Journal of Statistical Software

January 2018, Volume 83, Issue 6. doi: 10.18637/jss.v083.i06

Temporal Exponential Random Graph Models with btergm: Estimation and Bootstrap Confidence Intervals

Philip Leifeld
University of Glasgow

Skyler J. Cranmer
The Ohio State University

Bruce A. Desmarais
Pennsylvania State University

Abstract

The **xergm** package is an implementation of extensions to the exponential random graph model (ERGM). It acts as a meta-package for multiple constituent packages. One of these packages is **btergm**, which implements bootstrap methods for the temporal ERGM estimated by maximum pseudolikelihood. Here, we illustrate the temporal exponential random graph model and its implementation in the package **btergm** using data on international alliances and a longitudinally observed friendship network in a Dutch school.

Keywords: relational data, network analysis, temporal exponential random graph model, TERGM, **xergm**, **btergm**, R.

1. The xergm package

The **xergm** package (Leifeld, Cranmer, and Desmarais 2017b) for the R (R Core Team 2017) statistical computing environment is dedicated to extensions of exponential random graph models (ERGM). These extensions include the so-called temporal exponential random graph model (TERGM), which was implemented in the sub-package **btergm** (Leifeld, Cranmer, and Desmarais 2017a). The TERGM was proposed by Hanneke, Fu, and Xing (2010) owing to Robins and Pattison (2001). Desmarais and Cranmer (2010, 2012c) proposed a bootstrap approach to assessing uncertainty. This article is designed as a hands-on tutorial for specifying and estimating TERGMs for binary networks, assessing goodness-of-fit, and examining predictive performance (Cranmer and Desmarais 2011; Leifeld and Cranmer 2014) using the **btergm** package version 1.9.0.

Below, we briefly review the TERGM and the use of bootstrap methods with estimation via maximum pseudolikelihood, then move on to the practical matters of data preparation, the

software implementation for TERGM estimation, the assessment of model fit, out-of-sample prediction, degeneracy checking, and micro-level interpretation.

Throughout the examples provided below, two datasets are used, which are both delivered with the **xergm.common** package (Leifeld 2017b), another sub-package of **xergm**. The first dataset illustrates the network of international alliances over 20 years and was collected by Cranmer, Desmarais, and Kirkland (2012a) and Cranmer, Desmarais, and Menninga (2012b). The second dataset, collected by Andrea Knecht, illustrates the dynamics of adolescent friendship networks in a Dutch school class (Knecht 2006, 2008; Knecht, Snijders, Baerveldt, Steglich, and Raub 2010; Steglich and Knecht 2009). This dataset is also a classic example for estimating stochastic actor-oriented models using **SIENA** and **RSiena** (Ripley, Snijders, Boda, Vörös, and Preciado 2017b; Ripley, Boitmanis, Snijders, and Schoenenberger 2017a; Snijders, Steglich, and Van de Bunt 2010).

2. ERGM with temporal extensions

The feature of network data that is both scientifically and methodologically distinct is that the probability of a tie forming between any two nodes in the network – often the dependent variable of interest – depends upon the structure of the rest of the network (Cranmer and Desmarais 2011). Additionally, the likelihood of two nodes tying can also depend upon attributes of nodes, which makes tie prediction in part analogous to regression modeling with covariates. To provide an example of the integration of interdependence and covariate theories, consider the dynamics of partner selection within an inter-organizational policy network. Berardo and Scholz (2010) hypothesize that highly connected policy organizations are likely to be sought after due to their coordinating potential. They also hypothesize that actors are likely to seek out trustworthy partners. These hypotheses represent (1) theory about dependence among ties in that the connectedness of a node generates additional connections, and (2) theory about the influence of a nodal attribute (trustworthiness) on the network structure. The methodological research challenge in statistical inference with networks is the development of methods that can simultaneously incorporate interdependence and covariate effects. One flexible method for incorporating a wide range of theories into the analysis of network generation is an ERGM. An ERGM treats a network as a single multivariate observation in which the relations in the network depend on covariates (i.e., exogenous data) as well as each other (i.e., endogenous processes).

2.1. The model

The temporal exponential random graph model, usually called TERGM, is an extension of the ERGM designed to accommodate inter-temporal dependence in longitudinally observed networks. The extension is accomplished by incorporating parameters into an ERGM specification that reflect the ways in which previous realizations of the network determine current features of the network.

To understand the TERGM, begin with the single-network (i.e., cross-sectional) ERGM. An ERGM of the network N, where N is the adjacency matrix in which the (i, j) element $N_{ij} = 1$ if actor/node i sends a tie to j and 0 otherwise, is specified as

$$\mathsf{P}(N, \boldsymbol{\theta}) = \frac{\exp(\boldsymbol{\theta}^\top \mathbf{h}(N))}{c(\boldsymbol{\theta})}, \tag{1}$$

where $\boldsymbol{\theta}$ is the vector of model coefficients, $\mathbf{h}(N)$ is a vector of statistics computed on N that include endogenous (e.g., mutual dyad counts, triangle counts) and exogenous (e.g., within-demographic-group tie counts) dependencies, and $c(\boldsymbol{\theta}) = \sum_{i=1}^{\mathcal{N}} \exp(\boldsymbol{\theta}^\top(N_i))$. Note that the normalizing constant $c(\boldsymbol{\theta})$ is problematic for estimation because the set of all possible permutations of the network with the same number of nodes, denoted $\mathcal{N}$, is extremely large, even with small to moderate sized networks. Note, too, that we use the notation $\boldsymbol{\theta}^\top$ to indicate the transpose of the $\boldsymbol{\theta}$ vector. The transpose symbol $\top$ is different from the number of time steps T introduced in the following equations. A detailed review of ERGMs is provided by a number of sources including Goodreau, Kitts, and Morris (2008b), Hunter, Handcock, Butts, Goodreau, and Morris (2008b), Cranmer and Desmarais (2011), Cranmer, Leifeld, McClurg, and Rolfe (2017), and Lusher, Koskinen, and Robins (2013).

The ERGM in (1) for network N at time t, denoted N^t, can be modified to include dependencies on some number, $K \in \{0, 1, \ldots, T-1\}$, of previously observed networks by introducing lagged networks into $\mathbf{h}$

$$\mathsf{P}(N^t | N^{t-K}, \ldots, N^{t-1}, \boldsymbol{\theta}) = \frac{\exp(\boldsymbol{\theta}^\top \mathbf{h}(N^t, N^{t-1}, \ldots, N^{t-K}))}{c(\boldsymbol{\theta}, N^{t-K}, \ldots, N^{t-1})}, \tag{2}$$

where the specification of K is important because it must fully encompass the temporal dependencies of N^t. In other words, it must be the case that networks in the time series occurring prior to N^{t-K} are independent of N^t, conditional upon $\{N^{t-K}, \ldots, N^{t-1}\}$.

But Equation 2 only specifies a TERGM for a single network at a single point in time, N^t. We can model the joint probability of observing the networks between times $K+1$ and T by taking the product of the probabilities of the individual networks conditional on the others, which we may do since they are independent if K has been chosen appropriately:

$$\mathsf{P}(N^{K+1}, \ldots, N^T | N^1, \ldots, N^K, \boldsymbol{\theta}) = \prod_{t=K+1}^{T} \mathsf{P}(N^t | N^{t-K}, \ldots, N^{t-1}, \boldsymbol{\theta}). \tag{3}$$

In this way, a TERGM can account for high (or low)-order time dependence in a single network or over a series of networks. For more detailed methodological treatment of TERGMs, see Hanneke *et al.* (2010), Cranmer and Desmarais (2011), and Desmarais and Cranmer (2012c). For TERGM applications, see, e.g., Cranmer *et al.* (2012a,b), Czarna, Leifeld, Śmieja, Dufner, and Salovey (2016), Almquist and Butts (2013), Cranmer, Heinrich, and Desmarais (2014) and Ingold and Leifeld (2016).

2.2. Exogenous and endogenous dependencies

As with a cross-sectional ERGM, a TERGM can incorporate both exogenous covariates and endogenous dependencies through the $\mathbf{h}()$ function. It is useful to explore some important examples. We begin with an exogenous covariate X with no temporal dependencies. That is, contemporaneous observations of X, rather than earlier observations of X, affect contemporaneous observations of N. This is included as an edge covariate that models the relationship between ties and covariate values as,

$$h_X(N^t, X^t) = \sum_{i \neq j} N_{ij}^t X_{ij}^t, \tag{4}$$

where X is a relational matrix of the same dimensions as N.

Intertemporal dependencies are also included in a similar way as in the cross-sectional ERGM, but now, functions of the lagged networks are incorporated into the statistics. For example, in many social networks, reciprocity may be a generative feature of the model, but will incorporate a temporal lag (i.e., it may take time for a node to develop a response to a sent tie). We can specify a single-period delayed reciprocity term as:

$$h_r(N^t, N^{t-1}) = \sum_{ji} N^t_{ij} N^{t-1}_{ji}. \tag{5}$$

Naturally, the temporal dependencies involved with the specification of endogenous effects can vary in terms of their order, from 1 to K. For example, if $K = 3$, we could model the effects of inter-temporal triad closure over three periods, but specify over-time reciprocation as only a one-period process. See Morris, Handcock, and Hunter (2008) for an extensive review of endogenous effect specification. The **btergm** package is fully compatible with the **ergm** package, and any endogenous dependencies available in the **ergm** package may be used in **btergm**. Custom model terms added to **ergm** through the **ergm.userterms** package (Hunter, Goodreau, and Handcock 2013; Handcock, Hunter, Butts, Goodreau, Krivitsky, and Morris 2015) or the **ergm.graphlets** package (Yaveroglu, Fitzhugh, Kurant, Markopoulou, Przulj, and Butts 2015) can be used in **btergm** as well. Additionally, **btergm** contains several temporal statistics designed specifically for TERGMs.

2.3. Memory terms

So-called "memory terms" play a special role in the specification of TERGMs. "Memory terms" refer to a class of intertemporal dependencies designed to capture temporal processes without capturing additional network structure. Below we introduce the notion of change statistics for the easy coding and implementation of such effects.

First, note that an ERGM, as articulated in Equation 1, may be equivalently expressed through the conditional probability of an edge from i to j:

$$\pi_{ij}(\boldsymbol{\theta}) = \mathsf{P}(N_{ij} = 1 | N_{-ij}, \boldsymbol{\theta}) = \text{logit}^{-1}\left(\sum_{r=1}^{R} \theta_r \delta_r^{(ij)}(N)\right), \tag{6}$$

where the notation N_{-ij} refers to the entire network N except for the dyad N_{ij}, logit^{-1} is the inverse logistic function such that $\text{logit}^{-1}(x) = 1/(1 + \exp(-x))$, the subscript r refers to any effect included in the $\mathbf{h}$ vector, of length R, and – most importantly – $\delta_r^{(ij)}(N)$ is equal to the change in h_r when N_{ij} is changed from zero to one. The change statistic $\delta_r^{(ij)}(N)$ is the result of the thought experiment in which the value of $h_r(N^-_{ij})$ is subtracted from the value of $h_r(N^+_{ij})$, where $h_r(N^+_{ij})$ indicates the network with the ij dyad – possibly directed – equal to 1 and $h_r(N^-_{ij})$ indicates the network with the ij dyad equal to 0. Because these change values are summed if included as an edgewise covariate in Equation 4, one may compute the change statistics for a given element of the generative model and include them directly. This is useful because it is generally easier to compute change statistics for a given dependency than to code new structural parameters. See Goodreau, Handcock, Hunter, Butts, and Morris (2008a) for a more detailed exposition of change statistics.

What follows is a short taxonomy of "memory terms" that can be included in a TERGM. This list is by no means exhaustive. For example, Krivitsky and Handcock (2014) briefly

discuss a "two-timing" term, and other, more elaborate, memory terms are possible. As the term is vague, we give specific names to the different temporal effects and avoid the use of the generic "memory." To use a statistic in the **btergm** package that has not already been hard-coded (e.g., `memory`, `delrecip`), users need to calculate the change statistic for the statistic and include it as an edge covariate. Note that if the statistics meet the condition that $\delta_r^{(ij)}\left(N^t|N_{l,k}^t=0\right) = \delta_r^{(ij)}\left(N^t|N_{l,k}^t=1\right)\ \forall\{i,j,l,k\}$ (i.e., dyadic independence within time points), then the change statistics involve only past networks, and the statistics themselves are special cases of $h_X(N^t, X^t)$, where X^t is defined by the networks preceding t.

Positive autoregression (lagged outcome network)

This is the simplest memory term to include and is especially useful in cases where edge persistence is of particular interest, but it is not necessary to account for the stability of non-edges. There are many cases where this will be appropriate, but it may be particularly useful when dealing with sparse networks.

