

Native exponential-family random graph models in Stata

Thomas U. Grund

Institute of Sociology, RWTH Aachen University, Germany

Abstract. I describe `nwergm`, the exponential-family random graph model (ERGM) command of the `nwcommands` family of packages for the analysis of network data in Stata. `nwergm` is a port of the `ergm` package of the statnet project: it implements 125 of the 139 terms of `ergm` 4.12.0 (structural, covariate, curved, two-mode and operator terms, with sampling and observation constraints), estimates models by maximum pseudolikelihood, Monte Carlo maximum likelihood, stochastic approximation or contrastive divergence with the control defaults of `ergm`, and offers Monte Carlo diagnostics, goodness of fit, information criteria, prediction and simulation after estimation. The Monte Carlo sampler and the change statistics run in a compiled Stata plugin; no R or other external software is called. I describe the syntax, the term library and the estimation methods, show worked examples on a small, a directed and a two-mode network, and report how the estimates compare with those of `ergm` on the same data and how fast the two programs are: with the defaults of both programs, `nwergm` 1.3 fits eight benchmark models 2.9 to 9.4 times faster than `ergm` 4.12.0.

Keywords: st0003, `nwergm`, exponential-family random graph models, ERGM, network statistics, Monte Carlo maximum likelihood, Stata plugins, `nwcommands`

1 Introduction

An exponential-family random graph model (ERGM) treats a network’s ties as the joint outcome of a small set of structural and covariate-driven tendencies – density, reciprocity, homophily, triadic closure, degree heterogeneity – estimated simultaneously rather than approximated tie by tie; Lusher et al. (2013) give the standard treatment of the theory, estimation, and interpretation involved. The `ergm` package of the statnet project (Handcock et al., 2008; Hunter et al., 2008b; Krivitsky et al., 2023) has been the standard implementation for close to two decades. A Stata user who wanted to fit an ERGM had to leave Stata, export the network, fit the model in R and import the results back; that breaks the one-environment workflow of the rest of a Stata-based analysis and puts a second piece of software between a user and the model.

`nwergm` removes that step. It is the ERGM command of `nwcommands` (Grund, 2026a), a family of Stata packages for network analysis, and it fits models to a network that has been declared with `nwset` or loaded with one of the `nwcommands` import commands. It implements 125 of the 139 terms of `ergm` 4.12.0 (Table 1), estimation by maximum pseudolikelihood, Monte Carlo maximum likelihood, stochastic approximation and contrastive divergence, the sampling and observation constraints, the treatment of missing dyads, and the postestimation tools `mcmc.diagnostics()`, `gof()`, `simulate()`, `logLik()` and `predict()`, each under a Stata name. The remaining 14 terms are listed in Section 10.

Provenance. `nwergm` was developed from `ergm`: the term definitions and change statistics, the pseudolikelihood and Monte Carlo estimators, the control defaults and the diagnostics follow the R package and its published description (Hunter and Handcock, 2006; Hunter, 2007; Morris et al., 2008; Hummel et al., 2012; Geyer and Thompson, 1992), and `nwergm` is distributed under the same licence as `ergm`, the GNU General Public License, version 3. It is not affiliated with or endorsed by the statnet project, and errors are mine. Where `ergm` and `nwergm` differ, Section 10 says so.

`nwergm` is a companion to the article on `nwcommands` (Grund, 2026a). This article is self-contained. Further model packages and extensions are described on the website <https://nwcommands.org>.

The article is organized as follows. Section 2 gives the installation, Section 3 recalls the model, Section 4 describes the syntax and the term library, Section 5 the estimation methods and Section 6 the postestimation tools. Section 7 gives worked examples, Section 8 describes the implementation, Section 9 reports the comparison with `ergm` in estimates and in speed, and Section 10 states what is not implemented.

2 Installation

`nwergm` needs Stata 17 or later; it is tested with Stata 17 and 18. It is a separate package that needs the core `nwcommands`. From within Stata:

```
. net install nwcommands, from(https://nwcommands.org/net/)  
replace  
. nwinstall, all  
. nwinstall, modelpkg(nwergm)
```

The first line installs a small bootstrap package with `nwinstall` itself. `nwinstall, all` installs the rest of the core: the command set, the help files, the dialog boxes and the Mata library. The third line installs `nwergm`, which has its own repository and version, its own Mata library and the compiled plugin for the platform (macOS, Linux or Windows). No compiler is needed. The command `nwergm version` shows the installed release, the core, the plugin in use, and the version of `ergm` the package was compared with; `help nwergm` opens the documentation, and the dialog `db nwergm` sets every term and option. (`nwinstall, everything` installs the core and all four model packages in one step.) The results in this article were produced with `nwergm` 1.3.1 (which needs `nwcommands` 3.6.0 or later) and `nwcommands` 3.10.0, in Stata 18; the Mata libraries are built with Stata 17, the oldest tested version, because a library built by a newer Stata cannot be used by an older one. The manual of `nwergm` (Grund, 2026c) documents every option and term; it is archived at Zenodo under the DOI of all versions, <https://doi.org/10.5281/zenodo.23661331>.

3 ERGM framework

An ERGM models an entire network y , not tie by tie, as a draw from an exponential-family distribution over the set $\mathcal{Y}$ of all possible networks on a fixed set of nodes:

$$(1) \quad P_{\theta}(Y = y) = \frac{\exp(\theta^{\top} g(y))}{\kappa(\theta)}, \quad y \in \mathcal{Y},$$

where $g(y)$ is a vector of network statistics – density, reciprocity, triadic closure, and so on –, θ is the corresponding vector of parameters, and $\kappa(\theta) = \sum_{y' \in \mathcal{Y}} \exp(\theta^{\top} g(y'))$ normalizes the distribution over every network on the same node set: a sum with $2^{n(n-1)}$ terms for a directed network on n nodes, intractable at any realistic size. In a curved model, θ is a function of a smaller parameter vector, as for the geometrically weighted terms whose decay is estimated (Hunter and Handcock, 2006).

Every term enters the model through its *change statistic*: how much the term’s contribution to $g(y)$ changes if a single tie y_{ij} is toggled, holding every other tie fixed,

$$(2) \quad \Delta g_{ij}(y) = g(y_{ij}^{+}) - g(y_{ij}^{-}),$$

where y_{ij}^{+} and y_{ij}^{-} denote y with the ij tie forced present and absent. The conditional log odds of a single tie reduce to a linear form in the change statistic regardless of how complicated $g(y)$ is:

$$(3) \quad \log \frac{P_{\theta}(Y_{ij} = 1 \mid Y_{-ij} = y_{-ij})}{P_{\theta}(Y_{ij} = 0 \mid Y_{-ij} = y_{-ij})} = \theta^{\top} \Delta g_{ij}(y).$$

So a coefficient is the change in the conditional log odds of a tie per unit of the statistic, given the rest of the network. This conditional-logit form is what makes pseudolikelihood estimation possible: the ERGM’s own log odds of a tie become an ordinary logistic-regression predictor, with $\Delta g_{ij}(y)$ playing the role of the covariates of dyad ij .

Whether the pseudolikelihood is also the likelihood depends on whether the terms are dyad-independent or dyad-dependent. If every term is dyad-independent – its change statistic depends only on nodal attributes and dyadic covariates of i and j , never on another tie – the joint likelihood factors into independent per-dyad logistic terms, and the maximum pseudolikelihood estimate *is* the maximum likelihood estimate (van Duijn et al., 2009). Dyad-dependent terms – **mutual**, the geometrically weighted and shared-partner terms, the degree-distribution and triad terms – break the factorization, and $\kappa(\theta)$ cannot be avoided. **nwergm** then maximizes the likelihood by Monte Carlo: it simulates networks from a trial θ by a Metropolis–Hastings sampler and updates θ until the simulated statistics match the observed ones on average (Geyer and Thompson, 1992; Hummel et al., 2012). The log-likelihood at the estimate, needed for the information criteria, is evaluated by bridge sampling from the dyad-independent submodel.

4 The nwergm command

4.1 Syntax overview

```
. nwergm [netname] , edges [terms] [structural(termlist)]
[operators(oplist)] [estimation options]
```

```

. nvergm simulate [nodes] , [terms theta(numlist)] [options]
. estat gof | estat mcmcdiag | estat ic

```

A model is specified as a list of term options, the way a Stata regression command's option list specifies covariates. **edges** is required; *netname* defaults to the current network, so a model is fitted directly to whatever **nwset** or **nwebuse** last produced, with no export step. Node attributes are numeric variables of the current dataset (a string attribute is first converted with **encode**); dyadic covariates are networks in memory. Coefficients carry the names of **ergm**'s coefficients with the dot written as an underscore, such as **nodematch_seat**. An estimation result is an ordinary Stata estimation result: **e(b)**, **e(V)**, **estimates store**, **lrtest**, **test** and **predict** work as usual.

Network types. **nvergm** fits directed, undirected and two-mode (bipartite) networks. A directed two-mode network is refused, as are temporal networks. A network with tie values is fitted as the binary graph in which a tie is present when its value is not zero, as **ergm** does without its option **response=**; **nvergm** prints one note that the values are ignored (and no note when every value is 1). The terms for valued networks of **ergm** are not available (Section 10).

4.2 Term library

Table 1 lists the terms of **ergm** 4.12.0 by family and the number that **nvergm** implements. Of the 139 terms, 123 are implemented as in **ergm**, two in part, and 14 are not implemented (13 terms for valued networks and the operator **B**). Apart from these, the library is complete. Thirty-three of the terms apply to two-mode networks only. The definitions and the interpretation of the terms are those of **ergm** (Morris et al., 2008); Lusher et al. (2013) discuss when to prefer one over another. Figure 1 shows the structure behind six representative families.

How the terms are written. The most common terms have options of their own: **mutual**, **nodematch()**, **nodecov()**, **nodefactor()**, **nodemix()**, **edgecov()**, **degree()**, **kstar()**, **triangle**, **gwap()**, **esp()** and many more; the table gives the spelling. Terms whose R name contains a mode digit followed by a letter keep the name with the digit moved to the end, because Stata does not allow a digit followed by a letter in an option name: **b1degree** is **bdegree1()**, **gwb2degree** is **bgwdegree2()**. The less common structural terms of **ergm** (**twopath**, **isolates**, **cycle**, **degree** with **by=**, **ttriple**, **simmelian**, **triadcensus**, and others) are written in R's own notation inside **structural()**, with R's argument names and defaults (a vector such as R's **c(1,2)** is written **1 2**): **structural(isolates cycle(k=3 4) degree(d=1 2 by=sex))**. The terms of the form **gwap(α)** have a fixed decay α ; **gwapfree()**, **gwdegreefree()**, **gwdspfree()**, **gwnspfree()**, **gwdegreefree()** and **gwidegreefree()** estimate the decay together with the other coefficients (R's **fixed=FALSE**); the number in parentheses is then the starting value. On a directed network the shared-partner terms take **type(OTP|ITP|OSP|ISP|RTP)**, selecting one of the five definitions of a directed shared partner, the default being **OTP** as in R.

