

Fast estimation of stochastic actor-oriented network models in Stata

Thomas U. Grund

Institute of Sociology, RWTH Aachen University, Germany

Abstract. `nwsaom` fits stochastic actor-oriented models (Siena models) to panel network data in Stata. It uses the same method-of-moments algorithm and the same effects as RSiena, entirely within Stata, and implements 1,058 of the 1,121 effects in the effects table of RSiena 1.6.6 that it covers (about 94%). Beyond one network, it handles undirected and two-mode networks, several networks and behaviors, and network-behavior co-evolution; it also offers maximum-likelihood and Bayesian estimation, goodness of fit, simulation, score tests, joint estimation of several groups, and meta-analysis across groups. The simulation runs in a compiled, multithreaded plugin. On 28 benchmark models the estimates agree with RSiena's to within 0.08 standard errors, and on a single thread `nwsaom` is 1.5 to 9.0 times as fast (median 2.1). The article describes the syntax, the effect library, the estimation, and the diagnostics, with a published adolescent-friendship panel as the running example.

Keywords: st0004, `nwsaom`, stochastic actor-oriented models, SAOM, Siena models, network dynamics, method of moments, Stata plugins, `nwcommands`

1 Introduction

Most network data collected to test theories of tie formation are not a single snapshot but a short panel: the same actors observed at two or more points in time. A stochastic actor-oriented model (SAOM) is built for exactly that situation. Rather than explaining one observed network in terms of density, reciprocity, or triadic closure the way an exponential-family random graph model (ERGM) does (Frank and Strauss, 1986; Wasserman and Pattison, 1996; Robins et al., 2007; Lusher et al., 2013), it explains the transition between waves as the sum of many small, unobserved moves – one actor at a time creating or dropping a single outgoing tie, or doing nothing, guided only by how that choice affects its own position in the network (Snijders, 1996, 2001). The two model families share more vocabulary than their different aims might suggest – both are estimated from a linear combination of effects over network structure – and Block et al. (2019) trace exactly where that resemblance breaks down; Block et al. (2018) weigh SAOMs against other longitudinal network models on consistency, interpretability, and predictive power, useful background for choosing between a SAOM and a cross-sectional model such as an ERGM, which the companion package `nwergm` fits. Snijders (2011) places both within the wider family of statistical network models. Because a SAOM represents change explicitly, it can also do something a cross-sectional model cannot: separate an actor choosing friends who already share a trait (selection) from an actor coming to share that trait because of the friends it already has (influence) (Steglich et al., 2010).

R’s **RSiena** package (Ripley et al., 2024) has held this territory for roughly two decades, and Stata has had nothing of its own to offer: fitting a SAOM has meant exporting the panel network out of Stata, running **RSiena** in R, and bringing the result back in.

nwsaom closes that gap. It is a Stata package for stochastic actor-oriented models that builds on the network objects and commands of **nwcommands** (Grund, 2026a). It fits the same models as **RSiena**, with the same effects, the same three-phase method-of-moments algorithm, conditional and unconditional estimation, and joint network-behavior co-evolution (Snijders, 1996, 2001; Snijders et al., 2010; Huisman and Snijders, 2003), and it gives the same estimates (Section 7.3). Networks are handled by the network objects of **nwcommands**, and the simulations run in a compiled plugin that uses all processor cores. Further model packages and extensions are described on the website <https://nwcommands.org>.

nwsaom is meant for researchers who work in Stata or prefer it to R; it is not a replacement for **RSiena**, which remains the reference implementation and the more complete program.

Speed is the main practical reason to prefer **nwsaom** for the models it covers. A SAOM fit is a simulation-based estimate, and applied work rarely needs only one: models are compared and refined, fits that have not converged are restarted from their estimates, results are checked with different seeds, and goodness of fit is assessed by further simulation. In the benchmark of Section 7.3, **nwsaom** gives the same estimates as **RSiena** on every model and, on a single thread, is 1.7 to 2.4 times as fast for the network models with **gwesp()** and 3.0 to 9.0 times as fast for the models without it; it also uses all processor cores without being asked, which on the realistic models makes it 2.7 to 4.5 times as fast as **RSiena** at its default (a single R process). **RSiena** remains the more complete program, but the gap is small: the effects table of **RSiena** 1.6.6 (as **getEffects()** returns it for twelve data configurations) has 1,201 rows, 80 of which belong to a model for continuous behavior that **nwsaom** does not cover, and **nwsaom** implements 1,058 of the other 1,121 (94%), every one of them compared with **siena07** (Section 6.4 says what is missing; the effects that **RSiena** 1.6.7 to 1.6.14 added or changed are implemented and compared with **RSiena** 1.6.14 but are not in these counts, and the counts change with **RSiena** releases). For the models **nwsaom** supports, the same analysis simply takes less time, and it stays within Stata.

The package as a whole, including its network objects and sparse computational backend, is described in Grund (2026a). A manual of the **nwsaom** command, with all effects, the comparison with **RSiena** and further examples, is archived at Zenodo (Grund, 2026) (<https://doi.org/10.5281/zenodo.23623807>); the network analysis with Stata, including a chapter on stochastic actor-oriented models, is the subject of a forthcoming book (Grund, 2027).

2 Installation

nwsaom is a package of its own. It uses the network objects, data handling and plotting commands of **nwcommands** (Grund, 2026a), which must be installed first; installation is repeated here in full so this article stands on its own. Stata 17 or later is needed (the

software is tested with Stata 17 and 18; StataBE is enough). From within Stata:

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

The first command installs **nwinstall**. The second installs the **nwcommands** core (the full command set, the help files and the dialog boxes in one pass; **nwinstall, gui** adds a “Network Analysis” menu to Stata), and the third installs **nwsaom** alone, with its commands, help files, two Mata libraries and the native plugin. The other model packages are not needed for this article; **nwinstall, models** would add them. Repeating the lines updates the installation. The command **nwversion** lists the installed versions at any time. No compiler is needed on the user’s own machine. The native backend (Section 7.2) ships as precompiled plugins for macOS, Linux and Windows; the Linux and Windows builds are produced by the package’s continuous integration, and every result in this article was produced on macOS. The more recent effects exist only in the plugin, and **nwsaom** stops with a message that says so when the plugin is unavailable; the older effects have a Mata implementation of the simulation that gives identical results but is much slower, and a fit that falls back to it prints one warning line that says why (plugin file not found, plugin too old, model beyond the plugin’s capacity); **quietly** suppresses it. The manual (Grund, 2026) documents every option; it is archived at Zenodo under the DOI of all versions, <https://doi.org/10.5281/zenodo.23623807>.

Versions

This article describes **nwsaom** version 1.2.0 (27 October 2026) with **nwcommands** version 3.10.0; it needs **nwcommands** version 3.6.0 or later. Each release of a package is a tagged version on GitHub, and the tag can be given in place of **main** to install exactly that release:

```
. net install nwsaom, ///  
from("https://raw.githubusercontent.com/  
thomas-u-grund/nwsaom/v1.2.0/") replace
```

The command **nwsaom version** reports the installed release of **nwsaom**, the version of the **nwcommands** core, the plugin in use and the version of RSiena that the results were compared with; the same number is stored in **e(version)** after each fit and printed at the top of every help file, so a published analysis can state exactly which release produced it. **nwversion** does the same for the core and lists all installed model packages. Releases follow the usual major.minor.patch scheme: a change in the first number may change results or syntax, the second adds features, and the third fixes errors.

nwsaom is free software under the GNU General Public License, version 3; so is the **nwcommands** core. **nwsaom** is derived from RSiena (Ripley et al., 2024) (GPL-2 or GPL-3) and, for Bayesian estimation, from the algorithm of the multiSiena package (Snijders et al., 2024) (GPL-3); the source code is public. The other model packages and **nwcommands** itself are free software under the GNU General Public License, version 3 as well. Further worked tutorials are at <https://nwcommands.org/nwsaom/>.

3 SAOM framework

A SAOM does not describe an observed network so much as the process assumed to have produced its change (Snijders, 1996, 2001; Snijders et al., 2010; Snijders, 2017). Between two waves, network x is taken to evolve through a chain of *ministeps*: at times set by a *rate function*, one actor i is chosen and may create or drop a single one of its own outgoing ties – never anyone else’s – or leave things as they are. This is where the contrast with an ERGM is sharpest (Block et al., 2019): an ERGM’s change statistic (Wasserman and Pattison, 1996; Robins et al., 2007) weighs a hypothetical toggle against the network’s entire joint distribution, while a SAOM ministepe is judged only against actor i ’s own *local* statistic, as seen from i ’s position in the network. Myopia, in this framework, means exactly that: i ’s choice can depend on how an option changes i ’s own circumstances, never on how it would change anyone else’s.

Once activated, actor i weighs its options – one for each tie it could create or drop, plus doing nothing – through an *evaluation function*: a linear combination of effects, the same basic form as an ERGM’s linear predictor, but built from i ’s own local statistics $s_{ki}(x)$ rather than a statistic of the whole network:

$$(1) \quad f_i(\theta, x) = \sum_k \theta_k s_{ki}(x).$$

Actor i then moves to x' with a multinomial (conditional) logit probability over its C available options (McFadden, 1973):

$$(2) \quad P(x \rightarrow x') = \frac{\exp\{f_i(\theta, x')\}}{\sum_{x'' \in C} \exp\{f_i(\theta, x'')\}}.$$

Whether an actor gets a turn at all is a separate question, governed by the rate function. Under a constant, actor-homogeneous rate (the default here), the time until the next ministepe, pooled across all n actors, is exponential with rate $n \times \text{rate}$, and whichever actor acts is drawn uniformly once that moment arrives; `ratecov()` (Table 1) lets the rate itself vary with a covariate, $\text{rate}_i = \text{rate}_0 \exp(\beta x_i)$. The rate parameter is answering an entirely different question from any coefficient in θ – how often an actor gets to move, not what it does once it can – and this article always reports it apart from the effect table.

3.1 Method of moments, maximum likelihood, and Bayesian estimation

A SAOM’s likelihood cannot be evaluated directly, for much the same reason an ERGM’s normalizing constant cannot: writing it down would mean summing over every ministepe sequence that could have produced the two observed waves, and there is no closed form for that sum. By default `nwsaom` therefore estimates θ by the *method of moments*, using Robbins-Monro stochastic approximation – RSiena’s default, and the estimator every benchmark in this article compares against. In practice: simulate the ministepe process forward from the first wave under some trial θ , then adjust θ until the simulated end-of-period statistics line up with the observed ones on average. `nwsaom` uses RSiena’s three-phase procedure: Phase 1 runs `k0()` simulations at the starting values to estimate,

via a score-function identity, how expected statistics move with θ ; Phase 2 works through four Robbins-Monro subphases, each with its own iteration count and a gain that begins at `firstg()` and is halved at every subphase; Phase 3 draws `k3()` replicates at the final θ to compute standard errors (a sandwich-form covariance matrix) and a convergence *t-ratio* for each parameter, rates included: the deviation of the simulated from the observed statistic, divided by its standard deviation. This checks whether the moment conditions are met, a different question from the Wald test reported alongside each coefficient. RSiena’s criteria are used: every t-ratio below 0.1 in absolute value and an overall maximum convergence ratio below 0.25. Below the coefficient and rate tables, `nwsaom` prints the overall ratio and names any parameter whose t-ratio is 0.1 or more; if the fit has not converged, it says so and suggests restarting from the estimates with `theta0()`. The option `detail` lists every t-ratio, and all of them are stored in `e(tconv)`. Section 5.3 below shows how to read this summary.

