PLOS COMPLEX SYSTEMS

RESEARCH ARTICLE

ALAAMEE: Open-source software for fitting autologistic actor attribute models

Alex Stivala[1]*, **Peng Wang**[1,2], **Alessandro Lomi**[3]

1 Institute of Computing, Università della Svizzera italiana, Lugano, Switzerland, **2** Centre for Transformative Innovation, Swinburne University of Technology, Melbourne, Australia, **3** Università della Svizzera italiana, Lugano, Switzerland

* alexander.stivala@usi.ch

OPEN ACCESS

Citation: Stivala A, Wang P, Lomi A (2024) ALAAMEE: Open-source software for fitting autologistic actor attribute models. PLOS Complex Syst 1(4): e0000021. https://doi.org/10.1371/journal.pcsy.0000021

Editor: Marian-Gabriel Hâncean, University of Bucharest: Universitatea din Bucuresti, ROMANIA

Received: May 6, 2024

Accepted: September 16, 2024

Published: December 5, 2024

Data Availability Statement: The "Project 90" data is available upon registration from https://oprdata.princeton.edu/archive/p90/. The excerpt of 50 girls from the "Teenage friends and lifestyle study" data is available from https://www.stats.ox.ac.uk/~snijders/siena/s50_data.htm. The "Deezer" data is available from the Stanford large network dataset collection at https://snap.stanford.edu/data/gemsec-Deezer.html. All other data, source code, and scripts are freely available from https://github.com/stivalaa/ALAAMEE.

Abstract

The autologistic actor attribute model (ALAAM) is a model for social influence, derived from the more widely known exponential-family random graph model (ERGM). ALAAMs can be used to estimate parameters corresponding to multiple forms of social contagion associated with network structure and actor covariates. This work introduces ALAAMEE, open-source Python software for estimation, simulation, and goodness-of-fit testing for ALAAM models. ALAAMEE implements both the stochastic approximation and equilibrium expectation (EE) algorithms for ALAAM parameter estimation, including estimation from snowball sampled network data. It implements data structures and statistics for undirected, directed, and bipartite networks. We use a simulation study to assess the accuracy of the EE algorithm for ALAAM parameter estimation and statistical inference, and demonstrate the use of ALAAMEE with empirical examples using both small (fewer than 100 nodes) and large (more than 10 000 nodes) networks.

Author summary

If we observe a social network, along with some attributes of the actors within it, including an opinion, behaviour, or belief (we will call this the outcome attribute) that we might suppose to be socially contagious, how might we test this supposition? The situation is complicated by the fact that, if the outcome attribute is indeed socially contagious, then its value for one actor will depend on its value for that actor's network neighbours (friends, for instance). Even further complexity arises if we suppose that the outcome is not simply contagious like a disease, where it is likely to be transmitted from a single network neighbour, but its adoption might instead depend on particular patterns of adoption of the outcome attribute in the local network structure surrounding an actor. The "autologistic actor attribute model" (ALAAM) is a statistical model that, unlike some better-known models, can handle such situations. In this work we describe open-source software called ALAAMEE that implements this model, and demonstrate its use on both small and large networks.

Funding: This work was funded by the Swiss National Science Foundation (https://www.snf.ch/en) grant numbers 167326 (NRP 75) and 200778 to AL. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Competing interests: The authors have declared that no competing interests exist.

Introduction

Social contagion is a form of social influence supported by social contact. Specifically, it is the process whereby actors in a social network adopt the attitudes, opinions, behaviours, or beliefs (which we refer to generically as outcome attributes here) of their network neighbours. This process may also be known as diffusion, one possible result of what is known in economics as peer effects [1]. One source of complexity in social networks and diffusion on them is that this process results in autocorrelation in such outcome variables. That is, the outcome variable is correlated across connected network nodes. Such processes are therefore often inferred from observed data with the network autocorrelation model [2–7], in which a parameter associated with network contagion is estimated.

Unlike a "simple contagion", such as, for example, disease, or diffusion of information, in which a single tie to a network neighbour with the outcome in question is likely sufficient for transmission, a "complex contagion" [8] may require a certain threshold of an actor's network neighbours to adopt a behaviour before that actor is influenced to do so [9–11]. Such a process results in an even more complex relationship between network structure, other attributes of the actors in the network, and the outcome variable [6, 12, 13].

The autologistic actor attribute model (ALAAM) is a model for social contagion which allows for such complexity. Unlike the more widely known network autocorrelation model, in which social contagion is associated with a single parameter [4, 7], the ALAAM can be used to estimate parameters corresponding to multiple mechanisms of social contagion associated with more complex relationships between network structure, the outcome variable, and other actor attributes. This is discussed further in Stivala et al. [14], and the dependency assumptions and the types of terms these allow in the ALAAM are discussed in Daraganova [15] and Koskinen & Daraganova [16].

Originally introduced by Robins et al. [17], the ALAAM is a variant of the exponential-family random graph model (ERGM) for social networks [15, 18–23]. As such, the ALAAM is a cross-sectional model: given a single observation of the social network and the outcome attribute (assumed to be binary in the ALAAM) which is hypothesized to be socially contagious, the model is used to estimate parameters relating to this contagion. In the ALAAM, the network is assumed to be fixed (exogenous), and the binary outcome variable is modeled (the variable is endogenous). This in contrast to the ERGM, in which the tie variables are modeled (tie formation is an endogenous process), based on fixed actor attributes. Note that the outcome (binary) attribute of an actor in the ALAAM is allowed to depend on its values for other actors connected to it in the social network, hence the "autologistic" in the name.

ALAAMs can also be estimated for network data obtained via snowball sampling [14, 15, 19, 24]. For a recent introduction and review of ALAAM usage, see Parker, Pallotti, & Lomi [25], and for a comprehensive survey of ALAAM applications, Stivala [26].

The most commonly used software for ALAAM modeling is the IPNet Windows application [27], or its successor, the Windows application MPNet [28, 29], which allows for ERGM and ALAAM modeling with undirected, directed, bipartite, and multilevel networks. Note that the widely-used R package statnet [30–34] for ERGM modeling, does not implement ALAAM. Although, as noted in Barnes et al. [35], an ALAAM can be considered as an ERGM for a two-mode network, with the N nodes of one mode representing the actors, and a tie from an actor to the single node of the other mode representing that actor having the outcome attribute, with the observed N node social network as a fixed covariate. And hence this could be implemented as a statnet ergm model, but whether or not it is practical is another question. This conception of an ALAAM could also be a way of implementing a multivariate ALAAM, by having more than one "outcome" node.

The exponential-family random network model (ERNM), a generalization of the ERGM and ALAAM, which models both social selection and social contagion simultaneously [36–38] is implemented using statnet. The only other publicly available code for ALAAM modeling is the R code for estimating Bayesian ALAAMs described by Koskinen & Daraganova [16].

These existing implementations are limited, both by the algorithms implemented and details of their implementation, in the size of the networks on which they can practically be used. Therefore, in order to be able use ALAAMs with large (tens of thousands of nodes or more) networks, such as, for example, those that can be collected from online social network data, an alternative is required.

This paper describes one such alternative, ALAAMEE, open-source Python code for ALAAM parameter estimation, simulation, and goodness-of-fit testing. The ALAAMEE software implements the same stochastic approximation algorithm for ALAAM parameter estimation as IPNet and MPNet do [39], which is practical for networks of the order of thousands, or in some cases tens of thousands, of nodes in size or smaller. For larger networks, it also implements the "equilibrium expectation" (EE) algorithm [40–42], which has previously been used for estimating ERGM parameters for very large networks [41, 43–45]. Because it implements both the stochastic approximation and EE algorithms, the ALAAMEE software can be used to estimate ALAAM models for both small and large (tens of thousands of nodes or more) networks. It is not practical to do the latter with existing software, and hence ALAAMEE extends the range of network data for which ALAAMs are a practical modeling choice.

The autologistic actor attribute model (ALAAM)

The ALAAM, modeling the probability of outcome attribute Y (taking the form of a binary vector y) given the network X (a matrix of binary tie variables) can be expressed as [19]:

$$\Pr_\theta(Y = y|X = x) = \frac{1}{\kappa(\theta)} \exp\left(\sum_I \theta_I z_I(y, x, w)\right) \tag{1}$$

where θ_I is the parameter corresponding to the network-attribute statistic z_I, in which the "configuration" I is defined by a combination of dependent (outcome) attribute variables y, network variables x, and actor covariates w, and $\kappa(\theta)$ is the normalizing quantity which ensures a proper probability distribution.

Just as for ERGM, parameter estimation for ALAAM is a computationally intractable problem, and so the maximum likelihood estimate (MLE) of the ALAAM parameters is found using Markov chain Monte Carlo (MCMC) methods (see Hunter et al. [46] for an overview in the ERGM context).

The sign and significance of the ALAAM parameters θ_I support inferences about the statistical relationship between the corresponding configurations $z_I(y, x, w)$ and the outcome attribute binary vector y, each conditional on all the other effects included in the model.

For example, consider the frequently used contagion effect, using an undirected network for simplicity. The statistic for contagion is the number of pairs of directly connected nodes, i and j ($i \neq j$) where $x_{ij} = 1$, in which both nodes have the outcome attribute $y_i = y_j = 1$ (see Fig 1). If the contagion parameter (that is, the parameter corresponding to the contagion statistic just defined) is found to be positive and significant, this means the contagion configuration (a directly connected pair of nodes both with the outcome attribute) occurs more frequently than expected by chance, given all the other effects included in the model. So there is a statistically significant correlation between a pair of nodes being directly connected, and both having the outcome attribute (conditional on the other effects in the model).

Name	Illustration	Description
Density		Baseline attribute density (incidence). Also used with directed networks.
Activity		Tendency for actor with the attribute to have ties.
Contagion		Tendency for actor with the attribute to be tied to an actor also with the attribute.
Two-star		Tendency of actor with attribute to have additional ties over Activity.
GWActivity		Geometrically weighted activity. This is used to account for the degree distribution of nodes with the outcome attribute, avoiding problems of near-degeneracy that can occur when using the Activity configuration.
Alter-2Star1		Partner activity two-path; tendency of an actor with the attribute to have a tie to another actor with a tie to a third actor.
Alter-2Star2		Also known as indirect partner attribute. Structural equivalence of actors with the attribute (two-path equivalence); tendency of actors with the attribute to have the same network partner in common.
Partner-Activity		Also known as partner attribute activity. Tendency of the attribute to be present on two directly connected nodes in a two-path.
Partner-Resource		Also known as partner-partner attribute; tendency of the attribute to be present in all three nodes in a two-path.
T1		Actor triangle; tendency of actor with attribute to be involved in a triangle.
T2		Partner attribute triangle; tendency of attribute to be present in actors involved in triangles with another node that also has the attribute.
T3		Partner-partner attribute triangle; tendency of attribute to be present in all three nodes in a triangle.
*attribute*_oOc		Covariate effect for continuous covariate *attribute*. The "oOc" notation is from IPNet, and we may omit this when there is no ambiguity, e.g. "Age_oOc" may also be written simply as "Age". Also used in directed and bipartite networks.
*attribute*_o_Oc		Covariate effect for continuous covariate *attribute* on partner node (outcome attribute related to continuous attribute on partner node). The "o_Oc" notation is from IPNet, and we may instead write this as "partner *attribute*".
*attribute*_oOb		Covariate effect for binary covariate *attribute*. The "oOb" notation is from IPNet, and we may omit this when there is no ambiguity, e.g. "Binary_oOb" may also be written simply as "Binary". Also used in directed and bipartite networks.

Legend:

- Node with outcome attribute
- Node irrespective of outcome attribute

Fig 1. Configurations used in ALAAMs for undirected networks.

https://doi.org/10.1371/journal.pcsy.0000021.g001

The distinction between the ERGM and the ALAAM arises from the assumptions and hypotheses behind the model. In the ERGM, modeling tie formation, we might hypothesize that having the same attribute value affects (increases, for homophily) the probability of a tie between two nodes, and test for this. In the ALAAM, modeling the outcome attribute, we might hypothesize that having a network tie affects (increases, for contagion) the probability that both nodes will have the outcome attribute. More generally, the difference between the ERGM and the ALAAM reflects the distinction introduced by Borgatti & Halgin [47] between theories of networks and network theories. The former concern the possible antecedents of network ties. The latter concern the potential outcomes of network ties. Distinguishing homophily and contagion in an observational network study is in general a difficult problem [48], and ALAAM parameters indicate only (auto -)correlation, not causation.

Once parameters for an ALAAM model of an observed network and outcome binary attribute have been estimated, these can be used to simulate, again using MCMC, a set of outcome binary vectors from the model. These simulated outcome attributes can be used for goodness-of-fit testing and model evaluation. For further explanation and examples, see Parker, Pallotti, & Lomi [25].