Table 1: The terms of `ergm` 4.12.0 by family, and how many `nwergm` 1.3.1 implements.

Family	<code>ergm</code>	<code>nwergm</code>	Examples (<code>nwergm</code> syntax)
Density, reciprocity	5	5	<code>edges</code> , <code>mutual</code> , <code>hamming()</code> , <code>structural(density meandeg)</code>
Degrees, stars, concurrency	35	35	<code>degree()</code> , <code>odegree()</code> , <code>idegree()</code> , <code>kstar()</code> , <code>degrange()</code> , <code>concurrent</code> , <code>structural(isolates twopath</code> <code>degree1.5)</code> , two-mode <code>bdegree1()</code> , <code>bstar2()</code>
Shared partners	5	5	<code>esp()</code> , <code>dsp()</code> , <code>structural(nsp())</code> , <code>bdsp1()</code> ; <code>type()</code> on directed networks
Triads, cycles, paths	18	18	<code>triangle</code> , <code>ctriple</code> , <code>transitiveties</code> , <code>cyclicalities</code> , <code>structural(cycle()</code> <code>ttriple threetrail simmelian</code> <code>triadcensus balance)</code>
Curved (geometrically weighted, alternating)	11	10 + 1 part	<code>gwersp()</code> , <code>gwdsp()</code> , <code>gwnsp()</code> , <code>gwdegree()</code> , <code>gwodegree()</code> , <code>gwidegree()</code> , two-mode <code>bgwdegree1()</code> , <code>bgwdsp1()</code> ; estimated decay: <code>gwerspfree()</code> and five more; <code>altkstar</code>
Node covariates	34	34	<code>nodecov()</code> , <code>nodefactor()</code> , <code>nodematch()</code> , <code>nodemix()</code> , <code>absdist()</code> , <code>sociality()</code> , <code>sender</code> , <code>receiver</code> , <code>directed nodeicov()</code> , <code>nodeofactor()</code> , two-mode <code>bcov1()</code> , <code>bfactor1()</code>
Dyadic covariates	3	3	<code>edgescov()</code> , <code>dyadcov()</code> , <code>attrcov()</code>
Operators	15	13 + 1 part	<code>operators(Sum() Prod() S() F()</code> <code>NodematchFilter() Symmetrize()</code> <code>Project() I() For() Label() Log()</code> <code>Exp() Offset() Curve())</code>
Valued networks only	13	0	—
All	139	123 + 2 part	125 of 139 (89.9%)

“part” means implemented in part: `altkstar` with an estimated decay (by pseudolikelihood only) and `Curve()` for linear maps (Section 10).

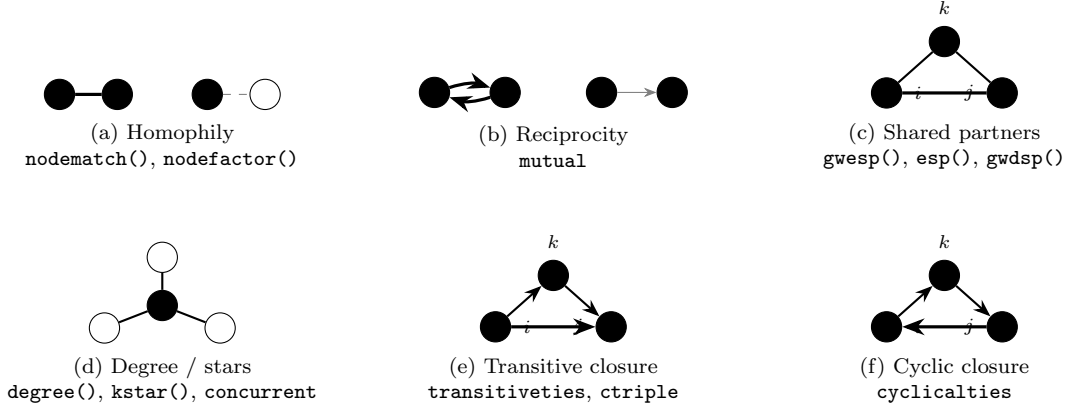


Figure 1: Structural motifs behind representative term families from Table 1. Thick lines mark the tie whose presence the term is sensitive to; thin/dashed lines mark ties whose presence or absence sets up that sensitivity but do not themselves change the statistic. Panel (c) carries over unchanged to `gwensp()` and `dsp()`, which count the same i - k - j configuration when the i - j dyad is untied rather than tied.

Operators. The option `operators()` applies terms to a derived network or combines them, written like R’s operator terms: `Sum`, `Prod`, `S` (the subgraph induced by a set of nodes), `F` (the network without the ties that fail a filter), `NodematchFilter`, `Symmetrize`, `Project` (the one-mode projection of a two-mode network), `I`, `For`, `Label`, `Log`, `Exp`, `Offset`, and `Curve` for linear maps. For example, `operators(Symmetrize(triangle, rule(strong)))` counts triangles in the undirected network of reciprocated ties (Section 7.4). Operators cannot be combined with constraints, and a curved term cannot be placed inside an operator.

Constraints, offsets and observation. `constraints()` restricts the set of networks on which the model is defined, written as in R’s `constraints=`: `edges` (the number of ties is fixed), `degrees`, `odegrees`, `idegrees`, `b1degrees`, `b2degrees`, `degreedist`, `bd()` (bounded degrees), `blocks()`, `blockdiag()`, `fixedas()`, `fixallbut()`, `observed`, `egocentric()` and `dyadnoise()`, joined by `+` or `-`. A statistic that a constraint holds constant is dropped from the estimation with a note, as in R. `obsconstraints()` describes how the network was observed (for example which dyads were not observed); `estat` `gof` and `nvergm` `simulate` use the constraints of the fit. The option `offset(coefname #)` holds a coefficient fixed at a known value. `hamming`, `ChangeStats()`, `Dyads()` and `I()` are not available as constraints.

Missing data. As in R, a missing dyad is not treated as a non-tie. Missing dyads leave the pseudolikelihood; for a dyad-dependent model a second, constrained sample in which only the missing dyads are free enters the estimating equations (Handcock and Gile, 2010). `e(N)` is the number of informative dyads and `e(nmissing)` the number of missing dyads. A missing value in a node attribute or a dyadic covariate is an error, as

in R.

5 Estimation

Choosing the method. If every term is dyad-independent, maximum pseudolikelihood is the maximum likelihood estimate, and `nwergm` detects this automatically, reports the fit as an MPLE and runs no simulation. Any dyad-dependent term, a dyad-dependent constraint, a curved term or an observation constraint switches the default to Monte Carlo maximum likelihood, with the pseudolikelihood estimate as the starting value. The option `estimate(mle|mple|cd)` chooses the estimator, and `mainmethod(mcmle|sa)` chooses the Monte Carlo algorithm.

Pseudolikelihood. The pseudolikelihood is maximized by logistic regression on the matrix of change statistics. `mpletype(penalized)` gives the penalized (Firth, 1993) estimate, which is finite when the ordinary estimate does not exist; `mplecheck(on)` tests whether the estimate exists and warns as R does; `mplecovariance(godambe|bootstrap)` replaces the covariance of a pseudolikelihood fit of a dyad-dependent model.

Monte Carlo maximum likelihood. The algorithm follows Hummel et al. (2012): each iteration simulates a sample at the current coefficients, takes a Newton step of adaptive length, and stops when a test on the simulated statistics (`mcmletermination()`: `confidence`, `hummel`, `hotelling` or `none`) says that the observed statistics lie inside the sampled distribution. The Metropolis-Hastings proposal is the tie/no-tie proposal of Morris et al. (2008); `proposal(uniform)` is also available. The controls have the names and defaults of R's `control.ergm()`, and the algorithm that uses them is R's, including its adaptive sample size. The first iteration starts with `mcmcinterval()` 1,024 proposals between recorded draws and a burn-in of 16 times the interval (`mcmcburnin()`, 16,384); an `mcmcinterval()` given without `mcmcburnin()` also sets the burn-in to 16 times its value. Each iteration records a base sample of $\max(256, 32p)$ draws for p parameters (`mmcscsamplesize()`) and continues sampling until the effective sample size of the estimating equations reaches the target of 64 (`mcmleeffectivesize()`, R's `MCMLE.effectivesize`); the burn-in is estimated from the sample itself, the chain is thinned by two whenever the sample has doubled, and the next iteration starts with half the interval that the sample needed; the thinning interval that `estat mcmcdiag` reports is therefore that of the final sample, which can be larger than the interval requested. The default termination is R's test `confidence`: the fit stops when, with 99% confidence, the estimating equations at the updated coefficients lie inside a region of 0.1 of their covariance; a failed test raises the target effective size and the base sample size, by at most a factor of 2. The covariance matrix is taken from the sample of the last iteration, and `mcmleiterations()` is 60. The number of recorded draws is therefore not a fixed setting: `e(mcmc.samplesize)` gives the number in the final sample. `mcmleeffectivesize(0)` asks instead for fixed-size samples (3,000 draws in every iteration unless `mmcscsamplesize()` says otherwise); much shorter chains, an interval of 50 and a burn-in of 3,000, can be set with `mcmcburnin(3000) mcmcinterval(50)`, at the risk of poor mixing. The log-likelihood is evaluated by bridge sampling (`bridgesteps()`,

16 as in R; skipped with `nologlik`), with the sample size, interval and burn-in of the last iteration of the fit, as in R; it is not evaluated after a curved fit by Monte Carlo maximum likelihood.

Contrastive divergence and stochastic approximation. `estimate(cd)` is contrastive divergence (Krivitsky, 2017): short chains started from the observed network, used to find a starting point or a rough estimate; the covariance reported is that of the short-chain sample. `mainmethod(sa)` is the stochastic approximation algorithm of Snijders (2002) in the form of R's `main.method="Stochastic-Approximation"`. `init()` supplies starting values, one per coefficient, with a missing value for a coefficient to be estimated.

Reproducibility. With `seed()` the same call gives the same numbers in any session, on the same platform and plugin build. A different operating system, a different plugin build or a different value of a control option generally changes the chain and, within Monte Carlo error, the estimates.