The likelihood can be handled by treating the ministep sequence between two waves as missing data. A Markov chain over such sequences, with Metropolis-Hastings moves that insert, delete and permute ministeps, gives the complete-data score of the likelihood; this is the machinery behind RSiena’s maximum-likelihood estimator (`maxlike = TRUE`) and behind its Bayesian estimation, which is provided by the `multiSiena` package (Snijders et al., 2024) (`multi_siena`, formerly `sienaBayes`), following Koskinen and Snijders (2007). `nwsaom` offers both for a single directed network, with or without one co-evolving behavior. The option `ml` starts from the method-of-moments fit, runs a Robbins-Monro algorithm on the complete-data score, and replaces the estimates, covariance matrix and rates by the maximum-likelihood ones, so that `estat` commands keep working. On the Glasgow data with `outdegree`, `reciprocity` and `transtrip` it gives -2.65 and -2.64 , 2.27 and 2.23 , and 0.68 and 0.69 over two seeds, against -2.66 , -2.63 and -2.65 , 2.22 , 2.19 and 2.22 , and 0.71 , 0.70 and 0.70 from RSiena’s maximum-likelihood runs (three seeds) on the same data; a co-evolution model agrees to within about 0.1 standard errors. The subcommand `nwsaom bayes` runs after a fit and samples the posterior of the fitted model with the schedule and controls of `multi_siena` for one group (flat prior for the effects, a normal prior around the starting rate for the rates). Its posterior means and standard deviations match RSiena’s maximum-likelihood estimates and standard errors to within Monte Carlo error (for the same model, posterior means -2.662 , 2.277 and 0.693 with standard deviations 0.110 , 0.181 and 0.064), and an exact enumeration on four actors reproduces the expected number of ministeps. Both are much slower than the method of moments (about half a second for the model above): the maximum-likelihood fit took about one minute, including the method-of-moments start, against about five minutes for RSiena’s maximum-likelihood fit of the same model, and a default Bayesian run took two to three minutes. This is why the method of moments remains the default, as in RSiena. Several groups with a hierarchical prior, and Bayesian estimation for non-directed and two-mode networks, are not implemented, and neither maximum-likelihood nor Bayesian estimation works with the joint estimation of several groups (Section 5.9).

3.2 Telling selection from influence apart

A network-only SAOM already answers something a cross-sectional ERGM cannot touch: whether these actors' own tendencies plausibly account for how the network actually changed. Adding a behavior variable opens a second, different question. `nwsaom` can then estimate a network-side effect (does similarity on some attribute predict which new ties form – selection) and a behavior-side effect (does an actor's own value drift toward its network alters' average – influence) in one joint fit over the same waves. This is the standard way around a confound no cross-sectional design escapes on its own: a correlation between an attribute and tie structure could equally mean actors sought out similar others, or actors grew similar to people they already knew, and only an explicitly dynamic model can tell which (Steglich et al., 2010; Kalish, 2020). Section 5.5 below works through exactly this joint model on a published panel of adolescent friendships.

4 The `nwsaom` command

4.1 Basic syntax

```
. nwsaom , {wave1(netname) wave2(netname) | waves(namelist)}  
outdegree [effect ...] [control options]
```

A model is written as a plain list of effect options, the way any Stata regression command lists covariates: `outdegree` is required, playing the part `edges` plays in `nwergm`, and any number of further effects can follow. Two waves go in `wave1()/wave2()`; three or more go in `waves()`, which strings every consecutive pair into its own inter-wave period and pools the effect coefficients across all of them into one method-of-moments fit, as in RSiena, while the rate parameter is still estimated separately for each period (`e(rates)/e(rate_ratios)`), since how often actors get a turn need not stay constant across a panel even when what they do with a turn does.

What the networks must look like. As in RSiena, the ties of every wave are binary. A wave that holds any tie value other than 0 and 1 (shares, counts or other real values) is refused with an error, whether or not the network was declared valued with `nwset`; such a network is dichotomized first with `nwdichotomize`. Dyads whose value is fixed by design, which RSiena codes as 10 and 11 in the network array, are marked instead with a separate 0/1 matrix in `structural()`. All waves must list the same actors in the same order.

4.2 Effect library

Table 1 sets out `nwsaom`'s current effect library, organized by family and by which evaluation function each effect belongs to. Rows above the rate-function entry belong to the network's evaluation function; rows below it belong to the behavior's, and require `behavior()` (Section 4.3). Several node-covariate options double as RSiena aliases alongside their `nwergm`-style names: `samex()` for `nodematch()`, `altx()` for `nodeicov()`, `egox()` for `nodeocov()`, and `simx()` for `simcov()` – so a reader arriving from RSiena's own effects table does not have to relearn a new vocabulary. Every node-covariate

option takes a varlist and adds one effect per variable, centered as RSiena centers covariates; `e(rsiena_labels)` holds RSiena’s name for each effect, and the option `detail` lists them below the table. `interact()` adds two- and three-way interactions of effects already in the model, following RSiena’s `includeInteraction()` rules.

Table 1: `nwsaom`’s effect library, by family and evaluation role.

Effect	Role	Description
Baseline (one required)		
outdegree	network	Tie-creation propensity (density/intercept)
Reciprocity		
reciprocity	network	Reciprocated-tie propensity
reciprocityendow	/	Split into loss- and gain-side coefficients
reciprocitycreation	network	
Node covariates (one varname each)		
nodematch()	network	Homophily on a categorical attribute (alias samex())
nodecov()	network	Sum of sender's and receiver's covariate value
nodeicov()	network	Receiver (in-)covariate effect (alias altx())
nodeocov()	network	Sender (out-)covariate effect (alias egox())
simcov()	network	Covariate similarity (alias simx())
Degree-based structural		
indegpopularity	network	Preferential attachment to high in-degree alters (sqrt)
outpopularity	network	Preferential attachment to high out-degree alters (sqrt)
outactivity	network	Sender's own out-degree activity (squared)
inactivity	network	Sender's own in-degree activity (sqrt)
Triadic closure		
transtrip	network	Transitive triplets ($i \rightarrow k \rightarrow j, i \rightarrow j$)
transmedtrip	network	Transitive mediated triplets
transties	network	Existence-indicator transitive closure
transrectrip	network	Transitive reciprocated triplets
cycle3	network	Directed 3-cycles
cycle4	network	Directed 4-cycles (base parameterization)
gwap()	network	Geometrically weighted edgewise shared partners
Isolates and balance		
balance	network	Structural balance (data-derived constant, no user parameter)
isolatenet / isolatepop	network	Isolate avoidance / isolate popularity
outiso / antiiso	network	Out-isolate count / anti-isolation
antiiniso / antiiniso2	network	In-isolation avoidance (two parameterizations)
Degree assortativity		
outoutass / ininass	network	Out-out / in-in degree assortativity
outinass / inoutass	network	Out-in / in-out degree assortativity
Rate function		
ratecov()	rate	Covariate-dependent activation rate, $\text{rate}_i = \text{rate}_0 e^{\beta x_i}$
Behavior co-evolution (require behavior())		

continued on next page

Table 1 (continued)

Effect	Role	Description
linear	behavior	Tendency toward higher/lower values (required baseline)
quadratic	behavior	Tendency toward extreme values (self-reinforcement)
quadraticendow / quadraticcreation	behavior	Split into decrease- and increase-side coefficients
avalt	behavior	Assimilation toward network alters' average value (influence)
avaltendow / avaltcreation	behavior	Split into decrease- and increase-side coefficients
avsim	behavior	Tendency to maximize similarity to alters (influence)
avsimendow / avsimcreation	behavior	Split into decrease- and increase-side coefficients

The table shows the main families; the manual (<https://doi.org/10.5281/zenodo.23623807>) lists all 1,058 implemented effects with their RSiena names, and Section 6.4 says what is not implemented.

Snijders et al. (2010) and Snijders (2017) cover what each family means substantively – when a finding is better expressed through **transtrip** than **cycle3**, say, or how isolate-avoidance behaves differently on a sparse network than a dense one – well beyond what a syntax table like this one can show. Effects in the isolate and assortativity families can make estimation unstable on small or sparse networks, in **nwsaom** as in RSiena: a single tie change can alter other actors' isolate status or degrees, and such networks contain few of the configurations these effects describe.

4.3 Co-evolution, composition change, and undirected networks

behavior(varlist) – one variable per wave – switches **nwsaom** into co-evolution mode, where **linear** becomes a required baseline (the behavior-side counterpart to **outdegree**) and **quadratic**, **avalt**, and **avsim** are available as optional behavior effects, all estimated together with the network-side effects in one method-of-moments fit (Section 5.5). Several behaviors can be listed in one call (**behavior(a1 a2 a3 : linear avalt | s1 s2 s3 : linear)**). **present()**, **joinat()/leaveat()**, **missnet()** and **missbeh()** cover composition change (actors joining or leaving between waves or, with **joinat()/leaveat()**, at any time within a period, as RSiena's **sienaCompositionChange()**) and missing tie or behavior data; with **present()** an actor can act in a period if it is present at the wave at which the period starts, as in RSiena; **groups()** fits one model to several groups of actors at once (Section 5.9); **structural()** marks dyads whose value is fixed by design (two waves, network only). **symmetric** moves to a mutual-consent minstep for a network whose ties are already symmetric at both waves; **symtype()** chooses among RSiena's model types for a non-directed relation: **forcing**, the default as in RSiena, lets the actor who chooses a tie impose it; **confirmation** needs the other actor to accept a new tie; **force**, **agree** and **joint** are RSiena's pairwise model types, in which a pair of actors decides together. Table 2 lays out these and **nwsaom**'s remaining controls, grouped as in **help nwsaom**.

Table 2: `nwsaom` control options, by group.

Option	Description
Observed waves (exactly one required)	
<code>wave1(netname) wave2(netname)</code>	Exactly two panel waves
<code>waves(namelist)</code>	Two or more waves, chained into $n-1$ pooled periods
Co-evolution	
<code>behavior(varlist)</code>	One behavior variable per wave; enables joint network+behavior estimation
<code>behtheta0(numlist)</code>	Starting values for behavior-side coefficients; default 0
Composition change and missing data	
<code>present(varlist)</code>	One 0/1 joiner/leaver indicator per wave
<code>missnet(matlist)</code>	One 0/1 missing-tie indicator matrix per wave
<code>missbeh(varlist)</code>	One 0/1 missing-behavior indicator per wave (requires <code>behavior()</code>)
Undirected networks	
<code>symmetric</code>	Non-directed relation (automatic for undirected or tie-symmetric waves; any number of waves)
<code>symtype(string)</code>	RSiena's model type: <code>forcing</code> (default), <code>confirmation</code> , <code>force</code> , <code>agree</code> , <code>joint</code>
Covariates, rates, and constraints	
<code>nocenter</code>	Use raw covariate values (default: centered by their means, as RSiena)
<code>ratecov(varlist)</code>	Covariates that change how often actors may change ties
<code>structural(matname)</code>	0/1 matrix of dyads that cannot change (structural zeros/ones)
Estimation control	
<code>unconditional</code>	Unconditional estimation (default for network-only models: conditional, as RSiena)
<code>theta0(numlist)</code>	Starting values, network-side effects; default 0
<code>rate0(numlist)</code>	Starting values of the rates (unconditional estimation)
<code>k0(int)</code>	Phase 1 replicate count; default 50
<code>k3(int)</code>	Phase 3 replicate count; default 1,000
<code>firstg(real)</code>	Phase 2 starting Robbins-Monro gain; default 0.2
<code>seed(int)</code>	Random-number seed
<code>cores(int)</code>	Number of threads; default all physical cores (results do not depend on it)
Estimator	
<code>ml</code>	Maximum likelihood instead of the method of moments (single directed network, with or without one behavior)
<code>mloptions(string)</code>	Controls of the maximum-likelihood estimator (<code>n1()</code> , <code>nsub()</code> , <code>n3()</code> , <code>gain()</code> , ...)
Reporting	
<code>detail</code>	Also list RSiena's effect names, covariate means, and every convergence t-ratio
Postestimation (estat gof)	
<code>nsim(int)</code>	Simulated replicates at fitted coefficients; default 50
<code>stats(namelist)</code>	Auxiliary statistics to check: <code>outdegree</code> <code>indegree</code> <code>geodesic</code> <code>triad</code> <code>behavior</code>

continued on next page

Table 2 (continued)

Option	Description
<code>maxdeg(int)</code> / <code>maxdist(int)</code>	Truncate the degree / geodesic-distance distribution; default 15 / 6
<code>twotailed</code>	Two-tailed rather than the default one-tailed test
<code>join(off)</code>	Test each period separately rather than pooled

5 Remarks and examples

The examples that follow use the Glasgow data: a 50-pupil excerpt of the Teenage Friends and Lifestyle Study (Michell and West, 1996; Pearson et al., 2006), three annual waves of directed friendship nominations among Scottish schoolgirls, with smoking, alcohol, and sport attributes recorded at each wave. It is the same panel behind RSiena’s `s50` tutorial data, and the source of every benchmark reported in Section 7. It ships with `nwcommands` and loads directly with `nwebuse glasgow`.