Sampling from the ALAAM distribution

Simulating outcome attribute vectors from a model specified by a parameter vector θ requires drawing from the ALAAM distribution defined by Eq (1). Drawing from this distribution is also necessary for the estimation algorithms (described in the following section). Due to the intractable normalizing constant $\kappa(\theta)$ in Eq (1), the Metropolis–Hastings algorithm is used for this purpose. This MCMC algorithm proposes transitions from the current outcome binary vector y to a proposed vector y' with acceptance probability

$$p_\alpha = \min\left\{1, \frac{q(y' \to y)\Pr_\theta(Y = y'|X = x)}{q(y \to y')\Pr_\theta(Y = y|X = x)}\right\} \tag{2}$$

$$= \min\left\{1, \frac{q(y' \to y)}{q(y \to y')}\exp\left(\sum_I \theta_I \delta_I(y, x, w)\right)\right\} \tag{3}$$

where q is the proposal distribution and $\delta_I(y, x, w) = z_I(y', x, w) - z_I(y, x, w)$ is the change statistic [46, 49, 50] corresponding to the statistic z_I for each configuration I in the model. Note that this means that only these change statistics δ need to be computed, rather than the statistics z themselves.

If the proposal distribution is symmetric, that is $q(y' \to y) = q(y \to y')$, then Eq (3) can be simplified to

$$p_\alpha = \min\left\{1, \exp\left(\sum_I \theta_I \delta_I(y, x, w)\right)\right\} \tag{4}$$

and this is an instance of the Metropolis, rather than the more general Metropolis–Hastings, algorithm.

The simplest choice of proposal, which we refer to as the basic ALAAM sampler, is to choose a node uniformly at random and toggle its outcome attribute value, that is, change it to 1 if it was 0, and to 0 if it was 1. In the former case the change statistic is $\delta_I(y, x, w)$ and in the latter case it is $-\delta_I(y, x, w)$. This proposal distribution is symmetric and so the simplified Metropolis acceptance probability, Eq (4), can be used.

In the case of ERGMs, the analogous basic sampler (toggling a dyad) often results in very low acceptance rates, due to the sparsity of the network, and more efficient samplers (with asymmetric proposal distributions) such as the TNT ("tie / no tie") sampler [51] or IFD (improved fixed density) sampler [40] are frequently used. In the case of ALAAM, however, the frequency (density or incidence) of the outcome variable is typically orders of magnitude higher the graph density of a sparse graph such as those to which ERGMs are usually applied, and so more advanced samplers are not necessary. We implemented an ALAAM analogue of the TNT sampler, the ZOO ("zero or one") sampler, but found that it had little or no practical application (it did not speed up the process of sampling from the ALAAM distribution). The basic ALAAM sampler is used for all experiments and examples in this work.

Estimation algorithms

Stochastic approximation (SA)

Stochastic approximation of ERGM parameters using the Robbins–Monro algorithm [52] is described in detail in Snijders [39], and recapitulated more briefly in Koskinen & Snijders [53] and very briefly in Hunter et al. [46]. Here we briefly describe this algorithm and its application to the estimation of ALAAM parameters. As this is a relatively straightforward application of the same algorithm in a slightly different context, the preceding references can referred to for further detail.

The aim of the algorithm is to find the MLE of the parameter vector θ, which solves

$$\nabla_\theta \log \Pr_\theta(Y = y) = z(y) - \mathrm{E}_\theta(z(Y)) = 0 \tag{5}$$

where $\nabla_\theta \log \Pr_\theta(Y = y)$ is the gradient of the log-likelihood $\log \Pr_\theta(Y = y)$ [46] (but note here Y and y are random binary outcome attribute vectors and instances of binary outcome attribute vectors, respectively, not matrices of tie variables as used in ERGM). That is, we aim to find the parameter vector θ such that $\mathrm{E}_\theta(z(Y))$, the expected value of the statistics with respect to the probability distribution (1), is equal to the observed statistics $z(y) = z(y_{\text{obs}}) = z_{\text{obs}}$.

The SA algorithm to do this consists of three phases. Phase 1 estimates the covariance matrix based on a small number of samples M sampled from the ALAAM distribution at the initial parameter value $\theta^{(0)}$ as

$$D = \frac{1}{M} U^{\mathsf{T}} U \tag{6}$$

where U is a matrix such that each row $U_{i,\cdot} = Z_{i,\cdot} - \bar{z}$ where Z is a matrix where each row is the vector of statistics for a sample and $\bar{z}$ is the vector of mean statistics over the M samples. We use $M = M_1 = 7 + 3p$ where p is the number of parameters in the model.

Phase 2 is the main part of the algorithm, where the parameter estimates θ are updated using the Robbins–Monro algorithm according to:

$$\theta^{(t+1)} = \theta^{(t)} - a_i D^{-1}(z^{(t)} - z_{\text{obs}}) \tag{7}$$

where $z^{(t)} = z(y^{(t)})$ is the vector of statistics of a sample $y^{(t)}$ drawn from the ALAAM distribution with parameter vector $\theta^{(t)}$ and a_i is a sequence of positive step sizes converging to zero, satisfying the Robbins–Monro convergence criteria: at each subphase this value is halved. We use five subphases and $a_0 = 0.01$.

Phase 3 estimates the covariance matrix of the estimator to estimate the standard error, checks the approximate validity of Eq (5) and tests for convergence. This is done by estimating the covariance matrix D just as in phase 1 using Eq (6), but with a larger number of samples $M = M_3 = 1000$ drawn from the ALAAM distribution at the final estimated value of θ. If this

matrix is singular then the model may be degenerate. Otherwise the standard error is estimated as

$$\mathrm{se}(\theta) = \sqrt{\mathrm{diag}(D^{-1})} \tag{8}$$

and the t-ratio as

$$t_i = \frac{\bar{z}_i - z_{\mathrm{obs},i}}{\mathrm{sd}(z_i)} \tag{9}$$

for each parameter i in the model. If the absolute value of the t-ratio $|t_i| < 0.1$ for each parameter i then the estimation is considered to be converged: the difference between the expected value of the statistic under the model with the estimated parameters θ and its observed value is small enough that we consider Eq (5) to be satisfied. If this convergence criterion is not satisfied, then the entire algorithm may be run again, starting at the current estimated value of the parameters θ.

In this algorithm, whenever samples are drawn from the ALAAM distribution (using the Metropolis–Hastings algorithm described in section "Sampling from the ALAAM distribution") an interval of $10N$ iterations, where N is the number of nodes, is used between each sample to reduce autocorrelation between samples. In phase 3, a burn-in period of M_3N iterations is used to try to ensure that it has reached equilibrium before samples are taken.

Equilibrium expectation (EE)

The stochastic approximation algorithm is a robust and reliable algorithm that does not require particularly good initial estimates [53]. However, it has the shortcoming that it is not scalable to very large networks since it involves running the MCMC algorithm for sampling from the ALAAM distribution to convergence each time the parameter estimates are updated.

The EE algorithm [40–42] overcomes this scalability problem by not requiring potentially very long MCMC simulations between parameter updates [44]. Instead, it attempts to find the MLE, that is, solve Eq (5), by taking only a small number ($M_s = 1000$) of steps of the Metropolis–Hastings algorithm (see section "Sampling from the ALAAM distribution") between updates of the estimated parameter vector θ according to the divergence of the simulated statistics from the observed statistics. After many iterations (default value $M_{\mathrm{EE}} = 50000$) of this process, if on average this divergence is zero, then we have estimated θ values for the MLE. Note that in the EE algorithm, the Markov chain of simulated outcome attribute vectors starts at the observed value (not, for example, a zero vector or random vector), and the Markov chain continues across iterations of the EE algorithm (that is, the simulated outcome attribute vector is not reset after the parameters are updated).

The parameter update step in the EE algorithm is

$$\theta^{(t+1)} = \theta^{(t)} - \mathrm{sign}(z^{(t)} - z_{\mathrm{obs}}) \cdot r \cdot \max\{|\theta^{(t)}|, c\} \tag{10}$$

where $z^{(t)} = z(y^{(t)})$ is the vector of statistics at the current state of the simulated outcome attribute vector $y^{(t)}$ (initially $z^{(0)} = z_{\mathrm{obs}}$), $r > 0$ is the learning rate ($r = 0.01$ by default) and $c = 0.01$ is a constant that ensures the algorithm does not get "stuck" at zero values of the parameters. The parameter update is actually performed as

$$\theta^{(t+1)} = \theta^{(t)} - \mathrm{sign}(d_z^{(t)}) \cdot r \cdot \max\{|\theta^{(t)}|, c\} \tag{11}$$

where $d_z^{(t+1)} = d_z^{(t)} + \delta_M$ (initially, $d_z^{(0)} = 0$), and δ_M are the change statistics for accepted moves summed over the M_s iterations of the Metropolis–Hastings algorithm in this EE

iteration. Since d_z accumulates change statistics for accepted Metropolis–Hastings moves, starting at zero for the observed outcome attribute vector, then $d_z^{(t)} = z^{(t)} - z_{\text{obs}}$ and so Eq (11) is equivalent to Eq (10), and z_{obs} need not be computed.

Note that the version of the EE algorithm used here is the simplified version of Borisenko et al. [42] rather than the original version [41, 44].

This algorithm results in chains of $\theta^{(t)}$ and $d_z^{(t)}$ values, where $0 \leq t < M_{\text{EE}}$. To compute the point estimate and standard error for θ we first thin the chains by discarding the first 1000 iterations (burn-in) and using an interval of 100 iterations between each value, resulting in $N_m = (M_{\text{EE}} - 1000)/100$ values for each.

Similar to the situation discussed in Hunter & Handcock [49] for ERGM estimation, there are two sources of error that need to be considered: the MCMC error in our point estimate of θ, and the error inherent in using the MLE. The point estimate for θ and its asymptotic covariance matrix T are estimated by the multivariate batch means method [54–58] from the thinned $\theta^{(t)}$ chain, as is the asymptotic covariance matrix for the simulated statistics, V, from the thinned $d_z^{(t)}$ chain. The estimated standard errors for the θ estimates are then estimated by

$$\text{se}(\theta) = \sqrt{\text{diag}(W)} \tag{12}$$

where

$$W = \frac{1}{N_m} T + \left(\frac{1}{N_m} V\right)^{-1}. \tag{13}$$

In doing the computations described above, several convergence checks can be applied. If the covariance matrix V is computationally singular, then the model is possibly degenerate, and this estimation is not valid. The estimation can also be discarded as non-converged if, heuristically, it appears not to converge due to θ values that are infinite or NaN ("not a number") or simply "too large" (heuristically, $|\theta| > 10^{10}$), or if the average of the d_z values after the burn-in period is not approximately zero (for this purpose we use the heuristic that an estimation is considered non-converged if $|\bar{d}_z/\text{sd}(d_z)| > 0.3$).

Unlike the stochastic approximation algorithm (but not unlike the MCMC MLE algorithm used for ERGM parameter estimation in statnet [30–34, 49, 53, 59]), it is important to have a reasonable initial estimate of the parameters for the EE algorithm, and for that purpose we use "Algorithm S" from the EE algorithm publications [41, 44], which, as discussed in those papers and also the paper describing the simplified EE algorithm [42] used here, is equivalent to contrastive divergence (CD) [60], which has been applied to ERGM parameter estimation [59, 61, 62]. Essentially this algorithm amounts to running a small number (we use 100 here) of the EE parameter update steps (10), but without actually performing any accepted moves in the Metropolis–Hastings algorithm (that is, leaving the outcome attribute vector at the observed value).

When using the original version of this algorithm to estimate ERGM parameters [41, 44], it was found that it tended to have low statistical power because of large estimated standard errors. For this reason, a meta-analysis technique, similar to that used for combining ERGM parameter estimates from multiple snowball samples of large networks [63] is used. Specifically, the EE algorithm is run multiple times (which can be done in parallel, taking advantage of modern multicore or parallel computing clusters), and the estimates combined by the inverse-variance weighted average [64, Ch. 4]. From the vector of N_C (the number of converged estimates) estimated parameter values θ and a vector of their associated estimated

standard errors s, the point estimate is

$$\hat{\theta} = \frac{\sum_{j=1}^{j=N_C} \theta_j / s_j^2}{\sum_{j=1}^{j=N_C} 1/s_j^2} \tag{14}$$

and the estimated standard error is

$$\mathrm{se}(\hat{\theta}) = \frac{1}{\sqrt{\sum_{j=1}^{j=N_C} 1/s_j^2}} \tag{15}$$

for each of the parameters in the model. In this work we use this method, running multiple EE algorithm runs in parallel for each ALAAM estimation.

In order to test that the parameter estimates are properly converged and the model is not degenerate or near-degenerate, ALAAM vectors can be simulated from the model with estimated parameter vector $\hat{\theta}$ and the statistics of these vectors compared to those of the observed outcome attribute vector, to verify that Eq (5) is (approximately) satisfied. Here we do this by plotting degeneracy check plots, showing trace plots and histograms of the simulated vector statistics, along with the observed values, from which it can be verified that the simulation has sufficient burn-in and large enough interval to avoid excessive autocorrelation between samples, and that the observed values are within the 95% confidence interval of sample distribution. Note that, unlike the EE algorithm itself, these simulations require sufficient burn-in and iterations to ensure the simulation has reached the stationary distribution and the samples are not too autocorrelated. For very large networks, therefore, this can mean that this degeneracy check simulation could possibly take longer than the estimation with the EE algorithm.

