is only relevant if `fixed=FALSE`. In that case it only uses this number of terms in computing the statistics to reduce the computational burden. This term can only be used with undirected bipartite networks.

`gwdegree(decay, fixed=FALSE, cutoff=30)` (binary) (undirected) (curved) (frequently-used)

> *Geometrically weighted degree distribution:* This term adds one network statistic to the model equal to the weighted degree distribution with decay controlled by the decay parameter. The decay parameter is the same as theta_s in equation (14) in Hunter (2007). The value supplied for this parameter may be fixed (if `fixed=TRUE`), or it may be used instead as the starting value for the estimation of decay in a curved exponential family model (when `fixed=FALSE`, the default) (see Hunter and Handcock, 2006). The optional argument `cutoff` is only relevant if `fixed=FALSE`. In that case it only uses this number of terms in computing the statistics to reduce the computational burden. This term can only be used with undirected networks.

`gwdsp(decay=0, fixed=FALSE, cutoff=30)` (binary) (directed) (undirected) (curved)

> *Geometrically weighted dyadwise shared partner distribution:* This term adds one network statistic to the model equal to the geometrically weighted dyadwise shared partner distribution with decay parameter decay parameter, which should be non-negative. The value supplied for this parameter may be fixed (if `fixed=TRUE`), or it may be used instead as the starting value for the estimation of decay in a curved exponential family model (when `fixed=FALSE`, the default) (see Hunter and Handcock, 2006). For directed networks the count is over homogeneous shared partners only (i.e., only partners on a directed two-path connecting the nodes in the dyad). The optional argument `cutoff` is only relevant if `fixed=FALSE`. In that case it only uses this number of terms in computing the statistics to reduce the computational burden.

`gwesp(decay=0, fixed=FALSE, cutoff=30)` (binary) (frequently-used) (directed) (undirected) (curved)

> *Geometrically weighted edgewise shared partner distribution:* This term is just like `gwdsp` except it adds a statistic equal to the geometrically weighted *edgewise* (not dyadwise) shared partner distribution with decay parameter decay parameter, which should be non-negative. The value supplied for this parameter may be fixed (if `fixed=TRUE`), or it may be used instead as the starting value for the estimation of decay in a curved exponential family model (when `fixed=FALSE`, the default) (see Hunter and Handcock, 2006). This term can be used with directed and undirected networks. For directed networks the geometric weighting is over homogeneous shared partners only (i.e., only partners on a directed two-path connecting the nodes in the edge and in the same direction). The optional argument `cutoff` is only relevant if `fixed=FALSE`. In that case it only uses this number of terms in computing the statistics to reduce the computational burden.

`gwidegree(decay, fixed=FALSE, cutoff=30)` (binary) (directed) (curved)

> *Geometrically weighted in-degree distribution:* This term adds one network statistic to the model equal to the weighted in-degree distribution with decay parameter decay parameter, which should be non-negative. (this parameter was called `alpha` prior to `ergm 3.7`). The value supplied for this parameter may be fixed (if `fixed=TRUE`), or it may be used instead as the starting value for the estimation of decay in a curved exponential family model (when `fixed=FALSE`, the default) (see Hunter and Handcock, 2006). This term can only be used with directed networks. The optional argument `cutoff` is only relevant if `fixed=FALSE`. In that case it only uses this number of terms in computing the statistics to reduce the computational burden.

`gwnsp(decay=0, fixed=FALSE, cutoff=30)` (binary) (directed) (undirected) (curved)

Geometrically weighted nonedgewise shared partner distribution: This term is just like `gwesp` and `gwdsp` except it adds a statistic equal to the geometrically weighted *nonedgewise* (that is, over dyads that do not have an edge) shared partner distribution with weight parameter decay parameter, which should be non-negative. (this parameter was called `alpha` prior to `ergm 3.7`). The optional argument `fixed` indicates whether the decay parameter is fixed at the given value, or is to be fit as a curved exponential-family model (see Hunter and Handcock, 2006). The default is `FALSE`, which means the scale parameter is not fixed and thus the model is a CEF model. This term can be used with directed and undirected networks. For directed networks the geometric weighting is over homogeneous shared partners only (i.e., only partners on a directed two-path connecting the nodes in the non-edge and in the same direction). The optional argument `cutoff` is only relevant if `fixed=FALSE`. In that case it only uses this number of terms in computing the statistics to reduce the computational burden.

`gwodegree(decay, fixed=FALSE, cutoff=30)` (binary) (directed) (curved)

Geometrically weighted out-degree distribution: This term adds one network statistic to the model equal to the weighted out-degree distribution with decay parameter `decay` parameter, which should be non-negative. (this parameter was called `alpha` prior to `ergm 3.7`). The value supplied for this parameter may be fixed (if `fixed=TRUE`), or it may be used instead as the starting value for the estimation of `decay` in a curved exponential family model (when `fixed=FALSE`, the default) (see Hunter and Handcock, 2006). This term can only be used with directed networks. The optional argument `cutoff` is only relevant if `fixed=FALSE`. In that case it only uses this number of terms in computing the statistics to reduce the computational burden.