5.1 Looking at the data first

Figure 1 draws the three waves of the friendship network. Each dot is a pupil, and an arrow from one pupil to another means that the first named the second as a friend in that wave. Every pupil has the same position in all three panels, so the pupils do not move from panel to panel: what changes is the set of arrows. Pupils who are close together are linked by friendships, so the clusters in the plot are groups of friends; the shade and shape of a dot show the pupil’s smoking at the first wave (nonsmoker, occasional smoker, regular smoker). Look for the arrows that appear or disappear between waves while the groups stay broadly in place, and for the two pupils at the right who name nobody and are named by nobody in any wave.

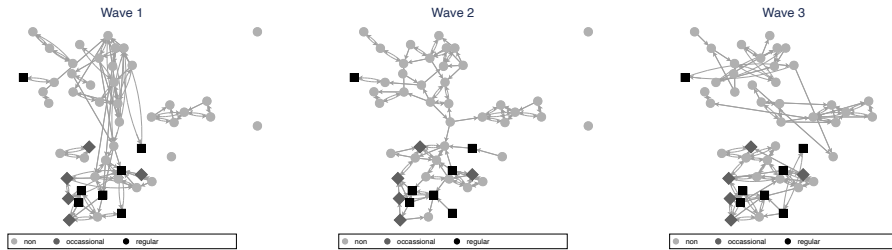


Figure 1: The Glasgow friendship network at waves 1, 2 and 3 (50 pupils; an arrow points from the pupil who names a friend to the pupil named). The positions are the same in all panels (Kamada-Kawai layout of wave 2). Shade and shape: smoking at wave 1.

Before a fit, `nwsaom describe` prints what RSiena prints in its data report: for every wave the number of ties, the density and the average degree, and for every period the

numbers of ties kept, created and dissolved, the Jaccard index (kept ties as a share of all ties present at either wave) and the Hamming distance. With `behavior()` it also gives the behavior's means and how many pupils move up, down or stay. On the Glasgow data:

```
. nwsaom describe, waves(glasgow1 glasgow2 glasgow3)
wave      ties      density  avg. degree
1          113      0.0461      2.260
2          116      0.0473      2.320
3          122      0.0498      2.440
period    kept  created dissolved  Jaccard  Hamming
1->2       57    59      56      0.331    115
2->3       66    56      50      0.384    106
```

About a third of the ties of one wave are still there at the next. The usual rule of thumb in the SAOM literature is that a Jaccard index below about 0.3 means that too few ties are stable for the data to say much about the process; `nwsaom describe` warns when a period falls below it.

5.2 A first fit

`outdegree` on its own – tie-formation propensity, nothing else – would be the SAOM equivalent of an intercept-only model. The example below goes further, asking whether smoking status predicts new friendship formation once reciprocity and cyclical triadic closure are accounted for, chaining all three Glasgow waves into a single pooled fit through `waves()`:

```
. nwwebuse glasgow, nwcLEAR
. nwsaom, waves(glasgow1 glasgow2 glasgow3) outdegree reciprocity cycle3 nodemat
> ch(smoke1) k0(50) k3(1000) seed(12345)
```

SAOM (Method of Moments, conditional), waves: glasgow1 glasgow2 glasgow3
Actors: 50 Periods: 2

glasgow1_to-3	Coefficient	Std. err.	z	P> z	[95% conf. interval]	
outdegree	-2.810307	.1546159	-18.18	0.000	-3.113349	-2.507266
reciprocity	2.613267	.1882569	13.88	0.000	2.24429	2.982244
nodematch_s-1	.124002	.1437741	0.86	0.388	-.1577901	.405794
cycle3	1.170845	.1528352	7.66	0.000	.8712939	1.470397

Rate parameters (estimated, one per inter-wave period):

	period1	period2
rate	6.6075	5.3038
se	1.1791	0.8962

Overall maximum convergence ratio: 0.132 (criterion: below 0.25; met)
Convergence t-ratios (criterion: all below 0.1 in absolute value): not met for c
> ycle3 (0.106)
Convergence is good only if both criteria are met. Refit starting from these est
> imates with `theta0()`; see `nwsaom`.

Reciprocity and cyclical closure (`cycle3`) are both strongly positive, while smoking homophily is not significant (`nodematch_smoke1`, $p = 0.388$). The rate table gives the estimated rate of each period: on average, a pupil had about 6.6 opportunities to change a friendship between the first and second waves and 5.3 between the second and third. The last lines summarize convergence: the overall maximum convergence ratio is 0.132, below RSiena's limit of 0.25, but the t-ratio of `cycle3` (0.106) is just above 0.1, so the fit meets the first of RSiena's two criteria and narrowly misses the second. This is typical of a first run with the default settings; the next section shows what to do about it. The estimates, standard errors and rates agree with RSiena's (Section 5.7).

5.3 Reading the convergence summary

When convergence is not perfect, the summary says which criterion is not met and which parameters need attention. Replacing `cycle3` by transitive triplets (`transtrip`) and shortening Phase 3 to 200 replicates, which makes the convergence statistics noisy, gives a fit that has not converged; the second command restarts from its estimates with `theta0()` and the default length of Phase 3:

```
. nwsaom, waves(glasgow1 glasgow2 glasgow3) outdegree reciprocity transtrip k0(5
> 0) k3(200) seed(12345)
```

SAOM (Method of Moments, conditional), waves: glasgow1 glasgow2 glasgow3
Actors: 50 Periods: 2

glasgow1_t-3	Coefficient	Std. err.	z	P> z	[95% conf. interval]	
outdegree	-2.685775	.1280667	-20.97	0.000	-2.936781	-2.434769
reciprocity	2.460293	.2128949	11.56	0.000	2.043026	2.877559
transtrip	.6264058	.0794441	7.88	0.000	.4706982	.7821133

Rate parameters (estimated, one per inter-wave period):

	period1	period2
rate	6.5086	5.1681
se	1.1288	0.8208

Overall maximum convergence ratio: 0.278 (criterion: below 0.25; not met)

Convergence t-ratios (criterion: all below 0.1 in absolute value): not met for o
> utdegree (0.171) reciprocity (0.224) transtrip (0.227)

Convergence is good only if both criteria are met. Refit starting from these est
> imates with `theta0()`; see `nwsaom`.

```
. nwsaom, waves(glasgow1 glasgow2 glasgow3) outdegree reciprocity transtrip k0(5
> 0) k3(1000) theta0(-2.686 2.460 0.626) seed(12346)
```

SAOM (Method of Moments, conditional), waves: glasgow1 glasgow2 glasgow3
Actors: 50 Periods: 2

glasgow1_t-3	Coefficient	Std. err.	z	P> z	[95% conf. interval]	
outdegree	-2.690742	.1071331	-25.12	0.000	-2.900719	-2.480765
reciprocity	2.476054	.1995521	12.41	0.000	2.084939	2.867169
transtrip	.6234451	.0715554	8.71	0.000	.4831991	.7636912

Rate parameters (estimated, one per inter-wave period):

	period1	period2
rate	6.5214	5.1890
se	1.1054	0.8490

Overall maximum convergence ratio: 0.145 (criterion: below 0.25; met)

Convergence t-ratios (criterion: all below 0.1 in absolute value): met

The overall maximum convergence ratio of the first fit (0.278) is above 0.25 and three t-ratios are above 0.1 (0.171, 0.224 and 0.227), so `nwsaom` says that both criteria are not met. The remedy is the one in the log: run the model again, starting from these estimates with `theta0()`, which is the counterpart of RSiena's `prevAns` (RSiena skips Phase 1 when the model is unchanged; `nwsaom` repeats it). The second fit has an overall ratio of 0.145 and every t-ratio below 0.1. A longer Phase 3 (`k3()`) gives more precise t-ratios, and the estimates hardly move. If a restart does not help, or if the overall ratio stays above 0.25, `nwsaom` says the fit has not converged. When repeating the run does not help, the usual advice is to simplify the model: add an effect that captures the feature the model misses (for example `gwersp()` for closure), include rate effects, fix a parameter that wanders to a very large value with `fix()` and test it with `estat score`, or drop an effect that is nearly collinear with another; `estat vce`, `corr` shows the correlations of the estimates. The Wald tests of linear hypotheses about the coefficients are Stata's `test`, `lincom` and `testparm`, which use `e(b)` and `e(V)`.

5.4 Checking fit after estimation

`estat gof`, available after any `nwsaom` fit, draws fresh networks at the fitted coefficients and compares their auxiliary statistics to the observed network with a Mahalanobis-distance test (Lospinoso and Snijders, 2019), the same construction behind RSiena's `sienaGOF()`. Run on the well-behaved model above:

```
. estat gof, seed(12345) name(glasgow_gof_plot)
Goodness of fit: Mahalanobis distance test, 50 simulations at the estimates (2 p
> eriods pooled)

-----
Outdegree | Mahalanobis dist. = 11.087   p = 0.300 (no evidence against f
> it)
Indegree  | Mahalanobis dist. = 10.005   p = 0.380 (no evidence against f
> it)
Geodesic   | Mahalanobis dist. = 38.536   p = 0.000 (evidence AGAINST fit)
-----

p < 0.05: the observed statistic is unlikely under the fitted model (one-tailed
> test).
Violin plots: glasgow_gof_plot_outdegree glasgow_gof_plot_indegree glasgow_gof_p
> lot_geodesic
```

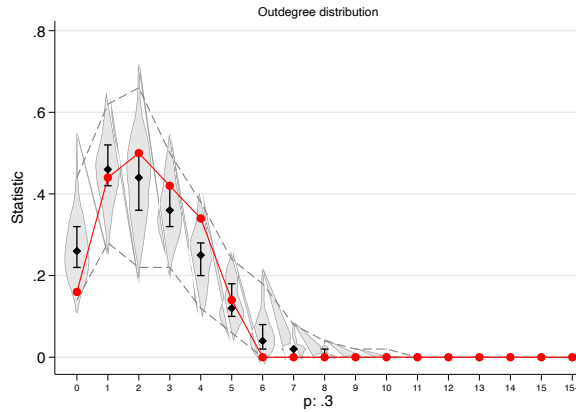


Figure 2: `estat gof`'s outdegree-distribution violin plot for the model of Section 5: simulated distribution (violin), interquartile range (bar), 95% envelope (dashed), and the observed network (red).

Neither the outdegree nor the indegree distribution shows evidence against the fitted model ($p = 0.300$, $p = 0.380$), but the distribution of geodesic distances does ($p < 0.02$; none of the 50 simulations was as far from the center of the simulated distribution as the observed network): the model reproduces who has how many friends, but not how far apart pupils are in the network. With 1,000 simulations in place of 50 the conclusions are the same ($p = 0.602$, 0.599 and < 0.001), and `sienaGOF` in `RSiena` gives $p = 0.600$, 0.689 and < 0.001 for the same model. Effects that shape the network's larger structure, such as `gwesp()`, would be the next step.

5.5 Co-evolution: separating selection from influence

The model below carries the same three Glasgow waves into a full co-evolution fit. `behsim` asks whether similarity in alcohol use, as it currently is, predicts which new friendships form – selection, the network-side counterpart to `nodematch(smoke1)` above (`RSiena`'s `simX` with the behavior as the interacting variable) – while `behavior(alcohol1 alcohol2 alcohol3)` together with `avalt` asks whether an actor's own alcohol use drifts toward its friends' average – influence. Both are estimated together, in one fit. Alcohol stands in for smoking here because smoking changes little between waves (76% of pupils are non-smokers at the first wave), which leaves too little information to estimate influence:

```
. nwsaom, waves(glasgow1 glasgow2 glasgow3) outdegree reciprocity behsim behavior
> r(alcohol1 alcohol2 alcohol3) linear avalt k0(50) k3(1000) seed(12345)
```

```
SAOM co-evolution (Method of Moments), waves: glasgow1 glasgow2 glasgow3
Actors: 50                               Periods: 2
Behavior: alcohol1 alcohol2 alcohol3
```

glasgow1_t-3	Coefficient	Std. err.	z	P> z	[95% conf. interval]	
outdegree	-2.386483	.116201	-20.54	0.000	-2.614233	-2.158734
reciprocity	2.780867	.18437	15.08	0.000	2.419508	3.142225
behsim	1.177022	.5130099	2.29	0.022	.1715414	2.182503
beh_linear	.3541812	.1338318	2.65	0.008	.0918756	.6164868
beh_avalt	.2508615	.2069595	1.21	0.225	-.1547716	.6564947

Rate parameters (estimated, one per inter-wave period):

	period1	period2
network_rate	5.8160	4.5319
se	0.9282	0.7035
behavior_r-e	1.2309	1.8186
se	0.3166	0.4458

Overall maximum convergence ratio: 0.187 (criterion: below 0.25; met)

Convergence t-ratios (criterion: all below 0.1 in absolute value): not met for o
> utdegree (0.103) reciprocity (0.133)

Convergence is good only if both criteria are met. Refit starting from these est
> imates with theta0(); see nwsaom.