If the estimation does not converge, it may be useful to reduce the value of the learning rate r, which affects how large the parameter update steps are. As discussed in Giacomarra et al. [65], it needs to be sufficiently small for the EE algorithm to converge to the MLE, but a smaller learning rate r may come at the expense of requiring a larger number of iterations M_{EE}. For example, in estimating ALAAM parameters for the very large (approximately 1.6 million node) network in Stivala [26], the learning rate was set to $r = 0.001$. For all estimations in this work, the default values of the learning rate ($r = 0.01$) and iterations ($M_{\mathrm{EE}} = 50000$) were used.

Materials and methods

Implementation

ALAAMEE is implemented in Python 3 [66], and uses the NumPy [67] package for array data types and linear algebra. The Python code does not require any other packages, simply using a "dictionary of dictionaries" data structure to implement graphs. Just as described in Bianchi et al. [68], this allows simple and efficient graph construction and implementation of the graph operations required, such as testing for the existence of an edge or arc, and iterating over the neighbours of a node.

ALAAM parameters are usually estimated by stochastic approximation with the Robbins–Monro algorithm [39, 52]. This is the method used in MPNet, and is also implemented in ALAAMEE, as described in the "Stochastic approximation (SA)" section. For the stochastic approximation algorithm, all stages of the estimation, including point estimation, estimation of standard errors from the Fisher information matrix [39], and simulation-based goodness-of-fit testing, are implemented in Python.

For larger networks, on the order of tens of thousands of nodes or more, this method may no longer be practical. For such networks, the EE algorithm can be used, and ALAAMEE also

implements this algorithm, as described in the "Equilibrium expectation (EE)" section. The EE algorithm works differently [44]. In this algorithm, a number of estimation processes (each of which is a separate Python task) are run independently. Because these runs are independent, they can be run in parallel to minimize the elapsed time taken.

From the results of these multiple estimation runs, a point estimate and estimated standard error are computed using an R [69] script that uses the mcmcse R package [70] to estimate the Fisher information matrix and the asymptotic covariance matrix for the MCMC standard error with the multivariate batch means method [54–58]. The overall estimate and its estimated standard error are then computed from these multiple independent runs as the inverse variance weighted average [64, Ch. 4].

Scripts are provided to run the parallel estimations using either GNU Parallel [71] or, for Linux compute clusters, SLURM [72] job arrays. Scripts for processing network data and converting it between different formats, taking snowball samples from networks, making plots, and computing the Wilson score interval [73] for the binomial proportion score interval (used to find confidence intervals for Type I and II error rates in the simulation study described in the "Results and discussion" section) are written in R and use the igraph [74], ggplot2 [75], and PropCIs [76] R packages.

ALAAMEE also implements estimation (and simulation) conditional on snowball sampled network structure, as described in Pattison et al. [77] and Stivala et al. [63] in the context of ERGM, and in Daraganova [15], Daraganova & Pattison [78], Kashima et al. [24], and Stivala et al. [14] for ALAAM.

Change statistics

It is a property of the ALAAM that the odds of a node having the outcome attribute equal to one, conditional on the values of the outcome attribute for the other nodes, is a function of the change in the vector of statistics associated with switching the outcome attribute value of that node from 0 to 1. It follows that, in implementing the MCMC process for ALAAM simulation and estimation, only these change statistics need be implemented (see section "Sampling from the ALAAM distribution" for details). That is, rather than writing functions to count each of the configurations in the data, we need only write functions that compute the change statistic value resulting from changing the outcome attribute value of a given node from 0 to 1. The value of the statistics in observed data can be computed by summing the change statistics for each element of the outcome attribute vector that is equal to 1.

For example, consider the simplest statistic, attribute Density, sometimes instead called Incidence [25] to avoid confusion with graph density. This statistic is simply the number of nodes with the outcome attribute equal to 1. Its corresponding change statistic is simply the constant 1, since for any node, if its outcome attribute value is switched from 0 to 1, then that increases the Density statistic by 1.

Every change statistic in ALAAMEE is implemented as a function of the form `change`*`Statname`*` (G, A, i)` where `G` is a `Graph` (or `Digraph` or `BipartiteGraph`, as appropriate to the change statistic) object, `A` is a binary outcome attribute vector, and `i` is a node identifier. The function returns the change statistic value for switching the node outcome attribute value `A[i]` from 0 to 1. It is a precondition of the change statistic functions that `A[i] == 0`. Some examples are shown in Fig 2. Python "docstrings" are used to document the change statistic functions, including ASCII diagrams illustrating the corresponding configurations. This allows the documentation to be viewed at the interactive Python prompt with the built-in `help()` function.

```python
def changeSender(G, A, i):
    """
    change statistic for Sender

    *->o
    """
    return G.outdegree(i)

def changeReceiver(G, A, i):
    """
    change statistic for Receiver

    *<-o
    """
    return G.indegree(i)

def changeEgoInTwoStar(G, A, i):
    """
    Change statistic for EgoIn2Star

    *<--o
     <
      \
       o
    """
    return (G.indegree(i) * (G.indegree(i) - 1)
    )/2.0 if G.indegree(i) > 1 else 0

def changeReciprocity(G, A, i):
    """
    change statistic for Reciprocity

    *<->o
    """
    delta = 0
    for u in G.outIterator(i):
        if G.isArc(u, i):
            delta += 1
    return delta

def changeContagionReciprocity(G, A, i):
    """
    change statistic for Contagion Reciprocity
    (mutual contagion)

    *<->*
    """
    delta = 0
    for u in G.outIterator(i):
        if A[u] == 1 and G.isArc(u, i):
            delta += 1
    return delta
```

Fig 2. Example change statistic implementations. Some change statistics for directed networks.

https://doi.org/10.1371/journal.pcsy.0000021.g002

The first two functions shown in Fig 2 implement, respectively, the Sender and Receiver change statistics (see Fig 3). These are quite straightforward. The Sender statistic counts the number of outgoing ties from actors that have the outcome attribute equal to 1. Therefore, the change statistic when a node i has its outcome variable changed from 0 to 1 is the out-degree of i, and similarly for Receiver and in-degree.

The third function, `changeEgoInTwoStar`, is slightly more complicated. This statistic (see Fig 3) counts the number of pairs of incoming arcs to nodes that have the outcome attribute equal to 1. Hence the change statistic when a node i has its outcome variable changed from 0 to 1 is the number of pairs of incoming arcs to node i, that is $\binom{d_i^{(\text{in})}}{2} = d_i^{(\text{in})}(d_i^{(\text{in})} - 1)/2$, where $d_i^{(\text{in})}$ is the in-degree of node i (and $d_i^{(\text{in})} > 1$).

The last function shown in Fig 2 implements the change statistic for the Contagion reciprocity statistic, also known as mutual contagion (see Fig 3). This statistic counts the number of pairs of nodes with a reciprocated (mutual) tie between them, in which both nodes have the outcome attribute equal to 1. Hence the change statistic, computing the change in the statistic when a node i has its outcome variable changed from 0 to 1, counts the number of out-neighbours of i, i.e. nodes u such that there is an arc $i \to u$, such that u has the outcome variable equal to 1 and there is also an arc $u \to i$. `changeReciprocity` is similar, but does not require that both nodes have the outcome attribute equal to 1, only that node i does. (Note that the code for `changeContagionReciprocity` in Fig 2 can be written less verbosely and more elegantly, and arguably in more idiomatic Python style, in a single line using a list comprehension: `return sum([(G.isArc(u, i)and A[u] == 1) for u in G.outIterator(i)])`, however this turns out to be slower; similar considerations apply to `changeReciprocity`.) The ALAAMEE source code repository (https://github.com/stivalaa/ALAAMEE) includes unit tests for verifying the correctness of change statistic implementations against known correct values, verifying that alternative implementations give the same results, and comparing their execution speeds.

Some change statistics make use of nodal attribute values. These change statistic functions take also as their first parameter the name of the attribute to use, used as the key in the relevant attribute dictionary in the `Graph` object. So that these functions have the same signature as the structural statistics, the higher-order function `functools.partial()` is used to create a function with the `(G, A, i)` signature. For example, `partial(changeoOc, "age")` returns a function with the required `(G, A, i)` signature, implementing the change statistic for outcome attribute related to nodal continuous attribute `"age"`, given the original generic change statistic function `changeoOc(attrname, G, A, i)`. This usage is illustrated in the empirical example in the "Results and discussion" section. The same technique is used for statistics that use an auxiliary ("setting") network (that is, statistics that depend on edges in a second network, as used for example in Diviák et al. [79], where the network is criminal collaboration, and the setting network is pre-existing ties such as kinship or friendship), a distance matrix, the decay value for geometrically weighted statistics [26], and for the node type (mode) in two-mode graphs using the `BipartiteGraph` object.

For some change statistics, computation can be made far more efficient and scalable by using a sparse matrix counting two-paths between each pair of nodes in the network. A similar technique, implementing the sparse matrix as a hash table, has been used for ERGM change statistics [44]. This technique is far simpler and more advantageous for ALAAM estimation and simulation in ALAAMEE, for two reasons. First, in ALAAM the network is exogenous, and hence, once constructed, the two-path lookup sparse matrix need not be modified. Second, the sparse matrix data structure is very easily and efficiently implemented in Python as a

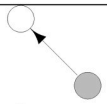
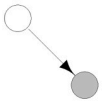
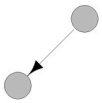
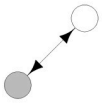
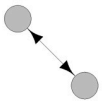
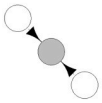
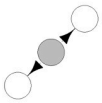
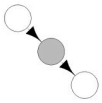
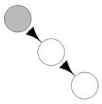
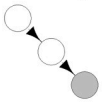
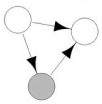
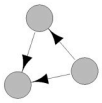
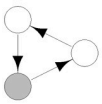
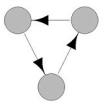
Name	Illustration	Description
Sender		Tendency of actors with the attribute to have outgoing ties (activity).
Receiver		Tendency of actors with the attribute to have incoming ties (popularity).
Contagion		Tendency of the attribute to be present in both actors connected by directed tie.
Reciprocity		Tendency of the attribute to be present in an actor connected to another by a reciprocated (mutual) tie.
Contagion reciprocity		Also known as mutual contagion. Tendency of the attribute to be present in both actors connected by a reciprocated tie.
Ego in-two-star		Tendency of the attribute to be present in an actor with additional incoming ties over Receiver.
Ego out-two-star		Tendency of the attribute to be present in an actor with additional outgoing ties over Sender.
Mixed-two-star		Tendency of the attribute to be present in an actor in the broker position between two other nodes (local brokerage).
Mixed-two-star source		Tendency of the attribute to be present in an actor in the source position in local brokerage.
Mixed-two-star sink		Tendency of the attribute to be present in an actor in the sink position in local brokerage.
Transitive triangle T1		Tendency of the attribute to be present in an actor in a transitive triangle, the broker position in Mixed-two-star bypassed by a transitive tie.
Transitive triangle T3		Contagion clustering: tendency of the attribute to be present in all three actors in a transitive triangle.
Cyclic triangle C1		Tendency of the attribute to be present in an actor in a cyclic triangle.
Cyclic triangle C3		Contagion cycle: tendency of the attribute to be present in all three actors in a cyclic triangle.

Fig 3. Configurations used in ALAAMs for directed networks.

https://doi.org/10.1371/journal.pcsy.0000021.g003

"dictionary of dictionaries" data structure, just as the graph data structures are, since dictionaries are a built-in data type in Python.

Currently, change statistics for undirected and directed one-mode networks, and undirected two-mode (bipartite) networks are implemented. These statistics include all of those used in the ALAAM models published in Kashima et al. [24], Letina [80], Letina et al. [81], and Diviák et al. [79] for undirected networks, and Gallagher [82] and Parker et al. [25] for directed networks, for example. ALAAMEE has been used to estimate ALAAM parameters for a director interlock (bipartite) network [83]. Change statistics for the new geometrically weighted ALAAM statistics described in Stivala [26] are also implemented in ALAAMEE.

We hope that the use of Python will facilitate the implementation of further user-defined change statistics, since adding ALAAM change statistics to the Python code, for example as shown in Fig 2, would appear to be considerably easier than the procedure for adding new ERGM change statistics in the statnet ergm package, which requires writing both R and C code [84].

Data sources

Simulation study. To evaluate the performance of the EE algorithm implemented in ALAAMEE for ALAAM parameter estimation and statistical inference, we apply it to estimating parameters for ALAAM outcome attribute vectors with known parameters. These are obtained by generating the outcome attribute vectors by ALAAM simulation from a fixed network (and covariates) with a given set of ALAAM parameters. This is done over a set of 100 simulated ALAAMs, allowing us to measure the point estimate bias and RMSE (root mean square error), as well as the type I (false positive) and type II (false negative) error rates in inference.

This technique was used to evaluate ERGM estimation from snowball sampled network data in Stivala et al. [63] and for ERGM estimation by the EE algorithm in Stivala et al. [44]. It was used to evaluate ALAAM estimation from sampled network data in Stivala et al. [14].