`hamming(x, cov, attrname=NULL)` (binary) (dyad-independent) (directed) (undirected)

Hamming distance: This term adds one statistic to the model equal to the weighted or unweighted Hamming distance of the network from the network specified by `x`. (If no argument is given, `x` is taken to be the observed network, i.e., the network on the left side of the ~ in the formula that defines the ERGM.) Unweighted Hamming distance is defined as the total number of pairs *(i,j)* (ordered or unordered, depending on whether the network is directed or undirected) on which the two networks differ. If the optional argument `cov` is specified, then the weighted Hamming distance is computed instead, where each pair *(i,j)* contributes a pre-specified weight toward the distance when the two networks differ on that pair. The argument `cov` is either a matrix of edgewise weights or a network; if the latter, the optional argument `attrname` provides the name of the edge attribute to use for weight values.

`hammingmix(attrname, x, base=0)` (binary) (directed) (dyad-independent)

Hamming distance within mixing: This term adds one statistic to the model for every possible pairing of attribute values of the network for the vertex attribute named `attrname`. Each such statistic is the Hamming distance (i.e., the number of differences) between the appropriate subset of dyads in the network and the corresponding subset in `x`. The ordering of the attribute values is alphabetical. The option `base` gives the index of statistics to be omitted from the tabulation. For example `base=2` will omit the second statistic, making it the de facto reference category. This term can only be used with directed networks.

`idegrange(from, to=+Inf, by=NULL, homophily=FALSE)` (binary) (directed) (categorical nodal attribute)

In-degree range: The `from` and `to` arguments are vectors of distinct integers (or `+Inf`, for `to` (its default)). If one of the vectors has length 1, it is recycled to the length of the other. Otherwise, they must have the same length. This term adds one network statistic to the model for each element of `from` (or `to`); the *i*th such statistic equals the number of nodes in the network of in-degree greater than or equal to `from[i]` but strictly less than `to[i]`, i.e. with in-edge count in semiopen interval `[from,to)`. The optional argument `by` is a character string giving the name of an attribute in the network's vertex attribute list. If this is specified and `homophily` is TRUE, then degrees are calculated using the subnetwork consisting of only edges whose endpoints have the same value of the `by` attribute. If `by` is specified and `homophily` is FALSE (the default), then separate degree range statistics are calculated for nodes having each separate value of the attribute.

This term can only be used with directed networks; for undirected networks (bipartite and not) see `degrange`. For degrees of specific modes of bipartite networks, see `b1degrange` and `b2degrange`. For in-degrees, see `idegrange`.

`idegree(d, by=NULL, homophily=FALSE)` (binary) (directed) (categorical nodal attribute) (frequently-used)

In-degree: The `d` argument is a vector of distinct integers. This term adds one network statistic to the model for each element in `d`; the *i*th such statistic equals the number of nodes in the network of in-degree `d[i]`, i.e. the number of nodes with exactly `d[i]` in-edges. The optional term `by` is a character string giving the name of an attribute in the network's vertex attribute list. If this is specified and `homophily` is TRUE, then degrees are calculated using the subnetwork consisting of only edges whose endpoints have the same value of the `by` attribute. If `by` is specified and `homophily` is FALSE (the default), then separate degree statistics are calculated for nodes having each separate value of the attribute. This term can only be used with directed networks; for undirected networks see `degree`.

`idegree1.5` (binary) (directed)

In-degree to the 3/2 power: This term adds one network statistic to the model equaling the sum over the actors of each actor's indegree taken to the 3/2 power (or, equivalently, multiplied by its square root). This term is analogous to the term of Snijders et al. (2010), equation (12). This term can only be used with directed networks.

`idegreepopularity` (binary) (directed) (deprecated)

In-degree popularity (deprecated): see `idegree1.5`.

`ininterval(lower=-Inf, upper=+Inf, open=c(TRUE,TRUE))` (valued) (directed) (undirected) (dyad-independent)

Number of dyads whose values are in an interval Adds one statistic equaling to the number of dyads whose values are between `lower` and `upper`. Argument `open` is a `logical` vector of length 2 that controls whether the interval is open (exclusive) on the lower and on the upper end, respectively.

`intransitive` (binary) (directed) (triad-related)

Intransitive triads: This term adds one statistic to the model, equal to the number of triads in the network that are intransitive. The intransitive triads are those of type `111D`, `201`, `111U`, `021C`, or `030C` in the categorization of Davis and Leinhardt (1972). For details on the 16 possible triad types, see `triad.classify` in the sna package. Note the distinction from the `ctriple` term. This term can only be used with directed networks.

`isolates` (binary) (directed) (undirected) (frequently-used)

Isolates: This term adds one statistic to the model equal to the number of isolates in the network. For an undirected network, an isolate is defined to be any node with degree zero. For a directed network, an isolate is any node with both in-degree and out-degree equal to zero.

`istar(k, attrname=NULL)` (binary) (directed) (categorical nodal attribute)

In-stars: The `k` argument is a vector of distinct integers. This term adds one network statistic to the model for each element in `k`. The *i*th such statistic counts the number of distinct `k[i]`-instars in the network, where a *k*-instar is defined to be a node *N* and a set of *k* different nodes *\{O_1, …, O_k\}* such that the ties *(O_j, N)* exist for *j=1, …, k*. The optional argument `attrname` is a character string giving the name of an attribute in the network's vertex attribute list. If this is specified then the count is over the number of *k*-instars where all nodes have the same value of the attribute. This term can only be used for directed networks; for undirected networks see `kstar`. Note that `istar(1)` is equal to both `ostar(1)` and `edges`.