6 Postestimation

estat mcmcdiag. After a Monte Carlo fit, `estat mcmcdiag` reports diagnostics of the final simulated sample: for each statistic the mean, standard deviation, autocorrelation at the thinning interval and effective sample size (the header gives the thinning interval of the final sample and, when the adaptive sampler raised it, a line with the requested interval), the Geweke statistic and the Heidelberger–Welch test, followed by the report of R's `mcmc.diagnostics()` (sample statistics centered at the observed ones, the test that they differ from zero, cross- and autocorrelations). With `plot` it draws trace and density plots. A satisfied convergence test is a necessary condition, not proof of a good fit; the note at the end of the report advises `mcmleffectivesize()` (or `mcmcsamplesize()` and `mcmcinterval()` for a fixed-size sample).

estat gof. `estat gof` follows R's `gof()`: it simulates networks from the fitted model and, for each chosen statistic, shows the observed value, the minimum, mean and maximum over the simulations and a Monte Carlo p -value. The statistics are the model's own statistics and the distributions of degree, in-degree, out-degree, two-mode degrees, edgewise and dyadwise shared partners, geodesic distance and the triad census, chosen with `gof()`; the default is that of R for the type of network. `plot` draws box-and-whisker plots of the simulated distributions with the observed one overlaid, as R's `plot.gof()` (Hunter et al., 2008a).

Fit statistics and prediction. After every fit the log-likelihood, its Monte Carlo standard error, deviance, AIC and BIC are stored (BIC with N the number of informative dyads); `estat ic` displays them, `estimates stats` and `lrtest` compare models. `predict`, `matrix()` and `predict, frame()` return the conditional tie probabilities or log odds of each dyad given the rest of the observed network, as `predict.ergm`.

nwergm simulate. `nwergm simulate` draws networks from a model. Without a number of nodes it draws from the last fit, starting from the fitted network and using the fitted coefficients and constraints, as R's `simulate()`; `coef()` and `basis()` change the coefficients and the starting network. With a number of nodes, the terms and `theta()` it draws from a fully specified model. `output(network)` creates ordinary networks, `output(stats)` returns the statistics of the draws and `output(edgelist)` an edge list. A fitted two-mode model gives two-mode networks.

Stored results. Besides `e(b)` and `e(V)`, `nwergm` stores the numbers of informative and missing dyads (`e(N)`, `e(nmissing)`), of nodes and ties, the log-likelihood and its relatives (`e(ll)`, `e(ll_0)`, `e(ll_se)`, `e(df_m)`, `e(aic)`, `e(bic)`, `e(deviance)`), and, after Monte Carlo estimation, `e(converged)`, `e(mcmle.iterations)`, the acceptance rate, the burn-in, interval and sample size used, `e(native)` (1 if the plugin ran), and the matrix `e(mcmcsample)` of the final sample of statistics. `e(version)` holds the package version; it should be quoted with an analysis.

7 Remarks and examples

All logs were produced with `nwergm` 1.3.1 and `nwcommands` 3.10.0 in Stata 18 (do-file `gen_logs.do`); every fit that uses the Monte Carlo sampler has a `seed()` and can be reproduced on the same platform.

7.1 A small network: the Florentine marriages

The marriage network among 16 Renaissance Florentine families (Padgett and Ansell, 1993) has 20 ties; `seat` records whether a family had a seat on the city council. In network terms, the families are the nodes, and a marriage between two families is an undirected tie between them; the network can be drawn with `nwplot` (Figure 2).

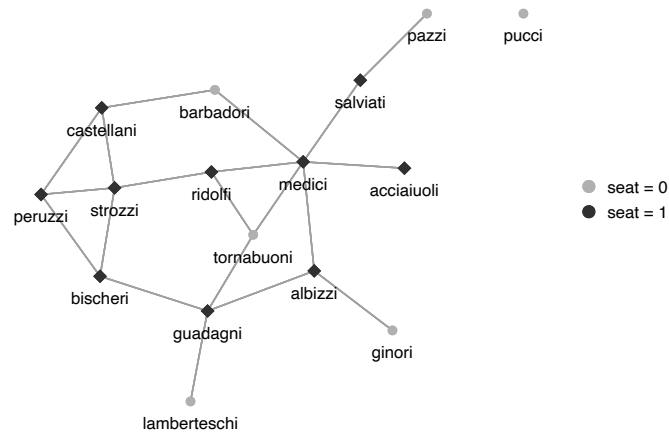


Figure 2: Marriage ties among 16 Florentine families (`nwplot flomarriage`, do-file `plot_networks.do`). Each marker is a family and each line a marriage. Dark diamonds are families with a seat on the city council (`seat = 1`), light circles families without one. The position of a family on the page has no meaning of its own; the layout only places linked families near each other.

Read the figure as a map of who is connected to whom: a family with many lines (the Medici have six) has many marriage partners, and two families joined by a line are married to each other. Pucci has no marriage tie in these data and floats free. The picture suggests a theory-relevant pattern, namely that most council families are linked to other council families, but it cannot tell whether this is more than chance given how many ties there are; that is what the model below is for. An edges-only model is the ERGM analogue of an intercept-only regression:

```
. nwwebuse florentine, nwcLEAR
. nwerGM flomarriage, edges
Exponential-family random graph model
Network:          = flomarriage
Nodes:            = 16
Ties:             = 20
Directed:         = No
Estimation:       = MPLE
```

flomarriage	Coefficient	Std. err.	z	P> z	[95% conf. interval]	
edges	-1.609438	.244949	-6.57	0.000	-2.089529	-1.129347

Maximum pseudolikelihood estimate (not full ERGM maximum likelihood unless the model is dyad-independent).
> dent).

Null Deviance: 166.4 on 120 degrees of freedom
Residual Deviance: 108.1 on 119 degrees of freedom
Log-likelihood: -54.0673 (df = 1) AIC: 110.1 BIC: 112.9 (MC std. err. = 0)

The coefficient is the log odds of a tie, $\log(20/100) = -1.609$, and the fit is an MPLE that is also the maximum likelihood estimate. Dyad-independent terms keep the estimation at the pseudolikelihood:

```
. nwergm flomarriage, edges nodematch(seat)
Exponential-family random graph model
Network:      = flomarriage
Nodes:        = 16
Ties:         = 20
Directed:     = No
Estimation:    = MPLE
```

flomarriage	Coefficient	Std. err.	z	P> z	[95% conf. interval]	
edges	-1.871802	.3797773	-4.93	0.000	-2.616152	-1.127452
nodematch_seat	.4855078	.4983949	0.97	0.330	-.4913282	1.462344

Maximum pseudolikelihood estimate (not full ERGM maximum likelihood unless the model is dyad-independent).
> dent).

Null Deviance: 166.4 on 120 degrees of freedom
Residual Deviance: 107.2 on 118 degrees of freedom
Log-likelihood: -53.5846 (df = 2) AIC: 111.2 BIC: 116.7 (MC std. err. = 0)

Adding `gwesp()`, which models triadic closure and is dyad-dependent, moves the estimation to Monte Carlo maximum likelihood without any change in how the model is specified:

```
. nwergm flomarriage, edges nodematch(seat) gwesp(.5) seed(2026)
Exponential-family random graph model
Network:      = flomarriage
Nodes:        = 16
Ties:         = 20
Directed:     = No
Estimation:    = MCMLE
MCMC sample size: = 243
```

flomarriage	Coefficient	Std. err.	z	P> z	[95% conf. interval]	
edges	-1.963784	.4874806	-4.03	0.000	-2.919229	-1.00834
nodematch_seat	.4917407	.4837383	1.02	0.309	-.4563689	1.43985
gwesp_.5	.1068421	.2845549	0.38	0.707	-.4508752	.6645594

MCMLE converged after 1 iteration(s).

Evaluating log-likelihood at the estimate (bridge sampling with 16 bridges) ...

Null Deviance: 166.4 on 120 degrees of freedom
Residual Deviance: 107.1 on 117 degrees of freedom
Log-likelihood: -53.5278 (df = 3) AIC: 113.1 BIC: 121.4 (MC std. err. = .01305)

The closure coefficient is small relative to its standard error, so the example shows the

syntax and not a finding. The same network illustrates a constraint, the choice of the estimator and the treatment of missing dyads. With `constraints(edges)` the number of ties is fixed, so `edges` cannot be estimated and is shown as omitted:

```
. nwergm flomarriage, edges nodecov(wealth) triangle constraints(edges) seed(2026)
Note: the specified model's sample space constraint holds statistic(s) edges constant. They will be
> ignored.
```

Exponential-family random graph model

```
Network:      = flomarriage
Nodes:        = 16
Ties:         = 20
Directed:     = No
Estimation:   = MCMLE
MCMC sample size: = 243
```

flomarriage	Coefficient	Std. err.	z	P> z	[95% conf. interval]	
edges	0 (omitted)					
nodecov_wealth	.01033	.0052377	1.97	0.049	.0000642	.0205957
triangle	.0876229	.6320863	0.14	0.890	-1.151244	1.326489

MCMLE converged after 1 iteration(s).

Evaluating log-likelihood at the estimate (bridge sampling with 16 bridges) ...

```
Null Deviance:    166.4 on 120 degrees of freedom
Residual Deviance: 111.0 on 118 degrees of freedom
```

Log-likelihood: -55.5019 (df = 2) AIC: 115.0 BIC: 120.6 (MC std. err. = .17931)

The estimators give similar answers for one model with a triangle term (the pseudo-likelihood estimate is not the maximum likelihood estimate once the model is dyad-dependent); the contrastive divergence estimate is the roughest, with larger standard errors:

```
. quietly nwergm flomarriage, edges nodecov(wealth) triangle seed(2026)
. estimates store mcmle
. quietly nwergm flomarriage, edges nodecov(wealth) triangle mainmethod(sa) seed(2026)
. estimates store sa
. quietly nwergm flomarriage, edges nodecov(wealth) triangle estimate(cd) seed(2026)
. estimates store cd
. quietly nwergm flomarriage, edges nodecov(wealth) triangle estimate(mple)
. estimates store mple
. estimates table mple mcmle sa cd, b(%8.3f) se(%8.3f)
```

Variable	mple	mcmle	sa	cd
edges	-2.603	-2.589	-2.604	-2.711
	0.545	0.542	0.550	0.964
nodecov_wealth	0.010	0.011	0.011	0.013
	0.005	0.005	0.005	0.011
triangle	0.036	-0.048	-0.119	-0.168
	0.447	0.632	0.630	0.931

Legend: b/se

Missing dyads are left out of the likelihood and counted:

```

. nwtomata flomarriage, mat(A)
. mata: A[1,2] = A[2,1] = A[3,5] = A[5,3] = .
. mata: st_matrix("M", A)
. nwset, mat(M) undirected name(flo_na)
. nwergm flo_na, edges nodematch(seat)
Note: network flo_na has 2 missing dyad(s); they are excluded from the (pseudo)likelihood and handle
> d as in R's ergm (missing-data MCMLE).
Exponential-family random graph model
Network:          = flo_na
Nodes:            = 16
Ties:             = 19
Directed:         = No
Estimation:       = MPLE