Here we use the simulated ALAAM outcome attribute values on the "Project 90" network, a sexual contact network of high-risk heterosexuals in Colorado Springs [85–88]. These are exactly the simulations described in Stivala et al. [14]. The network is the giant component of the Project 90 network, consisting of 4430 nodes, with mean degree 8.31. Binary and continuous attributes are generated for the nodes: the binary attribute is assigned the nonzero value for 50% of the nodes, chosen at random, and the continuous attribute value v_i at each node i is $v_i \overset{iid}{\sim} N(0, 1)$. From this network and nodal attributes, ALAAM outcome attribute vectors are simulated with parameters (Density, Activity, Contagion, Binary, Continuous) = (−15.0, 0.55, 1.00, 1.20, 1.15). As described in Stivala et al. [14], these parameters were chosen so that approximately 15% of the nodes have the outcome attribute value equal to 1.

Small network. To demonstrate the implementation of the stochastic approximation algorithm for estimating ALAAM parameters, we will use the excerpt of 50 girls from the "Teenage friends and lifestyle study" data [89–94], which is used as an illustrative example for the SIENA software [95] for stochastic actor-oriented models (SAOMs) [96]. The data consists of an excerpt of 50 girls from panel data recording friendship networks and substance use over a three year period from 1995, when the pupils were aged 13, to 1997, in the West of Scotland. The data includes information on smoking (tobacco), alcohol, and cannabis consumption, as well as sporting activities.

This data was also used as a tutorial example for the Bayesian ALAAM [16] implementation in R [97]. As noted in the description for this data [98], this is not a properly delineated network, and is used only for illustrative purposes.

Table 1. Network descriptive statistics for the Deezer networks.

Network	Nodes	Mean degree	Max. degree	Density	Clustering coefficient	Likes jazz %	Likes alternative %
Deezer Croatia	54573	18.26	420	0.00033	0.11463	5	38
Deezer Hungary	47538	9.38	112	0.00020	0.09292	5	37
Deezer Romania	41773	6.02	112	0.00014	0.07527	6	36

Each network is a single connected component. Network statistics were computed using the igraph R package. "Clustering coefficient" is the global clustering coefficient (transitivity).

https://doi.org/10.1371/journal.pcsy.0000021.t001

Unlike SAOMs, we cannot use ALAAMs to model the co-evolution of network and actor covariates based on longitudinal data: we can only model a single binary outcome attribute given a fixed network and other (fixed) covariates, as well as the outcome attribute itself on other nodes. As in the Bayesian ALAAM R tutorial, we will use smoking at the second wave as the outcome variable, and use the network and other covariates from the first wave. Hence the network assumed to diffuse social contagion is observed before the outcome variable, as discussed in Parker et al. [25].

The categorical variable for smoking is converted to binary by treating any amount of smoking other than completely non-smoking as the nonzero binary outcome.

Large networks. For networks with tens of thousands of nodes or more, estimating ALAAM parameters with the stochastic approximation algorithm may no longer be practical. For such networks, we can instead use the EE algorithm. We will use three such networks as examples for ALAAMEE parameter estimation: undirected online friendship networks for the "Deezer" music streaming service in Croatia, Hungary, and Romania [99]. These networks are publicly available from the Stanford large network dataset collection [100]. Descriptive statistics of these networks are shown in Table 1.

Each node in these three networks represents a user, and an undirected edge represents friendship in the Deezer online social network. Each node is annotated with a list of genres liked by the user [99]. Based on these genre annotations, we created two different binary outcome variables: one for liking jazz, and one for liking "alternative" music. The binary outcome variable for liking jazz is true if the user likes any one or more of the genres in the data that describe jazz music, namely "Jazz", "Instrumental jazz", "Jazz Hip Hop", or "Vocal Jazz". The binary outcome variable for liking alternative music is true if the user likes the genre in the data labelled "Alternative". Consistently across all three networks, approximately 5% of users like jazz, and between 35% and 40% of users like alternative music (Table 1).

Results and discussion

Simulation study for estimation using the equilibrium expectation algorithm

Fig 4 and Table 2 show the results for using the EE algorithm implemented in ALAAMEE to estimate parameters from the simulated ALAAMs. For all parameters other than Binary (the covariate effect for the simulated binary attribute described under "Simulation study" in the "Data sources" section), the type II error rate was less than 5%. For the Binary parameter, however, the type II error rate was estimated to be 9%, with a 95% confidence interval [5%, 16%]. When using the stochastic approximation algorithm on the same data, used as the baseline for comparing against results from sampled data in Stivala et al. [14], the type II error rate was less than 5% for all parameters, including Binary (S1 Fig). Comparing the results for the EE algorithm (Fig 4) and the stochastic approximation algorithm (S1 Fig) for the Binary parameter, it

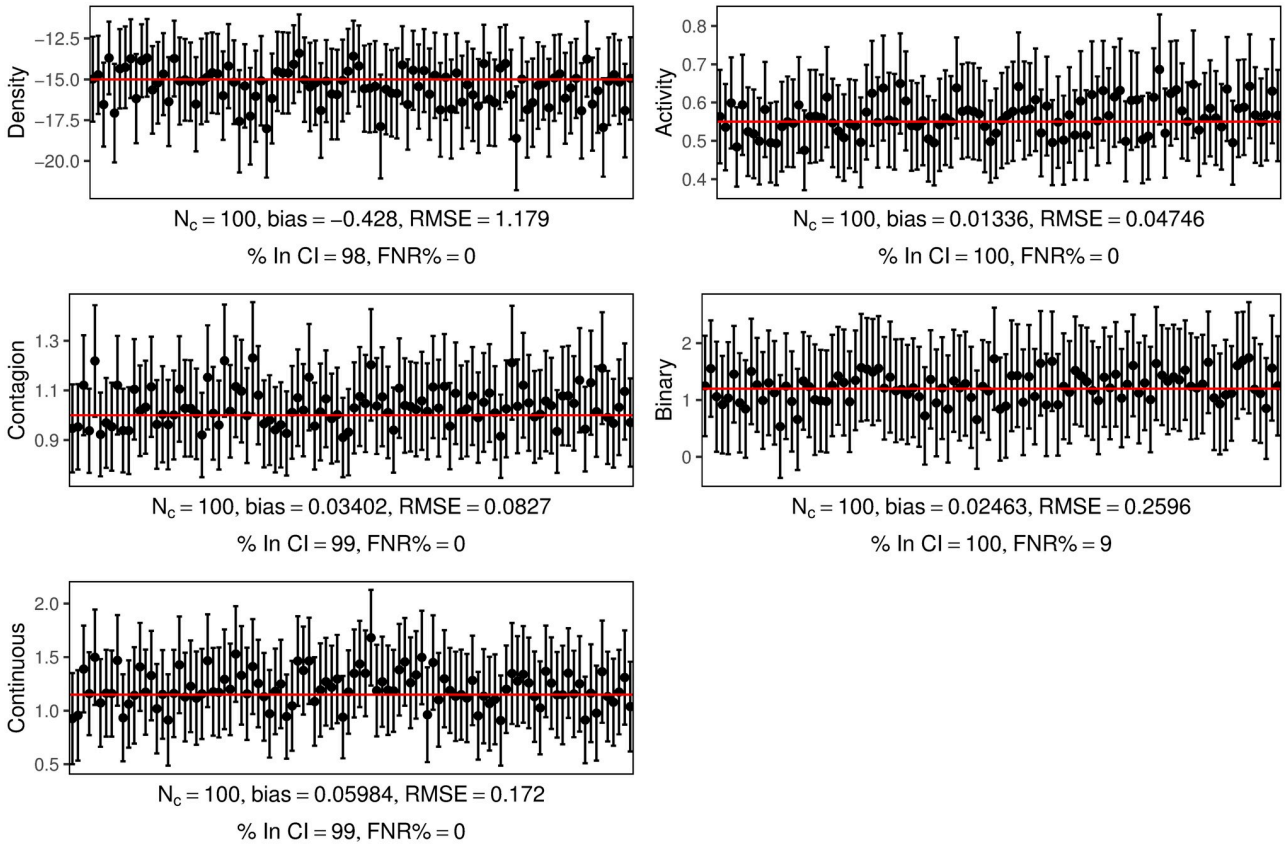


Fig 4. Parameter estimates and estimated standard errors from the EE algorithm. The algorithm was used to estimate the known ALAAM parameters for the Project 90 network with simulated attributes. The error bars show the nominal 95% confidence interval. The horizontal line shows the true value of the parameter, and each plot is annotated with the mean bias, root mean square error (RMSE), the percentage of samples for which the true value is inside the confidence interval (coverage rate), and the Type II error rate (False Negative Rate, FNR). N_C is the number of samples (of the total 100) for which a converged estimate was found.

https://doi.org/10.1371/journal.pcsy.0000021.g004

seems that the problem is likely not the point estimate, but rather that the estimated standard errors (and hence confidence interval) from the EE algorithm were too large, giving a high type II (false negative) error rate on the Binary parameter. Although this problem only occurs on the Binary parameter, the coverage rate (the percentage of samples for which the true value

Table 2. Type II error rate from estimation of simulated outcomes using the EE algorithm.

Effect	Bias	RMSE	False negative rate (%)			in C.I. (%)	Total samples converged	Mean runs converged	Total runs per sample
			Estim.	95% C.I.					
				lower	upper				
Density	-0.4280	1.1790	0	0	4	98	100	200.00	200
Activity	0.0134	0.0475	0	0	4	100	100	200.00	200
Contagion	0.0340	0.0827	0	0	4	99	100	200.00	200
Binary	0.0246	0.2596	9	5	16	100	100	200.00	200
Continuous	0.0598	0.1720	0	0	4	99	100	200.00	200

The "estim.", "lower", and "upper" columns show the point estimate and lower and upper 95% confidence interval (C.I.), respectively, of the Type II error rate (false negative rate). This C.I. is computed as the Wilson score interval. The "in C.I. (%)" column is the coverage rate for the nominal 95% confidence interval. Results are over 100 simulated ALAAM outcome attribute vectors (samples), each of which is estimated with 200 parallel estimation runs.

https://doi.org/10.1371/journal.pcsy.0000021.t002

Table 3. Type I error rate from estimation of simulated outcomes using the EE algorithm.

Effect	Bias	RMSE	False positive rate (%)			in C.I. (%)	Total samples converged	Mean runs converged	Total runs per sample
			estim.	95% C.I.					
				lower	upper				
Activity	0.0048	0.0149	0	0	4	100	100	200.00	200
Contagion	0.0011	0.0386	0	0	4	100	100	200.00	200
Binary	0.0191	0.2922	1	0	5	99	100	200.00	200
Continuous	0.0475	0.1638	0	0	4	100	100	200.00	200

The "estim.", "lower", and "upper" columns show the point estimate and lower and upper 95% confidence interval (C.I.), respectively, of the Type I error rate (false positive rate). This C.I. is computed as the Wilson score interval. The "in C.I. (%)" column is the coverage rate for the nominal 95% confidence interval. Results are over 100 simulated ALAAM outcome attribute vectors (samples), each of which is estimated with 200 parallel estimation runs.

https://doi.org/10.1371/journal.pcsy.0000021.t003

is inside the confidence interval) was higher than the nominal 95% on all the parameters: in fact was between 98% and 100% for the EE algorithm, while for the stochastic approximation algorithm, it was still higher than 95%, but less than 99% (S1 Fig).

In order to measure the type I error rate on each parameter, an ALAAM simulation without the corresponding effect (the parameter value is zero) was required. Hence for each of the ALAAM parameters considered (other than Density, which if zero results in almost all nodes having the outcome attribute equal to 1), another set of 100 ALAAM outcome attribute vectors were simulated, with the parameter set to zero and the other parameters unchanged (apart from Activity, which if zero results in very few nodes having the outcome attribute equal to 1, and so Density was changed to −7.0 for this case only).

The results for estimating the type I error rate for ALAAM estimation by the EE algorithm are shown in Table 3. In all cases the type I error rate was less than the nominal 5%. Just as for the type II error rate results, however, the coverage rate was higher than the nominal 95%, indicating that the standard error estimates might be too large.

In the results discussed so far, the point estimates and standard errors were estimated, as described in the "Materials and methods" section, from 200 parallel estimation runs. Fig 5 shows the effect on the type II error rate when the number of runs was varied from 1 up to 500 (the data point for 200 runs therefore corresponding to the results shown in Table 2 and Fig 4). This shows that for all parameters other than Binary, 50 runs (indeed, for the Density, Activity, and Contagion parameters, fewer than 20 runs) were more than sufficient to obtain a type II error rate of less than 5%. For the Binary parameter, however, the type II error rate declined far more slowly, and was 9% with 200 runs, as we have seen.

Each of these estimation runs, with the default EE algorithm parameters, took on average 49 minutes (sd = 12.7, min. = 44, max. = 126, median = 46, N = 8000) on an Intel Xeon E5–2650 v3 2.30 GHz processor on a Linux cluster, and on average 9.46 minutes (sd = 0.49, min. = 3.59, max. =12.85, median = 9.56, N = 40000) on AMD EPYC 7543 2.8 GHz processors on a newer and larger Linux cluster. Less than 2 GB of memory was required (for all parallel runs occupying a single node) for the Python code, and less than 8 GB in total for the R scripts to do the final statistical analyses and diagnostic plots.