Pupils with similar alcohol use are more likely to become friends (`behsim`, $p = 0.022$), while there is no evidence that pupils' alcohol use moves toward that of their friends (`beh_avalt`, $p = 0.225$). On this panel, there is evidence for selection and none for influence. A co-evolution model has two rates per period, one for the network and one for the behavior, and both are estimated with the other parameters (unconditional estimation, as in RSiena). The convergence summary flags `outdegree` (0.103) and `reciprocity` (0.133); with an overall ratio of 0.187 the fit is close to converged, and a restart from the estimates with `theta0()` and `rate0()` is the usual next step. The estimates, standard errors and rates agree with RSiena's (Section 5.7). `estat gof` adds the distribution of the behavior after a co-evolution model:

```
. estat gof, seed(12345) name(glasgow_coev_gof_plot)
```

Goodness of fit: Mahalanobis distance test, 50 simulations at the estimates (2 p
> eriods pooled)

Outdegree > it)	Mahalanobis dist. =	7.777	p =	0.340 (no evidence against f
Indegree > it)	Mahalanobis dist. =	9.790	p =	0.180 (no evidence against f
Geodesic	Mahalanobis dist. =	255.075	p =	0.000 (evidence AGAINST fit)
Behavior > it)	Mahalanobis dist. =	7.123	p =	0.100 (no evidence against f

p < 0.05: the observed statistic is unlikely under the fitted model (one-tailed
> test).

Violin plots: `glasgow_coev_gof_plot_outdegree` `glasgow_coev_gof_plot_indegree` gla
> `sgow_coev_gof_plot_geodesic` `glasgow_coev_gof_plot_behavior`

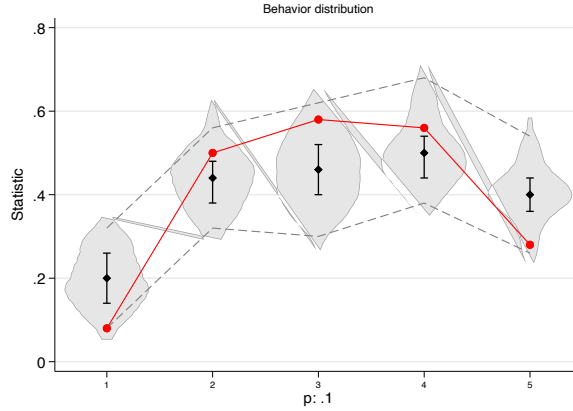


Figure 3: `estat gof`'s behavior-distribution violin plot for the co-evolution model above.

Outdegree and indegree show no evidence against the fit ($p = 0.340$, $p = 0.180$). Geodesic distance again does ($p < 0.02$). The behavior distribution is borderline: $p = 0.100$ with 50 simulations and $p = 0.056$ with 1,000 (RSiena's `sienaGOF`: $p = 0.077$), so the model may not fully reproduce how alcohol use is distributed at the later waves, and adding behavior effects such as `quadratic` would be a natural check.

5.6 Endowment and creation functions

Every effect discussed so far treats a step up and a step down (in the behavior) or creating and dropping a tie (in the network) symmetrically. Endowment and creation effects split an effect into two, so that, for example, the tendency to reduce alcohol use can differ from the tendency to increase it. `nwsaom` offers the split for `quadratic`, `avalt`, and `avsim` on the behavior side, and for `reciprocity` on the network side. The example splits the quadratic shape effect:

```
. nwsaom, wave1(glasgow1) wave2(glasgow2) outdegree reciprocity behavior(alcohol
> 1 alcohol2) linear quadraticendow quadraticcreation k0(50) k3(1000) seed(12345
> )
```

```
SAOM co-evolution (Method of Moments), waves: glasgow1 -> glasgow2
Actors: 50 Network rate: 5.538 (0.831)
Behavior: alcohol1 alcohol2 Behavior rate: 1.207 (0.357)
```

glasgow1_to-2	Coefficient	Std. err.	z	P> z	[95% conf. interval]	
outdegree	-2.225121	.1326602	-16.77	0.000	-2.485131	-1.965112
reciprocity	2.439252	.2669675	9.14	0.000	1.916006	2.962499
beh_linear	.4323116	.3913384	1.10	0.269	-.3346976	1.199321
beh_quadrat-w	.0975672	.4240626	0.23	0.818	-.7335802	.9287145
beh_quadrat-n	-.2168233	.595471	-0.36	0.716	-1.383925	.9502784

Overall maximum convergence ratio: 0.151 (criterion: below 0.25; met)

Convergence t-ratios (criterion: all below 0.1 in absolute value): met

Neither part is significant here ($p = 0.818$ and 0.716), which is common: separate endowment and creation effects need many changes in both directions, and RSiena's manual warns that they are harder to estimate than the symmetric effect. The linear shape effect and `outdegree` cannot be split: their endowment and creation statistics add up to the statistic of the rate, so with the rate estimated the model is not identified, and `nwsaom` refuses it with that explanation.

5.7 Checking the examples against RSiena

The models of this section were also fitted in RSiena 1.6.14 (R 4.6.0) on the same data, with the same effects and 1,000 Phase 3 simulations, five seeds each. Table 3 shows two of them. Over all coefficients of the four models fitted with several seeds (the first fit, the model of Section 5.3, the co-evolution model and the endowment and creation model) the mean estimates of the two programs differ by at most 0.08 RSiena standard errors, the ratio of the standard errors lies between 0.97 and 1.03, and the network rates agree to within 0.02 standard errors. For comparison, the seed-to-seed standard deviation of an RSiena estimate is below 0.04 in every case.

Table 3: The estimates of two examples, `nwsaom` and RSiena: mean over five seeds and mean standard error (Phase 3 with 1,000 simulations in both programs)

Effect	nwsaom		RSiena	
	Estimate	s.e.	Estimate	s.e.
<i>Model of Section 5, first fit</i>				
<code>outdegree</code>	-2.808	0.159	-2.805	0.162
<code>reciprocity</code>	2.610	0.193	2.617	0.191
<code>nodematch(smoke1)</code>	0.128	0.148	0.128	0.151
<code>cycle3</code>	1.157	0.149	1.150	0.149
<i>Co-evolution model</i>				
<code>outdegree</code>	-2.385	0.112	-2.385	0.108
<code>reciprocity</code>	2.773	0.188	2.770	0.184
<code>behsim</code>	1.172	0.496	1.154	0.484
<code>linear</code>	0.352	0.132	0.353	0.131
<code>avalt</code>	0.248	0.198	0.246	0.196
network rate, period 1	5.873	0.981	5.891	1.011
network rate, period 2	4.522	0.691	4.519	0.683

5.8 Fixing effects, simulating, testing, and combining groups

The commands of Section 6.3, on the same data. First a score-type test of two effects that are not in the model. They are held fixed at 0 in a restricted fit, and `estat score` asks what freeing them would do:

```
. nwsaom, wave1(glasgow1) wave2(glasgow2) outdegree reciprocity transtrip cycle3
>      theta0(-2.3 2 0 0) fix(transtrip cycle3) k0(100) k3(1000) seed(1)
. estat score
```

Generalised score-type test of the model restricted by:

```

(1) transtrip          = 0.0000
(2) cycle3             = 0.0000
Joint test: c = 82.2543 d.f. = 2 p-value = 0.0000
(1) transtrip separately: c = 82.0640 d.f. = 1 p-value = 0.0000
(2) cycle3 separately: c = 67.8301 d.f. = 1 p-value = 0.0000
One-step estimates (restricted estimate plus the one-step correction):
outdegree -2.6919 reciprocity 2.2197 transtrip 1.2905 cycle3 0.1595

```

Both effects would improve the fit; the one-step estimates show roughly what they would be if freed. With three waves, `estat timetest` asks whether the effects of a fitted model differ between the two periods:

```

. nwsaom, waves(glasgow1 glasgow2 glasgow3) outdegree reciprocity transtrip seed(12345)
. estat timetest
Joint test of all period-specific deviations: c = 3.6943 d.f. = 3 p-value = 0.2964
effect          c d.f. p-value
outdegree       0.106 1 0.7444
reciprocity     1.075 1 0.2999
transtrip       0.459 1 0.4980

```

and `estat importance` shows which effects matter most for the actors' choices (here for `outdegree reciprocity transtrip cycle3`, with the expected share of each effect in each period, `RI p1` and `RI p2`):

effect	estimate	s.e.	sigma	semi-std.	RI p1	RI p2
outdegree	-2.6983	0.1156	0.4384	-1.1829	0.4342	0.5049
reciprocity	2.4823	0.2102	0.2140	0.5312	0.3782	0.3101
transtrip	0.6735	0.1342	0.6514	0.4387	0.1740	0.1724
cycle3	-0.0968	0.2691	0.3374	-0.0326	0.0137	0.0126

Simulation at chosen coefficients needs no fit. The following makes four groups by simulating the model four times from the same start network, fits each group, and combines the four fits with `nwsaom meta`; the combined means should be close to the coefficients used (-2.3, 2, and 0.5):

```

. nwsaom simulate, wave1(glasgow1) wave2(glasgow2) outdegree reciprocity transtrip
> theta(-2.3 2 0.5) rates(5) nsim(4) seed(7) generate(g)
. forvalues s = 1/4 {
.   quietly nwsaom, wave1(glasgow1) wave2(g_`s'_p1) outdegree reciprocity transtrip k3(500) s
.   estimates store grp`s'
. }
. nwsaom meta grp1 grp2 grp3 grp4

```

effect	n	mean	s.e.	t	p	sd	Q	p(Q)
outdegree	4	-2.2557	0.0572	-39.47	0.000	0.1143	2.49	0.477
reciprocity	4	1.8729	0.0691	27.09	0.000	0.1383	0.99	0.804
transtrip	4	0.4749	0.0121	39.27	0.000	0.0242	0.16	0.983

No effect differs between the groups ($p(Q)$ between 0.48 and 0.98), as expected for groups simulated from one model. In an application the groups would be, for example, school classes, and a small $p(Q)$ would say that the effect differs between classes by more than the standard errors allow.

5.9 Several groups: joint estimation and meta-analysis

Many panels consist of several separate groups: school classes, teams, villages. Each group is its own network, with its own actors, and ties never run between groups. There are two ways to analyze such data, and `nwsaom` offers both, as `RSiena` does. One can fit the same model to each group separately and combine the estimates (a meta-analysis, `nwsaom meta`), or fit one model to all groups at once (joint estimation, the option `groups()`; `RSiena`'s `sienaGroupCreate()` followed by `siena07`).

Joint estimation. In a joint model all groups share *one set of effect parameters*: actors in every group are assumed to choose their ties by the same rule, with the same weight for reciprocity, transitive closure and so on. What is not shared is the speed of change: every period of every group has its own rate parameter, because classes differ in how much they change. The statistics of all periods of all groups are added up when the coefficients are estimated, just as the periods of one group are in a model with three or more waves. Small groups that are each too small for a model of their own can therefore be analyzed together, and the standard errors reflect the pooled information. The price is the assumption that the groups do not differ; the model does not estimate how much they do.

The groups are stacked in one data set. The rows are the actors of all groups, one variable (here `class`) says which group an actor belongs to, and each wave of the network is one `nwcommands` network on all actors that has no tie between two groups (a block-diagonal adjacency matrix). Covariates are ordinary variables; a tie between two groups stops the command with their number. The option `gwaves()` is for groups observed at different occasions: `gwaves(2: 1 2 | 3: 2 3)` says that group 2 was observed at the first two entries of `waves()` and group 3 at the last two. Rates are named with `RSiena`'s period numbers, in which the step from the last wave of one group to the first wave of the next is not a period: four groups with two waves each have `period1 period3 period5 period7`. Everything that `RSiena` keeps per group is per group here as well: the similarity mean of a covariate, the balance mean, the mean and variance of a behavior and the centering of a dyadic covariate. An actor covariate is centered at its mean over all actors.