`kstar(k, attrname=NULL)` (binary) (undirected) (categorical nodal attribute)

> *k-Stars:* The `k` argument is a vector of distinct integers. This term adds one network statistic to the model for each element in `k`. The *i*th such statistic counts the number of distinct `k[i]`-stars in the network, where a *k*-star is defined to be a node *N* and a set of *k* different nodes *\{O_1, ..., O_k\}* such that the ties *\{N, O_i\}* exist for *i=1, ..., k*. The optional argument `attrname` is a character string giving the name of an attribute in the network's vertex attribute list. If this is specified then the count is over the number of *k*-stars where all nodes have the same value of the attribute. This term can only be used for undirected networks; for directed networks, see `istar`, `ostar`, `twopath` and `m2star`. Note that `kstar(1)` is equal to `edges`.

`smallerthan(threshold=0)` (valued) (directed) (undirected) (dyad-independent)

> *Number of dyads with values strictly smaller than a threshold:* Adds one statistic equaling to the number of dyads whose values exceeded by `threshold`.

`localtriangle(x)` (binary) (triad-related) (directed) (undirected)

> *Triangles within neighborhoods:* This term adds one statistic to the model equal to the number of triangles in the network between nodes “close to” each other. For an undirected network, a local triangle is defined to be any set of three edges between nodal pairs *\{(i,j), (j,k), (k,i)\}* that are in the same neighborhood. For a directed network, a triangle is defined as any set of three edges *(i,j), (j,k)* and either *(k{\rightarrow}i)* or *(k{≤ftarrow}i)* where again all nodes are within the same neighborhood. The argument `x` is an undirected network or an symmetric adjacency matrix that specifies whether the two nodes are in the same neighborhood. Note that `triangle`, with or without an argument, is a special case of `localtriangle`.

`m2star` (binary) (directed)

> *Mixed 2-stars, a.k.a 2-paths:* This term adds one statistic to the model, equal to the number of mixed 2-stars in the network, where a mixed 2-star is a pair of distinct edges *(i,j), (j,k)*. A mixed 2-star is sometimes called a 2-path because it is a directed path of length 2 from *i* to *k* via *j*. However, in the case of a 2-path the focus is usually on the endpoints *i* and *k*, whereas for a mixed 2-star the focus is usually on the midpoint *j*. This term can only be used with directed networks; for undirected networks see `kstar(2)`. See also `twopath`.

`meandeg` (binary) (dyad-independent) (directed) (undirected)

> *Mean vertex degree:* This term adds one network statistic to the model equal to the average degree of a node. Note that this term is a constant multiple of both `edges` and `density`.

`mutual(same=NULL, diff=FALSE, by=NULL, keep=NULL)` (binary) (directed) (frequently-used), `mutual(form="min",threshold=0)` (valued) (directed)

> *Mutuality:* In binary ERGMs, equal to the number of pairs of actors *i* and *j* for which *(i,j)* and *(j,i)* both exist. For valued ERGMs, equal to *∑_{i<j} m(y_{i,j},y_{j,i})*, where *m* is determined by `form` argument: `"min"` for *\min(y_{i,j},y_{j,i})*, `"nabsdiff"` for *-|y_{i,j},y_{j,i}|*, `"product"` for *y_{i,j}y_{j,i}*, and `"geometric"` for *√{y_{i,j}}√{y_{j,i}}*. See Krivitsky (2012) for a discussion of these statistics. `form="threshold"` simply computes the binary `mutuality` after thresholding at `threshold`.
>
> This term can only be used with directed networks. The binary version also has the following capabilities: if the optional `same` argument is passed the name of a vertex attribute, only mutual pairs that match on the attribute are counted; separate counts for each unique matching value can be obtained by using `diff`=TRUE with `same`; and if `by` is passed the name of a vertex attribute, then each node is counted separately for each mutual pair in which it occurs and the counts are tabulated by unique values of the attribute. This means that the sum of the mutual statistics when `by` is used will equal twice the standard mutual statistic. Only one of `same` or `by` may be used, and only the former is affected by `diff`; if both `same` and `by` are passed, `by` is ignored. Finally, if `keep` is passed a numerical vector, this vector of integers tells which statistics should be kept whenever the `mutual` term would ordinarily result in multiple statistics.

`nearsimmelian` (binary) (directed) (triad-related)

> *Near simmelian triads:* This term adds one statistic to the model equal to the number of near Simmelian triads, as defined by Krackhardt and Handcock (2007). This is a sub-graph of size three which is exactly one tie short of being complete. This term can only be used with directed networks.

`nodecov(attrname, transform, transformname)` (binary) (dyad-independent) (frequently-used) (directed) (undirected) (quantitative nodal attribute) , `nodecov(attrname, transform, transformname, form="sum")` (valued) (dyad-independent) (directed) (undirected) (quantitative nodal attribute) , a.k.a. `nodemain` (binary) (directed) (undirected)

> *Main effect of a covariate:* The `attrname` argument is a character string giving the name of a numeric (not categorical) attribute in the network's vertex attribute list. This term adds a single network statistic to the model equaling the sum of `attrname(i)` and `attrname(j)` for all edges *(i,j)* in the network. For categorical attributes, see `nodefactor`. Note that for directed networks, `nodecov` equals `nodeicov` plus `nodeocov`.

`nodecovar` (valued) (directed) (undirected) (quantitative nodal attribute)

> *Uncentered covariance of dyad values incident on each actor:* This term adds one statistic equal to ∑*_{i,j,k} (y_{i,j}y_{i,k}+y_{k,j}y_{k,j})*. This can be viewed as a valued analog of the `kstar(2)` statistic.