```

	flo_na	Coefficient	Std. err.	z	P> z	[95% conf. interval]	
edges		-2.005334	.4026014	-4.98	0.000	-2.794418	-1.216249
nodematch_seat		.6400926	.516428	1.24	0.215	-.3720876	1.652273

```

Maximum pseudolikelihood estimate (not full ERGM maximum likelihood unless the model is dyad-indepen
> dent).
      Null Deviance:      163.6  on 118 degrees of freedom
      Residual Deviance:  102.6  on 116 degrees of freedom
Log-likelihood:  -51.2876 (df = 2)  AIC:      106.6  BIC:      112.1  (MC std. err. =      0)
. display "informative dyads: " e(N) "  missing dyads: " e(nmissing)
informative dyads: 118  missing dyads: 2

```

A network with tie values is fitted as the binary graph, with one note:

```

. nwtomata flomarriage, mat(A)
. mata: V = A :* (1 :+ mod((1::16)*J(1,16,1) + J(16,1,1)*(1..16), 3))
. mata: st_matrix("V", V)
. nwset, mat(V) undirected name(flo_val)
. nwsummarize flo_val

```

```

Network name: flo_val
Network id: 4
Directed: false
Valued: true
Two-mode: false
Nodes: 16
Selfloop: false
Edges: 20
Minimum value: 0
Maximum value: 3
Density: .167
Temporal: false
. nwergm flo_val, edges nodematch(seat)
(note: tie values are ignored; the model is fitted to the binary network)
Exponential-family random graph model
Network:          = flo_val
Nodes:            = 16
Ties:             = 20
Directed:         = No

```

Estimation: = MPLE

flo_val	Coefficient	Std. err.	z	P> z	[95% conf. interval]	
edges	-1.871802	.3797773	-4.93	0.000	-2.616152	-1.127452
nodematch_seat	.4855078	.4983949	0.97	0.330	-.4913282	1.462344

Maximum pseudolikelihood estimate (not full ERGM maximum likelihood unless the model is dyad-independent).

Null Deviance: 166.4 on 120 degrees of freedom
 Residual Deviance: 107.2 on 118 degrees of freedom
 Log-likelihood: -53.5846 (df = 2) AIC: 111.2 BIC: 116.7 (MC std. err. = 0)

7.2 A directed network with covariates: advice among lawyers

The network of a corporate law firm studied by Lazega and van Duijn (1997) has 71 lawyers, 35 associates and 36 partners, in three offices; an arc from i to j means that i names j as a source of advice. The data set is distributed with `nwcommands`; readers who reuse it elsewhere should check the terms of use of the original data. Figure 3 shows the network: with 892 arcs among 71 lawyers it is dense, so the picture is useful for the overall shape (the Boston lawyers form one large cluster, the Hartford and Providence lawyers lie to the side) and not for single ties. Of the 2,485 pairs, 192 are mutual, 508 asymmetric and 1,785 null:

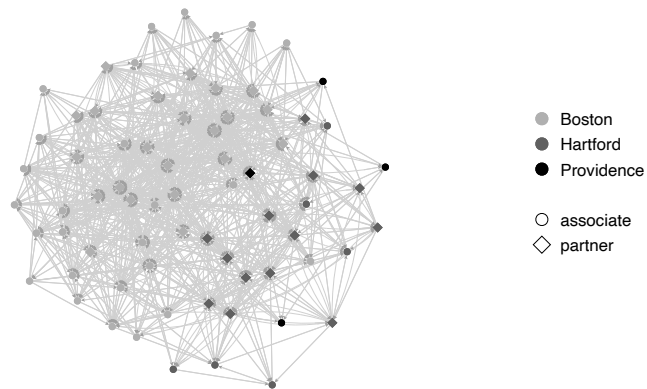


Figure 3: Advice network among 71 lawyers (nwplot lazega_adv, do-file plot_networks.do). Shading shows the office (light grey Boston, mid grey Hartford, black Providence), diamonds partners and circles associates. Arrowheads are too small to read at this density.

```
. nwwebuse lazega, nwclean
. nwsummarize lazega_adv
```

```

Network name: lazega_adv
Network id: 1
Directed: true
Valued: false
Two-mode: false
Nodes: 71
Selfloop: false
Arcs: 892
Minimum value: 0
Maximum value: 1
Density: .179
Temporal: false

. foreach v in office practice status {
2.     encode `v', generate(`v`n')
3. }

. nwdyads lazega_adv
Dyad census: lazega_adv

```

Mutual	Asym	Null
192	508	1785

Reciprocity: .274

Do lawyers seek advice in their own office and practice, and do partners ask and are they asked more often? A model with these terms is dyad-independent and is fitted by pseudolikelihood:

```
. nwergm lazega_adv, edges nodematch(officen practicen) nodeofactor(statusn) nodeifactor(statusn)
Exponential-family random graph model
Network:      = lazega_adv
Nodes:        = 71
Ties:         = 892
Directed:     = Yes
Estimation:   = MPLE
```

lazega_adv	Coefficient	Std. err.	z	P> z	[95% conf. interval]	
edges	-4.095448	.1309373	-31.28	0.000	-4.352081	-3.838816
nodematch_officen	1.422373	.0887433	16.03	0.000	1.248439	1.596306
nodematch_practicen	1.037945	.0837376	12.40	0.000	.8738223	1.202068
nodeofactor_statusn_2	1.125832	.0842818	13.36	0.000	.9606422	1.291021
nodeifactor_statusn_2	.7831546	.0819723	9.55	0.000	.6224919	.9438172

Maximum pseudolikelihood estimate (not full ERGM maximum likelihood unless the model is dyad-indepen
> dent).

```
Null Deviance:    6889.9 on 4970 degrees of freedom
Residual Deviance: 4009.0 on 4965 degrees of freedom
Log-likelihood: -2004.4927 (df = 5)  AIC:    4019.0  BIC:    4051.5  (MC std. err. =    0)
. estimates store m1
```

The coefficients are conditional log odds. Advice arcs within the same office are more likely by a factor of $\exp(1.42) = 4.1$ in odds than arcs across offices, and within the same practice by $\exp(1.04) = 2.8$, with the other terms held fixed. The factor terms refer to partners (**statusn** 2) against associates (1, the base): partners send advice arcs with $\exp(1.13) = 3.1$ times the odds of associates and receive them with $\exp(0.78) = 2.2$ times the odds. Adding **mutual** (not shown) and a geometrically weighted edgewise shared-partner term, which asks whether the advice arcs close triads beyond what attributes and reciprocity imply, gives a dyad-dependent model fitted by Monte Carlo maximum likelihood with the defaults of **ergm**:

```
. nwergm lazega_adv, edges mutual nodematch(officen practicen) nodeofactor(statusn) nodeifactor(stat
> usn) gwesp(.5) seed(12345)
Exponential-family random graph model
Network:      = lazega_adv
Nodes:        = 71
Ties:         = 892
Directed:     = Yes
Estimation:   = MCMLE
MCMC sample size: = 503
```

lazega_adv	Coefficient	Std. err.	z	P> z	[95% conf. interval]	
edges	-4.650677	.1645927	-28.26	0.000	-4.973273	-4.328081
mutual	1.026804	.1253031	8.19	0.000	.7812146	1.272394

nodematch_officen	1.013289	.0839443	12.07	0.000	.848761	1.177817
nodematch_practicen	.7668631	.0739739	10.37	0.000	.621877	.9118492
nodeofactor_statusn_2	.8255106	.0943506	8.75	0.000	.6405868	1.010434
nodeifactor_statusn_2	.3986274	.0841785	4.74	0.000	.2336407	.5636142
gwesp_.5	.5526487	.1049484	5.27	0.000	.3469536	.7583438

MCMLE converged after 9 iteration(s).

Evaluating log-likelihood at the estimate (bridge sampling with 16 bridges) ...

Null Deviance: 6889.9 on 4970 degrees of freedom

Residual Deviance: 3908.9 on 4963 degrees of freedom

Log-likelihood: -1954.4420 (df = 7) AIC: 3922.9 BIC: 3968.5 (MC std. err. = .17443)

. estimates store m3

Reciprocity ($\exp(1.03) = 2.8$ in odds, given the other terms) and closure ($0.55, z = 5.3$) are both clearly positive. The Monte Carlo algorithm needed nine iterations and the fit took about 5 seconds on the machine described in Section 9.3. Whether the simulation behind the estimates mixed is a separate question that `estat mcmcdiag` answers. The final sample has 503 draws, the number that the adaptive sample size reached, recorded 4,096 proposals apart (the sampler raised the requested 1,024, as the header of the listing below shows); its effective sample size is about 215–315. A fixed sample of 3,000 draws (`mcmleffectivesize(0)`) has an effective size of about 360–550; the same fit run both ways:

```
. quietly nwergm lazega_adv, edges mutual nodematch(officen practicen) nodeofactor(statusn) nodeifac
> tor(statusn) gwesp(.5) seed(12345) mcmleffectivesize(0)

. quietly estat mcmcdiag
. matrix ess_old = r(ess)

. quietly nwergm lazega_adv, edges mutual nodematch(officen practicen) nodeofactor(statusn) nodeifac
> tor(statusn) gwesp(.5) seed(12345)

. quietly estat mcmcdiag
. matrix ess_new = r(ess)
. matrix E = ess_old \ ess_new
. matrix rownames E = "fixed" "adaptive"
. matrix list E, format(%7.1f) noheader
```

	edges	mutual	nodematch_-n	nodematch_-n	nodeofacto-2	nodeifacto-2
fixed	519.4	499.8	507.8	362.7	500.9	545.5
adaptive	309.1	214.7	270.5	219.0	292.3	316.4

```

      gwesp_.5
      fixed    511.1
      adaptive   256.2
```

The diagnostics of the final fit show the same effective sizes, the Geweke statistics (one of seven flagged at the 5% level, `nodematch(practicen)` with $z = 2.4$) and the Heidelberger–Welch test (all passed); the report that follows in the output, the one of `mcmc.diagnostics()`, is omitted here:

```
. estat mcmcdiag

MCMC diagnostics for the final simulation (nwergm, method(mcmle))
Sample size:      =      503
Burn-in:         =     16384
Thinning interval: =      4096
```

Requested interval: = 1024 (raised by the adaptive sampler)
 Acceptance rate: = 0.5417

Statistic	Mean	SD	Autocorr	ESS
edges	894.0696	27.2439	0.2400	309.1
mutual	192.5686	11.9552	0.3171	214.7
nodematch_officen	673.4751	22.7139	0.3024	270.5
nodematch_practicen	612.5070	21.8276	0.3949	219.0
nodeofactor_statusn_2	616.7634	20.0480	0.2660	292.3
nodeifactor_statusn_2	561.4254	20.3215	0.3035	316.4
gwesp_.5	1289.0900	53.9639	0.2585	256.2

Geweke diagnostic (first 10% vs. last 50% of the retained sample):

Statistic	z	p-value
edges	1.748	0.0804
mutual	-0.129	0.8970
nodematch_officen	1.081	0.2798
nodematch_practicen	2.373	0.0176 *
nodeofactor_statusn_2	0.076	0.9397
nodeifactor_statusn_2	-0.253	0.8004
gwesp_.5	1.445	0.1484

Note: $|z| > 1.96$ (flagged *) rejects convergence of that parameter's own chain at the 5% level (Geweke & Porter 1992) - a large systematic difference between the retained sample's own early and late portions, not proof either way for a borderline z.