The example uses four Dutch school classes of 45, 37, 33 and 36 pupils, observed twice, from the Dutch Social Behavior Data Set of Chris Baerveldt that ships with `RSiena` (Snijders and Baerveldt, 2003). The stacked networks are `w1` and `w2`, and `class` is the group variable.

```
. nwsaom describe, waves(w1 w2)
Network data: 2 waves, 151 actors, directed
wave      ties      density  avg. degree
1          246       0.0109    1.629
```

```

2          335      0.0148      2.219
. nwsaom, waves(w1 w2) groups(class) outdegree reciprocity transtrip seed(5)
SAOM (Method of Moments, conditional), waves: w1 w2
Actors: 151                      Periods: 4
Groups: 4 (variable class), actors per group: 45, 37, 33, 36; waves per group: 2, 2, 2, 2

```

w1_to_w2	Coefficient	Std. err.	z	P> z	[95% conf. interval]	
outdegree	-2.247226	.0808946	-27.78	0.000	-2.405776	-2.088675
reciprocity	1.86988	.1595137	11.72	0.000	1.557239	2.182522
transtrip	.4650632	.0590168	7.88	0.000	.3493924	.5807339

Rate parameters (estimated, one per inter-wave period):

	period1	period3	period5	period7
rate	6.4599	5.3856	2.2902	8.1389
se	1.0048	1.3792	0.5385	1.9861

Overall maximum convergence ratio: 0.067 (criterion: below 0.25; met)
Convergence t-ratios (criterion: all below 0.1 in absolute value): met

There is one coefficient table, because the weights of the three effects are common to all 151 pupils: ties are reciprocated and closed beyond what density alone gives. There are four rates. Pupils in class 4 (rate 8.1) had about three and a half times as many opportunities to change a tie as pupils in class 3 (2.3). The matrices `e(groupn)`, `e(rates_group)` and `e(rates_period)` say which rate belongs to which group and to which of RSiena's period numbers. The same calls also work for a co-evolving behavior in all groups (the shared effects, one network and one behavior rate per period), for groups with different numbers of waves, and, after the fit, for `estat timetest` (joint, per effect, per group and per period) and `estat score`.

Meta-analysis. `nwsaom meta` takes the stored estimates (`estimates store`) of the same model fitted to each group, with the same effects in the same order. For every effect it reports the combined mean and its standard error (an iteratively reweighted least squares estimate with a random effect for the group, Snijders and Baerveldt, 2003), the estimated standard deviation of the effect between groups, a heterogeneity test Q , maximum-likelihood estimates of the mean and the between-group standard deviation with profile-likelihood intervals, and Fisher's combination of the one-sided tests. Here are the four classes, fitted separately with the same model, and combined:

```

. forvalues s = 1/4 {
.   quietly nwsaom, wave1(school's'_w1) wave2(school's'_w2) outdegree reciprocity
.   transtrip seed(5)
.   estimates store school's'
. }
. nwsaom meta school1 school2 school3 school4
Meta-analysis of 4 fits (RSiena siena08): estimates with a standard error of at least

```

10 are excluded. Tests of mean parameters use a t distribution with $n-1$ d.f.
IWLs estimates (Snijders & Baerveldt 2003)

effect	n	mean	s.e.	t	p	sd	Q	p(Q)
outdegree	4	-2.3772	0.2535	-9.38	0.003	0.5069	9.05	0.029
reciprocity	4	1.8653	0.2664	7.00	0.006	0.5328	5.91	0.116
transtrip	4	0.6958	0.1734	4.01	0.028	0.3468	8.39	0.039
Maximum likelihood under normality with 95% profile-likelihood intervals								
effect		mean	s.e.	lower	upper	sd	lower	upper
outdegree		-2.2339	0.0940	-2.7705	-2.0006	0.0829	0.0000	0.7810
reciprocity		1.8256	0.2162	1.3163	2.5174	0.2423	0.0000	1.0805
transtrip		0.5115	0.0514	0.4104	1.0354	0.0001	0.0000	0.6001
Fisher's combination of one-sided tests (chi-squared, 2n d.f.)								
effect		right	p	left	p			
outdegree		0.000	1.000	807.018	0.000			
reciprocity		138.037	0.000	0.003	1.000			
transtrip		127.633	0.000	0.000	1.000			

The separate fits agree in sign and differ in size: the outdegree effect is -2.01 to -2.33 in classes 4, 1 and 2 but -3.34 in class 3, and reciprocity is 2.59 in class 2 against 1.54 to 1.56 in classes 1 and 4. The random-effects mean of reciprocity is 1.87 with a between-class standard deviation of 0.53. The heterogeneity test Q rejects equal effects for outdegree ($p = 0.029$) and transitive triplets ($p = 0.039$). The ML intervals for the between-class standard deviations start at zero, which is what four groups can tell about a variance. Fisher's test combines the one-sided tests of the four classes; a small "right" p-value speaks for a positive effect (reciprocity and closure), a small "left" p-value for a negative one (outdegree). Because only four classes enter, the tests of the means use a t distribution with 3 degrees of freedom. The option `rates` combines the rate parameters as well.

Joint versus meta. The joint estimates are -2.25 for outdegree, 1.87 for reciprocity and 0.47 for transitive triplets (Table 4 also gives RSiena's), against the random-effects means -2.38 , 1.87 and 0.70 and the simple averages of the four class estimates -2.49 , 1.98 and 0.79. The standard errors of the joint model (0.08, 0.16, 0.06) are much smaller than those of the random-effects means (0.25, 0.27, 0.17), because the joint model takes the sameness of the classes for granted, while the meta-analysis lets the effects vary and estimates by how much. When Q is clearly significant, as for outdegree and transitive triplets here, the joint standard errors are too optimistic. The joint model is the first thing to try; the meta-analysis checks it; and the joint model can then be extended by letting selected effects differ between groups, by interacting them with a group indicator. Letting reciprocity differ for class 2 (`egox(c2)` with `interact(egox_c2#reciprocity)`) gives an interaction coefficient of 1.12 (standard error 0.40): pupils in class 2 reciprocate more than the others.

Agreement with RSiena. Table 4 shows the joint model of the four classes fitted by both programs: the largest difference between the programs in an effect is 0.002, and the rates agree within 0.014. Three further joint models were compared in the same

Table 4: Joint estimation of the four classes (151 pupils): `nwsaom` and RSiena 1.6.14. Means over ten runs of each program with seeds 1 to 10; standard errors are the means over the runs.

Parameter	Estimate		Standard error	
	<code>nwsaom</code>	RSiena	<code>nwsaom</code>	RSiena
outdegree	-2.247	-2.248	0.081	0.080
reciprocity	1.868	1.870	0.158	0.153
transtrip	0.465	0.464	0.055	0.054
rate, class 1	6.50	6.51	1.02	1.01
rate, class 2	5.38	5.38	1.38	1.16
rate, class 3	2.31	2.31	0.54	0.73
rate, class 4	8.17	8.16	2.07	1.42

way, ten seeds each. A co-evolution model of friendship and alcohol use in three groups of 15, 20 and 15 pupils (the Glasgow excerpt cut into artificial groups for the purpose; ties between the groups removed) gives, for example, reciprocity 2.933 against 2.936 and the linear shape of alcohol use 0.388 against 0.384, with its six network and six behavior rates within 0.02 and a largest difference of an effect of 0.008. For three groups with 3, 2 and 2 waves (`gwaves()`) the estimates differ by at most 0.003 (RSiena 1.6.14 stops with an error for a group object whose groups have different numbers of waves; the reference is the equivalent object of two-wave groups). `estat timetest` after the grouped co-evolution fit gives a joint test of 37.9 on 25 degrees of freedom (RSiena 38.7), and the score test of two fixed effects 0.17 on 2 degrees of freedom (RSiena 0.18).

What is not available with groups. Maximum-likelihood and Bayesian estimation, `estat gof`, `estat simulate` and `estat importance`, two-mode and non-directed networks, `nwsaom multinet`, several behaviors, the centered degree effects, `avdeg`, `transtriads`, `behrate()`, `structural()`, `joinat()/leaveat()` and time-varying dyadic covariates stop with a message that says so. Groups need the compiled plugin of this release; there is no Mata fallback. For models with missing behavior values and the selection effects `behegox` or `behaltx` in groups, the estimates differ from RSiena’s, and these models should be treated with caution.

6 Beyond the basic model

This section describes what `nwsaom` offers beyond the single directed network of the examples in Section 5. The rule behind the syntax is the usual Stata one: a property of the data is detected from the data, an option changes how a model is specified or fitted, and a different job is a subcommand or a postestimation command.

6.1 More kinds of networks and covariates

A network declared as two-mode (`nwset`, `bipartite`) is recognised and fitted with the effects that RSiena offers for bipartite networks, the covariate effects, changing covari-

ates and, with `behavior()`, a co-evolving behavior of the actors. An undirected network (undirected `nwset`, or waves that are symmetric) is fitted with RSiena’s model types for non-directed relations over any number of waves, also together with a behavior. Further covariate types are changing covariates (`tvcov()`), constant and changing dyadic covariates (`dyadcov()`, `dyadx()`), covariates with missing values, and the effects of a covariate on triads, popularity and activity (`netx()`). Dyads whose value is fixed by design are marked with `structural()`. `nwsaom describe` and `nwsaom simulate` also work for two-mode networks.

6.2 Several networks and a behavior: `nwsaom multinet`

`nwsaom multinet` fits two networks that evolve together on the same actors, as RSiena does for two dependent network variables: each network has its own effects and rates, and the cross-network effects (`crprod`, `from`, `closure` and many more, with or without a covariate) let the state of one network affect the choices in the other. One network may be non-directed, or two-mode (detected from the data, with the cross-network effects that RSiena offers for such pairs, including the dyadic-covariate effects `toU`, `sharedToU` and `avAltU`). A co-evolving behavior can be added with `behavior()`: `beheff1()` lists the effects of the first network on the behavior, `beheff2()` those of the second, and the behavior can be a covariate of the network and cross-network effects (`behsim`, `behegox`, `behaltx`, and `effect:behavior` in `cross1()`). The behavior can also depend on both networks (`beheff12()`, `beheff21()`: RSiena’s `FFDeg`, `BBDeg`, `indegMixedPop` and the other double-degree effects), on covariates, and on network effects with the behavior as covariate (`netbeh:`); behavior effects can have endowment and creation parts, and missing behavior values are allowed (`missbeh()`). Starting values and fixed effects are given with `theta0()` and `fix()`, so that `estat score` and `estat timetest` work after a `multinet` fit.

```
. nwsaom multinet, net1(friend1 friend2 friend3) net2(advice1
advice2 advice3)
    behavior(smoke1 smoke2 smoke3) beheff1(linear quadratic
avalt) beheff2(avalt)
```

All of these were compared with `siena07`: the observed statistics are equal and the estimates and rates agree within simulation noise. The older command `nwsaom multiplex` is deprecated and forwards to `nwsaom multinet`.

6.3 Holding effects fixed, simulating, and testing

`fix(effects)` keeps effects at their `theta0()` values (RSiena’s `fix = TRUE`). It is used to fit a restricted model for a score-type test, and with `nwsaom simulate` to simulate at chosen coefficients without estimating anything (RSiena’s `siena07` with `simOnly`). After a fit, `estat simulate` simulates the fitted model.

`estat score` is the generalised score-type test of Schweinberger (2012): it tests, from the simulations of the restricted model alone, whether freeing the fixed effects would improve the fit. `estat timetest` tests whether the effects have the same size in every period (RSiena’s `sienaTimeTest`), and `estat importance` computes the relative

importance of the effects (RSiena’s `sienaRI`, called `interpret_size` in current versions; it works for one-mode, non-directed and two-mode networks and, for co-evolution fits, also gives the importance of the behavior effects). `nwsaom meta` combines the estimates of the same model fitted to several groups, such as school classes, as RSiena’s `siena08` (Snijders and Baerveldt, 2003); the option `groups()` instead fits one model to all groups at once (Section 5.9). Section 5.8 shows each of them on the Glasgow data. All were compared with RSiena: `estat score` and `estat timetest` within simulation noise over eight seeds, `estat importance` to 10^{-7} , and `nwsaom meta` to five decimals.