`nodefactor(attrname, base=1)` (binary) (dyad-independent) (directed) (undirected) (categorical nodal attribute) (frequently-used) , `nodefactor(attrname, base=1, form="sum")` (dyad-independent) (valued) (directed) (undirected) (categorical nodal attribute)

> *Factor attribute effect:* The `attrname` argument is a character vector giving one or more names of categorical attributes in the network's vertex attribute list. This term adds multiple network statistics to the model, one for each of (a subset of) the unique values of the `attrname` attribute (or each combination of the attributes given). Each of these statistics gives the number of times a node with that attribute or those attributes appears in an edge in the network. In particular, for edges whose endpoints both have the same attribute values, this value is counted twice. To include all attribute values is usually not a good idea – though this may be accomplished if desired by setting `base=0` – because the sum of all such statistics equals twice the number of edges and hence a linear dependency would arise in any model also including `edges`. Thus, the `base` argument tells which value(s) (numbered in order according to the `sort` function) should be omitted. The default value, `base=1`, means that the smallest (i.e., first in sorted order) attribute value is omitted. For example, if the “fruit” factor has levels “orange”, “apple”, “banana”, and “pear”, then to add just two terms, one for “apple” and one for “pear”, then set “banana” and “orange” to the base (remember to sort the values first) by using `nodefactor("fruit", base=2:3)`. For an analogous term for quantitative vertex attributes, see `nodecov`.

`nodeicov(attrname, transform, transformname)` (binary) (directed) (quantitative nodal attribute) (frequently-used) , `nodeicov(attrname, transform, transformname, form="sum")` (valued) (directed) (quantitative nodal attribute)

Main effect of a covariate for in-edges: The `attrname` argument is a character string giving the name of a numeric (not categorical) attribute in the network's vertex attribute list. This term adds a single network statistic to the model equaling the total value of `attrname(j)` for all edges *(i,j)* in the network. This term may only be used with directed networks. For categorical attributes, see `nodeifactor`.

`nodeicovar` (valued) (directed) (quantitative nodal attribute)

Uncentered covariance of in-dyad values incident on each actor: This term adds one statistic equal to $\sum_{i,j,k} y_{k,j}y_{k,j}$. This can be viewed as a valued analog of the `istar(2)` statistic.

`nodeifactor(attrname, base=1)` (binary) (dyad-independent) (directed) (categorical nodal attribute) (frequently-used) ,
`nodeifactor(attrname, base=1, form="sum")` (valued) (dyad-independent) (directed) (categorical nodal attribute)

Factor attribute effect for in-edges: The `attrname` argument is a character vector giving one or more names of a categorical attribute in the network's vertex attribute list. This term adds multiple network statistics to the model, one for each of (a subset of) the unique values of the `attrname` attribute (or each combination of the attributes given). Each of these statistics gives the number of times a node with that attribute or those attributes appears as the terminal node of a directed tie. To include all attribute values is usually not a good idea – though this may be accomplished if desired by setting `base=0` – because the sum of all such statistics equals the number of edges and hence a linear dependency would arise in any model also including `edges`. Thus, the base argument tells which value(s) (numbered in order according to the `sort` function) should be omitted. The default value, base=1, means that the smallest (i.e., first in sorted order) attribute value is omitted. For example, if the “fruit” factor has levels “orange”, “apple”, “banana”, and “pear”, then to add just two terms, one for “apple” and one for “pear”, then set “banana” and “orange” to the base (remember to sort the values first) by using `nodefactor("fruit", base=2:3)`. For an analogous term for quantitative vertex attributes, see `nodeicov`.

`nodeisqrtcovar` (valued) (directed) (non-negative) (quantitative nodal attribute)

Uncentered covariance of square roots of in-dyad values incident on each actor: This term adds one statistic equal to $\sum_{i,j,k} \sqrt{y_{i,j}} \sqrt{y_{k,j}}$. This can be viewed as a valued analog of the `istar(2)` statistic.

`nodematch(attrname, diff=FALSE, keep=NULL)` (binary) (dyad-independent) (frequently-used) (directed) (undirected) (categorical nodal attribute) , `nodematch(attrname, diff=FALSE, keep=NULL, form="sum")` (valued) (dyad-independent) (directed) (undirected) (categorical nodal attribute) a.k.a. `match` (binary) (directed) (dyad-independent) (undirected) (categorical nodal attribute)

Uniform homophily and differential homophily: The `attrname` argument is a character vector giving one or more names of attributes in the network's vertex attribute list. When `diff=FALSE`, this term adds one network statistic to the model, which counts the number of edges *(i,j)* for which `attrname(i)==attrname(j)`. This is also called ”uniform homophily,” because each group is assumed to have the same propensity for within-group ties. When multiple attribute names are given, the statistic counts only ties for which all of the attributes match. When `diff=TRUE`, *p* network statistics are added to the model, where *p* is the number of unique values of the `attrname` attribute. The *k*th such statistic counts the number of edges *(i,j)* for which `attrname(i) == attrname(j) == value(k)`, where `value(k)` is the *k*th smallest unique value of the attrname attribute. This is also called ”differential homophily,” because each group is allowed to have a unique propensity for within-group ties. Note that a statistical test of uniform vs. differential homophily should be conducted using the ANOVA function.