The term is specified as a lagged network and included as a dyadic covariate:

$$h_a = \sum N_{ij}^t N_{ij}^{t-1}. \tag{7}$$

There is no difference between this and Equation 4 where $X_{ij}^t = 1$ if $N_{ij}^{t-1} = 1$ and 0 otherwise. In other words, $X^t = N^{t-1}$, in this case. However, since the lagged network is included as a dyadic covariate, the software computes the h statistic as in Equation 4 rather than Equation 7 when we include this term in an ERGM.

The change statistic for positive autocorrelation is $\delta_a^{(ij)} = 1$ when $N_{ij}^{t-1} = 1$ and $\delta_a^{(ij)} = 0$ otherwise. In other words, if we imagine $h_a(N_{ij}^+) - h_a(N_{ij}^-)$, the change will be 0 when $N_{ij}^{t-1} = 0$ and 1 when $N_{ij}^{t-1} = 1$. Directly coding the change statistic as δ_a captures the process of interest.

The substantive interpretation of this term is straightforward: it counts the number of edges that persist from $t-1$ to t. This is comparable to an `edges` term in the dissolution phase of a separable TERGM (Krivitsky and Handcock 2014).

Dyadic stability

One may be dissatisfied with positive autocorrelation because it does not treat the persistence of non-edges (e.g., the tendency of dyadic relationships that do not exist at $t-1$ to continue not to exist at t). When one wishes to treat the stability of existing and non-existing relationships equivalently, the following stability term is useful.

The statistic should count the number of dyads that are stable between $t-1$ and t. The term may be written as

$$h_s = \sum_{ij} N_{ij}^t N_{ij}^{t-1} + (1 - N_{ij}^t)(1 - N_{ij}^{t-1}), \tag{8}$$

where all notation is the same as above.

The substantive interpretation is also simple: one counts the number of stable dyads, both persistent edges and persistent non-edges, between two time periods.

Consider now the change statistic, $\delta_s^{(ij)}$, for this term. Reflecting on $h_s(N_{ij}^+) - h_s(N_{ij}^-)$ makes it clear that the statistic will lose 1 whenever $N_{ij}^{t-1} = 0$ and gain 1 whenever $N_{ij}^{t-1} = 1$. As

such, the change statistic can be coded as a covariate where the $X_{ij}^t = -1$ when $N_{ij}^{t-1} = 0$ and $X_{ij}^t = 1$ when $N_{ij}^{t-1} = 1$. Note that a stability term and a positive autocorrelation term should not be included in the same model; one must be excluded as a baseline.

Edge innovation and loss

The analyst may be interested in directly modeling the tendency of new edges to form between observed time periods. Such a tendency would be modeled with an edge innovation statistic, $h_e = \sum_{ij} N_{ij}^t(1 - N_{ij}^{t-1})$, which counts the number of edges that exist in time t and did not exist in time $t-1$. This would be incorporated into TERGM by specifying a dyadic covariate X^t coded 1 if $N_{ij}^{t-1} = 0$ and 0 if $N_{ij}^{t-1} = 1$. In other words, the covariate is coded 1 any time an edge in period t would be new with respect to the previous period.

The change statistic produced from this coding will be $\delta^{(ij)} = 1$ if $N_{ij}^{t-1} = 0$ and 0 otherwise (i.e., one would be added to the edge creation count if N_{ij}^{t-1} were 0 and N_{ij}^t were toggled from 0 to 1). The substantive interpretation is that we are counting the number of edges that were created between $t-1$ and t.

To do the inverse of this: modeling edge loss directly, let $h_l = \sum_{ij}(1-N_{ij}^t)N_{ij}^{t-1}$, which counts the number of edges that existed in $t-1$ and do not exist in t. The change statistic of h_l is coded -1 if $N_{ij}^{t-1} = 1$ and 0 if $N_{ij}^{t-1} = 0$.

Note, though, that each of these different memory parameterizations is functionally just shifting around the baseline of a categorical variable to ease interpretation. The fit of the model should not be affected, given the intercept and some function of N_{ij}^{t-1}.

2.4. Dynamic effects and pooling time periods

Temporal change in networks can be governed by two levels of dynamics: (1) variation according to an underlying TERGM or (2) change in the generative TERGM (i.e., temporal variation in $\boldsymbol{\theta}$). When a single TERGM is applied to a pooled time series of networks, the analyst either implicitly or explicitly assumes that all changes in the network are attributable to (1). However, in many applications, variation in the underlying model is equally plausible. There are several ways in which parameter/model heterogeneity over time can be diagnosed and addressed.

The major symptom of model variation is that the pooled model fits different time spans differently (e.g., is overly dense early on in the time series and too sparse in later periods). This can be diagnosed by evaluating model fit on a period-by-period basis. There are two basic approaches to addressing temporal heterogeneity in the underlying model. First, completely different models can be fitted in different time spans, as is done by Cranmer *et al.* (2012a). The other approach is to interact individual statistics with smooth functions of time (e.g., linear time trends, quadratics and higher order polynomials). This is similar to the approach taken by Cranmer *et al.* (2014).

More specifically, one may be interested in modeling time trends in the probability of edges over time by including temporal statistics of the form

$$h_t = \sum_{ij} N_{ij}^t f(t), \tag{9}$$

where $f(t)$ is a function of time, for example a linear time trend $f(t) = t$, a quadratic trend $f(t) = t^2$, arbitrary polynomial functions of the form $f(t) = a + bt + ct^2$, geometric decay

$f(t) = \sqrt{t}$, or step-wise functions of the form $f(t) = 1$ if $a < t < b$ and 0 otherwise, with user-defined a and b which capture exogenous events.

Moreover, one may be interested in interaction effects between exogenous covariates and time,

$$h_{tcov} = \sum_{ij} N_{ij}^t X_{ij}^t f(t), \tag{10}$$

which capture the increasing or decreasing role of a covariate X with time.

2.5. Maximum pseudolikelihood with bootstrap confidence intervals

The **btergm** package uses the bootstrapped pseudolikelihood inference methods described by Desmarais and Cranmer (2012c), and a much more detailed treatment of this estimation routine may be found there. Maximum pseudolikelihood estimation (MPLE) works by replacing the joint likelihood with the product over conditional (on the rest of the network) dyadic tie probabilities (Strauss and Ikeda 1990). These conditional dyadic tie probabilities were described above in Equation 6. Consider a slight modification of the notation in Equation 6, where we add a temporal index to π in order to define it for a given network as a function of the past $t - K$ networks and the non-ij entries in the network N^t. To find the MPLE, one need only to maximize

$$\arg\max_{\theta} \sum_{t=K+1}^{T} \sum_{\langle ij \rangle} \ln \left[\left(\pi_{ij}^t(\boldsymbol{\theta}) \right)^{N_{ij}^t} \left(1 - \pi_{ij}^t(\boldsymbol{\theta}) \right)^{1-N_{ij}^t} \right], \tag{11}$$

where T indicates networks in the time series and $\langle ij \rangle$ indicates the set of dyads. This optimization can be performed with a hill-climbing algorithm and does not involve simulation. The fact that MPLE does not require simulation represents a substantial advantage over the major alternative approach, which is based on Markov chain Monte Carlo (MCMC) and is much more computationally demanding. This computational advantage is more significant with voluminous data – either many nodes or large T – as simulations require more time and memory and the MPLE gains in efficiency as the volume of data increases (Desmarais and Cranmer 2012c). Indeed, the MPLE converges, asymptotically, to the MLE (Strauss and Ikeda 1990; Hyvarinen 2006), making it a consistent estimator as either the number of nodes or the number of network samples (T) goes to infinity (Strauss and Ikeda 1990; Arnold and Strauss 1991).

Yet the MPLE has a substantial drawback: uncertainty measures based on the MPLE are biased downwards, sometimes severely. Specifically, while the MPLE is a consistent estimator of the MLE, using the inverse of the negative Hessian $([-\mathbf{H}])^{-1}$ of the log likelihood at the MPLE to measure variance in the MPLE leads to substantial underestimation (e.g., on the order of 70%) (Van Duijn, Gile, and Handcock 2009).

Desmarais and Cranmer's (2012c) approach to bootstrapping the pseudolikelihood corrects this problem for TERGMs (not for single network ERGMs). The procedure works as follows: take a sample of s estimates of $\boldsymbol{\theta}$ by drawing s samples of $T - K$ networks from the longitudinally observed series of networks $\{N^{K+1}, \ldots, N^T\}$ and computing $\hat{\boldsymbol{\theta}}$. The efficacy of this approach is made apparent by the fact that

$$\sum_{t=K+1}^{T} \frac{\partial \sum_{\langle ij \rangle} \ln \left[\left(\pi_{ij}^t(\hat{\boldsymbol{\theta}}) \right)^{N_{ij}^t} \left(1 - \pi_{ij}^t(\hat{\boldsymbol{\theta}}) \right)^{1-N_{ij}^t} \right]}{\partial \hat{\boldsymbol{\theta}}} = \mathbf{0}, \tag{12}$$

the very definition of an M-estimator (Lahiri 1992). The consistency of the temporal bootstrap confidence intervals is based on the consistency and asymptotic normality of the MPLE as T goes to infinity (Arnold and Strauss 1991; Desmarais and Cranmer 2012c). Desmarais and Cranmer (2012c) validate the technique extensively with Monte Carlo studies.

2.6. Markov chain Monte Carlo maximum likelihood estimation

The **btergm** package also contains routines that employ Markov chain Monte Carlo maximum likelihood estimation (MCMC-MLE) for TERGMs. They serve as an alternative to MPLE with bootstrap confidence intervals. With MCMC-MLE, all networks are combined in a large block-diagonal matrix, and an ERGM is estimated on this matrix with structural zeros prohibiting cross-network edges in the off-diagonal blocks. The estimation procedure of Hunter *et al.* (2008b) is used for this. MCMC-MLE is slower than MPLE with bootstrap confidence intervals, even prohibitively slow for more than a few small consecutive networks. Therefore MPLE with bootstrapped confidence intervals is the only option in many empirical applications. MCMC-MLE is also prone to degeneracy when the model specification does not hit the true data-generating process. However, the estimates of MCMC-MLE may be more accurate precisely when only few networks are observed, as the consistency of MPLE and the bootstrap confidence intervals increases with more observations (see Section 2.5). Thus both estimators have complementary strengths. For example, applications in the field of international relations are often based on 20 to 50 annual time slices of the international system – a typical situation where MPLE with bootstrapped confidence intervals can play out its strengths –, while applications in public policy are often based on two or three survey waves of political actors and their collaboration ties – a typical situation where MCMC-MLE may be preferable.

3. Software implementation in R

The **xergm** package (Leifeld *et al.* 2017b) for the statistical computing environment R (R Core Team 2017) serves as a meta-package for extensions of exponential random graph models. Package **xergm** is available from the Comprehensive R Archive Network (CRAN) at `https://CRAN.R-project.org/package=xergm`. One of its constituent packages is the **btergm** package (Leifeld *et al.* 2017a), which implements TERGM estimation (see Section 2.5). Loading the **xergm** package automatically loads the **btergm** package as well, but **btergm** can also be loaded on its own without the other companion packages contained in **xergm**.

The package contains the `btergm` function, which implements MPLE estimation for TERGMs with bootstrap confidence intervals. The `btergm` syntax is highly compatible with that of the `ergm` function in the **ergm** package (Hunter *et al.* 2008b; Handcock, Hunter, Butts, Goodreau, Krivitsky, and Morris 2017). Moreover, the package contains the `mtergm` function, which has the same syntax as the `btergm` function and estimates TERGMs by MCMC-MLE. In addition to these estimation functions, there are several support functions that we illustrate in the following sections. They include goodness-of-fit assessment, degeneracy checks, and micro-level interpretation of fitted models. All of these functions are highly compatible with the **statnet** suite of packages (Handcock, Hunter, Butts, Goodreau, and Morris 2008; Goodreau *et al.* 2008a; Handcock, Hunter, Butts, Goodreau, Krivitsky, Bender-deMoll, and Morris

2016), in particular the **ergm** package (Hunter *et al.* 2008b; Morris *et al.* 2008; Handcock *et al.* 2017) and the **network** package (Butts 2008, 2015).

The next two sections provide an introduction to TERGM modeling using the **btergm** package. Two examples are presented along with the code necessary to replicate the examples: a network of international alliances, and a school friendship network.