6.4 How much of RSiena is there

Current versions of RSiena have renamed most functions (`siena` for `siena07`, `test_gof` for `sienaGOF`, `test_time` for `sienaTimeTest`, `interpret_size` for `sienaRI`, `meta.siena` for `siena08`); this article uses the older names, which are the ones in the literature. The effects table of RSiena 1.6.6, as `getEffects()` returns it for twelve data configurations, has 1,201 rows. The effects that RSiena 1.6.7 to 1.6.14 added or changed, such as `WXZX` and the two-mode version of `toU`, are implemented and were compared with RSiena 1.6.14, but they are not part of the counts below; the counts change with RSiena releases. A row is one RSiena short name for one variable type and context. 80 rows belong to the stochastic-differential-equation model for continuous behavior, which is outside the scope of `nwsaom`. Of the other 1,121 rows `nwsaom` implements 1,058 (94%), and every implemented row has been compared with `siena07`, by the observed statistic and by estimates over at least two seeds; for a few effects the comparison rests on the statistic alone because the estimates are degenerate in both programs. Table 5 lists what is missing.

Table 5: RSiena effects that `nwsaom` does not implement (rows of RSiena’s effects table)

Group	Rows
Statistics of the generalised method of moments (the estimator is not implemented)	31
RSiena’s settings model (no estimate to compare with in <code>siena07</code>)	12
Effects that <code>siena07</code> itself cannot evaluate or does not offer (for example the two-mode <code>sameXOutAct</code>)	15
Placeholder rows of RSiena’s table	2
Combinations <code>nwsaom</code> does not offer (a two-mode member of <code>multinet</code> together with a behavior; a rate effect for a continuous behavior)	2
Not implemented, total	62

Beyond effects, `nwsaom` has no generalised-method-of-moments estimator, no settings model and no hierarchical Bayesian model. Joint estimation of several groups (`groups()`) is limited to the method of moments for directed one-mode networks (Section 5.9 lists the rest). The time-heterogeneity test has no counterpart for the orthogonalised tests of `sienaTimeTest(condition = TRUE)` or its period-wise group tests. `estat timetest` re-estimates the rate parameters by default, which matches a score test of a fitted time dummy; its option `fixrates` holds them fixed, which matches `sienaTimeTest`.

7 Computational architecture

7.1 How a fit runs

`nwsaom` first reads the waves into network objects and builds the model: one entry per effect, each with two functions. The first computes the effect's statistic for a whole network; summed over the periods, the statistics of the observed waves are the targets that the method of moments matches. The second computes an actor's contribution to its evaluation function for each possible minstep: creating or dropping a tie to each other actor, or, for a behavior, moving up or down by one. In a minstep, the acting actor sees only its own contributions (the myopic actor of Section 3), so for effects such as `gwesp()`, popularity and activity, the contribution is not the change in the network statistic. Covariates of `egox()`, `altx()`, `nodecov()` and `ratecov()` are centered by their means, as RSiena's `coCovar()` does, unless `nocenter` is specified.

Simulating a period starts from the observed network at the beginning of the period. Opportunities to change arrive at the rate of the period; at each one, an actor is chosen and makes a minstep with the multinomial logit probabilities of Section 3. With two dependent variables, a network and a behavior, the waiting time is drawn from their combined rate and the variable that moves is chosen in proportion to its rate.

How a simulated period ends depends on how the rates are estimated. In conditional estimation, the default for network-only models and RSiena's default for a single dependent network, a period runs until the simulated network is as far from its starting point as the observed end-of-period network (the number of tie differences), and the rate is estimated from the time that takes: its mean over the Phase 3 simulations, with their standard deviation as standard error. In unconditional estimation, used for co-evolution models, with composition change, and with the option `unconditional`, a period runs for a fixed time, and each rate is a method-of-moments parameter like the effects, with the distance between the observed waves as its target statistic.

The three phases of Section 3 then follow RSiena's defaults: the gain sequence, the number of subphases and their lengths, and the score-function estimate of the derivatives in Phase 1. The estimates, their covariance matrix, the convergence t-ratios, and RSiena's name for each effect are stored in `e()`.

7.2 A compiled backend for simulation

Almost all of a SAOM fit's time goes into simulating the minstep process: Phase 1 and Phase 3 each simulate every period hundreds of times, and every Robbins-Monro step in Phase 2 simulates it again. `nwsaom` therefore runs the simulations, Phase 1's derivative estimates and the conditional-estimation step in a compiled C plugin. The data are passed to the plugin once per fit, and each phase hands it whole batches of simulations at a time. Every model `nwsaom` accepts runs this way, including co-evolution, undirected, two-mode, several-network, composition-change and missing-data models; there is no option to turn it on, and no difference in how results should be read. The older effects also have a Mata implementation of the simulation, with the same results but hundreds of times slower (model N1 of Section 7.3: a median of 47 seconds in the Mata engine against 0.23 seconds in the plugin); the effects added later exist only in the plugin, because a Mata version would be too slow to use, and `nwsaom` stops with a

message if the plugin is missing or too old for them; joint estimation of several groups exists only in the plugin as well. `e(engine)` says which of the two ran. The Mata code of `nwsaom` is compiled into two libraries that are built with Stata 17, which makes them work in Stata 17 and 18.

The plugin also uses several processor cores. Phase 1 and Phase 3 simulations are independent of one another and run on worker threads. Phase 2 cannot be split up the same way, because each Robbins-Monro step needs the previous step’s estimate, so it parallelizes only over the periods of a step, and only when the network is large enough for threads to pay off. `cores(#)` sets the number of threads; the default is all physical cores. Each simulation draws from its own random-number stream, derived from `seed()` and the simulation’s position in the algorithm, so a given seed gives identical estimates whatever the number of threads.

7.3 Benchmark against RSiena

To check both correctness and speed, `nwsaom` and RSiena 1.6.14 (R 4.6.0) were run on the same 28 models and the same data, with each program’s default algorithm settings: 20 models on the Glasgow data and larger copies of it (N1 to T200 in Table 6) and 8 further models of other families (X1 to X8, Section 7.6). The data of the first 20 are the 50-actor Glasgow panel used throughout this article (RSiena’s `s50` data); the larger networks are 2, 4 or 10 disconnected copies of it ($n = 100, 200, 500$). Network-only models use conditional estimation, RSiena’s default for a single dependent network, except N7, which uses unconditional estimation. Each model was fitted with seeds 1–7 in both programs (seeds 1–5 for the fits above one minute, seeds 1–3 for those above three minutes), and both ran on a single thread: RSiena’s default, and `cores(1)` for `nwsaom`. The programs were run alternately, one process at a time, each fit in a fresh process after an untimed warm-up fit that loads the program, with a kill switch of ten minutes per fit. All timings are wall time per fit on one machine (Apple M5 Pro (18 cores), macOS 26.5.1, battery power) that was otherwise idle, apart from the operating system and, during part of the multi-process runs of Table 7 and of the last fits of S500, short jobs of other processes; the one-minute load average read before each fit had a median of 2.7 (highest 8.1). The tables report medians over the fits of each program.

The realistic models R1–R5 are the kind of specification an applied analysis would use. R1 combines `outdegree`, `reciprocity`, `gwesp()`, transitive reciprocated triplets, indegree and outdegree popularity (square root), `samex(smoke1)`, and `egox()`, `altx()` and `simx()` on alcohol use. R2 keeps R1’s structural effects and smoking homophily and adds alcohol use as a co-evolving behavior (selection through `simx()`; `linear`, `quadratic` and `avalt` on the behavior side). R3 is an undirected model, and R4 and R5 repeat R1 and R2 on 200 actors. In RSiena, outdegree popularity (`outPopSqrt`) was given internal parameter 1, its documented definition. From version 1.6.1 until 1.6.12, RSiena’s default parameter of 0 made this effect’s statistic use the outdegrees at the start of the period, without the square root, while its ministep still used the square root of the current outdegree; with that default, RSiena did not converge on these models. Version 1.6.13 changed the default to parameter 1, which is the documented effect that `nwsaom` implements; the benchmark, run with version 1.6.14, sets it explicitly.

Table 6: `nwsaom` and RSiena 1.6.14 on the same models and data, both single-threaded: largest difference in a mean estimate (in RSiena standard errors) and median seconds per fit

	Specification	<i>n</i>	Waves	Max diff	<code>nwsaom</code>	RSiena	Speed-up
<i>Network</i>							
N1	<code>outdegree, reciprocity</code>	50	2	0.06	0.23	0.88	3.8
N2	+ <code>gwesp</code>	50	3	0.02	1.08	1.85	1.7
N4	+ <code>gwesp, samex()</code>	50	3	0.05	1.20	2.40	2.0
N5	+ <code>gwesp, indegpopularity, outactivity</code>	50	3	0.07	1.37	2.97	2.2
N6	+ <code>gwesp, egox(), altx(), simx()</code>	50	3	0.05	1.46	3.50	2.4
N7	N2, unconditional estimation	50	3	0.03	1.41	2.52	1.8
<i>Undirected</i>							
U1	<code>outdegree</code> , default model (forcing)	50	2	0.03	0.11	0.32	3.0
U2	<code>outdegree, symtype(joint)</code>	50	2	0.05	0.12	1.04	9.0
<i>Co-evolution</i>							
C1	N2 + alcohol: <code>linear, quadratic, avalt</code>	50	3	0.05	1.99	4.84	2.4
C2	N2 + smoking: <code>linear, quadratic, avsim</code>	50	3	0.08	2.07	5.51	2.7
C3	N1 + alcohol: <code>linear, avalt</code>	50	2	0.02	0.43	1.81	4.2
<i>Realistic</i>							
R1	full network model (see text)	50	3	0.03	2.93	5.64	1.9
R2	R1 network part + alcohol co-evolution	50	3	0.05	4.06	7.96	2.0
R3	undirected: <code>gwesp, samex(), simx()</code>	50	2	0.01	0.38	0.79	2.1
R4	R1	200	3	0.03	52.9	96.2	1.8
R5	R2	200	3	0.02	62.0	114	1.8
<i>Larger networks</i>							
S100	N2	100	3	0.02	4.64	7.35	1.6
S200	N2	200	3	0.04	20.1	31.4	1.6
S500	N2	500	3	0.07	159	236	1.5
T200	C1	200	3	0.05	30.2	69.4	2.3
<i>Two-mode</i>							
X1	<code>outdegree, indegpopularity, cycle4</code>	40×20	3	0.07	0.86	2.83	3.3
X2	<code>outdegree, indegpopularity, outactivity, cycle4, outiso, antiiniso2</code>	40×30	3	0.04	0.29	0.61	2.1
<i>Two networks</i>							
X3	both: <code>outdegree, reciprocity; crprod</code>	50	3	0.02	1.61	4.04	2.5
X4	X3 without <code>crprod</code> + smoking: <code>linear, quadratic, avalt</code> (both networks)	50	3	0.03	1.26	8.74	6.9
X5	X4 with <code>avalt</code> on network 1 only; <code>behegox, behaltx, behsim, samewwclosure:behavior</code>	50	3	0.05	4.13	12.0	2.9

continued on the next page

Table 6 continued

	Specification	<i>n</i>	Waves	Max diff	<i>nwsaom</i>	RSiena	Speed-up
X6	undirected network 1 (<i>sym1</i>) + directed network 2; <i>crprod</i>	50	3	0.04	1.12	3.22	2.9
<i>Single network</i>							
X7	<i>outdegree</i> , <i>reciprocity</i> , <i>gwersp</i> , <i>samextranstrip</i> (), <i>altxoutact</i> (), <i>egoplusaltsqx</i> ()	50	3	0.03	2.98	4.49	1.5
X8	<i>outdegree</i> , <i>reciprocity</i> , <i>ownx</i> (<i>transtrip1</i> <i>transrectrip2</i> <i>jout</i>)	50	3	0.07	1.65	2.96	1.8

Medians over seven fits (seeds 1–7; five above one minute, three above three minutes) per program, run alternately. Max diff includes the rate parameters and uses the fits that converged (overall convergence ratio < 0.25) in both programs. N7, X1, X2: unconditional estimation in both programs (*unconditional*; *cond* = FALSE); X3–X6: *nwsaom* *multinet* with *k0*(100) *k3*(1000) against RSiena *n3* = 1000, *nsub* = 4. Fits that converged, RSiena / *nwsaom*, for the models with at least one that did not: R1 7/7, 6/7; R2 6/7, 7/7; R4 4/5, 5/5; R5 4/5, 5/5; X1 1/7, 7/7; X4 6/7, 7/7; X5 6/7, 7/7; X8 6/7, 6/7.