If set to non-NULL, the optional keep argument should be a vector of integers giving the values of k that should be considered for matches; other values are ignored (this works for both `diff=FALSE` and `diff=TRUE`). For instance, to add two statistics, counting the matches for just the 2nd and 4th categories, use `nodematch` with `diff=TRUE` and `keep=c(2,4)`.

`nodemix(attrname, base=NULL)` (binary) (dyad-independent) (frequently-used) (directed) (undirected) (categorical nodal attribute) ,
`nodemix(attrname, base=NULL, form="sum")` (valued) (dyad-independent) (directed) (undirected) (categorical nodal attribute)

Nodal attribute mixing: The `attrname` argument is a character vector giving the names of categorical attributes in the network's vertex attribute list. By default, this term adds one network statistic to the model for each possible pairing of attribute values. The statistic equals the number of edges in the network in which the nodes have that pairing of values. (When multiple names are given, a statistic is added for each combination of attribute values for those names.) In other words, this term produces one statistic for every entry in the mixing matrix for the attribute(s). The ordering of the attribute values is alphabetical (for nominal categories) or numerical (for ordered categories). The optional `base` argument is a vector of integers corresponding to the pairings that should not be included. If `base` contains only negative integers, then these integers correspond to the only pairings that should be included. By default (i.e., with `base=NULL` or `base=0`), all pairings are included.

`nodeocov(attrname, transform, transformname)` (binary) (directed) (dyad-independent)(quantitative nodal attribute) ,
`nodeocov(attrname, transform, transformname, form="sum")` (valued) (directed) (dyad-independent) (quantitative nodal attribute)

Main effect of a covariate for out-edges: The `attrname` argument is a character string giving the name of a numeric (not categorical) attribute in the network's vertex attribute list. This term adds a single network statistic to the model equaling the total value of `attrname(i)` for all edges *(i,j)* in the network. This term may only be used with directed networks. For categorical attributes, see `nodeofactor`.

`nodeocovar` (valued) (directed) (quantitative nodal attribute)

Uncentered covariance of out-dyad values incident on each actor: This term adds one statistic equal to $\sum_{i,j,k} y_{i,j}y_{i,k}$. This can be viewed as a valued analog of the `ostar(2)` statistic.

`nodeofactor(attrname, base=1)` (binary) (dyad-independent) (directed) (categorical nodal attribute) , `nodeofactor(attrname, base=1, form="sum")` (valued) (dyad-independent) (categorical nodal attribute) (directed)

Factor attribute effect for out-edges: The `attrname` argument is a character string giving one or more names of categorical attributes in the network's vertex attribute list. This term adds multiple network statistics to the model, one for each of (a subset of) the unique values of the `attrname` attribute (or each combination of the attributes given). Each of these statistics gives the number of times a node with that attribute or those attributes appears as the node of origin of a directed tie. To include all attribute values is usually not a good idea – though this may be accomplished if desired by setting `base=0` – because the sum of all such statistics equals the number of edges and hence a linear dependency would arise in any model also including `edges`. Thus, the base argument tells which value(s) (numbered in order according to the `sort` function) should be omitted. The default value, base=1, means that the smallest (i.e., first in sorted order) attribute value is omitted. For example, if the “fruit” factor has levels “orange”, “apple”, “banana”, and “pear”, then to add just two terms, one for “apple” and one for “pear”, then set “banana” and “orange” to the base (remember to sort the values first) by using `nodefactor("fruit", base=2:3)`. For an analogous term for quantitative vertex attributes, see `nodeocov`.

`nodeosqrtcovar` (valued) (directed) (non-negative) (quantitative nodal attribute)

Uncentered covariance of square roots of out-dyad values incident on each actor: This term adds one statistic equal to $\sum_{i,j,k} \sqrt{y_{i,j}}\sqrt{y_{i,k}}$. This can be viewed as a valued analog of the `ostar(2)` statistic.

`nodesqrtcovar(center=TRUE)` (valued) (non-negative) (directed) (undirected) (quantitative nodal attribute)

Covariance of square roots of dyad values incident on each actor: This term adds one statistic equal to ∑_{i,j,k} (√{y_{i,j}} √{y_{i,k}}+√{y_{k,j}}√{y_{k,j}}) if `center=FALSE`. This can be viewed as a valued analog of the `kstar(2)` statistic. If `center=FALSE` (the default), the statistic is instead ∑_{i,j,k} ((√{y_{i,j}}-\bar{√{y}})(√{y_{i,k}}-\bar{√{y}})+(√{y_{k,j}}-\bar{√{y}})(√{y_{k,j}}-\bar{√{y}})), where \bar{√{y}} is the mean of the square root of dyad values.

`nsp(d)` (binary) (directed) (undirected)

Nonedgewise shared partners: This is just like the `dsp` and `esp` terms, except this term adds one network statistic to the model for each element in `d` where the *i*th such statistic equals the number of *non-edges* (that is, dyads that do not have an edge) in the network with exactly `d[i]` shared partners. This term can be used with directed and undirected networks. For directed networks the count is over homogeneous shared partners only (i.e., only partners on a directed two-path connecting the nodes in the non-edge and in the same direction).