Several R packages should be loaded for running the examples presented below: the **texreg** package (Leifeld 2013, 2017a) will be used for displaying estimation results; the **statnet** suite of network analysis packages provides basic functions for handling network data (Handcock *et al.* 2008; Butts 2008); and the **xergm** package (Leifeld *et al.* 2017b). It is important to load the **btergm** package *after* loading the **statnet** package because **btergm** provides add-on functionality that masks some of the functions contained in the latter. We also set the random seed for exact reproducibility.

```
R> library("statnet")
R> library("texreg")
R> library("xergm")
R> set.seed(10)
```

4. A simple example: International alliances

The first example is a model of international alliances (see also Cranmer *et al.* 2012a,b). The package contains the international defense alliance network among 164 countries, covering the years 1981–2000. In this toy dataset, the same countries are present across all years. This makes it a simple example without the need for any data preparation. The alliance network is stored in 20 '`network`' objects wrapped in a '`list`' object called `allyNet`. These objects can be manipulated and plotted using functions from the **network** package (Butts 2008). For example, the following code loads the dataset and plots the last three alliance networks (as displayed in Figure 1).

```
R> data("alliances", package = "xergm.common")
R> pdf("alliances.pdf", width = 9, height = 3)
R> par(mfrow = c(1, 3), mar = c(0, 0, 1, 0))
R> for (i in (length(allyNet) - 2):length(allyNet)) {
+    plot(allyNet[[i]], main = paste("t =", i))
+  }
R> dev.off()
```

Network data for use with package **btergm** should be stored as lists of network objects in chronological order, with one network per time period. Dyadic covariates can be stored either as lists of network objects or matrices (in the case of time-varying covariates) or as single network objects or matrices (in the case of covariates that do not change over time). Node-level data should be stored as vertex attributes of the networks that comprise the outcome network time series (i.e., the temporal network to be modeled).

The alliance dataset contains the constant dyadic covariate `contigMat`, which is a 164×164 binary matrix in which a 1 indicates that two countries share a border. Furthermore, it

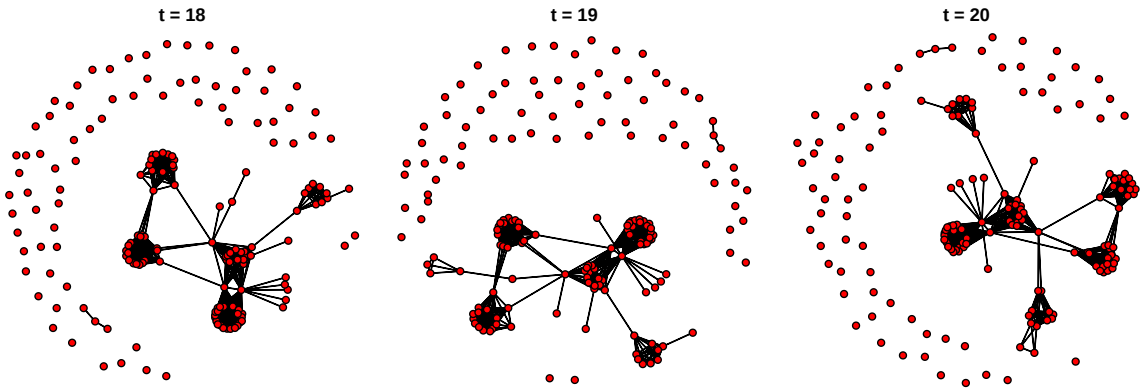

Figure 1: The last three years in the network of international alliances.

contains two time-varying dyadic covariates: `LSP` records the number of shared partners between countries in the alliance network from the previous year, and `warNet` contains matrices that indicate whether two states were in a militarized interstate dispute in the respective year. Each network in the `allyNet` list also contains several node-level attributes, for example `cinc` (indicating military capabilities) and `polity` (the "polity" score, which indicates regime authority on a scale from -10 = hereditary monarchy to $+10$ = consolidated democracy). Nodal attribute data can be accessed using commands from the **network** package (Butts 2008), e.g., `get.vertex.attribute` and related functions. More information about the dataset can be found on the help page `?alliances`.

In a first simple model, a TERGM without any temporal dependencies is estimated. This corresponds to a pooled ERGM – pooling across T networks that are assumed to be independent of each other – where the estimates reflect the average effects across the 20 time points. The (somewhat unrealistic) assumption here is that the consecutive networks are independent from each other. Note that this functionality could be used to fit a single set of ERGM parameters to many independently sampled networks (e.g., as in Goodreau *et al.* 2008b, who apply ERGM to friendship networks measured in 59 schools). The model and the coefficients therefore describe a single data-generating process that applies to all 20 time steps independently.

A TERGM is estimated with the `btergm` function, which implements the bootstrapping procedure described above and stands for "bootstrapped TERGM." The first argument of the `btergm` function is a formula like in the `ergm` function, but accepting lists of networks or matrices instead of a single network or matrix as the dependent variable. In all other regards, the syntax of `btergm` and the syntax of `ergm` are identical.

The model is estimated with 50 bootstrap replications (argument `R = 50`). More replications means less simulation error in the confidence intervals calculated, but also longer runtime. As a rule of thumb, at least 100 replications are necessary for testing purposes, while something on the order of 1,000 replications should be used for publication purposes. Here, we use a smaller sample for purposes of illustration. Options for parallel processing on multicore CPUs or HPC servers are available (see `?btergm`). In the following example, we use two cores and an ad-hoc PSOCK cluster, which is not the fastest option but works also in Windows environments without any additional preparation (while `multicore` forking is restricted to Unix systems).

```
R> model.1a <- btergm(allyNet ~ edges + gwesp(0, fixed = TRUE) +
+    edgecov(LSP) + edgecov(warNet) + nodecov("polity") +
+    nodecov("cinc") + absdiff("polity") + absdiff("cinc") +
+    edgecov(contigMat), R = 50, parallel = "snow",  ncpus = 2)
R> summary(model.1a, level = 0.95)

==========================
Summary of model fit
==========================

Formula:   allyNet ~ edges + gwesp(0, fixed = TRUE) + edgecov(LSP) +
edgecov(warNet) + nodecov("polity") + nodecov("cinc") + absdiff("polity") +
absdiff("cinc") + edgecov(contigMat)

Time steps: 20

Bootstrapping sample size: 50

Estimates and 95% confidence intervals:
                        Estimate    2.5%   97.5%
edges                 -5.0498497 -5.1861 -4.8859
gwesp.fixed.0          1.6242771  1.5516  1.6938
edgecov.LSP[[i]]       2.1065906  1.8301  2.4422
edgecov.warNet[[i]]    0.2866346  0.1953  0.3893
nodecov.polity        -0.0050764 -0.0121  0.0012
nodecov.cinc           8.3308516  2.6219 12.7612
absdiff.polity        -0.1257948 -0.1379 -0.1141
absdiff.cinc          -4.8494093 -8.5083  0.5427
edgecov.contigMat[[i]] 3.2145679  3.1287  3.3238
```

In Model 1a, the collection of 20 alliance networks is jointly explained by a number of endogenous and exogenous statistics. The model contains network dependencies within time steps, such as `gwesp(0, fixed = TRUE)`, which controls for triadic closure by at least one shared partner. Any model term available in the **ergm** package (Hunter *et al.* 2008b; Handcock *et al.* 2017) can also be used with **btergm**. Details on available model terms can be found on the help page `?"ergm-terms"`. Moreover, the model contains several nodal and dyadic covariates. Note that nodal covariates – e.g., `nodecov("polity")` – are wrapped in quotes because they refer to the attribute stored in the network objects contained in `allyNet`, while dyadic covariates – e.g., `edgecov(warNet)` – are stored in separate objects in the work space.

The coefficients and confidence intervals resulting from the estimation are stored in a '`btergm`' object, which is an instance of an S4 class for which several methods like `coef`, `getformula`, `nobs`, `summary`, `print`, and `confint` are available. The results can be displayed using the `summary` method, and its `level` argument can be used to adjust the confidence level. The results are shown in Table 1. It is not possible to set interval or inequality constraints on the coefficients using any of the estimation functions in package **btergm**, so the estimated coefficients can take on any real value.

	Model 1a	Model 1b
`edges`	−5.05 [−5.19; −4.89]*	−3.14 [−3.76; −2.34]*
`gwesp.fixed.0`	1.62 [1.55; 1.69]*	1.79 [1.46; 2.04]*
`edgecov.LSP[[i]]`	2.11 [1.83; 2.44]*	0.11 [0.06; 0.46]*
`edgecov.warNet[[i]]`	0.29 [0.20; 0.39]*	−1.14 [−2.32; 0.21]
`nodecov.polity`	−0.01 [−0.01; 0.00]	−0.02 [−0.05; 0.02]
`nodecov.cinc`	8.33 [2.62; 12.76]*	−0.16 [−9.63; 17.15]
`absdiff.polity`	−0.13 [−0.14; −0.11]*	−0.07 [−0.13; −0.01]*
`absdiff.cinc`	−4.85 [−8.51; 0.54]	3.27 [−10.23; 10.70]
`edgecov.contigMat[[i]]`	3.21 [3.13; 3.32]*	1.60 [0.56; 2.49]*
`edgecov.memory[[i]]`		5.10 [4.68; 5.89]*
`edgecov.timecov[[i]]`		0.07 [0.00; 0.16]*
`edgecov.timecov.warNet[[i]]`		0.05 [−0.11; 0.11]

* 0 outside the confidence interval.

Table 1: International alliance TERGM examples.

By default, a 95 % confidence interval is reported around the estimates. For example, the `absdiff("polity")` term has an estimate of −0.13, which means that an additional difference in the polity scores of two countries of one point decreases their odds of being allies by $100 \cdot (\exp(0.13) - 1) \approx 13.88\%$ on average conditional on the rest of the network. The effect is significant because 0 is outside the confidence interval of $[-0.14; -0.11]$. See Section 6 and Desmarais and Cranmer (2012a) for further details on interpretation of ERGMs and TERGMs. The `summary` and `confint` methods have a `type` argument, where the confidence interval type can be changed from quantile-based confidence intervals (the default `type = "perc"`) to bias-corrected confidence intervals (`type = "bca"`), as implemented in the **boot** package (Canty and Ripley 2017).

The results indicate, among other findings, that direct contiguity of two countries makes them more likely to be allied, as indicated by the positive and significant `edgecov.contigMat` effect, and that two countries that fight each other are more likely to be allied than countries that do not have a war in the respective year (`edgecov.warNet`). But are these results valid? Statistical theory suggests that the joint likelihood is only valid if the model encompasses *all* relevant cross-sectional and temporal dependencies (see Sections 2.1 and 2.4). This can be diagnosed by examining model fit. We can assess the endogenous goodness-of-fit of the model using the `gof` function in the **btergm** package. `gof` is a generic function that has methods for several kinds of network models, including '`btergm`', '`mtergm`', '`ergm`', and **RSiena** models, both for within-sample and out-of-sample fit assessment, following the procedures suggested by Hunter, Goodreau, and Handcock (2008a). Details on out-of-sample predictive fit are provided in Section 5.6.

```
R> gof.1a <- gof(model.1a, nsim = 50, statistics = c(esp, geodesic, deg))
R> pdf("gof-1a.pdf", width = 8, height = 2.5)
R> plot(gof.1a)
R> dev.off()
```

The `gof` method accepts arguments `nsim` (for the number of networks simulated from each time step to compare to the observed networks) and `statistics` (for user-supplied or hard-

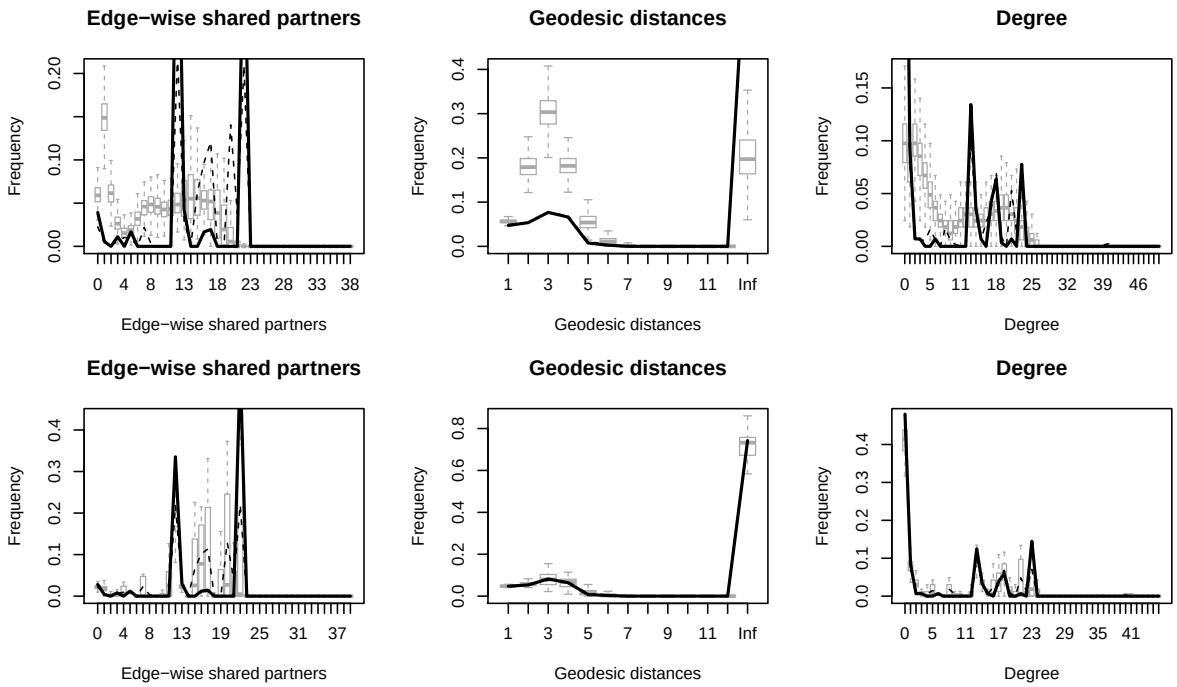

Figure 2: Goodness-of-fit assessment for the non-temporal Model 1a (upper row) and Model 1b with temporal dependencies (lower row).

coded functions to compare simulated and observed network matrices and condense the comparison into a single quantity or a vector of values). The goodness-of-fit statistics supplied with the package are listed on the help page `?"gof-statistics"`. The `gof` function also supports parallel processing (see `?"gof-methods"`).