S2 Fig shows the corresponding results for the type I error rate. It shows that it is "safe" to vary the number of runs, in the sense that the type I error rate remained less than 5% at all values tested.

The results of these simulation experiments indicate that it is desirable for the purposes of decreasing the type II error rate, without increasing the type I error rate to an unacceptable level, to use as many runs of the ALAAMEE estimation process as practical (at least up to a

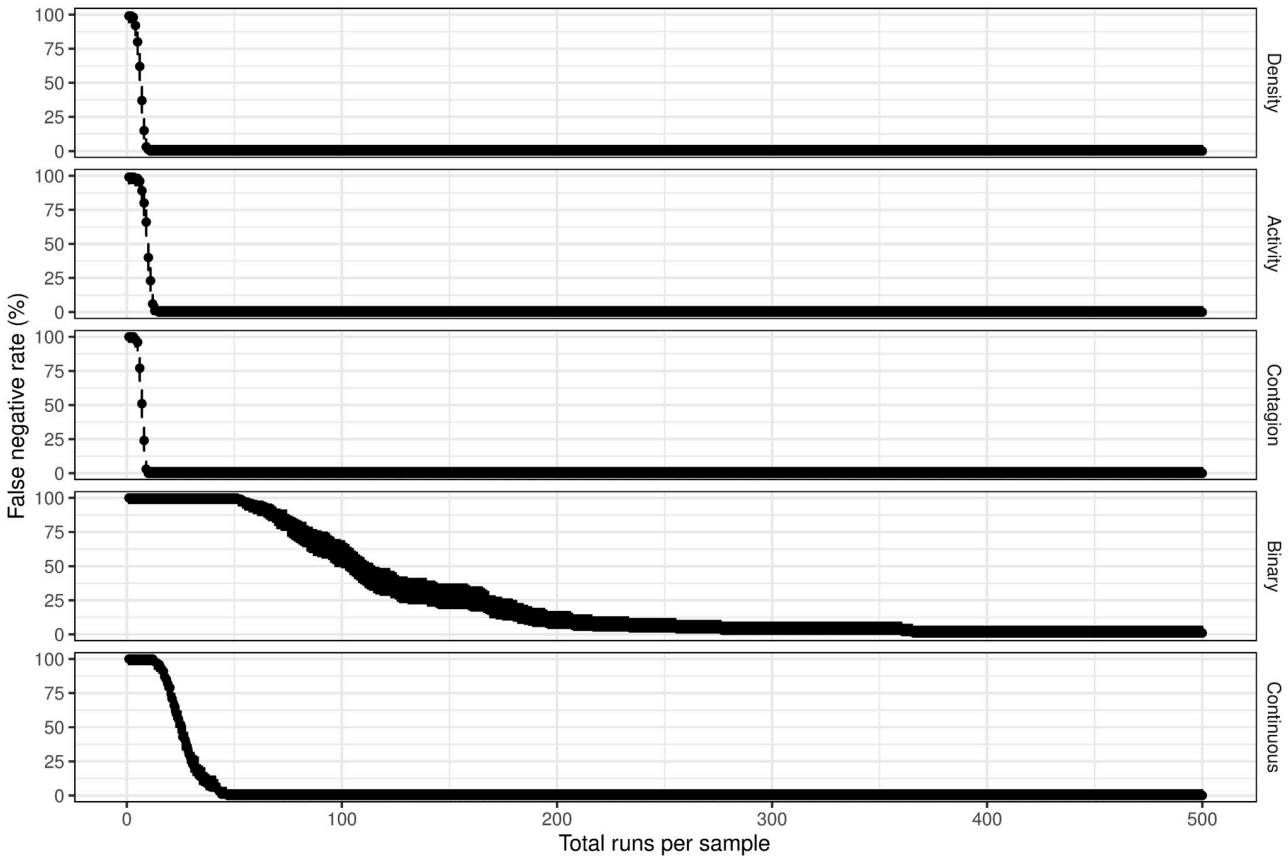


Fig 5. Type II error rate as the number of runs used for each sample is varied.

https://doi.org/10.1371/journal.pcsy.0000021.g005

maximum of 500). Because these runs are independent, they can be run in parallel to minimize elapsed time, taking advantage of as many processor cores as may be available. The results here indicate that 200 runs is a reasonable number, however for some parameters this might still result in an undesirably high type II error rate.

Empirical examples

Small network. Python code to use ALAAMEE to specify an ALAAM model for the excerpt of 50 girls from the "Teenage friends and lifestyle study" and estimate its parameters using the stochastic approximation algorithm [39] is shown in Fig 6 (this code is located in the ALAAMEE repository as https://github.com/stivalaa/ALAAMEE/blob/master/examples/simple/directed/glasgow_s50/runALAAMSAs50.py). It also automatically does a goodness-of-fit test for the converged model (if found). This code took less than two minutes (and less than 300 MB of memory) to run on a Windows 10 personal computer with an Intel Core i5–10400 2.90 GHz processor, giving the model shown in Fig 7. The results are consistent (only the contagion and alcohol effects are significant, and positive) with those from estimating the same model using MPNet [28]. For this small network, MPNet was faster, taking approximately one minute (and less than 100 MB of memory), on the same PC.

The model shown in Fig 7 indicates a significant and positive contagion effect: smokers tend to be directly connected to other smokers. The only other significant effect is for alcohol consumption; this is positive, indicating that drinkers are more likely to smoke.

```
from functools import partial
import  estimateALAAMSA
from changeStatisticsALAAMdirected import *
from changeStatisticsALAAM import changeDensity, changeoOc

model_param_funcs = [changeDensity, changeSender,
                     changeReceiver, changeContagion,
                     changeReciprocity,
                     changeContagionReciprocity,
                     changeEgoInTwoStar,
                     changeEgoOutTwoStar,
                     changeMixedTwoStar,
                     changeTransitiveTriangleT1,
                     partial(changeoOc, "sport"),
                     partial(changeoOc, "alcohol")]

estimateALAAMSA.run_on_network_attr(
        's50-friendships-directed.net',
        model_param_funcs,
        [param_func_to_label(f)
          for f in model_param_funcs],
        outcome_bin_filename = 's50-outcome.txt',
        binattr_filename = 's50-binattr.txt',
        contattr_filename = 's50-contattr.txt',
        catattr_filename = 's50-catattr.txt',
        directed = True
    )
```

Fig 6. Python code for estimating parameters of an ALAAM model. This code uses ALAAMEE with the stochastic approximation algorithm to estimate ALAAM parameters and their standard errors for the teenage friends and lifestyle data excerpt, with smoking as the outcome variable. After a converged model is found, this will also do a goodness-of-fit test.

https://doi.org/10.1371/journal.pcsy.0000021.g006

```
                    Estimate  Std.Error t-ratio
             Density  -1.150      2.375    0.021
              Sender   0.340      1.213   -0.022
            Receiver  -0.933      1.403    0.049
           Contagion   1.861      0.910    0.010 *
         Reciprocity   1.171      1.465    0.029
ContagionReciprocity  -2.913      2.054    0.031
        EgoInTwoStar   0.328      0.484    0.049
       EgoOutTwoStar   0.214      0.593   -0.053
        MixedTwoStar  -0.278      0.446   -0.010
TransitiveTriangleT1  -0.958      0.584    0.065
           sport_oOc  -1.541      1.026    0.001
          alcohol_oOc   0.821      0.398   -0.023 *
```

Fig 7. Example ALAAMEE output. This is the output from using ALAAMEE to estimate ALAAM parameters by stochastic approximation for the teenage friends and lifestyle data excerpt, with smoking as the outcome variable. Asterisks indicate statistical significance at $p < 0.05$.

https://doi.org/10.1371/journal.pcsy.0000021.g007

	t-ratio
Density	0.075
Sender	0.024
Receiver	0.084
Contagion	0.051
Reciprocity	0.075
ContagionReciprocity	0.083
EgoInTwoStar	0.079
EgoOutTwoStar	-0.012
MixedTwoStar	0.030
TransitiveTriangleT1	0.084
sport_oOc	0.066
alcohol_oOc	0.018
MixedTwoStarSource	-0.036
MixedTwoStarSink	-0.372
TransitiveTriangleT3	-0.045
TransitiveTriangleD1	-0.056
TransitiveTriangleU1	-0.050
CyclicTriangleC1	0.053
CyclicTriangleC3	-0.490
AlterInTwoStar2	0.002
AlterOutTwoStar2	-0.539

Fig 8. Example ALAAMEE goodness-of-fit output. This is the goodness-of-fit output from ALAAMEE for the model it estimated by stochastic approximation for the teenage friends and lifestyle data excerpt, with smoking as the outcome variable.

https://doi.org/10.1371/journal.pcsy.0000021.g008

The goodness-of-fit statistics for this model are shown in Fig 8. The effects in the model are at the top, followed by additional effects which are not included in the model. For effects in the model, the convergence statistic (t-ratio) should be less than 0.1 in magnitude (just as they must be for the model to be considered converged by the estimation algorithm). Fig 8 shows that all the effects included in the model met this criterion.

For effects not included in the model, a rule of thumb is that the t-ratio should be less than 2.0 in magnitude for an acceptable fit [25], although some authors have used a stricter threshold of 1.0 [24, 78], or even 0.3 [79]. Parker et al. [25] use 1.645, signifying statistical significance at the 5% level (one-tailed). All of the effects not included in the model, but included in the goodness-of-fit test shown in Fig 8, were less than 1.0 in magnitude, indicating this model had an acceptable fit for all the statistics included in the goodness-of-fit test.

Table 4. Models estimated using ALAAMEE for the Deezer networks, with liking jazz as the outcome variable.

Effect	Croatia	Hungary	Romania
Density	**−4.727** (**−4.866**,**−4.589**)	**−5.055** (**−5.209**,**−4.901**)	**−4.691** (**−4.870**,**−4.513**)
GWActivity [$\alpha = 2.0$]	−2.945 (−7.280,1.390)	**5.234** (**3.269**,**7.200**)	1.606 (−0.835,4.047)
Contagion	−0.008 (−0.130,0.115)	**0.265** (**0.153**,**0.377**)	**0.613** (**0.403**,**0.823**)
TriangleT1	—	—	−0.034 (−0.076,0.008)
TriangleT2	—	—	0.004 (−0.177,0.185)
TriangleT3	—	—	−0.020 (−0.749,0.710)
num. genres	**0.194** (**0.186**,**0.201**)	**0.193** (**0.184**,**0.201**)	**0.194** (**0.184**,**0.203**)
partner num. genres	−0.000 (−0.001,0.001)	—	**−0.006** (**−0.011**,**−0.001**)

Parameter estimates are shown with 95% confidence interval. Estimates that are statistically significant at the nominal $p < 0.05$ level are shown in bold. Results are from 200 converged runs (of a total 200).

https://doi.org/10.1371/journal.pcsy.0000021.t004

Large networks. Table 4 shows the results of using the EE algorithm implemented in ALAAMEE to estimate parameters for ALAAM models of the three Deezer online social networks, with liking jazz as the outcome variable. The model parameters were selected to test for social contagion of liking jazz on this social network (the Contagion parameter), including specifically in closed triads (the three Triangle parameters), while also accounting for the degree distribution of users who like jazz (the GWActivity parameter [26]), and the number of genres liked by a user and their friends. Degeneracy check plots for these models are shown in S3, S4 and S5 Figs.

This model was estimated first on the Romania network (the smallest of the three). Initially the Activity parameter, rather than GWActivity, was used, however the degeneracy check showed that this model was not properly converged, and hence the geometrically weighted activity parameter GWActivity (with decay parameter $\alpha = 2.0$) was used instead of Activity to overcome this problem, as described in Stivala [26]. This model, however, did not converge for the Croatia and Hungary data, but removing the triangle parameters solved this problem, resulting in the models shown in Table 4 (for Hungary, the partner number of genres parameter also had to be removed to find a converged model).

Density (or incidence) is negative and statistically significant in all models. This effect, analogous to the intercept in logistic regression, is simply the baseline presence of the outcome attribute (liking jazz). Its negative value merely reflects the fact that jazz is a minority interest in the data for all three countries.

The only parameter, apart from Density, that is significant across all three networks is the number of genres liked by a user: the more genres a user likes, the more likely they are to like jazz. This could simply be a consequence of the fact that, even if we assume liking jazz is completely random, the more genres someone likes, the more likely it is that jazz is one of them (simply by chance), and this parameter controls for this effect. It also has a more substantive interpretation: we might suppose that a preference for jazz is a highly specific taste that

excludes also liking other genres, and so people who like jazz are unlikely to also like other genres. The negative and statistically significant estimate of this parameter refutes that hypothesis.

The Contagion parameter is positive and significant for both Hungary and Romania, indicating that (given the assumptions of the model) a liking for jazz is socially contagious in these countries' networks. However this parameter is not significant for Croatia.

The negative and significant partner number of genres parameter (for Romania only) indicates that users who have friends who like many genres are less likely to like jazz (while the positive and significant number of genres parameter means that a user who likes many genres is more likely to like jazz).