`odegrange(from, to=+Inf, by=NULL, homophily=FALSE)` (binary) (directed) (categorical nodal attribute)

Out-degree range: The `from` and `to` arguments are vectors of distinct integers (or `+Inf`, for `to` (its default)). If one of the vectors has length 1, it is recycled to the length of the other. Otherwise, they must have the same length. This term adds one network statistic to the model for each element of `from` (or `to`); the *i*th such statistic equals the number of nodes in the network of out-degree greater than or equal to `from[i]` but strictly less than `to[i]`, i.e. with out-edge count in semiopen interval `[from,to)`. The optional argument by is a character string giving the name of an attribute in the network's vertex attribute list. If this is specified and `homophily` is TRUE, then degrees are calculated using the subnetwork consisting of only edges whose endpoints have the same value of the by attribute. If by is specified and `homophily` is FALSE (the default), then separate degree range statistics are calculated for nodes having each separate value of the attribute.

This term can only be used with directed networks; for undirected networks (bipartite and not) see `degrange`. For degrees of specific modes of bipartite networks, see `b1degrange` and `b2degrange`. For in-degrees, see `idegrange`.

`odegree(d, by=NULL, homophily=FALSE)` (binary) (directed) (categorical nodal attribute) (frequently-used)

Out-degree: The `d` argument is a vector of distinct integers. This term adds one network statistic to the model for each element in `d`; the *i*th such statistic equals the number of nodes in the network of out-degree `d[i]`, i.e. the number of nodes with exactly `d[i]` out-edges. The optional argument by is a character string giving the name of an attribute in the network's vertex attribute list. If this is specified and `homophily` is TRUE, then degrees are calculated using the subnetwork consisting of only edges whose endpoints have the same value of the by attribute. If by is specified and `homophily` is FALSE (the default), then separate degree statistics are calculated for nodes having each separate value of the attribute. This term can only be used with directed networks; for undirected networks see `degree`.

`odegree1.5` (binary) (directed)

Out-degree to the 3/2 power: This term adds one network statistic to the model equaling the sum over the actors of each actor's outdegree taken to the 3/2 power (or, equivalently, multiplied by its square root). This term is analogous to the term of Snijders et al. (2010), equation (12). This term can only be used with directed networks.

`odegreepopularity` (binary) (directed) (deprecated)

Out-degree popularity (deprecated): see `odegree1.5`.

`opentriad` (binary) (undirected) (triad-related)

Open triads: This term adds one statistic to the model equal to the number of 2-stars minus three times the number of triangles in the network. It is currently only implemented for undirected networks.

`ostar(k, attrname=NULL)` (binary) (directed) (categorical nodal attribute)

k-Outstars: The `k` argument is a vector of distinct integers. This term adds one network statistic to the model for each element in `k`. The *i*th such statistic counts the number of distinct `k[i]`-outstars in the network, where a *k*-outstar is defined to be a node *N* and a set of *k* different nodes *\{O_1, ..., O_k\}* such that the ties *(N,O_j)* exist for *j=1, ..., k*. The optional argument `attrname` is a character string giving the name of an attribute in the network's vertex attribute list. If this is specified then the count is the number of *k*-outstars where all nodes have the same value of the attribute. This term can only be used with directed networks; for undirected networks see `kstar`. Note that `ostar(1)` is equal to both `istar(1)` and `edges`.

`receiver(base=1)` (binary) (directed) (dyad-independent)

Receiver effect: This term adds one network statistic for each node equal to the number of in-ties for that node. This measures the popularity of the node. The term for the first node is omitted by default because of linear dependence that arises if this term is used together with `edges`, but its coefficient can be computed as the negative of the sum of the coefficients of all the other actors. That is, the average coefficient is zero, following the Holland-Leinhardt parametrization of the p_1 model (Holland and Leinhardt, 1981). The `base` argument allows the user to determine which nodes' statistics should be omitted. The `base` argument can also be a vector of negative indices, to specify which should be added instead of deleted, and `base=0` specifies that all statistics should be included. This term can only be used with directed networks. For undirected networks, see `sociality`.

`sender(base=1)` (binary) (directed) (dyad-independent)

Sender effect: This term adds one network statistic for each node equal to the number of out-ties for that node. This measures the activity of the node. The term for the first node is omitted by default because of linear dependence that arises if this term is used together with `edges`, but its coefficient can be computed as the negative of the sum of the coefficients of all the other actors. That is, the average coefficient is zero, following the Holland-Leinhardt parametrization of the p_1 model (Holland and Leinhardt, 1981). The `base` argument allows the user to determine which nodes' statistics should be omitted. The `base` argument can also be a vector of negative indices, to specify which should be added instead of deleted, and `base=0` specifies that all statistics should be included. This term can only be used with directed networks. For undirected networks, see `sociality`.

`simmelian` (binary) (directed) (triad-related)

Simmelian triads: This term adds one statistic to the model equal to the number of Simmelian triads, as defined by Krackhardt and Handcock (2007). This is a complete sub-graph of size three. This term can only be used with directed networks.

`simmelianties` (binary) (triad-related) (directed)

Ties in simmelian triads: This term adds one statistic to the model equal to the number of ties in the network that are associated with Simmelian triads, as defined by Krackhardt and Handcock (2007). Each Simmelian has six ties in it but, because Simmelians can overlap in terms of nodes (and associated ties), the total number of ties in these Simmelians is less than six times the number of Simmelians. Hence this is a measure of the clustering of Simmelians (given the number of Simmelians). This term can only be used with directed networks.