The `nsim = 50` argument causes the `gof` function to simulate a total of 1,000 networks (50 from each of the 20 time steps). Naturally, when estimating models for publication, more simulations are preferable. Here, however, exposition is accomplished with fewer simulations and less computing time.

The resulting `gof.1a` object can be printed to the console in order to obtain comparison tables of the edge-wise shared partner, geodesic distance, and degree distributions of the simulated versus the observed networks. The object can also be plotted, as shown in the upper row of Figure 2. Interpretation of these plots is straightforward; the model is said to fit better the closer the medians of the box plots (based on the simulated networks) come to the line that plots the actual value of these statistics in the observed network. Obviously, the endogenous model fit of Model 1a is poor. Therefore we estimate a model with some simple temporal dependencies as follows:

```
R> model.1b <- btergm(allyNet ~ edges + gwesp(0, fixed = TRUE) +
+    edgecov(LSP) + edgecov(warNet) + nodecov("polity") +
+    nodecov("cinc") + absdiff("polity") + absdiff("cinc") +
+    edgecov(contigMat) + memory(type = "stability") +
+    timecov(transform = function(t) t) +
+    timecov(warNet, transform = function(t) t), R = 50)
```

Model term	Arguments with default values	Details
`delrecip`	`mutuality = FALSE`, `lag = 1`	Delayed reciprocity or mutuality.
`timecov`	`x = NULL`, `minimum = 1`, `maximum = NULL`, `transform = function(t) 1`	Time trend or interaction with time.
`memory`	`type = "stability"`, `lag = 1`	Positive autoregression; dyadic stability; edge innovation or loss.

Table 2: Temporal network statistics.

```
R> texreg(list(model.1a, model.1b), single.row = TRUE,
+    include.nobs = FALSE, file = "alliance-table.tex",
+    caption = "International alliance TERGM examples.",
+    label = "alliance-table",
+    custom.model.names = c("Model~1a", "Model~1b"),
+    use.packages = FALSE, booktabs = TRUE, dcolumn = TRUE)
R> gof.1b <- gof(model.1b, nsim = 50, statistics = c(esp, geodesic, deg))
R> pdf("gof-1b.pdf", width = 8, height = 2.5)
R> plot(gof.1b)
R> dev.off()
```

The results of Model 1b along with Model 1a are displayed in Table 1. The table is created using the **texreg** package (Leifeld 2013, 2017a). Goodness-of-fit is assessed like in the previous model, and the resulting diagrams can be found in the lower row of Figure 2. The model with temporal dependencies fits considerably better, although there is still some room for improvement. With increasing endogenous model fit, the estimates become less biased.

The new model is identical to the previous model, except for three additional model terms that represent temporal dependence:

- `memory(type = "stability")` is a dyadic stability memory term (see Section 2.3), which controls whether ties and non-ties at one time point carry over to ties and non-ties, respectively, at the next time point. As indicated by the positive and significant coefficient, there is a great amount of stability in alliance ties over time.

- `timecov(transform = function(t) t)` is a time covariate (see Section 2.4) that simply checks whether there is a (linear) time trend in the number of alliance edges over time. The `transform` argument accepts any functional form provided as a function, including linear time trends (`transform = function(t) t`), quadratic time trends (`transform = function(t) t^2`), and geometric decay (`transform = sqrt`). The default is `transform = function(t) 1 + (0 * t) + (0 * t^2)`). Additionally, stepwise functions of time are possible by setting the `minimum` and `maximum` arguments. Time points below and above the minimum and maximum, respectively, are set to 0 while the function given in the `transform` argument is applied to the remaining time steps. This functionality serves to model structural breaks in the data-generating process, for example due to political reforms or new knowledge entering a system. Here,

the linear time trend has a positive and significant coefficient, which indicates that the network gets slightly denser over time.

- `timecov(warNet, transform = function(t) t)` is an interaction between a linear time effect and the `warNet` exogenous covariate. The model term tests whether dyadic conflicts become more (or less) important over time in explaining alliances. This is not the case, as shown in Table 1.

Besides `memory` and `timecov`, the package contains a third hard-coded temporal network statistic, `delrecip`, which tests for delayed reciprocity (i.e., whether an edge from node j to node i in one time period leads to an edge from i to j later). More temporal dependencies may be included in the future. Table 2 gives an overview of the temporal model terms that are currently implemented. The help page for `?"tergm-terms"` provides additional details.

While some of the temporal model terms are significant, some of the other model terms that were previously significant have lost their explanatory power, such as the `warNet` covariate and the nodal effects for `polity` and `cinc`. Simply controlling for previous states of the system provides a better-fitting account of who is allied with whom than these variables. However, the majority of model terms are still significant after controlling for the temporal effects.

5. Advanced example: Friendship networks in a school class

This first TERGM example illustrated the use of the `btergm` and `gof` functions with and without temporal dependence, and introduced the data structures and model syntax. The next example will serve to introduce advanced functions, such as `mtergm` estimation, changes in actor composition and other data preparation issues, and out-of-sample prediction with TERGMs.

The dataset used for the second example contains four waves of a friendship network in a Dutch school class (Knecht 2006, 2008; Knecht *et al.* 2010; Steglich and Knecht 2009). It is delivered with the **xergm** package and is also the classic example for estimating stochastic actor-oriented models using **SIENA** and **RSiena** (Ripley *et al.* 2017b,a; Snijders *et al.* 2010).

The four network waves are stored as matrices in a '`list`' object called `friendship`. There are several nodal and dyadic covariates (see `help("knecht")` for details). Of particular interest are the sex of the pupils (stored in a data frame called `demographics`) and a network called `primary`, which contains dyadic information on which pupils co-attended the same primary school.

We replicate a basic model described in Snijders *et al.* (2010). For the sake of simplicity, we replicate the specification used by Snijders *et al.* (2010) exactly, even though it might be possible to achieve a better fit by altering this specification. In this model, the topology of the networks is determined by the following quantities (with the corresponding model terms of the **ergm** package given in brackets – see `help("ergm-terms")` for details):

- a baseline probability of establishing edges (`edges`);
- the in-degree and out-degree of the nodes in the network (computed using the `degree` function and modeled using `nodeocov` and `nodeicov` terms);

- the sex of ego, the sex of alter, and sex match of ego and alter (`nodeofactor`, `nodeifactor`, and `nodematch`, respectively);
- primary school co-attendance (`edgecov`);
- the tendency of edges to be reciprocated (`mutual`);
- and the number of cyclic triples, transitive triples, and transitive ties (`ctriple`, `ttriple`, and `transitiveties`, respectively).

5.1. Addressing missing data

After attaching the dataset to the workspace, some missing data need to be addressed. The dataset in its original form contains four friendship matrices with 26 nodes (equaling the number of rows and columns) each. However, one pupil dropped out of the class, which results in missing data for node 21 at time steps 3 and 4. In the matrices, these missing values are coded using values of `10`. Moreover, at time step 2, one pupil does not provide any answers, and at time step 3, two pupils do not provide any answers. This kind of unit non-response is coded as `NA` in the dataset.

Because the `btergm` estimation function cannot handle any missing values, something must be done to address such values. One has several options. Perhaps most common is to replace `NA` entries with the modal value (usually `0`). This makes sense in situations where one can assume that all present edges will be observed, though we may not have a specific recording of `0` for absent edges. Note, however, that the erroneous introduction of `0`s where edges actually exist is a form of measurement error that will result in biased statistical models. A second option is to remove nodes with incomplete edge profiles from the dataset. This strategy, at best, is inefficient because removing a node – perhaps because of a single missing value in one of its edges – requires removing all incoming and outgoing ties from that node. In other words, the network nature of the data means that an entire row and column of the data matrix will be removed any time we remove a node. This compounds the inefficiency of the case-wise deletion process compared to that same inefficiency in a normal, rectangular, data frame. At worst, case-wise deletion of nodes with missing edges results in bias. Bias will occur from this procedure any time the occurrence of missing edges is not completely random (e.g., if the occurrence of missing values is related to any attributes of the edge or node, observed or unobserved, bias will be the result). Lastly, missing edge values may be imputed with one of several techniques from a new and budding literature (Handcock and Gile 2010; Koskinen, Robins, Wang, and Pattison 2013; Robins, Pattison, and Woolcock 2004). With this cautionary note, the **xergm** package provides the `handleMissings` function to aid the user in removing or imputing missing data by either removing nodes with missing data or imputing modal values.

In this example, we choose to remove nodes who dropped out (coded as `10`) because they are no longer part of the network, while we impute the values affected by unit non-response using the modal value. To do this, it is necessary to be able to identify the respective nodes across network waves and covariates. Removing a node would otherwise make the networks incomparable because they would have different dimensions, with some networks having 25 and others 26 nodes. By labeling the rows and columns before removing the respective nodes, we ensure that the nodes can be assigned correctly across network or matrix objects even after

removing rows and columns. Therefore the first step is the assignment of row and column names to all matrices, and names to all vectors containing attribute data:

```
R> data("knecht", package = "xergm.common")
R> for (i in 1:length(friendship)) {
+    rownames(friendship[[i]]) <- 1:nrow(friendship[[i]])
+    colnames(friendship[[i]]) <- 1:ncol(friendship[[i]])
+  }
R> rownames(primary) <- rownames(friendship[[1]])
R> colnames(primary) <- colnames(friendship[[1]])
R> sex <- demographics$sex
R> names(sex) <- 1:length(sex)
```

The second step is the application of the `handleMissings` function to the list of friendship networks. The first function call removes all nodes that have values of `10`, and the second function call replaces `NA` values by the modal value in the respective matrix:

```
R> friendship <- handleMissings(friendship, na = 10, method = "remove")

t = 1: 0% of the data (= 0 ties) were dropped due to 0% (= 0) missing ties.
t = 2: 0% of the data (= 0 ties) were dropped due to 0% (= 0) missing ties.
t = 3: 7.54% of the data (= 51 ties) were dropped due to 7.54% (= 51)
       missing ties.
t = 4: 7.54% of the data (= 51 ties) were dropped due to 7.54% (= 51)
       missing ties.

R> friendship <- handleMissings(friendship, na = NA, method = "fillmode")

t = 1: 0% of the data (= 0 ties) were replaced by the mode (0) because they
       were NA.
t = 2: 3.85% of the data (= 26 ties) were replaced by the mode (0) because
       they were NA.
t = 3: 8% of the data (= 50 ties) were replaced by the mode (0) because they
       were NA.
t = 4: 0% of the data (= 0 ties) were replaced by the mode (0) because they
       were NA.
```

5.2. Data preparation and manual adjustment of dimensions

The model will contain node-level covariates. To accommodate them in the model formula, we need to convert the matrices at each time step into '`network`' objects and embed the node attributes in these networks. However, the nodal covariates like `sex` are constant over time and have 26 observations while we have just removed nodes from some of the time steps. Therefore we need to adjust the dimensions of the covariates to the dependent network at each time step by removing observations from the node attribute that are not present in the network object. To do that, package **btergm** provides a helper function called `adjust`.