Heidelberger-Welch stationarity + halfwidth test (pvalue=.05, eps=.1):

Statistic	Stationary	Start	Halfwidth	HW test
edges	passed	1	3.0371	passed
mutual	passed	1	1.5990	passed
nodematch_officen	passed	1	2.7068	passed
nodematch_practicen	passed	1	2.8910	passed
nodeofactor_statusn_2	passed	1	2.2985	passed
nodeifactor_statusn_2	passed	1	2.2393	passed
gwesp_.5	passed	1	6.6081	passed

Note: Stationary = passed a Cramer-von-Mises test on the chain's own cumulative-sum path (Heidelberger & Welch 1983), discarding an increasing initial fraction (up to 50%) until it does - Start is the first retained iteration once it passes, FAILED means no discard fraction achieved stationarity. HW test additionally requires the retained portion's own 95% CI halfwidth to be within `eps()` (default 10%) of its own mean - a separate, stricter precision check, only meaningful once stationarity itself has passed.

MCMLE's own convergence test was satisfied after 9 iteration(s).

Note: a satisfied convergence test is a necessary check, not proof of a converged fit - inspect the autocorrelation/ESS above, and consider raising `mcmleeffectivesize()` (the target effective sample size of the adaptive sample; `mcmleiterations()` if the iterations ran out) if acceptance rate or ESS looks poor. For a fit with a fixed-size sample (`mcmleeffectivesize(0)`) raise `mcmcsamplesize()` or `mcmcinterval()` instead.

(output of the R-style report of `mcmc.diagnostics()` omitted)

Figure 4 shows the trace and density of each statistic: a trace that drifts or sticks at one level would be evidence of poor mixing even if the numerical diagnostics looked acceptable.

MCMC diagnostics: trace and density per statistic

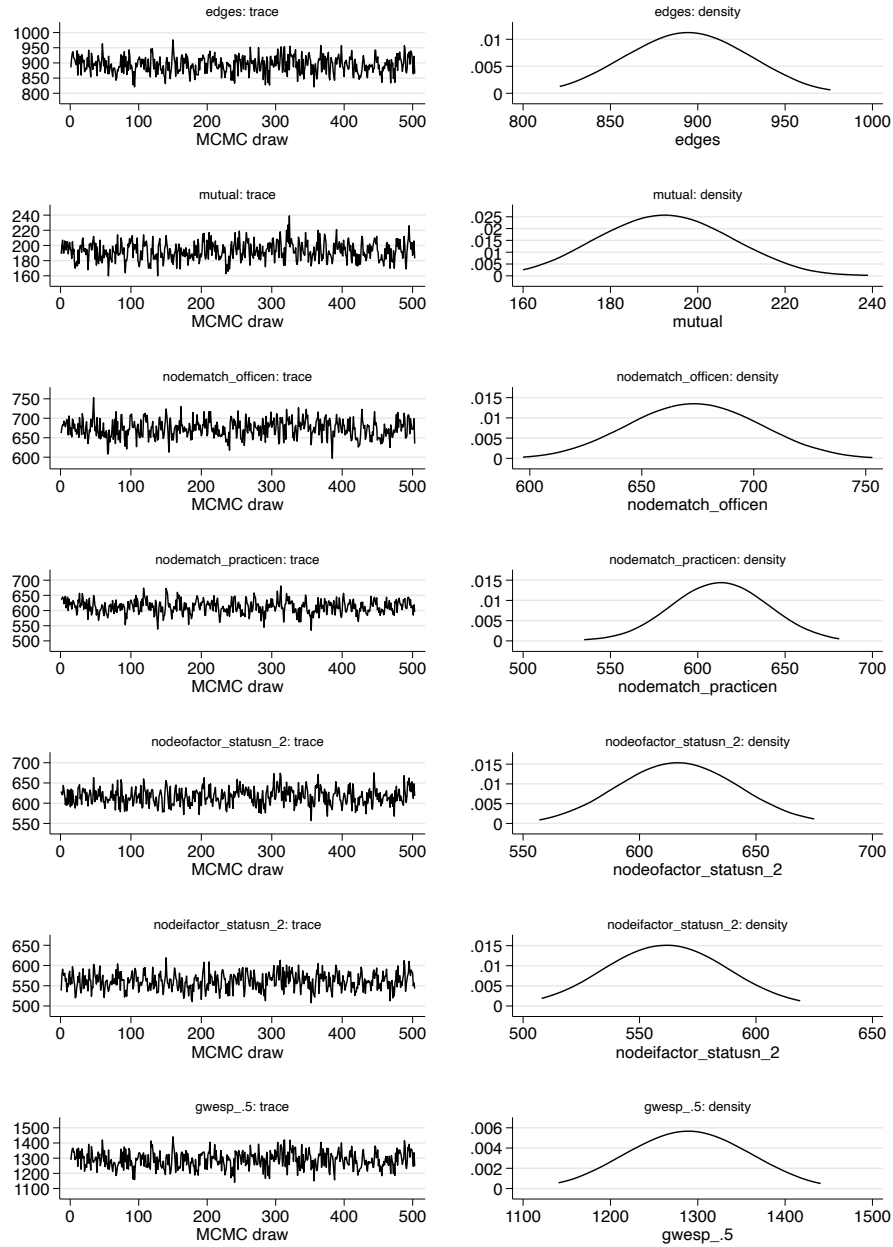


Figure 4: `estat mcmcdiag`, plot after the final advice model: trace (left) and density (right) of each model statistic in the final sample.

Information criteria and the likelihood-ratio test compare the three models (the second adds `mutual`); the closure term improves on reciprocity alone:

```
. estimates stats m1 m2 m3
Akaike's information criterion and Bayesian information criterion
```

Model	N	ll(null)	ll(model)	df	AIC	BIC
m1	4,970	-3444.941	-2004.493	5	4018.985	4051.541
m2	4,970	-3444.941	-1970.714	6	3953.428	3992.495
m3	4,970	-3444.941	-1954.442	7	3922.884	3968.462

```
Note: BIC uses N = number of observations. See [R] IC note.

. lrtest m2 m3
Likelihood-ratio test
Assumption: m2 nested within m3
LR chi2(1) = 32.54
Prob > chi2 = 0.0000
```

Goodness of fit asks whether the model reproduces features of the observed network that it was not fitted to. The model's own statistics are reproduced (by construction, at the estimate) and so are most of the geodesic distances except the pairs at distance 2 and 3, where the model produces too many pairs at distance 2 (3,287 simulated against 3,074 observed) and too few at distance 3 (731 against 991), with Monte Carlo p -values of 0.00 and 0.02:

```
. estat gof, gof(model distance) seed(12345)
Goodness-of-fit for model statistics
```

	obs	min	mean	max	MC p-value
edges	892	830	890.64	961	1.00
mutual	192	159	191.18	214	1.00
nodematch_offi-n	674	618	672.11	728	0.90
nodematch_prac-n	610	571	610.28	671	0.98
nodeofactor_st-2	616	576	617.87	672	1.00
nodeifactor_st-2	562	509	561.06	603	1.00
gwesp_.5	1284.861	1152.664	1282.78	1425.7373	0.92

```
Goodness-of-fit for minimum geodesic distance
```

	obs	min	mean	max	MC p-value
1	892	830	890.64	961	1.00
2	3074	3080	3286.76	3439	0.00
3	991	580	730.97	994	0.02
4	13	0	19.97	100	0.94
5	0	0	0.39	12	1.00
Inf	0	0	41.27	140	1.00

Figure 5 shows the degree and distance distributions. The observed degree distribution lies mostly inside the simulated ranges, whereas in the distance panel the observed proportion of pairs at distance 2 lies below, and at distance 3 above, the simulated boxes, as the table shows. Rough agreement is evidence for a model, not proof of it.

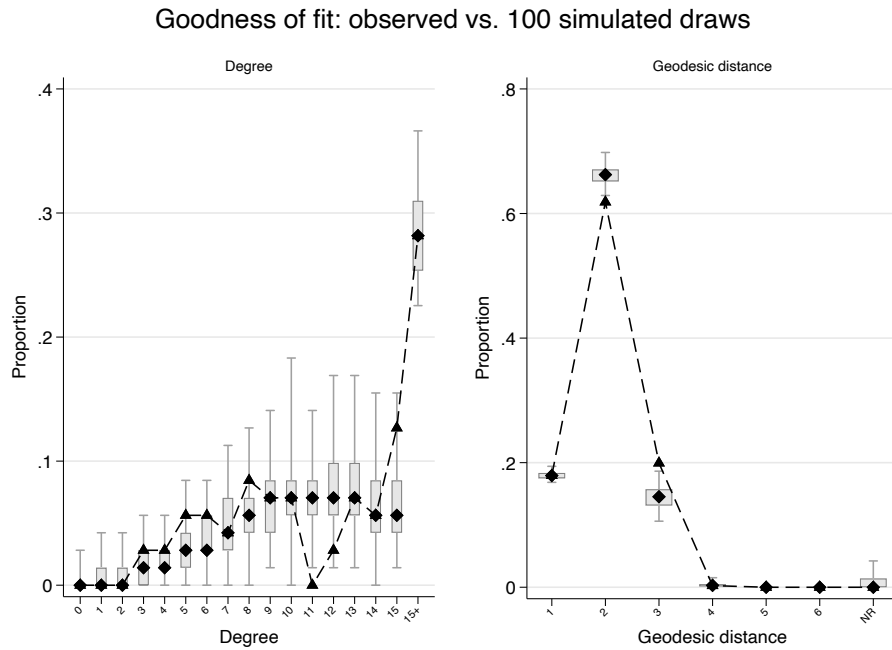


Figure 5: `estat gof`, plot after the final advice model: degree (left) and geodesic-distance (right) distributions, the observed network (dashed line) against 100 simulated networks (boxes and whiskers).