Estimating these models (200 parallel estimations) took approximately 21, 12, and 36 minutes, for Croatia, Hungary, and Romania, respectively, on AMD EPYC 7543 2.8 GHz processors on a Linux cluster. Less than 300 MB of memory per CPU was required for the Python code running these estimations, and less than 8 GB in total was required for the R scripts to do the final statistical computations and diagnostic plots.

We also estimated the same models with ALAAMEE using the stochastic approximation algorithm, and the results are shown in S1 Table. These estimations took approximately 32, 14, and 36 hours (and less than 500 MB memory) for Croatia, Hungary, and Romania, respectively, on the same system. The results are consistent with those of the estimation with the EE algorithm. However three parameter estimates were found to be statistically significant by the stochastic approximation algorithm that were not when using the EE algorithm: GWActivity for Croatia, and GWActivity and TriangleT1 for Romania.

For the Romania network, the negative and significant TriangleT1 parameter indicates that users who like jazz are less likely to be involved in a triangle structure than other users. This is evidence that liking jazz is less likely to be observed in more clustered regions of this network.

The TriangleT2 and TriangleT3 parameters need to be interpreted in conjunction with Contagion and TriangleT1. The positive and significant Contagion parameter indicates that liking jazz is socially contagious, while the TriangleT1 parameter controls for the incidence of liking jazz in closed triads. Given these effects in the model, a positive and significant TriangleT2 parameter would indicate a type of structural equivalence within a closed triad: a pair of users who are friends and are both also friends with a common third user is associated with that pair of users both liking jazz. A positive and significant TriangleT3 parameter would indicate that jazz is socially contagious within triads (over and above its contagion in dyads), as reflected in an over-representation of closed triads where all three users like jazz (given all the other effects in the model). However the TriangleT2 and TriangleT3 parameters are either not statistically significant, or models containing them do not converge, in all three countries, and so we cannot draw any conclusions about these effects in this data.

Note that, as discussed in the Introduction, by using the ALAAM model, in which the networks ties are fixed (exogenous to the model), we are assuming that only the process of social contagion is occurring, without the ability to account for the possibility that friends (network neighbours) might both like the same genre of music due to homophily.

Table 5 shows the results of using the EE algorithm to estimate ALAAM parameters for the same three networks, but this time with liking alternative music as the outcome variable. For this data, the full model was able to be estimated for all three countries. Degeneracy check plots for these models are shown in S6, S7 and S8 Figs.

The results are qualitatively quite similar to the model for jazz, in that the only parameter estimate (other than Density) that is statistically significant across all three networks is number of genres, which is positive, and, again, the Contagion parameter is statistically significant and positive for the Hungary and Romania (but not Croatia) networks. None of the Triangle parameters are statistically significant in any of the three countries.

Table 5. ALAAM models for the Deezer networks, with liking "alternative" music as the outcome variable.

Effect	Croatia	Hungary	Romania
Density	**−3.277** (**−3.364**,**−3.191**)	**−3.020** (**−3.111**,**−2.929**)	**−3.156** (**−3.262**,**−3.050**)
GWActivity [$\alpha = 2.0$]	−0.346 (−2.755,2.064)	0.088 (−1.777,1.954)	−0.381 (−2.039,1.276)
Contagion	−0.010 (−0.053,0.032)	**0.158** (**0.095**,**0.221**)	**0.109** (**0.028**,**0.190**)
TriangleT1	−0.001 (−0.007,0.005)	−0.006 (−0.022,0.009)	−0.018 (−0.054,0.017)
TriangleT2	0.003 (−0.012,0.017)	0.000 (−0.032,0.032)	−0.005 (−0.072,0.062)
TriangleT3	−0.008 (−0.046,0.030)	0.028 (−0.057,0.112)	0.044 (−0.107,0.195)
num. genres	**0.448** (**0.438**,**0.458**)	**0.419** (**0.408**,**0.430**)	**0.405** (**0.393**,**0.417**)
partner num. genres	0.001 (−0.002,0.003)	**−0.010** (**−0.014**,**−0.006**)	−0.002 (−0.007,0.003)

Parameter estimates are shown with 95% confidence interval. Estimates that are statistically significant at the nominal $p < 0.05$ level are shown in bold. Results are from 200 converged runs (of a total 200).

https://doi.org/10.1371/journal.pcsy.0000021.t005

Estimating these models (200 parallel estimations) took approximately 244, 74, and 47 minutes, for Croatia, Hungary, and Romania, respectively (on the same system as the jazz model, and with the same upper bounds on memory usage). We also attempted to estimate the same models using the stochastic approximation algorithm, however for the Croatia and Hungary networks these estimations did not complete within a 48 hour elapsed time limit. The estimation for the Romania network took approximately 30 hours (and less than 200 MB of memory), and the resulting parameter estimates are shown in S2 Table. These estimates are consistent with those estimated by the EE algorithm: the same three parameters are found to be statistically significant with the same sign, and all the other parameters are not statistically significant.

Conclusion

ALAAMEE is open-source Python software for the estimation, simulation, and goodness-of-fit testing of the ALAAM social contagion model. It currently supports ALAAMs on undirected and directed one-mode networks, and undirected two-mode (bipartite) networks. It also supports estimation from snowball sampled network data.

Models can be estimated using either stochastic approximation with the Robbins–Monro algorithm, or, for large networks, by the EE algorithm. For networks small enough that it is practical—on the order of thousands of nodes, but also depending on the model parameters and data—we recommend using stochastic approximation, a well-known and widely used method, which is implemented in ALAAMEE just as it is in MPNet for ALAAM parameter estimation. However for larger networks, stochastic approximation is likely to be too slow to be practical, and so the EE algorithm can be used instead. In this work we demonstrated its use on networks on networks of approximately 50 000 nodes, for which stochastic approximation was feasible in some, but not all, cases. The use of the EE algorithm implemented in ALAAMEE to estimate ALAAM parameters for a much larger network, with approximately 1.6 million nodes, for which estimation by stochastic approximation is certainly not feasible, is

Table 6. Comparison of MPNet and ALAAMEE software for ALAAM estimation.

Feature		MPNet	ALAAMEE
Download location		http://www.melnet.org.au/pnet	https://github.com/stivalaa/ALAAMEE
Estimation algorithms		Stochastic approximation (SA)	SA and equilibrium expectation (EE)
Max. network size (nodes)		Thousands	Tens of thousands (SA, EE); >1.5 million (EE only)
Platform		Microsoft Windows	Cross-platform (anywhere Python and R are available)
Implementation language		C#	Python and R
User interface		Windows GUI	Python functions
Open source		No	Yes
User-extensible		No	Yes
Can use parallel computing		No	Yes (EE algorithm only)
First release year		2014	2020
Most recent release year		2022	2024
Snowball sampled networks		No	Yes
Network types supported	Undirected	Yes	Yes
	Directed	Yes	Yes
	Two-mode	Yes	Yes (currently undirected only)
	Multilevel	Yes	No
Handles missing data	Nodal attributes	No	Simplistic (NA values ignored in statistic computation)
	Outcome variable	No	No (NA values mark outcome inapplicable in mode)
	Network ties	No	No

Note: MPNet supports both ERGM and ALAAM, while ALAAMEE is only for ALAAM. MPNet features listed are those applicable to ALAAM.

https://doi.org/10.1371/journal.pcsy.0000021.t006

described in Stivala [26]. This demonstrates that ALAAMEE allows estimation and goodness-of-fit testing of ALAAM models for networks that are too large for prior software implementations to estimate.

When using the EE algorithm, the results from the simulation experiments section indicate that it is always advantageous, and does not result in a significantly increased false positive rate (at least up to the maximum of 500 tested in this work), to use as many runs as practical, and 200 is probably more than sufficient in most cases. Because these runs are entirely independent, elapsed time can be minimized by running them in parallel using as many compute cores and nodes as may be available.

The EE algorithm is fast, scalable to far larger networks, and able to take advantage of multi-core and large-scale parallel computing. The simplified EE algorithm [42] used in ALAAMEE has recently been shown, in the context of ERGMs, to be guaranteed to converge to the MLE, if it exists, when the learning rate (a parameter of the EE algorithm) is sufficiently small [65]. The proof uses the uncertain energies framework of Ceperley & Dewing [101], as originally suggested in Borisenko et al. [42].

However, as shown in this work, using this algorithm with an insufficient number of estimation runs can result in very low statistical power on some parameters, due to larger estimated standard errors than the stochastic approximation method on the same data. This has also been observed in using the EE algorithm to estimate parameters for the "citation ERGM" (cERGM) ERGM variant [102], compared with using the default statnet MCMLE algorithm [45].

Because ALAAMEE is written in Python using only the NumPy package, it can easily be run on any system on which Python and NumPy are available. This also facilitates its use in

automated scripts, for example for large-scale computational experiments such as those described in this paper and in Stivala et al. [14], which is not practical with a Windows application like MPNet. A comparison of MPNet and ALAAMEE is shown in Table 6.

The relative ease of Python programming, along with the simple and internally documented implementations, with unit tests, of a variety of ALAAM change statistics already included in the open-source ALAAMEE software, should facilitate the creation of change statistics for any new configurations required by users of the software.

A demonstration implementation of the EE algorithm for ERGM parameter estimation was written in Python in the same style as ALAAMEE, that is, using NumPy for linear algebra and Python dictionaries for graph data structures. However this was found to be too slow for practical use, and the software [103] was completely re-written in C [44]. For ALAAM, however, where the MCMC process involves flipping binary variables in a vector rather than edges in a graph, we found that the Python implementation ALAAMEE was sufficiently fast for practical use not only with the EE algorithm, but also with the Robbins–Monro algorithm.

In theory, ALAAMEE could be made considerably faster with little effort by using a Python "just-in-time" (JIT) compiler, such as Numba [104] or PyPy [105]. However to date we have been unable to accelerate ALAAMEE this way, finding that either the code is not supported, or the "accelerated" version is actually slower than the original.

In Parker et al. [25], three limitations of ALAAMs are identified. First, the outcome variable is restricted to binary; second, the inherent assumption that the underlying social contagion process is at equilibrium; and third, their inadequacy for very large networks. We suggest that this work, in addition to the use of ALAAM estimation from network snowball samples [14] effectively addresses the third limitation. The second limitation can be mitigated as suggested in Parker et al. [25], by applying ALAAM models to data in which the social network is observed at an appropriate time before the outcome behavioural variable is. If longitudinal (panel) data is available, in which the social network ties and actor attributes of a population are observed at multiple time points, the stochastic actor attribute model (SAOM) is a more appropriate choice of model, enabling the estimation of parameters corresponding to the co-evolution of the social network and actor attributes [96]. This leaves the first limitation, that the outcome variable must be binary, to be addressed in future work. We have identified an additional limitation of ALAAMs in this work and in Stivala [26], namely that, like ERGMs, ALAAMs can suffer from problems of "near-degeneracy" when only simple statistics (such as Activity) are used, particularly on larger networks. The use of the GWActivity parameter can help overcome these problems, but as illustrated by the examples in the "Results and discussion" section, it still may not always be possible to fit a model with all the desired parameters. Some avenues for further work on this problem are suggested in Stivala [26].

One further limitation of ALAAM estimation as implemented in ALAAMEE is the handling of missing data. Missing nodal attributes can be specified as "NA" values, in which case they are handled in a simplistic way by simply not being counted towards the relevant change statistics. For more than very small amounts of missing data, however, it would be better to use some form of imputation. Missing values of the outcome binary attribute may also be specified by "NA" values, however in this case they act as "structural zeroes", which fix the outcome variable at the NA value. This is appropriate in the case of two-mode networks where the outcome variable is only defined for one mode, and so it is set to NA for all nodes in the other mode. To handle missing values of the outcome binary variable, the Bayesian estimation method described by Koskinen & Daraganova [16] is more appropriate. Missing tie variables are not handled at all, although it is possible to use network snowball samples [14], which is also possible with the Bayesian ALAAM estimation method [16].

An additional avenue of further work on ALAAM modeling was suggested in the Introduction: the conception of an ALAAM as a bipartite ERGM could be a way of implementing a multivariate ALAAM, that is, an ALAAM with more than one outcome variable.