In the following code block, we iterate over the four time steps, and at each time step, we adjust the dimensions of the `sex` covariate to the dimensions of the current friendship network and embed it in the network object. We also generate two nodal covariates `idegsqrt` and `odegsqrt`, which contain the square root of the in-degree or out-degree centrality of each node (as in the original contribution by Snijders *et al.* 2010), and embed them in the network object at each time step. Finally, we print the dimensions of the four networks to see whether we have indeed 26 nodes at the first two and 25 nodes at the last two time steps.

```
R> for (i in 1:length(friendship)) {
+    s <- adjust(sex, friendship[[i]])
+    friendship[[i]] <- network(friendship[[i]])
+    friendship[[i]] <- set.vertex.attribute(friendship[[i]], "sex", s)
+    idegsqrt <- sqrt(degree(friendship[[i]], cmode = "indegree"))
+    friendship[[i]] <- set.vertex.attribute(friendship[[i]],
+        "idegsqrt", idegsqrt)
+    odegsqrt <- sqrt(degree(friendship[[i]], cmode = "outdegree"))
+    friendship[[i]] <- set.vertex.attribute(friendship[[i]],
+        "odegsqrt", odegsqrt)
+  }
R> sapply(friendship, network.size)


t1 t2 t3 t4
26 26 25 25
```

To get a first visual impression of the networks after the data preparation step, we plot each network wave using methods from the **statnet** suite of packages. The result is displayed in Figure 3.

```
R> pdf("knecht.pdf")
R> par(mfrow = c(2, 2), mar = c(0, 0, 1, 0))
R> for (i in 1:length(friendship)) {
+    plot(network(friendship[[i]]), main = paste("t =", i),
+      usearrows = TRUE, edge.col = "grey50")
+  }
R> dev.off()
```

We see that the topology (i.e., the geometric shape) of the network varies considerably over the four periods of observation. The primary modeling challenge of the TERGM analysis is to capture the generative process that leads to these four observed networks.

5.3. Automatic adjustment of network dimensions

We begin by explaining edge formation at these four time steps, first by assuming independence between the time steps (Model 2a) and then by modeling network evolution as a process with temporal dependencies (Model 2b).

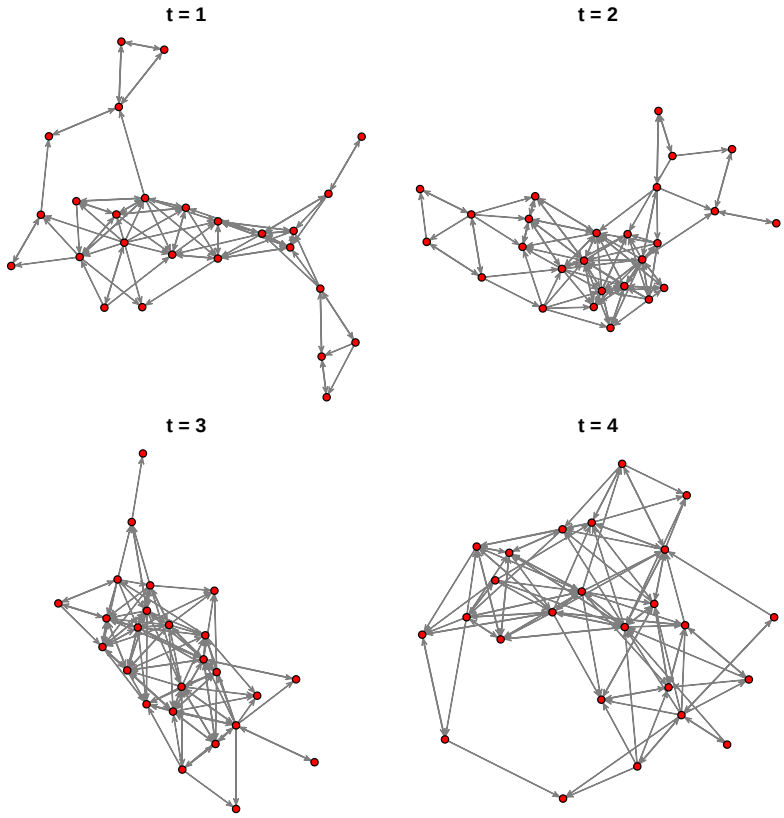

Figure 3: The four directed networks in the Knecht dataset.

```
R> model.2a <- btergm(friendship ~ edges + mutual + ttriple +
+     transitiveties + ctriple + nodeicov("idegsqrt") +
+     nodeicov("odegsqrt") + nodeocov("odegsqrt") +
+     nodeofactor("sex") + nodeifactor("sex") + nodematch("sex") +
+     edgecov(primary), R = 100)
```

In Model 2a, friendship is modeled as a function of several endogenous statistics, like reciprocity (`mutual`) and transitive triplets (`ttriple`), several node-level covariates like `sex` and popularity (`nodeicov("idegsqrt")`), and one dyadic covariate (`edgecov("primary")`), which is constant over time and contains a 1 if a row node and a column node attended the same primary school.

To estimate a TERGM, the dimensions of the networks and matrices need not be identical across time steps or even within time steps across covariates/networks – as long as row and column names or node labels are embedded in the respective objects that are referenced in the model formula. If all objects are appropriately labeled, which we did in Section 5.1 for the current example, the `btergm` function automatically adjusts the dimensions between objects within a single time step. I. e., if a node is present in one network but not in another network at the same time step, the node is automatically removed from the first network to ensure compatibility. To make this adjustment work, however, row and column names must be consistently present.

In the present example, the `primary` covariate is a constant-time covariate, meaning that the

same matrix is used as a covariate at each time step. As the dependent network has 26 nodes at time steps 1 and 2 and 25 nodes at time steps 3 and 4, the `btergm` function internally creates copies of the `primary` matrix for each time step and removes the nodes from the respective copy that are not present at a given time step. This is all done automatically, and the `btergm` function reports the result of this adjustment process when the above command is executed:

```
Initial dimensions of the network and covariates:
                 t=1 t=2 t=3 t=4
friendship (row)  26  26  25  25
friendship (col)  26  26  25  25
primary (row)     26  26  26  26
primary (col)     26  26  26  26

Dimensions differ across networks within time steps.
Trying to auto-adjust the dimensions of the networks. If this fails, provide
conformable matrices or network objects.

Absent nodes:
  label time   object where
1    21    3 networks   row
2    21    3 networks   col
3    21    4 networks   row
4    21    4 networks   col

Number of nodes per time step after adjustment:
                            t=1 t=2 t=3 t=4
maximum deleted nodes (row)   0   0   1   1
maximum deleted nodes (col)   0   0   1   1
remaining rows               26  26  25  25
remaining columns            26  26  25  25

Dimensions of the network and covariates after adjustment:
                 t=1 t=2 t=3 t=4
friendship (row)  26  26  25  25
friendship (col)  26  26  25  25
primary (row)     26  26  25  25
primary (col)     26  26  25  25

Starting pseudolikelihood estimation with 100 bootstrapping replications on a
single computing core...
Done.
```

A TERGM does *not* require the same nodes to be present across different time steps. Only compatibility of dimensions within time steps across covariates is necessary and automatically enforced. To see why different time steps may have different node sets, consider that a TERGM counts subgraph products within time steps, e.g., the number of reciprocal ties at

	Model 2a	Model 2b
edges	−9.18 [−10.50; −8.73]*	−9.54 [−12.42; −8.72]*
mutual	2.96 [2.26; 3.84]*	2.17 [1.84; 2.85]*
ttriple	0.21 [0.10; 0.37]*	0.13 [0.03; 0.24]*
transitiveties	0.37 [0.20; 0.42]*	0.32 [0.29; 0.39]*
ctriple	−0.68 [−0.96; −0.46]*	−0.55 [−0.82; −0.42]*
nodeicov.idegsqrt	1.18 [1.02; 1.52]*	1.28 [1.12; 1.62]*
nodeicov.odegsqrt	−0.28 [−0.65; −0.17]*	−0.13 [−0.31; −0.02]*
nodeocov.odegsqrt	1.20 [1.17; 1.43]*	1.50 [1.38; 2.02]*
nodeofactor.sex.2	0.61 [0.42; 0.83]*	0.53 [0.38; 0.83]*
nodeifactor.sex.2	0.23 [0.10; 0.37]*	0.29 [−0.13; 0.60]
nodematch.sex	1.77 [1.60; 2.10]*	1.49 [1.30; 2.03]*
edgecov.primary[[i]]	1.05 [0.76; 1.42]*	0.42 [−0.28; 0.98]
edgecov.delrecip[[i]]		0.67 [0.33; 1.50]*
edgecov.memory[[i]]		0.78 [0.68; 1.03]*

* 0 outside the confidence interval.

Table 3: TERGM examples on friendship networks in a school class.

each time step. Only if dependencies across at least two time steps are used in a model, the time steps involved in the dependency term must be compatible. However, this is also solved automatically by the `btergm` function because a `memory` term, for instance, is computed by creating an internal copy of the list of dependent networks, shifting the list by a time unit specified by the user (usually one step, as defined by the `lag` argument), using the time-shifted list of networks as a covariate, and auto-adjusting this copy rather than the original dependent network to the other networks. Thus the user needs only to take care of a proper labeling of rows and columns of all networks and matrices at all time steps. Providing such row and column names is not necessary if all objects are already conformable per time step, as in the alliance example (Section 4).

Model 2b adds a memory term and a delayed reciprocity term. Note that these two model terms are internally introduced into the model as time-shifted covariates. Therefore the auto-adjustment procedure reports that nodes are removed from these covariates to match the node sets of the dependent networks:

```
R> model.2b <- btergm(friendship ~ edges + mutual + ttriple +
+    transitiveties + ctriple + nodeicov("idegsqrt") + nodeicov("odegsqrt") +
+    nodeocov("odegsqrt") + nodeofactor("sex") + nodeifactor("sex") +
+    nodematch("sex") + edgecov(primary) + delrecip +
+    memory(type = "stability"), R = 100)

Initial dimensions of the network and covariates:
                 t=2 t=3 t=4
friendship (row)  26  25  25
friendship (col)  26  25  25
primary (row)     26  26  26
primary (col)     26  26  26
delrecip (row)    26  26  25
```

```
delrecip (col)    26  26  25
memory (row)      26  26  25
memory (col)      26  26  25

Dimensions differ across networks within time steps.
Trying to auto-adjust the dimensions of the networks. If this fails, provide
conformable matrices or network objects.

Absent nodes:
  label time   object where
1    21    3 networks   row
2    21    3 networks   col
3    21    4 networks   row
4    21    4 networks   col

Number of nodes per time step after adjustment:
                            t=2 t=3 t=4
maximum deleted nodes (row)   0   1   1
maximum deleted nodes (col)   0   1   1
remaining rows               26  25  25
remaining columns            26  25  25

Dimensions of the network and covariates after adjustment:
                 t=2 t=3 t=4
friendship (row)  26  25  25
friendship (col)  26  25  25
primary (row)     26  25  25
primary (col)     26  25  25
delrecip (row)    26  25  25
delrecip (col)    26  25  25
memory (row)      26  25  25
memory (col)      26  25  25
```

The results of Models 2a and 2b are reported in Table 3. While the inclusion of temporal model terms changes some of the effect sizes, the qualitative interpretation is not changed in this case.

5.4. Temporal dependencies and the treatment of initial networks

Another important detail is that the use of temporal network dependencies leads to a smaller number of usable observations. If temporal dynamics are modeled, at least one time step is lost. Usually, the first time step is dropped from the list of dependent networks. The reason is that the estimation is conditioned on the network at the previous time step(s) each time, and for the first observed network, there is simply no previous time step that could be used. Therefore, estimation of the friendship TERGM with temporal dynamics starts at $t = 2$ and ends at $t = 4$, and the covariates start at $t = 1$ and end at $t = 3$ (see the adjustment report above).

To see this procedure more clearly, it is possible to specify an equivalent model manually by estimating the model explicitly on time steps two to four and including two dyadic covariates that contain the change statistics for delayed reciprocity (see Section 2.2) and dyadic stability (see Section 2.3) based on time steps one to three as lists of matrices. As expected, Model 2b and Model 2c yield identical estimates:

```
R> delrecip <- lapply(friendship, function(x) t(as.matrix(x)))[1:3]
R> stability <- lapply(friendship, function(x) {
+    mat <- as.matrix(x)
+    mat[mat == 0] <- -1
+    return(mat)
+  })[1:3]
R> model.2c <- btergm(friendship[2:4] ~ edges + mutual + ttriple +
+    transitiveties + ctriple + nodeicov("idegsqrt") +
+    nodeicov("odegsqrt") + nodeocov("odegsqrt") +
+    nodeofactor("sex") + nodeifactor("sex") + nodematch("sex") +
+    edgecov(primary) + edgecov(delrecip) + edgecov(stability), R = 100)
```

In a similar way, it is possible to include custom change statistics for other temporal network effects manually. For example, one could design a temporal network statistic for delayed triadic closure over two or three time periods, analytically derive the change statistic (see Section 2.3), include it as a list of matrices in a dyadic covariate, and estimate the model on all but the first one or two time steps to ensure that the covariate and the networks to be modeled are of equal length. Delayed triadic closure has not been hard-coded in package **btergm** because there are many different variants of it one could be theoretically interested in.