`smalldiff(attrname, cutoff)` (binary) (dyad-independent) (directed) (undirected) (quantitative nodal attribute)

Number of ties between actors with similar (but not necessarily identical) attribute values: The `attrname` argument is a character string giving the name of a quantitative attribute in the network's vertex attribute list. This term adds one statistic, having as its value the number of edges in the network for which the incident actors' attribute values differ less than `cotoff`; that is, number of edges between `i` to `j` such that `abs(attrname[i]-attrname[j])<cutoff`.

`sociality(attrname=NULL, base=1)` (binary) (undirected) (categorical nodal attribute)

Undirected degree: This term adds one network statistic for each node equal to the number of ties of that node. The optional `attrname` argument is a character string giving the name of an attribute in the network's vertex attribute list that takes categorical values. If provided, this term only counts ties between nodes with the same value of the attribute (an actor-specific version of the `nodematch` term). This term can only be used with undirected networks. For directed networks, see `sender` and `receiver`. By default, `base=1` means that the statistic for the first node will be omitted, but this argument may be changed to control which statistics are included just as for the `sender` and `receiver` terms.

`sum(pow=1)` (valued) (directed) (undirected)

Sum of dyad values (optionally taken to a power): This term adds one statistic equal to the sum of dyad values taken to the power `pow`, which defaults to 1.

`threetrail(keep=1:4)` (binary) (directed) (undirected) (triad-related),

Three-trails: a.k.a. `threepath`. For an undirected network, this term adds one statistic equal to the number of 3-trails, where a 3-trail is defined as a “trail” of length three that traverses three distinct edges. Note that a 3-trail need not include four distinct nodes; in particular, a triangle counts as three 3-trails. For a directed network, this term adds four statistics (or some subset of these four specified by the keep argument), one for each of the four distinct types of directed three-paths. If the nodes of the path are written from left to right such that the middle edge points to the right (R), then the four types are RRR, RRL, LRR, and LRL. That is, an RRR 3-trail is of the form *i-->j-->k-->l*, and RRL 3-trail is of the form *i-->j-->k<--l*, etc. Like in the undirected case, there is no requirement that the nodes be distinct in a directed 3-trail. However, the three edges must all be distinct. Thus, a mutual tie *i<-->j* does not count as a 3-trail of the form *i-->j-->i<--j*; however, in the subnetwork *i<-->j-->k*, there are two directed 3-trails, one LRR (*k<--j-->i-->j*) and one RRR (*k<--j-->i-->j*).

This term used to be (inaccurately) called `threepath`. That name has been deprecated and may be removed in a future version.

`transitive` (binary) (directed) (triad-related)

Transitive triads: This term adds one statistic to the model, equal to the number of triads in the network that are transitive. The transitive triads are those of type `120D`, `030T`, `120U`, or `300` in the categorization of Davis and Leinhardt (1972). For details on the 16 possible triad types, see `triad.classify` in the `sna` package. Note the distinction from the `ttriple` term. This term can only be used with directed networks.

`transitiveties(attrname=NULL)` (binary) (directed) (triad-related) (categorical nodal attribute) , `transitiveties(threshold=0)` (valued) (directed) (undirected) (triad-related)

Transitive ties: This term adds one statistic, equal to the number of ties *i-->j* such that there exists a two-path from *i* to *j*. (Related to the `ttriple` term.) The binary version takes a nodal attribute `attrname`, and, if given, all three nodes involved (*i*, *j*, and the node on the two-path) must match on this attribute in order for *i-->j* to be counted. The binary version of this term can only be used with directed networks. The valued version can be used with both directed and undirected.

`transitiveweights(twopath="min",combine="max",affect="min")` (valued) (directed) (undirected) (non-negative) (triad-related)

Transitive weights: This statistic implements the transitive weights statistic defined by Krivitsky (2012), Equation 13. The currently implemented options for `twopath` is the minimum of the constituent dyads (`"min"`) or their geometric mean (`"geomean"`); for `combine`, the maximum of the 2-path strengths (`"max"`) or their sum (`"sum"`); and for `affect`, the minimum of the focus dyad and the combined strength of the two paths (`"min"`) or their geometric mean (`"geomean"`). For each of these options, the first (and the default) is more stable but also more conservative, while the second is more sensitive but more likely to induce a multimodal distribution of networks.

`triadcensus(d)` (binary) (triad-related) (directed) (undirected)

Triad census: For a directed network, this term adds one network statistic for each of an arbitrary subset of the 16 possible types of triads categorized by Davis and Leinhardt (1972) as `003, 012, 102, 021D, 021U, 021C, 111D, 111U, 030T, 030C, 201, 120D, 120U, 120C, 210,` and `300`. Note that at least one category should be dropped; otherwise a linear dependency will exist among the 16 statistics, since they must sum to the total number of three-node sets. By default, the category `003`, which is the category of completely empty three-node sets, is dropped. This is considered category zero, and the others are numbered 1 through 15 in the order given above. By specifying a numeric vector of integers from 0 to 15 as the `d` argument, the user may specify a set of terms to add other than the default value of `1:15`. Each statistic is the count of the corresponding triad type in the network. For details on the 16 types, see `?triad.classify` in the `{sna}` package, on which this code is based. For an undirected network, the triad census is over the four types defined by the number of ties (i.e., 0, 1, 2, and 3), and the default is to add 1:3, which is to say that the 0 is dropped; however, this too may be controlled by changing the `d` argument to a numeric vector giving a subset of *\{0, 1, 2, 3\}*.