A fitted model is also a generator of networks. The statistics of 100 networks drawn at the fitted coefficients (here the seven model statistics, in the order of the table) have the means of the goodness-of-fit table:

```
. nwergm simulate, nsim(100) output(stats) seed(12345)
. matrix S = r(stats)
. preserve
. clear
. set obs 100
Number of observations (_N) was 0, now 100.
. svmat double S, names(s)
number of observations will be reset to 100
Press any key to continue, or Break to abort
Number of observations (_N) was 0, now 100.
. summarize s1-s7
```

Variable	Obs	Mean	Std. dev.	Min	Max
s1	100	894.29	28.27973	837	968
s2	100	193.82	11.72787	169	232
s3	100	677.83	21.20037	632	730
s4	100	611.22	22.39344	568	674

s5	100	619.3	22.40062	554	682
s6	100	564.59	19.6834	514	626
s7	100	1291.051	56.57726	1165.243	1462.178

```
. restore
```

7.3 A two-mode network: the Southern women

The 18 women and 14 social events of the classic data set of Davis et al. (2022) form a two-mode network of 89 attendances (analyzed in Breiger, 1974). With `title` (Miss or Mrs) as the only attribute of the women, the terms of two-mode networks count the nodes of one mode: `bfactor1()` is R's `bifactor`, and `bgwdegree2()` and `bgwdsp1()` are the geometrically weighted degree of the events and the geometrically weighted dyadwise shared partners of the women. All three models have the same attribute and differ in the structural term; the bipartite goodness of fit adds the degree distributions of both modes:

```
. nwwebuse davis, nwcLEAR
. encode title, generate(titlen)
. nwsummarize davis
```

```

Network name:  davis
Network id:    1
Directed:     false
Valued:       false
Two-mode:     true
Nodes:        32
  Level 1:    18 (woman)
  Level 2:    14 (event)
Selfloop:     false
Edges:        89
Minimum value: 0
Maximum value: 1
Density:      .353
Temporal:     false

. nwerGM davis, edges bfactor1(titlen)
Exponential-family random graph model
Network:      = davis
Nodes:        = 32
Ties:         = 89
Directed:     = No
Estimation:   = MPLE
```

	davis	Coefficient	Std. err.	z	P> z	[95% conf. interval]
edges		-.5877867	.168175	-3.50	0.000	-.9174036 -.2581697
bifactor_titlen_2		-.0447359	.2707759	-0.17	0.869	-.5754469 .4859751

Maximum pseudolikelihood estimate (not full ERGM maximum likelihood unless the model is dyad-independant).

```

      Null Deviance:    349.3 on 252 degrees of freedom
    Residual Deviance:    327.3 on 250 degrees of freedom
Log-likelihood: -163.6325 (df = 2)  AIC:    331.3  BIC:    338.3  (MC std. err. =    0)
```

```
. estimates store d1
. nwergm davis, edges bfactor1(titlen) bgwdegree2(1) seed(1)
Exponential-family random graph model
Network:          = davis
Nodes:            = 32
Ties:             = 89
Directed:         = No
Estimation:       = MCMLE
MCMC sample size: = 486
```

	davis	Coefficient	Std. err.	z	P> z	[95% conf. interval]	
edges		-.3245924	.1749978	-1.85	0.064	-.6675817	.0183969
bfactor_titlen_2		-.041804	.2601426	-0.16	0.872	-.5516741	.4680662
bgwdegree2_1		-2.337552	.8092405	-2.89	0.004	-3.923634	-.75147

MCMLE converged after 1 iteration(s).

Evaluating log-likelihood at the estimate (bridge sampling with 16 bridges) ...

Null Deviance: 349.3 on 252 degrees of freedom

Residual Deviance: 323.3 on 249 degrees of freedom

Log-likelihood: -161.6496 (df = 3) AIC: 329.3 BIC: 339.9 (MC std. err. = .06)

```
. estimates store d2
. nwergm davis, edges bfactor1(titlen) bgwdsp1(.5) seed(1)
```

Exponential-family random graph model

```
Network:          = davis
Nodes:            = 32
Ties:             = 89
Directed:         = No
Estimation:       = MCMLE
MCMC sample size: = 486
```

	davis	Coefficient	Std. err.	z	P> z	[95% conf. interval]	
edges		-1.176527	.2987292	-3.94	0.000	-1.762026	-.5910288
gwbldsp_fixed_.5		.3024289	.1292092	2.34	0.019	.0491835	.5556743
bfactor_titlen_2		-.0664987	.3370617	-0.20	0.844	-.7271276	.5941301

MCMLE converged after 2 iteration(s).

Evaluating log-likelihood at the estimate (bridge sampling with 16 bridges) ...

Null Deviance: 349.3 on 252 degrees of freedom

Residual Deviance: 321.9 on 249 degrees of freedom

Log-likelihood: -160.9260 (df = 3) AIC: 327.9 BIC: 338.4 (MC std. err. = .04524)

```
. estimates store d3
. estimates stats d1 d2 d3
```

Akaike's information criterion and Bayesian information criterion

Model	N	ll(null)	ll(model)	df	AIC	BIC
d1	252	-174.6731	-163.6325	2	331.2651	338.324
d2	252	-174.6731	-161.6496	3	329.2992	339.8875
d3	252	-174.6731	-160.926	3	327.8519	338.4402

Note: BIC uses N = number of observations. See [R] IC note.

Whether a woman is a Miss or a Mrs has no effect. Women who attend the same events (shared partners, 0.30 with $z = 2.3$) cluster beyond the density of attendances, and the degrees of the events are more unevenly distributed than a model with only a density term implies (the coefficient of `bgwdegree2` is -2.3 , $z = -2.9$ in the second model); the information criteria slightly prefer the shared-partner model. The goodness of fit of that model:

```
. estat gof, seed(1)
```

Goodness-of-fit for model statistics

	obs	min	mean	max	MC p-value
edges	89	78	89.74	106	1.00
gwbids_fixed_5	185.25809	147.01635	185.62	220.81434	0.84
bifactor_title-2	34	27	34.68	45	1.00

Goodness-of-fit for minimum geodesic distance

	obs	min	mean	max	MC p-value
1	89	78	89.74	106	1.00
2	205	185	219.71	240	0.14
3	163	137	160.79	174	0.82
4	39	4	23.60	44	0.12
5	0	0	0.61	5	1.00
Inf	0	0	1.55	31	1.00

Goodness-of-fit for bipartition 1 degree

	obs	min	mean	max	MC p-value
0	4	4	4.15	5	1.00
1	0	0	0.48	4	1.00
2	2	0	1.20	5	0.62
3	3	0	1.97	6	0.58
4	2	0	2.37	7	1.00
5	2	0	2.32	6	1.00
6	2	0	2.09	5	1.00
7	1	0	1.52	4	1.00
8	0	0	0.90	3	0.78
9	0	0	0.62	3	1.00
10	0	0	0.24	2	1.00
11	2	0	0.11	1	0.00
12	0	0	0.02	1	1.00
13	0	0	0.01	1	1.00

Goodness-of-fit for bipartition 2 degree

	obs	min	mean	max	MC p-value
0	0	0	0.01	1	1.00
1	0	0	0.03	1	1.00
2	2	0	0.36	2	0.08
3	1	0	1.60	6	1.00
4	4	0	3.45	7	0.92
5	1	0	3.81	8	0.12
6	1	0	2.63	6	0.50
7	2	0	1.45	4	0.96
8	3	0	0.48	2	0.00
9	0	0	0.16	2	1.00
10	0	0	0.02	1	1.00

Goodness-of-fit for dyadwise shared partner

	obs	min	mean	max	MC p-value
0	91	68	86.79	108	0.56
1	40	24	54.88	90	0.16
2	50	38	54.82	72	0.50

3	32	12	31.67	53	1.00
4	17	2	11.69	33	0.32
5	7	0	3.24	11	0.22
6	5	0	0.72	4	0.00
7	1	0	0.14	2	0.24
8	1	0	0.03	1	0.06
9	0	0	0.01	1	1.00
11	0	0	0.01	1	1.00

The statistics of the model and most counts of the degree and shared-partner distributions are reproduced. Three counts at the upper end are not: the two women who attended 11 events (simulated mean 0.11), the three events with 8 attendees (0.48) and the five dyads with six shared partners (0.72) all have a Monte Carlo p -value of 0.00, so the model underproduces the extremes. With 18 women and 14 events the example shows the syntax and the output rather than a finding.

7.4 Curved terms and operators

A geometrically weighted term with an estimated decay is a curved term. On the friendship network of the students of a school (Faux Mesa High, 205 students, 203 ties, `mesa`), the closure term `gwespfree(.25)` estimates a weight and a decay; the number in parentheses is the starting value. This is a Monte Carlo fit that takes about 15 seconds on the machine used:

```
. nwwebuse mesa, nwcLEAR
. encode race, generate(racen)
. nwerGM mesa, edges nodematch(grade racen) gwespfree(.25) seed(1)
Exponential-family random graph model
Network:          = mesa
Nodes:            = 205
Ties:             = 203
Directed:         = No
Estimation:       = MCMLE
MCMC sample size: = 763
```

mesa	Coefficient	Std. err.	z	P> z	[95% conf. interval]	
edges	-6.358761	.1581786	-40.20	0.000	-6.668785	-6.048736
nodematch_grade	1.960923	.1788696	10.96	0.000	1.610345	2.311501
nodematch_racen	.2827783	.1125541	2.51	0.012	.0621763	.5033802
gwesp_weight	1.348847	.1375028	9.81	0.000	1.079347	1.618348
gwesp_decay	.3807975	.1008682	3.78	0.000	.1830995	.5784955

```
MCMLE converged after 6 iteration(s).
(decay is an estimated curved parameter here - see help nwerGM, gw_effects for how this differs from
> R ergm's own curved-model machinery)
```

Homophily in grade is strong, homophily in race is modest, and a clustering weight of 1.35 and a decay of 0.38 describe how fast the effect of an additional shared partner declines. The likelihood and the information criteria are not available after a curved fit by Monte Carlo maximum likelihood.