Supporting information

S1 Fig. Parameter estimates and estimated standard errors from the stochastic approximation algorithm. The stochastic approximation algorithm was used to estimate the known ALAAM parameters for the Project 90 network with simulated attributes. This is the baseline result for the Project 90 example in a study of the effect of network sampling on ALAAM estimation [14]. The error bars show the nominal 95% confidence interval. The horizontal line shows the true value of the parameter, and each plot is annotated with the mean bias, root mean square error (RMSE), the percentage of samples for which the true value is inside the confidence interval (coverage rate), and the Type II error rate (False Negative Rate, FNR). A converged estimate was found for 99 of the total 100 samples.
(PDF)

S2 Fig. Type I error rate as the number of runs used for each sample is varied.
(PDF)

S3 Fig. Degeneracy check for the Deezer Croatia network with liking jazz as the outcome variable. Trace plots and histograms show statistics of 100 networks simulated from the model. The blue dashed lines on histograms show mean and blue dotted lines on histograms and shaded areas on trace plots show 95% confidence interval, and red lines show the observed values.
(PDF)

S4 Fig. Degeneracy check for the Deezer Hungary network with liking jazz as the outcome variable. Trace plots and histograms show statistics of 100 networks simulated from the model. The blue dashed lines on histograms show mean and blue dotted lines on histograms and shaded areas on trace plots show 95% confidence interval, and red lines show the observed values.
(PDF)

S5 Fig. Degeneracy check for the Deezer Romania network with liking jazz as the outcome variable. Trace plots and histograms show statistics of 100 networks simulated from the model. The blue dashed lines on histograms show mean and blue dotted lines on histograms and shaded areas on trace plots show 95% confidence interval, and red lines show the observed values.
(PDF)

S6 Fig. Degeneracy check for the Deezer Croatia network with liking "alternative" music as the outcome variable. Trace plots and histograms show statistics of 100 networks simulated from the model. The blue dashed lines on histograms show mean and blue dotted lines on histograms and shaded areas on trace plots show 95% confidence interval, and red lines show the observed values.
(PDF)

S7 Fig. Degeneracy check for the Deezer Hungary network with liking "alternative" music as the outcome variable. Trace plots and histograms show statistics of 100 networks simulated from the model. The blue dashed lines on histograms show mean and blue dotted lines on histograms and shaded areas on trace plots show 95% confidence interval, and red lines show the

observed values.
(PDF)

S8 Fig. Degeneracy check for the Deezer Romania network with liking "alternative" music as the outcome variable. Trace plots and histograms show statistics of 100 networks simulated from the model. The blue dashed lines on histograms show mean and blue dotted lines on histograms and shaded areas on trace plots show 95% confidence interval, and red lines show the observed values.
(PDF)

S1 Table. Models estimated using with the stochastic approximation algorithm, with liking jazz as the outcome variable. Parameter estimates are shown with their estimated standard errors. Parameter estimates that are statistically significant at the nominal $p < 0.05$ level are shown in bold.
(PDF)

S2 Table. Model estimated using with the stochastic approximation algorithm, liking "alternative" music as the outcome variable. Asterisks indicate statistical significance at the nominal $p < 0.05$ level. Only the results for Romania are shown, as estimation for the other two networks (Croatia and Hungary) did not complete within the 48 hour elapsed time limit.
(PDF)

Acknowledgments

We used the high performance computing cluster at the Institute of Computing, Università della Svizzera italiana (https://www.ci.inf.usi.ch/), for initial computational experiments. Computational experiments for the final manuscript were performed on the OzSTAR national facility at Swinburne University of Technology (https://supercomputing.swin.edu.au/). The OzSTAR program receives funding in part from the Astronomy National Collaborative Research Infrastructure Strategy (NCRIS, https://www.education.gov.au/ncris) allocation provided by the Australian Government, and from the Victorian Higher Education State Investment Fund (VHESIF, https://www.vic.gov.au/projects-funded-victorian-higher-education-state-investment-fund) provided by the Victorian Government.

Author Contributions

Conceptualization: Alex Stivala, Peng Wang, Alessandro Lomi.

Data curation: Alex Stivala.

Formal analysis: Alex Stivala.

Funding acquisition: Alessandro Lomi.

Investigation: Alex Stivala.

Methodology: Alex Stivala.

Project administration: Alessandro Lomi.

Resources: Peng Wang, Alessandro Lomi.

Software: Alex Stivala, Peng Wang.

Validation: Alex Stivala.

Visualization: Alex Stivala.

Writing – original draft: Alex Stivala.

Writing – review & editing: Alex Stivala, Peng Wang, Alessandro Lomi.

References

1. Bramoullé Y, Djebbari H, Fortin B. Peer effects in networks: A survey. Annu Rev Econom. 2020; 12:603–629. https://doi.org/10.1146/annurev-economics-020320-033926
2. Ord K. Estimation methods for models of spatial interaction. J Am Stat Assoc. 1975; 70(349):120–126. https://doi.org/10.1080/01621459.1975.10480272
3. Cliff AD, Ord JK. Spatial processes: models & applications. Taylor & Francis; 1981.
4. Doreian P. Estimating linear models with spatially distributed data. Sociol Methodol. 1981; 12:359–388. https://doi.org/10.2307/270747
5. Anselin L. Some robust approaches to testing and estimation in spatial econometrics. Reg Sci Urban Econ. 1990; 20(2):141–163. https://doi.org/10.1016/0166-0462(90)90001-J
6. Friedkin NE. Social networks in structural equation models. Soc Psychol Q. 1990; 53(4):316–328. https://doi.org/10.2307/2786737
7. Leenders RTA. Modeling social influence through network autocorrelation: constructing the weight matrix. Soc Networks. 2002; 24(1):21–47. https://doi.org/10.1016/S0378-8733(01)00049-1
8. Centola D. Complex contagions. In: Manzo G, editor. Research Handbook on Analytical Sociology. Cheltenham, UK: Edward Elgar Publishing; 2021. p. 321–336.
9. Granovetter M. Threshold models of collective behavior. Am J Sociol. 1978; 83(6):1420–1443. https://doi.org/10.1086/226707
10. Centola D, Macy M. Complex contagions and the weakness of long ties. Am J Sociol. 2007; 113 (3):702–734. https://doi.org/10.1086/521848
11. Centola D. The spread of behavior in an online social network experiment. Science. 2010; 329 (5996):1194–1197. https://doi.org/10.1126/science.1185231 PMID: 20813952
12. Friedkin N, Johnsen EC. Social influence networks and opinion change. Adv Group Process. 1999; 16:1–29.
13. Centola D. The social origins of networks and diffusion. Am J Sociol. 2015; 120(5):1295–1338. https://doi.org/10.1086/681275 PMID: 26421341
14. Stivala AD, Gallagher HC, Rolls DA, Wang P, Robins GL. Using sampled network data with the autologistic actor attribute model; 2020. arXiv:2002.00849v2 [Preprint]. Available from: https://arxiv.org/abs/2002.00849v2 [cited 2024 April 24].
15. Daraganova G. Statistical models for social networks and network-mediated social influence processes: Theory and application [PhD thesis]. The University of Melbourne; 2009.
16. Koskinen J, Daraganova G. Bayesian analysis of social influence. J R Stat Soc Ser A Stat Soc. 2022; 185(4):1855–1881. https://doi.org/10.1111/rssa.12844
17. Robins G, Pattison P, Elliott P. Network models for social influence processes. Psychometrika. 2001; 66(2):161–189. https://doi.org/10.1007/BF02294834
18. Robins G, Elliott P, Pattison P. Network models for social selection processes. Soc Networks. 2001; 23(1):1–30. https://doi.org/10.1016/S0378-8733(01)00029-6
19. Daraganova G, Robins G. Autologistic actor attribute models. In: Lusher D, Koskinen J, Robins G, editors. Exponential Random Graph Models for Social Networks. New York: Cambridge University Press; 2013. p. 102–114.
20. Lusher D, Koskinen J, Robins G, editors. Exponential Random Graph Models for Social Networks. Structural Analysis in the Social Sciences. New York: Cambridge University Press; 2013.
21. Amati V, Lomi A, Mira A. Social network modeling. Annu Rev Stat Appl. 2018; 5:343–369. https://doi.org/10.1146/annurev-statistics-031017-100746
22. Koskinen J. Exponential Random Graph Modelling. In: Atkinson P, Delamont S, Cernat A, Sakshaug JW, Williams RA, editors. SAGE Research Methods Foundations. London: SAGE; 2020. Available from: https://doi.org/10.4135/9781526421036888175.
23. Koskinen J. Exponential Random Graph Models. In: McLevey J, Scott J, Carrington PJ, editors. The Sage Handbook of Social Network Analysis. 2nd ed. Sage; 2023. p. 474–500.
24. Kashima Y, Wilson S, Lusher D, Pearson LJ, Pearson C. The acquisition of perceived descriptive norms as social category learning in social networks. Soc Networks. 2013; 35(4):711–719. https://doi.org/10.1016/j.socnet.2013.06.002

25. Parker A, Pallotti F, Lomi A. New network models for the analysis of social contagion in organizations: an introduction to autologistic actor attribute models. Organ Res Methods. 2022; 25(3):513–540. https://doi.org/10.1177/10944281211005167

26. Stivala A. Overcoming near-degeneracy in the autologistic actor attribute model; 2023. arXiv:2309.07338v2 [Preprint]. Available from: https://arxiv.org/abs/2309.07338v2 [cited 2024 April 24].

27. Wang P, Robins G, Pattison P. PNet: A program for the simulation and estimation of exponential random graph models; 2009. Available from: http://www.melnet.org.au/s/PNetManual.pdf [cited 2024 April 25].

28. Wang P, Robins G, Pattison P, Koskinen J. MPNet: Program for the simulation and estimation of (p*) exponential random graph models for multilevel networks; 2014. Available from: http://www.melnet.org.au/s/MPNetManual.pdf [cited 2024 April 25].

29. Wang P, Stivala A, Robins G, Pattison P, Koskinen J, Lomi A. PNet: Program for the simulation and estimation of (p*) exponential random graph models for multilevel networks; 2022. Available from: http://www.melnet.org.au/s/MPNetManual2022.pdf [cited 2024 April 25].

30. Handcock MS, Hunter DR, Butts CT, Goodreau SM, Morris Martina. statnet: Software Tools for the Representation, Visualization, Analysis and Simulation of Network Data. J Stat Softw. 2008; 24(1):1–11. https://doi.org/10.18637/jss.v024.i01 PMID: 18618019

31. Hunter DR, Handcock MS, Butts CT, Goodreau SM, Morris M. ergm: A package to fit, simulate and diagnose exponential-family models for networks. J Stat Softw. 2008; 24(3):1–29. https://doi.org/10.18637/jss.v024.i03 PMID: 19756229

32. Hummel RM, Hunter DR, Handcock MS. Improving Simulation-Based Algorithms for Fitting ERGMs. J Comput Graph Stat. 2012; 21(4):920–939. https://doi.org/10.1080/10618600.2012.679224 PMID: 26120266

33. Handcock MS, Hunter DR, Butts CT, Goodreau SM, Krivitsky PN, Morris M. ergm: Fit, Simulate and Diagnose Exponential-Family Models for Networks; 2022. Available from: http://CRAN.R-project.org/package=ergm [cited 2024 April 29].

34. Krivitsky PN, Hunter DR, Morris M, Klumb C. ergm 4: New Features for Analyzing Exponential-Family Random Graph Models. J Stat Softw. 2023; 105(6):1–44. https://doi.org/10.18637/jss.v105.i06

35. Barnes ML, Wang P, Cinner JE, Graham NA, Guerrero AM, Jasny L, et al. Social determinants of adaptive and transformative responses to climate change. Nat Clim Chang. 2020; 10(9):823–828. https://doi.org/10.1038/s41558-020-0871-4

36. Fellows I, Handcock MS. Exponential-family Random Network Models; 2012. arXiv:1208.0121v1 [Preprint]. Available from: https://arxiv.org/abs/1208.0121v1 [cited 2024 April 24].

37. Fellows IE, Handcock MS. Analysis of partially observed networks via exponential-family random network models; 2013. arXiv:1303.1219v1 [Preprint]. Available from: https://arxiv.org/abs/1303.1219v1 [cited 2024 April 24].

38. Wang Z, Fellows IE, Handcock MS. Understanding networks with exponential-family random network models. Soc Networks. 2024; 78:81–91. https://doi.org/10.1016/j.socnet.2023.07.003

39. Snijders TAB. Markov chain Monte Carlo estimation of exponential random graph models. J Soc Struct. 2002; 3(2):1–40.

40. Byshkin M, Stivala A, Mira A, Krause R, Robins G, Lomi A. Auxiliary Parameter MCMC for Exponential Random Graph Models. J Stat Phys. 2016; 165(4):740–754. https://doi.org/10.1007/s10955-016-1650-5

41. Byshkin M, Stivala A, Mira A, Robins G, Lomi A. Fast maximum likelihood estimation via Equilibrium Expectation for large network data. Sci Rep. 2018; 8:11509. https://doi.org/10.1038/s41598-018-29725-8 PMID: 30065311

42. Borisenko A, Byshkin M, Lomi A. A simple algorithm for scalable Monte Carlo inference; 2020. arXiv:1901.00533v4 [Preprint]. Available from: https://arxiv.org/abs/1901.00533v4 [cited 2024 April 24].

43. Stivala A, Palangkaraya A, Lusher D, Robins G, Lomi A. ERGM parameter estimation of very large directed networks: implementation, example, and application to the geography of knowledge spillovers; 2019. Talk presented at INSNA Sunbelt XXXIX Conference. Available from: https://doi.org/10.5281/zenodo.7952037 [cited 2024 April 24].

44. Stivala A, Robins G, Lomi A. Exponential random graph model parameter estimation for very large directed networks. PLoS One. 2020; 15(1):e0227804. https://doi.org/10.1371/journal.pone.0227804 PMID: 31978150

45. Stivala A, Lomi A. A new scalable implementation of the citation exponential random graph model (cERGM) and its application to a large patent citation network; 2022. Talk presented at INSNA Sunbelt XLII conference. Available from: https://doi.org/10.5281/zenodo.7951927 [cited 2024 April 24].