`triangle(attrname=NULL)` (binary) (frequently-used) (triad-related) (directed) (undirected) (categorical nodal attribute)

Triangles: This term adds one statistic to the model equal to the number of triangles in the network. For an undirected network, a triangle is defined to be any set *\{(i,j), (j,k), (k,i)\}* of three edges. For a directed network, a triangle is defined as any set of three edges *(i,j)* and *(j,k)* and either *(k,i)* or *(i,k)*. The former case is called a “transitive triple” and the latter is called a “cyclic triple”, so in the case of a directed network, `triangle` equals `ttriple` plus `ctriple` — thus at most two of these three terms can be in a model. The optional argument `attrname` restricts the count to those triples of nodes with equal values of the vertex attribute specified by `attrname`.

`tripercent(attrname=NULL)` (binary) (undirected) (triad-related) (categorical nodal attribute)

Triangle percentage: This term adds one statistic to the model equal to 100 times the ratio of the number of triangles in the network to the sum of the number of triangles and the number of 2-stars not in triangles (the latter is considered a potential but incomplete triangle). In case the denominator equals zero, the statistic is defined to be zero. For the definition of triangle, see `triangle`. The optional argument `attrname` restricts the counts (both numerator and denominator) to those triples of nodes with equal values of the vertex attribute specified by `attrname`. This is often called the mean correlation coefficient. This term can only be used with undirected networks; for directed networks, it is difficult to define the numerator and denominator in a consistent and meaningful way.

`ttriple(attrname=NULL)` (binary) (directed) (triad-related) (categorical nodal attribute) , a.k.a. `ttriad` (binary) (directed) (triad-related) (categorical nodal attribute)

Transitive triples: This term adds one statistic to the model, equal to the number of transitive triples in the network, defined as a set of edges *{(i,j), (j,k), (i,k)}*. Note that `triangle` equals `ttriple+ctriple` for a directed network, so at most two of the three terms can be in a model. The optional argument `attrname` is a character string giving the name of an attribute in the network's vertex attribute list. If this is specified then the count is over the number of transitive triples where all three nodes have the same value of the attribute. This term can only be used with directed networks.

`twopath` (binary) (directed) (undirected)

> *2-Paths:* This term adds one statistic to the model, equal to the number of 2-paths in the network. For a directed network this is defined as a pair of edges *(i,j), (j,k)*, where *i* and *j* must be distinct. That is, it is a directed path of length 2 from *i* to *k* via *j*. For directed networks a 2-path is also a mixed 2-star but the interpretation is usually different; see `m2star`. For undirected networks a twopath is defined as a pair of edges *\{i,j\}, \{j,k\}*. That is, it is an undirected path of length 2 from *i* to *k* via *j*, also known as a 2-star.

References

- Bomiriya, R. P, Bansal, S., and Hunter, D. R. (2014). Modeling Homophily in ERGMs for Bipartite Networks. Submitted.
- Butts, CT. (2008). “A Relational Event Framework for Social Action.” *Sociological Methodology,* 38(1).
- Davis, J.A. and Leinhardt, S. (1972). The Structure of Positive Interpersonal Relations in Small Groups. In J. Berger (Ed.), *Sociological Theories in Progress, Volume 2*, 218–251. Boston: Houghton Mifflin.
- Holland, P. W. and S. Leinhardt (1981). An exponential family of probability distributions for directed graphs. *Journal of the American Statistical Association*, 76: 33–50.
- Hunter, D. R. and M. S. Handcock (2006). Inference in curved exponential family models for networks. *Journal of Computational and Graphical Statistics*, 15: 565–583.
- Hunter, D. R. (2007). Curved exponential family models for social networks. *Social Networks*, 29: 216–230.
- Krackhardt, D. and Handcock, M. S. (2007). Heider versus Simmel: Emergent Features in Dynamic Structures. *Lecture Notes in Computer Science*, 4503, 14–27.
- Krivitsky P. N. (2012). Exponential-Family Random Graph Models for Valued Networks. *Electronic Journal of Statistics*, 2012, 6, 1100-1128. doi: 10.1214/12-EJS696
- Robins, G; Pattison, P; and Wang, P. (2009). “Closure, Connectivity, and Degree Distributions: Exponential Random Graph (p*) Models for Directed Social Networks.” *Social Networks,* 31:105-117.
- Snijders T. A. B., G. G. van de Bunt, and C. E. G. Steglich. Introduction to Stochastic Actor-Based Models for Network Dynamics. *Social Networks*, 2010, 32(1), 44-60. doi: 10.1016/j.socnet.2009.02.004
- Morris M, Handcock MS, and Hunter DR. Specification of Exponential-Family Random Graph Models: Terms and Computational Aspects. *Journal of Statistical Software*, 2008, 24(4), 1-24. http://www.jstatsoft.org/v24/i04
- Snijders, T. A. B., P. E. Pattison, G. L. Robins, and M. S. Handcock (2006). New specifications for exponential random graph models, *Sociological Methodology*, 36(1): 99-153.

See Also

`ergm` package, `search.ergmTerms`, `ergm`, `network`, `%v%`, `%n%`

Examples

```
## Not run:
ergm(flomarriage ~ kstar(1:2) + absdiff("wealth") + triangle)

ergm(molecule ~ edges + kstar(2:3) + triangle
                      + nodematch("atomic type",diff=TRUE)
                      + triangle + absdiff("atomic type"))

## End(Not run)
```