An operator evaluates a term on a derived network. The advice network is a directed network; on the friendship network of the lawyers, whether friendship triads close among

reciprocated friendships alone is asked by applying `triangle` to the undirected network in which two lawyers are tied if the friendship is mutual:

```
. nwwebuse lazega, nwcLEAR
. foreach v in office practice status {
2.     encode `v', generate(`v`n')
3. }

. nwerGM lazega_fr, edges mutual nodematch(practicen) operators(Symmetrize(triangle, rule(strong)))
> seed(1)
```

Exponential-family random graph model

```
Network:      = lazega_fr
Nodes:        = 71
Ties:         = 575
Directed:     = Yes
Estimation:   = MCMLE
MCMC sample size: = 266
```

lazega_fr	Coefficient	Std. err.	z	P> z	[95% conf. interval]	
edges	-2.736101	.0829356	-32.99	0.000	-2.898652	-2.57355
mutual	.900128	.1705302	5.28	0.000	.5658949	1.234361
nodematch_practicen	.8942052	.1007393	8.88	0.000	.6967598	1.091651
Symmetrize(strong)-triangle	.7278742	.4769188	1.53	0.127	-.2068695	1.662618

MCMLE converged after 2 iteration(s).

Evaluating log-likelihood at the estimate (bridge sampling with 16 bridges) ...

Null Deviance: 6889.9 on 4970 degrees of freedom

Residual Deviance: 3404.2 on 4966 degrees of freedom

Log-likelihood: -1702.0947 (df = 4) AIC: 3412.2 BIC: 3438.2 (MC std. err. = .08466)

Closure among mutual friendships is positive but not significant at the 5% level (0.73, $z = 1.5$) on top of reciprocity and same-practice homophily.

7.5 Simulating from given coefficients

`nwerGM simulate` with a number of nodes draws networks from a fully specified model; the coefficients are given in the order of the terms:

```
. nwerGM simulate 16, edges gwesp(.5) theta(-2 .5) seed(42) nsim(3)
. nwsummarize ergmsim_1
```

```
Network name:  ergmsim_1
Network id:   4
Directed:     false
Valued:       false
Two-mode:     false
Nodes:        16
Selfloop:     false
Edges:        5
Minimum value: 0
Maximum value: 1
Density:      .0417
Temporal:     false
```

8 Implementation

Native plugin. The Metropolis–Hastings sampler, the change statistics of the terms, the observed statistics, the design matrix of the pseudolikelihood and the Newton iterations of curved pseudolikelihood fits are written in C and run in a Stata plugin, which `nwerm` uses without any option to set; `e(native)` records that it ran. The estimating step of the Monte Carlo algorithm, the covariance, the diagnostics and the output are written in Mata and Stata. The plugin is built by a continuous-integration workflow for macOS, Linux and Windows and is installed with the package. The sampling and observation constraints and the operators need the plugin and stop with a message if it is missing.

No slow fallback in use. The first release of `nwerm` was a pure Mata implementation. A Mata implementation of the original terms still exists and is used when the plugin is missing for the platform, when the option `nonative` is given (for testing and comparison) or when a model exceeds the capacity of the plugin. It is much slower than the plugin (the Florentine model with `gwesp(.5)` of Section 7.1 took 0.2 seconds in the plugin and 16 seconds with `nonative`, with the same seed and the defaults, a factor of about 70; the Lazega model of Section 7.2 with burn-in 3,000 and interval 50 took 5 seconds in the plugin, and with `nonative` did not finish in the 10 minutes allowed), and `nwerm` prints one warning line before a Monte Carlo fit, `estat gof` or `nwerm simulate` that uses it. The terms in `structural()`, the constraints, the observation constraints and the operators exist in the plugin only. `nwerm` is therefore a native implementation in the sense that the work is done in compiled code that ships with the package, not in an interpreted fallback.

Mata library. The commands share a Mata library (`lnwerm.mlib`), which contains the class definitions of the core library it was compiled against. The package checks at the start of a command that the core and the package libraries match and prints how to reinstall if they do not. Both libraries are built with Stata 17 and work in Stata 17 and 18.

9 Comparison with R

9.1 How the terms were validated

`nwerm` was developed against `ergm` 4.12.0 (R 4.6.0). For each term, the following were compared with `ergm`: (a) the observed statistic, as `summary(formula)`, on at least two networks (a random network and a real one) of each applicable type, to a tolerance of 10^{-6} or better; (b) the maximum pseudolikelihood coefficients and standard errors, as `ergm(estimate="MPLE")`, to 10^{-4} in the coefficient; and (c) the Monte Carlo maximum likelihood estimates, over several seeds on each side, which should differ by less than the seed-to-seed spread of `ergm`'s own runs. For the two-mode terms the whole design matrix of change statistics was compared with R's `ergmMPLE()`. Where a term of `ergm` 4.12.0 returns a wrong change statistic, `nwerm` uses the correct one and the statistic still equals R's; this applies to `opentriad`, for which the pseudolikelihood and Monte Carlo estimates therefore differ from those of `ergm` 4.12.0. These checks are repeated by the test suite of the package (about 60 test files), which runs as part of every release.

`nwergm` version prints the version of `ergm` used.

9.2 Estimates on the same data

Table 2 fits the models of Section 7 (and one small network) in `ergm` 4.12.0 and `nwergm` on the same networks, with the default settings of each program. For the Monte Carlo fits, each program was run with several seeds; the table gives the mean over seeds and the seed-to-seed standard deviation of each program, and the difference between the means in units of the standard error that `ergm` reports.

For the dyad-independent models (the Florentine and the Lazega fits by pseudolikelihood) the programs give the same numbers: the largest difference in a coefficient is 3×10^{-8} and in a standard error 10^{-5} . For the Monte Carlo fits, the means over seeds of the two programs differ by at most 0.09 standard errors of `ergm` for every coefficient in the table, including the operator term of the Lazega friendship model (0.08) and the curved terms (below), and the mean standard errors of the two programs agree within 4%. Both programs' own seed-to-seed standard deviations are far smaller than the standard errors of the estimates: for the final advice model, between 0.002 and 0.010 for every coefficient and for both programs, against standard errors of 0.08 to 0.17. Seed-to-seed differences between the two programs of a few hundredths of a standard error are therefore Monte Carlo noise of both programs, not a systematic difference. On the five-node network of the first block, which has 6 ties and essentially no information, both programs' means over ten seeds are close relative to the standard errors, which are 0.7 and 1.5.

Two comparisons are not like for like. On the Lazega friendship model, `ergm` stopped with “unconstrained MCMC sampling did not mix at all” for one of five seeds, so the `ergm` column rests on four fits; all five fits of `nwergm` converged. On the curved model for Faux Mesa High, `ergm` stopped for two of three seeds because the number of two-paths on an edge exceeded the cut-off of 30 of its curved shared-partner term (the term's argument `cutoff` can be raised), so the `ergm` column rests on one fit and the three `nwergm` fits all converged. That single `ergm` fit and the mean of the `nwergm` fits differ by 0.09 standard errors or less in every coefficient (0.02 in the weight of the curved term and 0.07 in its decay); with one `ergm` fit the table cannot show how large the seed-to-seed spread of `ergm` is for this model.

9.3 Speed

Table 3 gives the median time of seven fits of each program on identical networks, each with its default control settings, a default-workflow comparison rather than one with matched samplers. The machine is an Apple M5 Pro (18 cores, macOS 26.5.1) with R 4.6.0, `ergm` 4.12.0 and Stata/BE 18. It was otherwise idle: the one-minute load average read before each fit was 3.0 to 7.0 (median 3.3), of which about one is the fit itself. The two programs were run in turn, one fit at a time (`ergm`, then `nwergm`, then the next seed), so that a slow drift of the machine affects both equally; each fit is a separate process with one untimed warm-up fit, and the time is that of the estimation call alone (`ergm()` in R, the `nwergm` command in Stata). The first four networks are the exported synthetic networks of earlier benchmarks of the package, the next two were

Table 2: Coefficients of `ergm` 4.12.0 and `nwergm` 1.3.0 on the same networks, default settings. Mean over seeds (standard deviation over seeds in parentheses); Δ is the difference of the means in units of the standard error that `ergm` reports.

Term	ergm		nwergm		Δ
	mean	(sd)	mean	(sd)	
<i>5-node directed network, MCMLE (10 seeds)</i>					
edges	-0.954	(0.050)	-0.932	(0.048)	0.03
mutual	0.285	(0.093)	0.212	(0.096)	-0.05
<i>Florentine marriages, MPLE</i>					
edges	-2.825		-2.825		0.00
nodematch(seat)	0.454		0.454		0.00
nodecov(wealth)	0.010		0.010		0.00
<i>Florentine marriages, MCMLE (10 seeds)</i>					
edges	-1.961	(0.034)	-1.973	(0.019)	-0.03
nodematch(seat)	0.485	(0.040)	0.474	(0.018)	-0.02
gwesp(.5)	0.095	(0.010)	0.105	(0.012)	0.04
<i>Lazega advice, MPLE</i>					
edges	-4.095		-4.095		0.00
nodematch(office)	1.422		1.422		0.00
nodematch(practice)	1.038		1.038		0.00
nodeofactor(status=2)	1.126		1.126		0.00
nodeifactor(status=2)	0.783		0.783		0.00
<i>Lazega advice, MCMLE (5 seeds)</i>					
edges	-4.648	(0.007)	-4.658	(0.008)	-0.06
mutual	1.034	(0.010)	1.030	(0.006)	-0.03
nodematch(office)	1.006	(0.005)	1.008	(0.004)	0.02
nodematch(practice)	0.759	(0.006)	0.761	(0.003)	0.02
nodeofactor(status=2)	0.821	(0.004)	0.827	(0.004)	0.07
nodeifactor(status=2)	0.397	(0.005)	0.397	(0.006)	0.00
gwesp(.5)	0.557	(0.003)	0.561	(0.002)	0.04
<i>Lazega friendship, operator (see text)</i>					
edges	-2.736	(0.007)	-2.738	(0.001)	-0.02
mutual	0.921	(0.019)	0.915	(0.009)	-0.03
nodematch(practice)	0.895	(0.002)	0.896	(0.005)	0.00
Symmetrize(strong)~triangle	0.671	(0.036)	0.712	(0.018)	0.08
<i>Southern women, two-mode MCMLE (10 seeds)</i>					
edges	-0.316	(0.013)	-0.324	(0.016)	-0.04
b1factor(title=2)	-0.045	(0.022)	-0.043	(0.021)	0.01
gwb2degree(1)	-2.360	(0.053)	-2.315	(0.051)	0.05
<i>Faux Mesa High, MCMLE (5 seeds)</i>					
edges	-6.412	(0.004)	-6.412	(0.010)	0.00
nodematch(grade)	1.991	(0.005)	1.986	(0.007)	-0.03
nodematch(race)	0.285	(0.003)	0.288	(0.016)	0.02
gwesp(.25)	1.452	(0.007)	1.454	(0.007)	0.02
<i>Faux Mesa High, curved gwesp (see text)</i>					
edges	-6.369		-6.356	(0.003)	0.08
nodematch(grade)	1.973		1.969	(0.014)	-0.02
nodematch(race)	0.279		0.268	(0.013)	-0.09
gwesp weight	1.343		1.346	(0.006)	0.02
gwesp decay	0.389		0.382	(0.003)	-0.07

regenerated with the same seeds, and the last two are the Lazega and Faux Mesa High fits of Section 7. The undirected networks are loaded as undirected in both programs.