5.5. Estimation with `mtergm`

An alternative estimation strategy is MCMC-MLE (see Section 2.6). This is implemented in the `mtergm` function in the **btergm** package. `mtergm` has the same formula syntax as `btergm`, therefore Model 2b can be equivalently specified as follows:

```
R> model.2d <- mtergm(friendship ~ edges + mutual + ttriple +
+    transitiveties + ctriple + nodeicov("idegsqrt") +
+    nodeicov("odegsqrt") + nodeocov("odegsqrt") +
+    nodeofactor("sex") + nodeifactor("sex") + nodematch("sex") +
+    edgecov(primary) + delrecip + memory(type = "stability"),
+    control = control.ergm(MCMC.samplesize = 5000, MCMC.interval = 2000))
```

Note that arguments at the end of the function call are handed over to the `ergm` function, which permits adjusting the MCMC options as documented in the **ergm** package. The last line of the command illustrates this by increasing the MCMC sample size and the MCMC interval. It is also possible to pass on constraints to the `ergm` function, for example the argument `constraints = ~ bd(maxout = 3)` would limit the number of outgoing ties per node to three. Further details can be found in the documentation of the **ergm** package.

Note also that the estimates differ slightly from the `btergm` estimates and that they are likely more accurate with small numbers of networks like in this example.

With 'mtergm' objects, goodness-of-fit assessment, micro-level interpretation, degeneracy checks, and other operations are possible just like with 'btergm' objects.

5.6. Out-of-sample prediction with TERGMs

To draw valid inferences, the model must capture the endogenous properties of the data-generating process well. This can be checked by comparing simulated networks based on the model with the actual observed network(s). However, this bears the risk of overfitting as some of the endogenous properties may be very specific to the case study at hand. To draw inferences that are valid across networks, the gold standard is therefore the assessment of model fit with network data that were not used to create the model in the first place. For example, if a TERGM is applied to six independent networks, one could estimate the model based on four or five of them and try to predict the networks that were left out using the model.

In many settings, however, only one time series is available. For example, there is just one realization of international conflict per year. In such a situation, the best one can do is leave out some time steps when estimating the model, and predict the time steps that were left out using the model. If this works reasonably well, the model seems to fit well out-of-sample.

The **btergm** package offers both within-sample and out-of-sample prediction and goodness-of-fit assessment through the generic `gof` function and its methods. The within-sample case was demonstrated in Section 4. The following example introduces out-of-sample goodness-of-fit assessment with the friendship dataset.

We recompute Model 2b, but we omit the last time step from the estimation. That is, the model is based exclusively on time steps 1–3. As we are using temporal network statistics, the "dependent variable" consists of the networks at time steps 2 and 3 while the lagged temporal covariates consist of time steps 1 and 2:

```
R> model.2e <- btergm(friendship[1:3] ~ edges + mutual + ttriple +
+    transitiveties + ctriple + nodeicov("idegsqrt") +
+    nodeicov("odegsqrt") + nodeocov("odegsqrt") +
+    nodeofactor("sex") + nodeifactor("sex") + nodematch("sex") +
+    edgecov(primary) + delrecip + memory(type = "stability"), R = 100)
```

Next, we can employ the `gof` function to simulate 100 networks from the model and compare them to the observed network at $t = 4$. In the following code block, the argument `target = friendship[[4]]` causes the `gof` function to compare the simulated networks to the omitted and observed fourth time step. To simulate new networks, one needs to supply the covariates for the coefficients, including the temporal statistics. We do this by specifying the formula again and adding index `[3:4]` to the left-hand side of the formula. This computes the relevant covariates for the fourth time step, partly by making use of information contained in the third time step (in the case of the temporal variables). Using the `coef` argument, we tell the `gof` function to use the estimated coefficients from Model 2e. The `statistics` argument contains the functions that are used for comparing the simulated to the observed networks (see `?"gof-terms"`). Finally, the comparison is plotted.

```
R> gof.2e <- gof(model.2e, nsim = 100, target = friendship[[4]],
```

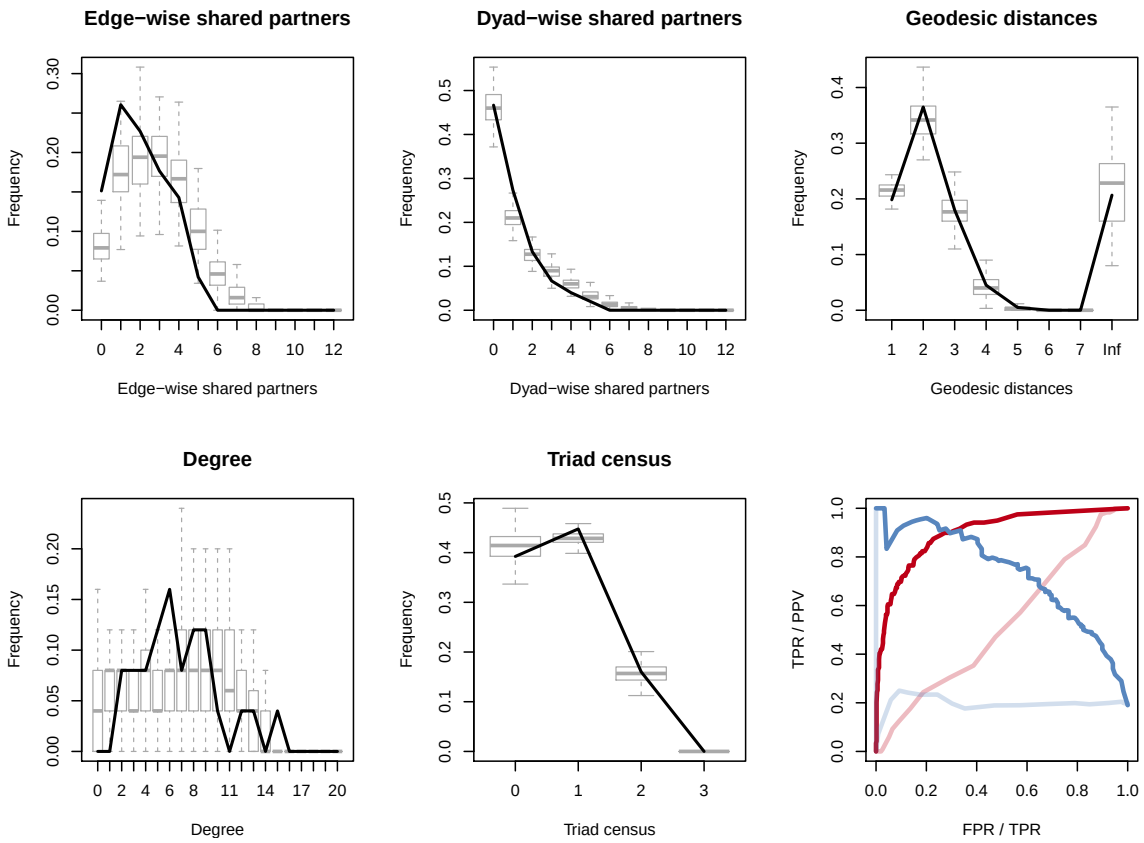

Figure 4: Out-of-sample goodness-of-fit assessment for Model 2b.

```
+    formula = friendship[3:4] ~ edges + mutual + ttriple +
+    transitiveties + ctriple + nodeicov("idegsqrt") +
+    nodeicov("odegsqrt") + nodeocov("odegsqrt") + nodeofactor("sex") +
+    nodeifactor("sex") + nodematch("sex") + edgecov(primary) +
+    delrecip + memory(type = "stability"), coef = coef(model.2e),
+    statistics = c(esp, dsp, geodesic, deg, triad.undirected, rocpr))
R> pdf("gof-2e.pdf", width = 8, height = 6)
R> plot(gof.2e, roc.rgraph = TRUE, pr.rgraph = TRUE)
R> dev.off()
```

The resulting diagram is shown in Figure 4. The first five subfigures compare the distribution of observed and simulated endogenous network statistics, as before (Hunter *et al.* 2008a). Indeed, the simulated networks predict the observed fourth time step reasonably well, with some room left for improvement.

The last subfigure presents receiver operating characteristics (ROC) and precision–recall (PR) curves (Davis and Goadrich 2006; Sing, Sander, Beerenwinkel, and Lengauer 2005). The dark red curve shows the ROC curve for Model 2b while the light red curve is the ROC curve of a random graph of the same size with the same density (i.e., the same model but only with an `edges` term). Model 2b clearly has a much better predictive performance than the null model. The dark blue line shows the PR curve for Model 2b while the light blue curve is the

PR curve of the null model. PR curves are better suited for sparse networks because absent ties are not taken into account. For the Knecht data, however, this is not an issue. The PR curve also shows that the predictive fit is good.

ROC and PR curves can be used to compare different model specifications, also for within-sample goodness-of-fit. To condense the performance into a single measure, the area under the curve (AUC) can be reported for both curves. AUC values are stored in the goodness-of-fit objects and are printed along with other goodness-of-fit measures when the object (here: `gof.2e`) is called. The following code prints the details of the six goodness-of-fit comparisons stored in `gof.2e`, including the AUC values for both curves (and random graphs for comparison), and plots only the ROC and PR curves, respectively. The AUC values can be accessed directly through their respective slots in the sixth object:

```
R> gof.2e
R> plot(gof.2e[[6]])
R> gof.2e[[6]]$auc.pr
```

Finally, the **btergm** package also provides '`btergm`' and '`mtergm`' methods for **statnet**'s generic `simulate` function. The `simulate` function can be used to simulate new networks from a '`btergm`' model or from an '`mtergm`' model. The following command simulates ten new networks based on the coefficients stored in Model 2e and for time step 3. The `index` argument refers to the third item in the list of dependent networks used in the model; if the model included time steps 2 to 4, `index = 3` would refer to time step 4. The resulting ten networks are stored in a list. The `simulate` methods use the '`formula`' method from the `ergm` package internally to create new networks via MCMC (Hunter *et al.* 2008b; Handcock *et al.* 2017).

```
R> nw <- simulate(model.2e, nsim = 10, index = 3)
```

6. Micro-level interpretation of ERGMs and TERGMs

In addition to the estimation and goodness-of-fit assessment of TERGMs, the **btergm** package provides tools for micro-level interpretation of ERGM or TERGM coefficients as described by Desmarais and Cranmer (2012a). The `interpret` function calculates probabilities of subgraph outcomes at the edge, dyad, and block level. For example, what is the probability that node i has a tie to node j (an edge-level question) conditional on the rest of the network? What is the probability that the dyad of i and j has a unidirectional, a reciprocated, or no tie at all (a dyad-level question)? Or what is the probability that node i is connected to nodes j, k, and l, but not to nodes m and n (a block-level question)? The `interpret` function computes these probabilities and facilitates interpretation and comparison of ERGM-type models.

A first simple example examines the dyad state probabilities between nodes 12 and 15 at the last time step, which is the third item in the `dep` list of networks used for Model 2 above:

```
R> interpret(model.2b, type = "dyad", i = 12, j = 15, t = 3)
```

```
$`t = 3`
           j->i = 0  j->i = 1
i->j = 0 0.15567374 0.2965098
i->j = 1 0.03089624 0.5169203
```

In this example, the probability that i and j are mutually tied is approximately 0.52. Note that nodes i and j can be provided as indices (in this case, the 12th row and 15th column of the matrix) or via their node labels.

In a second, more complex example, we would like to analyze the probabilities of same-sex edges over time, separately for both sexes. Provided that both nodes involved in a dyad are male (female), what is the probability that an edge is realized in this dyad at $t = 2$, $t = 3$, and $t = 4$, given the rest of the network and given Model 2b? This will give context to the coefficient of the `nodematch` term in Model 2b, which does not differentiate between sexes or time steps.

To answer the research question, we first need to compile the set of male-male and female-female dyads, apply the interpretation function to each of these dyads at the edge level to compute their probabilities of forming a tie, then sample a number of dyads from this set, and compute the median probability and confidence interval for each sex–time combination. With large numbers of networks or nodes per network, it would make sense to sample from the dyads of interest and then compute their probabilities using the `interpret` function. Here, we have relatively few dyads in total, so we just compute all dyadic probabilities and resample afterwards to obtain the confidence intervals.