Table 6 and figures 4 and 5 show the results. The two programs give the same estimates: across all 228 parameters, the mean estimates over seeds differ by at most 0.08 RSiena standard errors (on average 0.02). That is the size of the variation between seeds within one program: RSiena’s estimates vary between seeds by a median of 0.03 standard errors (at most 0.11). Standard errors, which are themselves simulation estimates, agree (as medians over seeds) to within 11% for the network models N1 to S500; in the co-evolution models, where the behavior parameters rest on less information, the ratio of the two programs’ standard errors ranges from 0.90 to 1.15, and over all 28 models from 0.84 to 1.15. Of the 186 *nwsaom* fits, 184 converged by RSiena’s criterion (overall maximum convergence ratio below 0.25; the exceptions are one fit each of R1 and X8); of RSiena’s 185 fits, 173 did (the exceptions are one fit each of R2, R4, X4, X5 and X8, two of R5 and six of the seven fits of X1). One fit of C2 failed in RSiena (seed 6) and is left out. Fits that did not converge are kept in the timings, but not in the comparison of the estimates.

On a single thread, *nwsaom* was faster than RSiena on every model, by a median factor of 2.1 (range 1.5–9.0). The factor depends more on the effects than on network size: the network models with *gwersp*() (N2, N4–N7) run 1.7 to 2.4 times as fast as in RSiena, while the models without it (N1, U1, U2, C3) run 3.0 to 9.0 times as fast. The advantage is smaller on the large networks: the 200-actor models R4 and R5 and the *gwersp*() models on 100 to 500 actors (S100, S200, S500) run 1.5 to 1.8 times as fast, and the co-evolution model T200 2.3 times as fast. In absolute terms the savings grow with the size of the task: for the realistic model on 200 actors (R4), one fit takes *nwsaom* 53 seconds on a single thread against 96 seconds in RSiena, so a model fitted a dozen times during an analysis saves about 9 minutes before any use of multiple cores.

Profiling RSiena shows where the difference comes from. A fit runs about 2,000 simulations of the network (the Phase 2 steps and the 1,000 Phase 3 replicates), and RSiena runs this loop in R, calling its compiled code once per simulation. For the model with *outdegree* and *reciprocity* only (N1), 38% of RSiena’s time is spent in the compiled simulation and 62% in the R code around it; with *gwersp*() (N2), the simulation takes 86% and the R code 14% (one fit each, R’s sampling profiler). *nwsaom* passes the data to its plugin once per fit and has the plugin run whole batches of simulations, so almost all of this overhead disappears; this is why the speed-up is largest for models that are cheap to simulate (N1, U2, C3). For N2, removing the overhead alone would make *nwsaom* at most $1/0.86 \approx 1.2$ times as fast; that it is 1.7 times as fast means its simulation code is also faster than RSiena’s on this model.

7.4 Why *nwsaom* is faster

Both programs run the same algorithm: the same Robbins-Monro phases, the same number of simulations, and the same ministep model. The speed-up therefore comes from how the simulation is carried out, and three differences account for it.

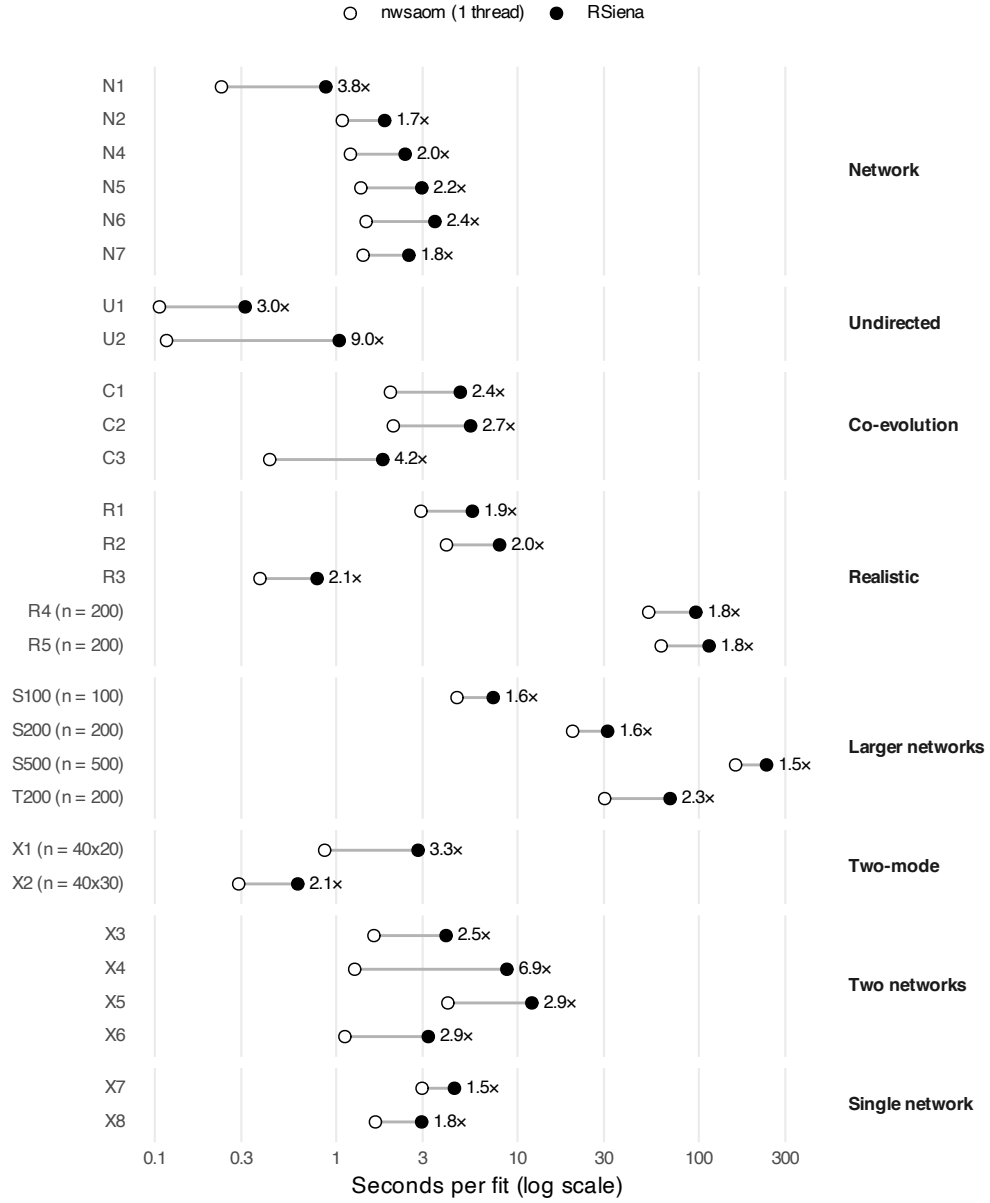


Figure 4: Median seconds per fit (log scale), *nwsaom* on one thread (open circles) and *RSiena* (filled circles), with the speed-up of *nwsaom*. Models as in Table 6.

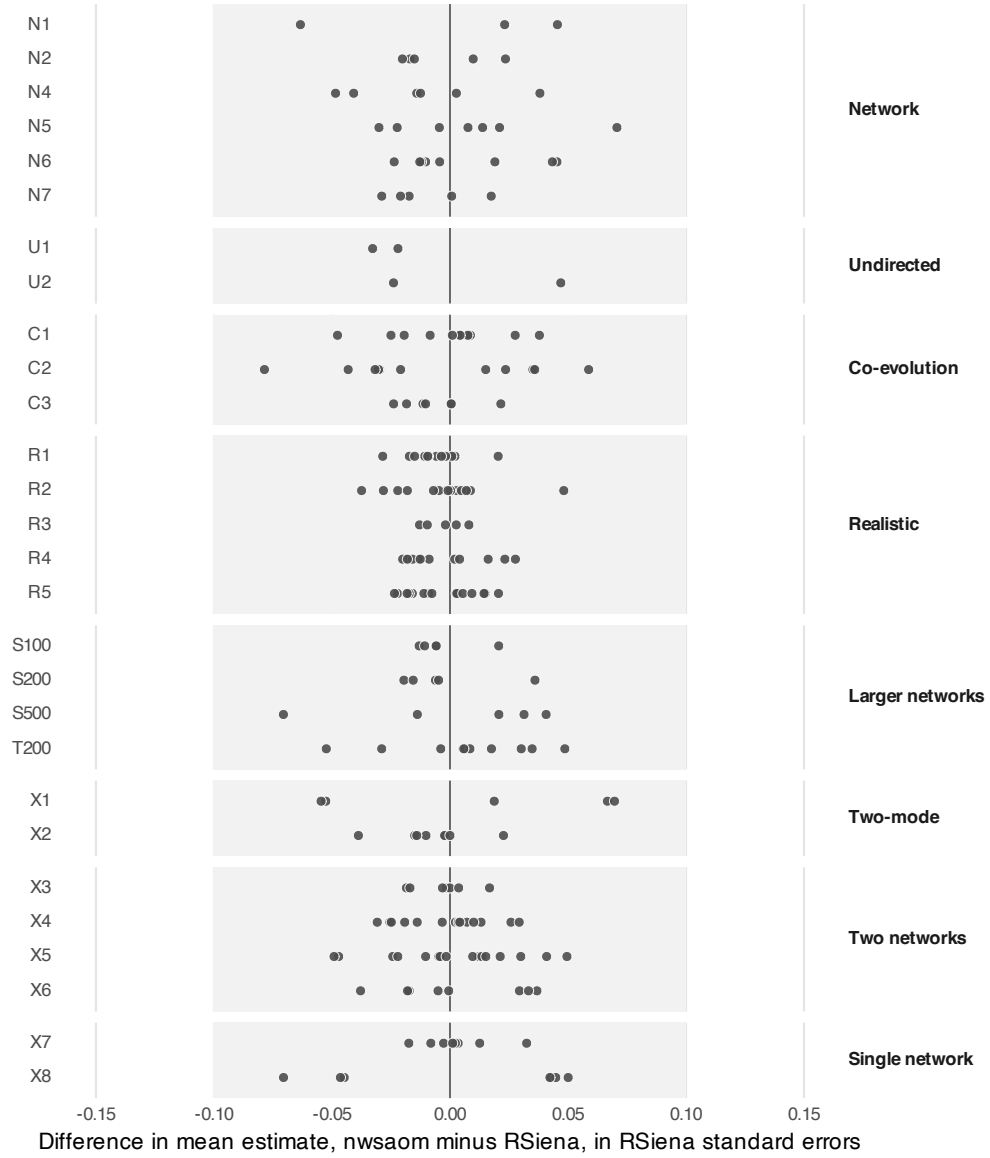


Figure 5: Difference between the mean estimates of `nwsaom` and `RSiena` for every parameter of every model, in `RSiena` standard errors. The shaded band marks ± 0.1 .

The simulation loop runs in compiled code. RSiena calls its compiled simulation once per simulated network from R, and the R code around it takes 14% to 62% of the time of the single-network models of Table 6 (see the profile above). `nwsaom` hands the plugin whole batches of simulations, so this overhead largely disappears. This matters most for models that are cheap to simulate, and it is why the largest single-network speed-ups are for `outdegree` and `reciprocity` models.

The change statistics of an actor are computed together. At each minstep an actor weighs all of its possible tie changes. RSiena’s simulation code (`NetworkVariable::calculateTieFlipContributions`) loops over the alternatives and, for each alternative, over the effects, and calls each effect’s method on its own. The plugin turns the loops around: for each effect it passes once over all alternatives of the actor, in tight loops over arrays, and quantities that several alternatives share are computed once per actor instead of once per alternative. For example, the number of two-paths through a tie is counted for all alternatives together, and the four-cycle count that the two-mode benchmark models use is accumulated for the whole row of alternatives in one sweep. The plugin keeps each network as a dense adjacency matrix together with neighbor lists, so a tie lookup is an array access.