Table 3: Median time of seven fits (seeds 1 to 7, fitted in turn on an idle machine) of **ergm** 4.12.0 and of **nvergm** 1.3.0 on identical networks, default settings; ratio of the medians, **ergm** divided by **nvergm**; Monte Carlo iterations in the seven fits of each program (minimum to maximum).

Benchmark	ergm (s)	nvergm (s)	ratio	ergm it.	nvergm it.
30 nodes, directed: edges, mutual	0.20	0.04	5.4	1	1
100 nodes, directed: + nodematch	0.30	0.06	4.7	1–3	1
100 nodes, undirected: edges, gwesp	0.32	0.07	4.6	1	1
500 nodes, sparse, undirected: + nodematch, gwesp	2.9	0.31	9.4	3–5	1–3
1000 nodes, directed: edges, mutual, nodematch	8.4	1.1	7.9	2–3	1–4
1000 nodes, sparse, undirected: edges, gwesp, gwnsp	7.1	2.5	2.9	2–6	1–4
Lazega advice, 71 nodes: seven terms	10.4	3.1	3.4	6–8	4–12
Faux Mesa High, 205 nodes: gwesp(.25)	4.0	1.4	2.9	5–11	5–11

With the defaults of both programs, **nvergm** was faster than **ergm** on all eight models, by factors of 2.9 (Faux Mesa High and the sparse 1000-node network) to 9.4 (the 500-node sparse network); the two fits of 1000 nodes take 1.1 and 2.5 seconds against 8.4 and 7.1 seconds, and the fits of the small networks 0.04 to 0.07 seconds against 0.20 to 0.32. Every call has a fixed overhead of the order of 0.05 to 0.25 seconds (parsing, set-up, returning the results), which is most of the time of the small models. The reason for the speed is, first, the sample size: **nvergm** follows **ergm** in adapting the sample to an effective size of 64 (Section 5) and iterates like it (1 to 4 iterations on the first six models and 4 to 12 on the Lazega model; **ergm**: 1 to 11 and 6 to 8), whereas a fixed sample of 3,000 draws per iteration, with a further sample for the covariance and 16 bridge samples of 3,000 draws for the log-likelihood, costs more whatever the model. The final sample of the Lazega fit has 503 draws, those of three **ergm** fits that I inspected 774 to 941, and the effective sizes of the two programs are of the same order: 215 to 316 for **nvergm** and 240 to 500 for **ergm** (Section 7.2); a larger effective size can be asked for with `mcmleeeffectivesize()` at the cost of time. Second, the compiled sampler is fast per proposal, in particular for **gwesp** and **gwnsp** on larger networks. The times of both programs vary with the seed, because the number of iterations does (for example the sparse 1000-node model took 1.7 to 3.0 seconds in **nvergm** and 3.6 to 11.3 seconds in **ergm**), and the order of the programs on a single fit could differ from the order of the medians; the ratios are those of the medians. The speed is not bought with agreement: the estimates are those compared with **ergm**'s in Table 2.

Three limits of this comparison. The timings were made with **nvergm** 1.3.0; version 1.3.1 changes the report of `estat mcmcdiag` and the documentation, not the sampler or the estimation, and was not timed again. They are for Stata 18 on macOS; Stata 17, Linux and Windows were not timed. And they are for the eight models of the table: the speed relative to **ergm** depends on the model and the data, and a model that runs in the Mata sampler instead of the plugin (Section 8) is much slower than **ergm**.

10 What is not implemented

Of the 139 terms of **ergm** 4.12.0, 14 are not implemented: the 13 terms for valued networks (**atleast**, **atmost**, **cdf**, **cyclicalweights**, **equalto**, **greaterthan**, **ininterval**, **nodecovar**, **nodeicovar**, **nodeocovar**, **smallerthan**, **sum**, **transitiveweights**) and the operator **B**. **nwergm** models the presence of ties; a valued network is fitted as the binary graph (Section 4), and the values are not an outcome. Two terms are implemented in part: **Curve()** for linear maps of the coefficients (**map(rep)** and **map(matrix())**; not **minpar()**, **maxpar()** or **cov()**), and **altkstar** with an estimated decay, which is available by pseudolikelihood with starting values from **init()** (**ergm** itself cannot fit it by Monte Carlo maximum likelihood on the test networks). Further limits are: the two-sided form of **attrs** in **S()**, dyadic covariates inside **S()** and **Project()**, and curved terms inside an operator; the constraints **hamming**, **ChangeStats()**, **Dyads()** and **I()**; missing dyads together with the constraints **freedyads()**, **blockdiag()** or **fixdensity**; directed two-mode networks; and the packages built on **ergm**, **tergm** for dynamic networks, **ergm.count** for count data and **ergm.multi** for several networks. The log-likelihood is not available after a curved fit by Monte Carlo maximum likelihood. Finally, **ergm** remains the reference implementation and the more complete program: where it is available it is the right tool for models that **nwergm** lacks, and the comparison above can be repeated on the user's own data. The manual (Grund, 2026c) lists the status of every term of **ergm**.

11 Conclusions

nwergm brings exponential-family random graph models to Stata: 125 of the 139 terms of **ergm** 4.12.0, four estimators with the defaults of **ergm**, constraints, missing-data handling, diagnostics, goodness of fit, information criteria, prediction and simulation, in a Monte Carlo sampler that runs as compiled code. On the same data, the estimates agree with those of **ergm** within Monte Carlo error. With the defaults of **ergm**, including its adaptive Monte Carlo sample size, the current version fits the eight benchmark models 2.9 to 9.4 times faster than **ergm**. Because an ERGM is fitted by simulation, the diagnostics deserve the attention of the user: the effective sample size, the trace plots and a goodness-of-fit check on features the model was not fitted to are as much part of the analysis as the coefficient table.

Also see

nwcommands (Grund, 2026a); further model packages and extensions: <https://nwcommands.org>.

Acknowledgments

nwergm would not exist without the statnet project. The **ergm** package (Hunter and Handcock, 2006; Handcock et al., 2008), developed by the statnet team and many contributors, is the reference implementation that **nwergm** follows and is compared with; its source code and documentation were the specification for the Stata code. **nwergm** is distributed under the GNU General Public License, as is **ergm**; it is not affiliated with

or endorsed by the statnet project, and errors are mine. I am grateful to the developers for making their software public.

Use of AI in developing the software

The software was developed with the help of an AI coding assistant (Claude Code, Anthropic), working under my direction. I specified the scope, reviewed the code, and am responsible for the package.

About the author

Thomas U. Grund is a sociologist at the Institute of Sociology, RWTH Aachen University, working on social network analysis and computational social science. He is the author of `nwcommands`, a Stata package for the analysis of network data.

12 References

- Breiger, R. L. 1974. The Duality of Persons and Groups. *Social Forces* 53(2): 181.
- Davis, A., B. B. Gardner, and M. R. Gardner. 2022. *Deep South*. University of Chicago Press. Reissue of the 1941 edition, with a foreword by Isabel Wilkerson.
- van Duijn, M. A. J., K. J. Gile, and M. S. Handcock. 2009. A Framework for the Comparison of Maximum Pseudo-Likelihood and Maximum Likelihood Estimation of Exponential Family Random Graph Models. *Social Networks* 31(1): 52–62.
- Firth, D. 1993. Bias Reduction of Maximum Likelihood Estimates. *Biometrika* 80(1): 27–38.
- Geyer, C. J., and E. A. Thompson. 1992. Constrained Monte Carlo Maximum Likelihood for Dependent Data. *Journal of the Royal Statistical Society, Series B* 54(3): 657–683.
- Grund, T. U. 2026a. `nwcommands`: Scalable and Extensible Network Analysis in Stata. Manuscript under review.
- . 2026c. `nwergm`: Exponential Random Graph Models in Stata. Manual. Zenodo. <https://doi.org/10.5281/zenodo.23661331>.
- Handcock, M. S., and K. J. Gile. 2010. Modeling Social Networks from Sampled Data. *The Annals of Applied Statistics* 4(1): 5–25.
- Handcock, M. S., D. R. Hunter, C. T. Butts, S. M. Goodreau, and M. Morris. 2008. `statnet`: Software Tools for the Representation, Visualization, Analysis and Simulation of Network Data. *Journal of Statistical Software* 24(1): 1–11.
- Hummel, R. M., D. R. Hunter, and M. S. Handcock. 2012. Improving Simulation-Based Algorithms for Fitting ERGMs. *Journal of Computational and Graphical Statistics* 21(4): 920–939.

- Hunter, D. R. 2007. Curved Exponential Family Models for Social Networks. *Social Networks* 29(2): 216–230.
- Hunter, D. R., S. M. Goodreau, and M. S. Handcock. 2008a. Goodness of Fit of Social Network Models. *Journal of the American Statistical Association* 103(481): 248–258.
- Hunter, D. R., and M. S. Handcock. 2006. Inference in Curved Exponential Family Models for Networks. *Journal of Computational and Graphical Statistics* 15(3): 565–583.
- Hunter, D. R., M. S. Handcock, C. T. Butts, S. M. Goodreau, and M. Morris. 2008b. ergm: A Package to Fit, Simulate and Diagnose Exponential-Family Models for Networks. *Journal of Statistical Software* 24(3).
- Krivitsky, P. N. 2017. Using Contrastive Divergence to Seed Monte Carlo MLE for Exponential-Family Random Graph Models. *Computational Statistics & Data Analysis* 107: 149–161.
- Krivitsky, P. N., D. R. Hunter, M. Morris, and C. Klumb. 2023. ergm 4: New Features for Analyzing Exponential-Family Random Graph Models. *Journal of Statistical Software* 105(6).
- Lazega, E., and M. van Duijn. 1997. Position in Formal Structure, Personal Characteristics and Choices of Advisors in a Law Firm: A Logistic Regression Model for Dyadic Network Data. *Social Networks* 19(4): 375–397.
- Lusher, D., J. Koskinen, and G. Robins, eds. 2013. *Exponential Random Graph Models for Social Networks: Theory, Methods, and Applications*. New York: Cambridge University Press.
- Morris, M., M. S. Handcock, and D. R. Hunter. 2008. Specification of Exponential-Family Random Graph Models: Terms and Computational Aspects. *Journal of Statistical Software* 24(4): 1–24.
- Padgett, J. F., and C. K. Ansell. 1993. Robust Action and the Rise of the Medici, 1400–1434. *American Journal of Sociology* 98(6): 1259–1319.