The first step is the creation of the set of dyads along with their probabilities (using the `interpret` function along with information on the sex of the dyad):

```
R> dyads <- list()
R> for (t in 2:length(friendship)) {
+    sex <- get.vertex.attribute(friendship[[t]], "sex")
+    mat <- as.matrix(friendship[[t]])
+    for (i in 1:nrow(mat)) {
+      for (j in 1:ncol(mat)) {
+        if (i != j && sex[i] == sex[j]) {
+          dyads[[length(dyads) + 1]] <- c(i, j, t, interpret(model.2b,
+            type = "tie", i = i, j = j, t = t - 1), sex[i])
+        }
+      }
+    }
+  }
R> dyads <- do.call("rbind", dyads)
R> dyads <- as.data.frame(dyads)
R> colnames(dyads) <- c("i", "j", "t", "prob", "sex")
```

In principle, one could plot the probability distributions for the six combinations of sex and time now and compare them. However, we resample dyads in each group in order to obtain the median along with the lower and upper bound of the confidence interval for each group:

```
R> samplesize <- 10000
R> results <- list()
R> for (t in 2:length(friendship)) {
+    for (s in 1:2) {
+      label <- ifelse(s == 1, paste0("f", t), paste0("m", t))
+      d <- dyads[dyads$sex == s & dyads$t == t, ]
```

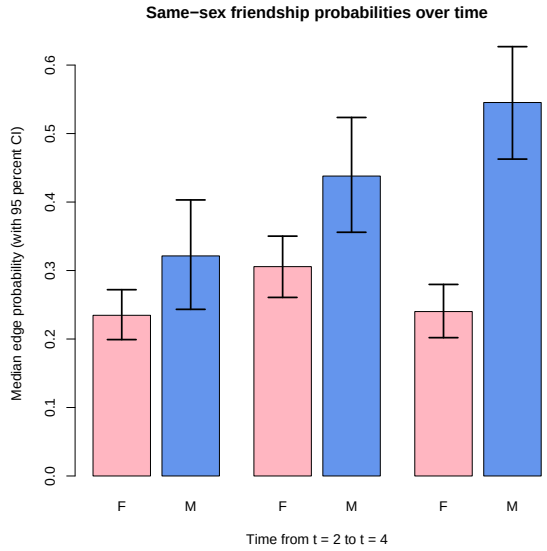

Figure 5: Micro-level interpretation exercise: Same-sex tie probabilities over time by sex.

```
+      n <- nrow(d)
+      means <- sapply(1:samplesize, function(x) {
+        samp <- sample(1:n, n, replace = TRUE)
+        mean(d[samp, ]$prob)
+      })
+      results[[label]] <- means
+    }
+  }
```

After the resampling of probabilities for each group, we compute the median and lower and upper confidence intervals from the resampled probabilities:

```
R> quantiles <- sapply(results, function(x) {
+    return(c(quantile(x, 0.025), quantile(x, 0.5), quantile(x, 0.975)))
+  })
```

Finally, we plot the median per group along with the confidence intervals:

```
R> library("gplots")
R> pdf("interpret.pdf")
R> barplot2(quantiles[2, ], col = c("lightpink", "cornflowerblue"),
+    plot.ci = TRUE, ci.l = quantiles[1, ], ci.u = quantiles[3, ],
+    names = rep(c("F", "M"), 3), space = c(0.2, 0.2, 0.6, 0.2, 0.6, 0.2),
+    xlab = "Time from t = 2 to t = 4",
+    ylab = "Median edge probability (with 95 percent CI)", ci.lwd = 2,
+    main = "Same-sex friendship probabilities over time")
R> dev.off()
```

Figure 5 shows the resulting plot and illustrates that male-male dyads become more likely to be tied over time whereas the probabilities for female-female dyads to be tied is approximately

constant over time. This qualifies the result for the `nodematch("sex")` model term: it might be useful to capture these processes in separate model terms for the two types of dyads. Estimating separate ERGMs for each time step with such separate model terms confirms this.

The **btergm** package also contains the `edgeprob` function, which returns a data frame of all dyadic tie probabilities along with the change statistics of all model terms. As the predicted probabilities are derived analytically with this approach, this is faster than applying the `interpret` function to all dyads separately. It can be used for creating plots of predicted probabilities (e.g., for visualizing interaction terms) as in Czarna *et al.* (2016).

7. Checking for model degeneracy

A simple degeneracy check compares the sufficient statistics of simulated networks to those of the observed networks at each observed time step (Hanneke *et al.* 2010, Section 4). If the statistics of the observed networks are extreme outliers in relation to the distribution of statistics in the simulated networks, the model specification needs to be modified such that the observed data is not an outlier with respect to predictions from the model.

The `checkdegeneracy` function computes target statistics (i.e., the counts for the h statistics included in the model) at each time step for the observed network and a range of simulated networks. The results can be printed to the R console as different quantiles of the simulated distribution versus the observed value, and a `plot` method permits plotting the observed statistic against a histogram of simulated statistics to aid the detection of extreme values.

The results of the degeneracy assessment could show extreme deviations between observed and simulated statistics for one of three reasons. First and foremost, the model is degenerate, which rarely occurs with `btergm`. Second, there is unmodeled temporal variation in parameter values. Third, the MPLE is substantially different from the MLE. This is not the case here as none of the observed target statistics (except delayed reciprocity at $t = 1$) is extreme with respect to the distribution of simulated statistics:

```
R> checkdegeneracy(model.2b, nsim = 1000)

Degeneracy check for network 1:
                           obs    2.5%     25%     50%     75%  97.5%
edges                  117.000  88.000  97.000 103.000 108.000 118.00
mutual                  33.000  21.975  26.000  28.000  30.000  35.00
ttriple                268.000 131.975 173.000 196.000 221.000 278.00
transitiveties          97.000  66.000  79.000  85.000  90.000 103.00
ctriple                 54.000  25.000  36.000  42.000  48.000  62.00
nodeicov.idegsqrt      285.284 217.306 239.613 252.785 265.228 289.13
nodeicov.odegsqrt      244.576 184.043 203.945 215.121 225.947 248.39
nodeocov.odegsqrt      288.245 216.992 241.773 253.712 266.237 290.89
nodeofactor.sex.2       39.000  25.000  31.000  34.000  37.000  43.00
nodeifactor.sex.2       31.000  23.000  27.000  30.000  32.000  38.00
nodematch.sex           95.000  72.000  80.000  84.500  89.000  97.00
edgecov.primary[[i]]    38.000  31.000  36.000  38.000  40.000  45.00
edgecov.delrecip[[i]]   55.000  40.000  45.000  47.000  49.000  54.00
edgecov.memory[[i]]      3.000  -8.000   2.000   7.000  12.000  22.00
```

```
Degeneracy check for network 2:
                          obs   2.5%    25%    50%    75%   97.5%
edges                  133.00 128.00 139.00 145.00 150.00 162.000
mutual                  36.00  30.00  35.00  37.00  40.00  44.025
ttriple                387.00 363.98 442.00 489.00 538.00 638.075
transitiveties         122.00 116.97 129.00 135.00 141.00 154.000
ctriple                 82.00  61.00  79.00  90.00 101.00 125.000
nodeicov.idegsqrt      340.53 330.73 355.40 368.75 382.23 410.919
nodeicov.odegsqrt      292.89 273.89 295.29 306.45 319.47 344.335
nodeocov.odegsqrt      383.34 367.77 397.35 412.98 428.83 461.046
nodeofactor.sex.2       53.00  47.00  54.00  57.00  61.00  69.000
nodeifactor.sex.2       44.00  35.00  41.00  44.00  48.00  53.000
nodematch.sex          102.00  95.00 102.00 106.00 111.00 119.000
edgecov.primary[[i]]    41.00  33.00  37.75  40.00  42.00  45.000
edgecov.delrecip[[i]]   59.00  52.00  56.00  59.00  61.00  66.000
edgecov.memory[[i]]      7.00 -21.00  -9.00  -4.00   1.00  11.000

Degeneracy check for network 3:
                          obs    2.5%     25%     50%     75%  97.5%
edges                  119.000 109.000 118.000 124.000 129.000 139.00
mutual                  33.000  31.000  36.000  39.000  41.000  46.00
ttriple                241.000 236.000 288.000 319.000 354.250 420.00
transitiveties         101.000  94.000 105.000 111.000 118.000 129.00
ctriple                 60.000  58.000  73.000  83.000  92.000 110.00
nodeicov.idegsqrt      285.411 264.668 285.118 299.128 310.387 332.74
nodeicov.odegsqrt      264.019 248.686 267.665 279.984 292.055 312.32
nodeocov.odegsqrt      299.939 275.066 298.810 313.153 324.830 349.10
nodeofactor.sex.2       54.000  49.000  55.000  58.000  61.000  67.00
nodeifactor.sex.2       47.000  41.000  46.000  48.500  51.000  57.00
nodematch.sex           92.000  83.000  92.000  96.000 100.000 109.00
edgecov.primary[[i]]    36.000  31.000  36.000  39.000  41.000  46.00
edgecov.delrecip[[i]]   62.000  60.975  66.000  69.000  72.000  78.00
edgecov.memory[[i]]     31.000  23.000  32.000  36.000  41.000  51.00
```

8. Discussion

The **btergm** package takes its place among two other packages that implement ERGM-like statistical models for longitudinal network data: the **RSiena** (Ripley *et al.* 2017a) and **tergm** (Krivitsky and Handcock 2017) packages.

The main differences between the **btergm** and **tergm** implementations of TERGMs are the estimation procedures and the specifications supported. First, estimation in package **tergm** is done by either MCMC-MLE or a simulated method of moments procedure. These are effective estimation procedures when the networks are modestly sized and/or T is small, but will be noticeably more computationally demanding than `btergm` when dealing with voluminous data. The first example provided in Section 4, for instance, is already too large to compute

by MCMC-MLE, making the bootstrap MPLE approach in the **btergm** package necessary. Second, the specifications supported in the **tergm** package are based on the separate edge loss and edge creation functions proposed by Krivitsky and Handcock (2014). Researchers may be interested in these specifications, but if not, the specification constraints in the **tergm** package may be seen as a limitation. An example from the study of international conflict may illustrate this point: it may be plausible that a country i starts attacking another country j as soon as a third country k's conflict with j is dissolved in order to keep up the pressure. However, if k does not dissolve its war with j, there is no reason for i to intervene because it can effectively free-ride. In this example, both the formation and dissolution may take place in close temporal order. Thus it may make sense in some applications to model both processes jointly, and the **xergm** package permits such joint modeling of temporal processes. In other situations, however, a separable TERGM as implemented in the **tergm** package may be theoretically plausible.

In terms of the stochastic actor-oriented models implemented in package **RSiena**, the differences with TERGM are numerous and interested readers are directed to Leifeld and Cranmer (2014) and Desmarais and Cranmer (2012a) for a detailed discussion as well as suggestions for empirical comparison between the two models.

There are many additional features available in the **btergm** package that we omit here to keep the exposition short, such as support for TERGMs for bipartite networks. There are several additional extensions in progress or planned for the **xergm** package. The **tnam** package (Leifeld and Cranmer 2017), another part of the **xergm** suite of packages, implements a dynamic model for vertex attributes which we refer to as "temporal network autocorrelation model" (TNAM) and which can be thought of as analogous to the vertex behavior components of the stochastic actor-oriented model implemented in package **RSiena** (Ripley *et al.* 2017a). We are actively working to implement the generalized ERGM for networks with weighted ties developed by Desmarais and Cranmer (2012b) in the **GERGM** package (Denny, Wilson, Cranmer, Desmarais, and Bhamidi 2017).

Acknowledgments

PL acknowledges that parts of this work were conducted at the Swiss Federal Institute of Aquatic Science and Technology (Eawag), the University of Bern, and the University of Konstanz. The Zukunftskolleg at the University of Konstanz provided research funding for the first author. SJC gratefully acknowledges the support of the National Science Foundation (SES-1357622, SES-1461493, and SES-1514750) and the Alexander von Humboldt Foundation. BD acknowledges that this work was supported in part by National Science Foundation grants SES-1558661, SES-1619644, SES-1637089, and CISE-1320219. Any opinions, findings, and conclusions or recommendations are those of the authors and do not necessarily reflect those of the sponsors.

References

Almquist ZW, Butts CT (2013). "Dynamic Network Logistic Regression: A Logistic Choice Analysis of Inter- And Intra-Group Blog Citation Dynamics in the 2004 US Presidential Election." *Political Analysis*, **21**(4), 430–448. doi:10.1093/pan/mpt016.

Arnold BC, Strauss D (1991). "Pseudolikelihood Estimation: Some Examples." *Sankhyā: The Indian Journal of Statistics B*, **53**(2), 233–243.

Berardo R, Scholz JT (2010). "Self-Organizing Policy Networks: Risk, Partner Selection, and Cooperation in Estuaries." *American Journal of Political Science*, **54**(3), 632–649. doi:10.1111/j.1540-5907.2010.00451.x.

Butts CT (2008). "**network**: A Package for Managing Relational Data in R." *Journal of Statistical Software*, **24**(2), 1–36. doi:10.18637/jss.v024.i02.

Butts CT (2015). ***network**: Classes for Relational Data*. The Statnet Project (http://statnet.org/). R package version 1.13.0, URL https://CRAN.R-project.org/package=network.

Canty A, Ripley B (2017). ***boot**: Bootstrap Functions*. R package version 1.3-20, URL https://CRAN.R-project.org/package=boot.

Cranmer SJ, Desmarais BA (2011). "Inferential Network Analysis with Exponential Random Graph Models." *Political Analysis*, **19**(1), 66–86. doi:10.1093/pan/mpq037.

Cranmer SJ, Desmarais BA, Kirkland JH (2012a). "Toward a Network Theory of Alliance Formation." *International Interactions*, **38**(3), 295–324. doi:10.1080/03050629.2012.677741.