Several networks share one simulation. In models with several dependent variables RSiena handles each variable as a separate object with its own caches, and the cross-network effects reach into the other variable’s cache. The plugin keeps all networks of a period in one structure and evaluates the cross-network terms on it directly. The models with two networks and with a behavior in Table 6 (2.5 to 6.9 times as fast) gain from this, but RSiena was not profiled on them, so this is not claimed to be the whole explanation.

Two-mode networks are handled as such. In a two-mode network the free alternatives of an actor are the nodes of the other mode only. For each actor the plugin keeps these as a list, loops over it, and evaluates the degree effects of the two-mode models (`indegpopularity`, `outactivity`, `outiso`, `antiiniso` and `antiiniso2`, `in3plus`) directly on it. The two-mode model X2 of Table 6 is the example: `nwsaom` needs 0.29 seconds per fit against 0.61 in RSiena.

These explanations rest on the profile of RSiena’s R and C++ time shares given above and on reading the RSiena source; RSiena’s C++ code itself was not instrumented. The speed-up is smallest for the covariate models X7 and X8 (1.5 and 1.8 times), where RSiena’s per-effect code does little work per alternative and the difference is mostly the simulation loop. The multi-core speed-up in Section 7.5 comes on top of all this.

7.5 Using several cores

Table 6 compares single-threaded runs because that is RSiena’s default. RSiena can also spread a fit over several R processes (`siena07(..., useCluster = TRUE, nbrNodes = k)`), which has to be requested explicitly. `nwsaom` uses threads by default. Table 7 fits the realistic models R1, R4 and R5 with 1, 2, 5 and 10 threads in `nwsaom` and, for comparison, with 1, 5 and 10 R processes in RSiena (three seeds each, run in turn).

Table 7: `nwsaom` with 1 to 10 threads (in parentheses, speed-up over one thread) and RSiena with 1, 5 and 10 R processes: median seconds per fit over three seeds. The estimates of `nwsaom` are identical for every number of threads.

Model	<i>n</i>	<code>nwsaom</code>				RSiena, processes		
		<code>cores(1)</code>	<code>cores(2)</code>	<code>cores(5)</code>	<code>cores(10)</code>	1	5	10
R1	50	2.96 (1.0)	2.38 (1.2)	2.02 (1.5)	1.91 (1.5)	5.08	3.65	4.02
R4	200	54.4 (1.0)	42.5 (1.3)	34.5 (1.6)	31.9 (1.7)	99.6	34.4	30.7
R5	200	60.4 (1.0)	34.8 (1.7)	28.8 (2.1)	25.9 (2.3)	117	47.9	49.4

More threads help, but less than in proportion: ten threads make the network models R1 and R4 about 1.5 and 1.7 times as fast, and the co-evolution model R5 2.3 times as fast. Phase 2, which takes much of a fit’s time, cannot be split across threads, because each Robbins-Monro step needs the estimate of the previous one; only Phases 1 and 3 run fully in parallel. Most of the gain comes from the first few threads. Comparing each program at its default, `nwsaom` with `cores(10)` is 2.7 to 4.5 times as

fast as single-threaded RSiena on these three models. The estimates of `nwsaom` are identical for every number of threads: each simulation’s random numbers depend only on the seed and the simulation’s position, not on which thread runs it.

RSiena run on several R processes closes much of this gap, and for the larger models it catches up. On R4, RSiena with five processes takes 34.4 seconds and with ten processes 30.7 seconds, against 31.9 seconds for `nwsaom` with ten threads, so the two programs are about equally fast there; on R5 the best RSiena time is 47.9 seconds against 25.9 seconds, and on R1 3.65 seconds against 1.91 seconds. The advantage of `nwsaom` on several cores is therefore real for small models and for the co-evolution model, but not for the realistic network model on 200 actors, where RSiena with several processes does as well. (RSiena’s processes need no setting beyond `useCluster`, but they have to be asked for.)

7.6 Further models

The last three groups of Table 6 (X1 to X8) repeat the comparison for model families that the first 20 models do not cover: two-mode networks, two networks with and without a behavior, a non-directed network next to a directed one, and some of the newer effects, with the same conventions (both programs single-threaded, default settings, a fit counts as converged when RSiena’s overall convergence ratio is below 0.25). The estimates agree with RSiena’s on every model, to within 0.07 RSiena standard errors. Models with several networks are 2.5 to 6.9 times as fast, the newer single-network effects 1.5 and 1.8 times. The two-mode models are 3.3 (X1) and 2.1 (X2) times as fast: on X2, `nwsaom` needs 0.29 seconds per fit against 0.61 seconds in RSiena. Some effect sets had to be chosen so that RSiena converges (see the manual for the details); even so, RSiena met the convergence criterion in only one of the seven fits of X1, and the timings in the table are medians over all fits. These fits take less than a second to twelve seconds, so their speed-ups are less precise than those of the longer fits.

8 Conclusions

`nwsaom` brings stochastic actor-oriented models to Stata: method-of-moments estimation with RSiena’s three-phase algorithm, an effect library of 1,058 of the 1,121 effects that RSiena 1.6.6 offers for the models it covers (about 94%), every one compared with `siena07`, undirected and two-mode networks, joint network-behavior co-evolution with one or several behaviors, several co-evolving networks, and after the fit goodness of fit, simulation, score-type tests, a test for time heterogeneity, relative importance of effects, joint estimation of several groups and meta-analysis of groups. Its simulations run in a compiled, multithreaded plugin; on every benchmark model it gives the same estimates as RSiena and on a single thread it is faster than RSiena on every one: by a median factor of 2.1 on the 28 models of Table 6 (up to 9.0 times, for the undirected pairwise model), by 1.5 to 6.9 times on the further models including the two-mode and the several-network models, and by a smaller factor on the largest networks. It is faster still on several cores, although RSiena run on several R processes narrows the gap. For applied researchers working within its effect library this makes `nwsaom` the quicker route to the same results: specification search, convergence restarts, robustness checks across seeds, and goodness-of-fit simulations all take less time, and the whole analysis stays in Stata. RSiena remains the reference implementation and the program of choice for what `nwsaom` does not offer.

What `nwsaom` leaves out is listed in Table 5: the 62 effects of RSiena’s table that are not implemented (the statistics of the generalised method of moments, the settings model, and effects that RSiena itself cannot evaluate), the generalised-method-of-moments estimator itself, the settings model and hierarchical Bayesian estimation. Joint estimation of several groups is available for the method of moments, but not with maximum likelihood or Bayesian estimation, `estat gof`, `estat simulate`, `estat importance`, two-mode and non-directed networks, `multinet` or several behaviors (Section 5.9). Inside `nwsaom multinet`, composition change and missing ties are not available, a two-mode member cannot be combined with a behavior, and `estat gof`, `estat importance` and `estat simulate` are not offered. Maximum-likelihood and Bayesian estimation cover a single directed network, with or without one behavior, and neither handles missing data or composition change. Valued and signed networks are refused with an error: the ties of every wave must be 0 or 1. All results in this article were produced on macOS; the Linux and Windows plugins are built by continuous integration. `nwsaom` refuses any model it does not support with an error that says why.

Also see

`nwcommands` (Grund, 2026a), which includes `nwplot` for plotting any of the observed or simulated waves a SAOM fit produces; further model packages and extensions: <https://nwcommands.org>.

Acknowledgments

`nwsaom` would not exist without the Siena project. The stochastic actor-oriented model is due to Tom Snijders (Snijders, 1996, 2001), and RSiena (Ripley et al., 2024), developed by Ruth Ripley, Tom Snijders, Zsofia Boda, Andras Voros, Paulina Preciado and many contributors, is the reference implementation that `nwsaom` follows and is compared with. The source code of RSiena and of multiSiena (Snijders et al., 2024) was the specification for the Stata code. `nwsaom` is distributed under the GNU General Public License, as are RSiena and multiSiena; it is not affiliated with or endorsed by the Siena project, and errors are mine. I am grateful to the developers for making their software and manual 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, and of the model packages `nwsaom`, `nwergm`, `nwdynam` and `nwrem` that build on it.

9 References

- Block, P., J. Koskinen, J. Hollway, C. Steglich, and C. Stadtfeld. 2018. Change We Can Believe In: Comparing Longitudinal Network Models on Consistency, Interpretability and Predictive Power. *Social Networks* 52: 180–191.
- Block, P., C. Stadtfeld, and T. A. B. Snijders. 2019. Forms of Dependence: Comparing SAOMs and ERGMs From Basic Principles. *Sociological Methods and Research* 48(1): 202–239.
- Frank, O., and D. Strauss. 1986. Markov graphs. *Journal of the American Statistical Association* 81(395): 832–842.
- Grund, T. U. 2026. `nwsaom`: Stochastic Actor-Oriented Models in Stata. Manual. Zenodo. <https://doi.org/10.5281/zenodo.23623807>.
- . 2026a. `nwcommands`: Scalable and Extensible Network Analysis in Stata. Manuscript under review.
- . 2027. *Network Analysis with Stata: Concepts, Methods, and Applications*. College Station, TX: Stata Press. Forthcoming.
- Huisman, M., and T. A. B. Snijders. 2003. Statistical Analysis of Longitudinal Network Data With Changing Composition. *Sociological Methods and Research* 32(2): 253–287.
- Kalish, Y. 2020. Stochastic Actor-Oriented Models for the Co-Evolution of Networks and Behavior: An Introduction and Tutorial. *Organizational Research Methods* 23(3): 511–534.
- Koskinen, J. H., and T. A. B. Snijders. 2007. Bayesian Inference for Dynamic Social Network Data. *Journal of Statistical Planning and Inference* 137(12): 3930–3938.

- Lospinoso, J., and T. A. B. Snijders. 2019. Goodness of Fit for Stochastic Actor-Oriented Models. *Methodological Innovations* 12(3).
- Lusher, D., J. Koskinen, and G. Robins, eds. 2013. *Exponential Random Graph Models for Social Networks: Theory, Methods, and Applications*. Cambridge: Cambridge University Press.
- McFadden, D. 1973. Conditional Logit Analysis of Qualitative Choice Behavior. In *Frontiers in Econometrics*, ed. P. Zarembka, 105–142. New York: Academic Press.
- Michell, L., and P. West. 1996. Peer Pressure to Smoke: the Meaning Depends on the Method. *Health Education Research* 11(1): 39–49.
- Pearson, M., C. Steglich, and T. A. B. Snijders. 2006. Homophily and Assimilation Among Sport-Active Adolescent Substance Users. *Connections* 27(1): 47–63.
- Ripley, R. M., T. A. B. Snijders, Z. Boda, A. Voros, and P. Preciado. 2024. Manual for RSiena. R package documentation.
- Robins, G., P. Pattison, Y. Kalish, and D. Lusher. 2007. An introduction to exponential random graph (p^*) models for social networks. *Social Networks* 29(2): 173–191.
- Schweinberger, M. 2012. Statistical modelling of network panel data: Goodness of fit. *British Journal of Mathematical and Statistical Psychology* 65: 263–281.
- Snijders, T. A. B. 1996. Stochastic Actor-Oriented Models for Network Change. *Journal of Mathematical Sociology* 21(1-2): 149–172.
- . 2001. The Statistical Evaluation of Social Network Dynamics. *Sociological Methodology* 31(1): 361–395.
- . 2011. Statistical Models for Social Networks. *Annual Review of Sociology* 37: 131–153.
- . 2017. Stochastic Actor-Oriented Models for Network Dynamics. *Annual Review of Statistics and Its Application* 4(1): 343–363.
- Snijders, T. A. B., and C. Baerveldt. 2003. A multilevel network study of the effects of delinquent behavior on friendship evolution. *Journal of Mathematical Sociology* 27: 123–151.
- Snijders, T. A. B., G. G. van de Bunt, and C. E. G. Steglich. 2010. Introduction to Stochastic Actor-Based Models for Network Dynamics. *Social Networks* 32(1): 44–60.
- Snijders, T. A. B., J. Koskinen, and R. M. Ripley. 2024. multiSiena: Bayesian and Multilevel Analysis of Stochastic Actor-Oriented Models. R package, version 1.2.40, <https://www.stats.ox.ac.uk/~snijders/siena/>.
- Steglich, C., T. A. B. Snijders, and M. Pearson. 2010. Dynamic Networks and Behavior: Separating Selection from Influence. *Sociological Methodology* 40(1): 329–393.
- Wasserman, S., and P. Pattison. 1996. Logit models and logistic regressions for social networks: I. An introduction to Markov graphs and p^* . *Psychometrika* 61(3): 401–425.