46. Hunter DR, Krivitsky PN, Schweinberger M. Computational statistical methods for social network models. J Comput Graph Stat. 2012; 21(4):856–882. https://doi.org/10.1080/10618600.2012.732921 PMID: 23828720

47. Borgatti SP, Halgin DS. On network theory. Organ Sci. 2011; 22(5):1168–1181. https://doi.org/10.1287/orsc.1100.0641

48. Shalizi CR, Thomas AC. Homophily and contagion are generically confounded in observational social network studies. Soc Meth Res. 2011; 40(2):211–239. https://doi.org/10.1177/0049124111404820 PMID: 22523436

49. Hunter DR, Handcock MS. Inference in Curved Exponential Family Models for Networks. J Comput Graph Stat. 2006; 15(3):565–583. https://doi.org/10.1198/106186006X133069

50. Snijders TAB, Pattison PE, Robins GL, Handcock MS. New specifications for exponential random graph models. Sociol Methodol. 2006; 36(1):99–153. https://doi.org/10.1111/j.1467-9531.2006.00176.x

51. Morris M, Handcock M, Hunter D. Specification of exponential-family random graph models: Terms and computational aspects. J Stat Softw. 2008; 24(4):1–24. https://doi.org/10.18637/jss.v024.i04 PMID: 18650964

52. Robbins H, Monro S. A stochastic approximation method. Ann Math Statist. 1951; 22(3):400–407. https://doi.org/10.1214/aoms/1177729586

53. Koskinen J, Snijders TAB. Simulation, estimation, and goodness of fit. In: Lusher D, Koskinen J, Robins G, editors. Exponential Random Graph Models for Social Networks. New York: Cambridge University Press; 2013. p. 141–166.

54. Dai N, Jones GL. Multivariate initial sequence estimators in Markov chain Monte Carlo. J Multivar Anal. 2017; 159:184–199. https://doi.org/10.1016/j.jmva.2017.05.009

55. Flegal JM, Jones GL. Batch means and spectral variance estimators in Markov chain Monte Carlo. Ann Stat. 2010; 38(2):1034–1070. https://doi.org/10.1214/09-AOS735

56. Jones GL, Haran M, Caffo BS, Neath R. Fixed-width output analysis for Markov chain Monte Carlo. J Am Stat Assoc. 2006; 101(476):1537–1547. https://doi.org/10.1198/016214506000000492

57. Vats D, Flegal JM, Jones GL. Strong consistency of multivariate spectral variance estimators in Markov chain Monte Carlo. Bernoulli. 2018; 24(3):1860–1909. https://doi.org/10.3150/16-BEJ914

58. Vats D, Flegal JM, Jones GL. Multivariate output analysis for Markov chain Monte Carlo. Biometrika. 2019; 106(2):321–337. https://doi.org/10.1093/biomet/asz002

59. Krivitsky PN. Using contrastive divergence to seed Monte Carlo MLE for exponential-family random graph models. Comput Stat Data Anal. 2017; 107:149–161. https://doi.org/10.1016/j.csda.2016.10.015

60. Hinton GE. Training products of experts by minimizing contrastive divergence. Neural Comput. 2002; 14(8):1771–1800. https://doi.org/10.1162/089976602760128018 PMID: 12180402

61. Asuncion A, Liu Q, Ihler A, Smyth P. Learning with blocks: Composite likelihood and contrastive divergence. In: Teh YW, Titterington M, editors. Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics. vol. 9 of Proceedings of Machine Learning Research. PMLR; 2010. p. 33–40.

62. Fellows IE. Why (and when and how) contrastive divergence works; 2014. arXiv:1405.0602v1 [Preprint]. Available from: https://arxiv.org/abs/1405.0602v1 [cited 2024 July 31].

63. Stivala AD, Koskinen JH, Rolls DA, Wang P, Robins GL. Snowball sampling for estimating exponential random graph models for large networks. Soc Networks. 2016; 47:167–188. https://doi.org/10.1016/j.socnet.2015.11.003

64. Hartung J, Knapp G, Sinha BK. Statistical meta-analysis with applications. Hoboken, NJ: John Wiley & Sons; 2008.

65. Giacomarra F, Bet G, Zocca A. Generating synthetic power grids using exponential random graph models. PRX Energy. 2024; 3(2):023005. https://doi.org/10.1103/PRXEnergy.3.023005

66. Python Software Foundation. Python Language Reference, version 3.9; 2020. Available from: http://www.python.org/ [cited 2024 April 29].

67. Harris CR, Millman KJ, van der Walt SJ, Gommers R, Virtanen P, Cournapeau D, et al. Array programming with NumPy. Nature. 2020; 585(7825):357–362. https://doi.org/10.1038/s41586-020-2649-2 PMID: 32939066

68. Bianchi F, Stivala A, Lomi A. Multiple clocks in network evolution. Method Innov. 2022; 15(1):29–41. https://doi.org/10.1177/20597991221077877

69. R Core Team. R: A Language and Environment for Statistical Computing; 2022. Available from: https://www.R-project.org/ [cited 2024 April 29].

70. Flegal JM, Hughes J, Vats D, Dai N, Gupta K, Maji U. mcmcse: Monte Carlo Standard Errors for MCMC; 2021. Available from: https://CRAN.R-project.org/package=mcmcse [cited 2024 April 29].

71. Tange O. GNU Parallel 2018; 2018. Available from: https://doi.org/10.5281/zenodo.1146014 [cited 2024 April 29].

72. Yoo AB, Jette MA, Grondona M. SLURM: Simple Linux Utility for Resource Management. In: Feitelson D, Rudolph L, Schwiegelshohn U, editors. Job Scheduling Strategies for Parallel Processing. vol. 2862 of Lecture Notes in Computer Science. Berlin, Heidelberg: Springer; 2003. p. 44–60.

73. Wilson EB. Probable inference, the law of succession, and statistical inference. J Am Stat Assoc. 1927; 22(158):209–212. https://doi.org/10.1080/01621459.1927.10502953

74. Csárdi G, Nepusz T. The igraph software package for complex network research. InterJournal. 2006; Complex Systems:1695.

75. Wickham H. ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York; 2016. Available from: https://ggplot2.tidyverse.org [cited 2024 April 29].

76. Scherer R. PropCIs: Various confidence interval methods for proportions; 2018. Available from: https://CRAN.R-project.org/package=PropCIs [cited 2024 April 29].

77. Pattison PE, Robins GL, Snijders TAB, Wang P. Conditional estimation of exponential random graph models from snowball sampling designs. J Math Psychol. 2013; 57(6):284–296. https://doi.org/10.1016/j.jmp.2013.05.004

78. Daraganova G, Pattison P. Autologistic actor attribute model analysis of unemployment: Dual importance of who you know and where you live. In: Lusher D, Koskinen J, Robins G, editors. Exponential Random Graph Models for Social Networks. New York: Cambridge University Press; 2013. p. 237–247.

79. Diviák T, Coutinho JA, Stivala AD. A Man's world? Comparing the structural positions of men and women in an organized criminal network. Crime Law Soc Change. 2020; 74(5):547–569. https://doi.org/10.1007/s10611-020-09910-5

80. Letina S. Network and actor attribute effects on the performance of researchers in two fields of social science in a small peripheral community. J Informetrics. 2016; 10(2):571–595. https://doi.org/10.1016/j.joi.2016.03.007

81. Letina S, Robins G, Maslić Seršić D. Reaching out from a small scientific community: the social influence models of collaboration across national and disciplinary boundaries for scientists in three fields of social sciences. Revija za sociologiju. 2016; 46(2):103–139. https://doi.org/10.5613/rzs.46.2.1

82. Gallagher HC. Social networks and the willingness to communicate: Reciprocity and brokerage. J Lang Soc Psychol. 2019; 38(2):194–214. https://doi.org/10.1177/0261927X18809146

83. Stivala A, Wang P, Lomi A. Numbers and structural positions of women in a national director interlock network; 2023. Talk presented at INSNA Sunbelt XLIII Conference. Available from: https://doi.org/10.5281/zenodo.8092829 [cited 2024 April 24].

84. Hunter DR, Goodreau SM, Handcock MS. ergm.userterms: A Template Package for Extending statnet. J Stat Softw. 2013; 52(2):1–25. https://doi.org/10.18637/jss.v052.i02 PMID: 24307887

85. Potterat J, Woodhouse DE, Muth SQ, Rothenberg RB, Darrow WW, Klovdahl AS, et al. Network dynamism: history and lessons of the Colorado Springs study. In: Morris M, editor. Network epidemiology: A handbook for survey design and data collection. Oxford University Press; 2004. p. 87–114.

86. Woodhouse DE, Rothenberg RB, Potterat JJ, Darrow WW, Muth SQ, Klovdahl AS, et al. Mapping a social network of heterosexuals at high risk for HIV infection. AIDS. 1994; 8(9):1331–1336. https://doi.org/10.1097/00002030-199409000-00018 PMID: 7802989

87. Klovdahl AS, Potterat JJ, Woodhouse DE, Muth JB, Muth SQ, Darrow WW. Social networks and infectious disease: The Colorado Springs study. Soc Sci Med. 1994; 38(1):79–88. https://doi.org/10.1016/0277-9536(94)90302-6 PMID: 8146718

88. Rothenberg RB, Woodhouse DE, Potterat JJ, Muth SQ, Darrow WW, Klovdahl AS. Social networks in disease transmission: the Colorado Springs Study. In: Needle RH, Coyle SL, Genser SG, Trotter RT, editors. Social Networks, Drug Abuse, and HIV Transmission. vol. 151. National Institute on Drug Abuse; 1995. p. 3–19.

89. Michell L, Amos A. Girls, pecking order and smoking. Soc Sci Med. 1997; 44(12):1861–1869. https://doi.org/10.1016/S0277-9536(96)00295-X PMID: 9194247

90. Pearson M, Michell L. Smoke Rings: social network analysis of friendship groups, smoking and drug-taking. Drug Educ Prev Polic. 2000; 7(1):21–37. https://doi.org/10.1080/dep.7.1.21.37

91. Pearson M, West P. Drifting smoke rings. Connections. 2003; 25(2):59–76.

92. Pearson M, Steglich C, Snijders TAB. Homophily and assimilation among sport-active adolescent substance users. Connections. 2006; 27(1):47–63.

93. Steglich C, Snijders TAB, West P. Applying SIENA. Methodology. 2006; 2(1):48–56. https://doi.org/10.1027/1614-2241.2.1.48

94. West P, Sweeting H. Background, rationale and design of the West of Scotland 11 to 16 Study; 1996. MRC Medical Sociology Unit Working Paper Number 52.

95. Ripley RM, Snijders TAB, Boda Z, Vörös A, Preciado P. Manual for RSiena; 2024. Available from: https://www.stats.ox.ac.uk/~snijders/siena/RSiena_Manual.pdf [cited 2024 April 25].

96. Snijders TAB. Stochastic actor-oriented models for network dynamics. Annu Rev Stat Appl. 2017; 4:343–363. https://doi.org/10.1146/annurev-statistics-060116-054035

97. Koskinen J. ALAAM; 2024. [Computer software]. Available from: https://github.com/johankoskinen/ALAAM [cited 2024 April 29].

98. Snijders TAB. Description excerpt of 50 girls from "Teenage Friends and Lifestyle Study" data;. Available from: https://www.stats.ox.ac.uk/~snijders/siena/s50_data.htm [cited 2024 April 29].

99. Rozemberczki B, Davies R, Sarkar R, Sutton C. GEMSEC: Graph Embedding with Self Clustering. In: Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2019). New York, NY, USA: Association for Computing Machinery; 2019. p. 65–72.

100. Leskovec J, Krevl A. SNAP Datasets: Stanford large network dataset collection; 2014. Available from: http://snap.stanford.edu/data [cited 2024 April 29].

101. Ceperley D, Dewing M. The penalty method for random walks with uncertain energies. J Chem Phys. 1999; 110(20):9812–9820. https://doi.org/10.1063/1.478034

102. Schmid CS, Chen THY, Desmarais BA. Generative Dynamics of Supreme Court Citations: Analysis with a New Statistical Network Model. Polit Anal. 2022; 30(4):515–534. https://doi.org/10.1017/pan.2021.20

103. Stivala A. EstimNetDirected; 2024. [Computer software]. Available from: https://github.com/stivalaa/EstimNetDirected [cited 2024 April 29].

104. Lam SK, Pitrou A, Seibert S. Numba: A LLVM-Based Python JIT Compiler. In: Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC. LLVM '15. New York, NY, USA: Association for Computing Machinery; 2015. p. 1–6.

105. Bolz CF, Cuni A, Fijalkowski M, Rigo A. Tracing the Meta-Level: PyPy's Tracing JIT Compiler. In: Proceedings of the 4th Workshop on the Implementation, Compilation, Optimization of Object-Oriented Languages and Programming Systems. ICOOOLPS '09. New York, NY, USA: Association for Computing Machinery; 2009. p. 18–25.