Cranmer SJ, Desmarais BA, Menninga EJ (2012b). "Complex Dependencies in the Alliance Network." *Conflict Management and Peace Science*, **29**(3), 279–313. doi:10.1177/0738894212443446.

Cranmer SJ, Heinrich T, Desmarais BA (2014). "Reciprocity and the Structural Determinants of the International Sanctions Network." *Social Networks*, **36**(January), 5–22. doi:10.1016/j.socnet.2013.01.001.

Cranmer SJ, Leifeld P, McClurg SD, Rolfe M (2017). "Navigating the Range of Statistical Tools for Inferential Network Analysis." *American Journal of Political Science*, **61**(1), 237–251. doi:10.1111/ajps.12263.

Czarna AZ, Leifeld P, Śmieja M, Dufner M, Salovey P (2016). "Do Narcissism and Emotional Intelligence Win Us Friends? Modeling Dynamics of Peer Popularity Using Inferential Network Analysis." *Personality and Social Psychology Bulletin*, **42**(11), 1588–1599. doi:10.1177/0146167216666265.

Davis J, Goadrich M (2006). "The Relationship between Precision-Recall and ROC Curves." In *Proceedings of the 23rd International Conference on Machine Learning*, pp. 233–240. ACM.

Denny MJ, Wilson JD, Cranmer SJ, Desmarais BA, Bhamidi S (2017). ***GERGM**: Estimation and Fit Diagnostics for Generalized Exponential Random Graph Models*. R package version 0.11.2, URL https://CRAN.R-project.org/package=GERGM.

Desmarais BA, Cranmer SJ (2010). "Consistent Confidence Intervals for Maximum Pseudolikelihood Estimators." In *Proceedings of the Neural Information Processing Systems 2010 Workshop on Computational Social Science and the Wisdom of Crowds*.

Desmarais BA, Cranmer SJ (2012a). "Micro-Level Interpretation of Exponential Random Graph Models with Application to Estuary Networks." *The Policy Studies Journal*, **40**(3), 402–434. doi:10.1111/j.1541-0072.2012.00459.x.

Desmarais BA, Cranmer SJ (2012b). "Statistical Inference for Valued-Edge Networks: The Generalized Exponential Random Graph Model." *PLoS ONE*, **7**(1), e30136. doi:10.1371/journal.pone.0030136.

Desmarais BA, Cranmer SJ (2012c). "Statistical Mechanics of Networks: Estimation and Uncertainty." *Physica A: Statistical Mechanics and its Applications*, **391**(4), 1865–1876. doi:10.1016/j.physa.2011.10.018.

Goodreau SM, Handcock MS, Hunter DR, Butts CT, Morris M (2008a). "A **statnet** Tutorial." *Journal of Statistical Software*, **24**(9), 1–26. doi:10.18637/jss.v024.i09.

Goodreau SM, Kitts J, Morris M (2008b). "Birds of a Feather, or Friend of a Friend? Using Exponential Random Graph Models to Investigate Adolescent Social Networks." *Demography*, **46**(1), 103–125. doi:10.1353/dem.0.0045.

Handcock MS, Gile KJ (2010). "Modeling Social Networks from Sampled Data." *The Annals of Applied Statistics*, **4**(1), 5–25. doi:10.1214/08-aoas221.

Handcock MS, Hunter DR, Butts CT, Goodreau SM, Krivitsky PN, Bender-deMoll S, Morris M (2016). ***statnet**: Software Tools for the Statistical Analysis of Network Data*. The Statnet Project (http://www.statnet.org/). R package version 2016.9, URL https://CRAN.R-project.org/package=statnet.

Handcock MS, Hunter DR, Butts CT, Goodreau SM, Krivitsky PN, Morris M (2015). ***ergm.userterms**: User-Specified Terms for the **statnet** Suite of Packages*. The Statnet Project (http://statnet.org/). R package version 3.1.1, URL https://CRAN.R-project.org/package=ergm.userterms.

Handcock MS, Hunter DR, Butts CT, Goodreau SM, Krivitsky PN, Morris M (2017). ***ergm**: Fit, Simulate and Diagnose Exponential-Family Models for Networks*. The Statnet Project (http://www.statnet.org/). R package version 3.8.0, URL https://CRAN.R-project.org/package=ergm.

Handcock MS, Hunter DR, Butts CT, Goodreau SM, Morris M (2008). "**statnet**: Software Tools for the Representation, Visualization, Analysis and Simulation of Network Data." *Journal of Statistical Software*, **24**(1), 1–11. doi:10.18637/jss.v024.i01.

Hanneke S, Fu W, Xing EP (2010). "Discrete Temporal Models of Social Networks." *Electronic Journal of Statistics*, **4**, 585–605. doi:10.1214/09-ejs548.

Hunter DR, Goodreau SM, Handcock MS (2008a). "Goodness of Fit of Social Network Models." *Journal of the American Statistical Association*, **103**(481), 248–258. doi:10.1198/016214507000000446.

Hunter DR, Goodreau SM, Handcock MS (2013). "**ergm.userterms**: A Template Package for Extending **statnet**." *Journal of Statistical Software*, **52**(2), 1–25. doi:10.18637/jss.v052.i02.

Hunter DR, Handcock MS, Butts CT, Goodreau SM, Morris M (2008b). "**ergm**: A Package to Fit, Simulate and Diagnose Exponential-Family Models for Networks." *Journal of Statistical Software*, **24**(3), 1–29. doi:10.18637/jss.v024.i03.

Hyvarinen A (2006). "Consistency of Pseudolikelihood Estimation of Fully Visible Boltzmann Machines." *Neural Computation*, **18**(10), 2283–2292. doi:10.1162/neco.2006.18.10.2283.

Ingold KM, Leifeld P (2016). "Structural and Institutional Determinants of Influence Reputation: A Comparison of Collaborative and Adversarial Policy Networks in Decision Making and Implementation." *Journal of Public Administration Research and Theory*, **26**(1), 1–18.

Knecht A (2006). "Networks and Actor Attributes in Early Adolescence." *ICS Codebook 61*, The Netherlands Research School ICS, Department of Sociology, Utrecht University, Utrecht.

Knecht A (2008). *Friendship Selection and Friends' Influence. Dynamics of Networks and Actor Attributes in Early Adolescence*. PhD Dissertation, University of Utrecht, Utrecht. URL https://dspace.library.uu.nl/handle/1874/25950.

Knecht A, Snijders TAB, Baerveldt C, Steglich CEG, Raub W (2010). "Friendship and Delinquency: Selection and Influence Processes in Early Adolescence." *Social Development*, **19**(3), 494–514. doi:10.1111/j.1467-9507.2009.00564.x.

Koskinen JH, Robins GL, Wang P, Pattison PE (2013). "Bayesian Analysis for Partially Observed Network Data, Missing Ties, Attributes and Actors." *Social Networks*, **35**(4), 514–527. doi:10.1016/j.socnet.2013.07.003.

Krivitsky PN, Handcock MS (2014). "A Separable Model for Dynamic Networks." *Journal of the Royal Statistical Society B*, **76**(1), 29–46. doi:10.1111/rssb.12014.

Krivitsky PN, Handcock MS (2017). ***tergm**: Fit, Simulate and Diagnose Models for Network Evolution Based on Exponential-Family Random Graph Models*. The Statnet Project (http://www.statnet.org/). R package version 3.4.1, URL https://CRAN.R-project.org/package=tergm.

Lahiri SN (1992). "On Bootstrapping M-Estimators." *Sankhyā: The Indian Journal of Statistics A*, **54**(2), 157–170.

Leifeld P (2013). "**texreg**: Conversion of Statistical Model Output in R to LaTeX and HTML Tables." *Journal of Statistical Software*, **55**(8), 1–24. doi:10.18637/jss.v055.i08.

Leifeld P (2017a). ***texreg**: Conversion of R Regression Output to LaTeX or HTML Tables*. R package version 1.36.23, URL https://CRAN.R-project.org/package=texreg.

Leifeld P (2017b). ***xergm.common**: Common Infrastructure for Extensions of Exponential Random Graph Models*. R package version 1.7.7, URL https://CRAN.R-project.org/package=xergm.common.

Leifeld P, Cranmer SJ (2014). "A Theoretical and Empirical Comparison of the Temporal Exponential Random Graph Model and the Stochastic Actor-Oriented Model." Paper

presented at the 7th Political Networks Conference, McGill University, Montreal, Canada, May 30. URL http://arxiv.org/abs/1506.06696.

Leifeld P, Cranmer SJ (2017). ***tnam**: Temporal Network Autocorrelation Models*. R package version 1.6.5, URL https://CRAN.R-project.org/package=tnam.

Leifeld P, Cranmer SJ, Desmarais BA (2017a). ***btergm**: Extensions of Exponential Random Graph Models*. R package version 1.9.0, URL https://CRAN.R-project.org/package=btergm.

Leifeld P, Cranmer SJ, Desmarais BA (2017b). ***xergm**: Extensions of Exponential Random Graph Models*. R package version 1.8.2, URL https://CRAN.R-project.org/package=xergm.

Lusher D, Koskinen J, Robins G (2013). *Exponential Random Graph Models for Social Networks*. Cambridge University Press, New York.

Morris M, Handcock MS, Hunter DR (2008). "Specification of Exponential-Family Random Graph Models: Terms and Computational Aspects." *Journal of Statistical Software*, **24**(4), 1–24. doi:10.18637/jss.v024.i04.

R Core Team (2017). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Ripley R, Boitmanis K, Snijders TAB, Schoenenberger F (2017a). ***RSiena**: Simulation Investigation for Empirical Network Analysis*. R package version 1.2-3, URL https://CRAN.R-project.org/package=RSiena.

Ripley R, Snijders TA, Boda Z, Vörös A, Preciado P (2017b). *Manual for **SIENA** Version 4.0*. University of Oxford, Department of Statistics, Oxford. URL http://www.stats.ox.ac.uk/siena/.

Robins G, Pattison P (2001). "Random Graph Models for Temporal Processes in Social Networks." *Journal of Mathematical Sociology*, **25**(1), 5–41. doi:10.1080/0022250x.2001.9990243.

Robins G, Pattison P, Woolcock J (2004). "Missing Data in Networks: Exponential Random Graph (p^*) Models for Networks with Non-Respondents." *Social Networks*, **26**(3), 257–283. doi:10.1016/j.socnet.2004.05.001.

Sing T, Sander O, Beerenwinkel N, Lengauer T (2005). "**ROCR**: Visualizing Classifier Performance in R." *Bioinformatics*, **21**(20), 3940–3941. doi:10.1093/bioinformatics/bti623.

Snijders TAB, Steglich CEG, Van de Bunt GG (2010). "Introduction to Actor-Based Models for Network Dynamics." *Social Networks*, **32**(January), 44–60. doi:10.1016/j.socnet.2009.02.004.

Steglich CEG, Knecht A (2009). "Die statistische Analyse dynamischer Netzwerkdaten." In C Stegbauer, R Häussling (eds.), *Handbuch der Netzwerkforschung*. VS Verlag für Sozialwissenschaften, Wiesbaden.

Strauss D, Ikeda M (1990). "Pseudolikelihood Estimation for Social Networks." *Journal of the American Statistical Association*, **85**(409), 204–212. doi:10.2307/2289546.

Van Duijn MAJ, Gile KJ, Handcock MS (2009). "A Framework for the Comparison of Maximum Pseudo-Likelihood and Maximum Likelihood Estimation of Exponential Family Random Graph Models." *Social Networks*, **31**(1), 52–62. doi:10.1016/j.socnet.2008.10.003.

Yaveroglu ON, Fitzhugh SM, Kurant M, Markopoulou A, Przulj N, Butts CT (2015). ***ergm.graphlets**: A Package for ERG Modeling Based on Graphlet Statistics*. R package version 1.0.3, URL https://CRAN.R-project.org/package=ergm.graphlets.

Affiliation:

Philip Leifeld
University of Glasgow
School of Social and Political Sciences
Adam Smith Building, 40 Bute Gardens
Glasgow, G12 8RT, Scotland, United Kingdom
E-mail: philip.leifeld@glasgow.ac.uk
URL: http://www.philipleifeld.com

Skyler J. Cranmer
The Ohio State University
Department of Political Science
2032 Derby Hall, 154 N Oval Mall
Columbus OH 43210, United States of America
E-mail: cranmer.12@osu.edu
URL: http://www.skylercranmer.net

Bruce A. Desmarais
Pennsylvania State University
Department of Political Science
230 Pond Lab
University Park, PA 16801, United States of America
E-mail: bdesmarais@psu.edu
URL: http://sites.psu.edu/desmaraisgroup/

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
January 2018, Volume 83, Issue 6 *Submitted:* 2014-11-10
doi:10.18637/jss.v083.i06 *Accepted:* 2017-04-05