*Journal of Statistical Software*

*October 2014, Volume 61, Issue 2.* *http://www.jstatsoft.org/*

# Bergm: Bayesian Exponential Random Graphs in R

**Alberto Caimo**
University of Lugano

**Nial Friel**
University College Dublin

**Abstract**

In this paper we describe the main features of the **Bergm** package for the open-source R software which provides a comprehensive framework for Bayesian analysis of exponential random graph models: tools for parameter estimation, model selection and goodness-of-fit diagnostics. We illustrate the capabilities of this package describing the algorithms through a tutorial analysis of three network datasets.

*Keywords*: exponential random graph models, Bayesian inference, Bayesian model selection, Markov chain Monte Carlo.

## 1. Introduction

Interest in statistical network analysis has grown massively in recent decades and its perspective and methods are now widely used in many scientific areas which involve the study of various types of networks for representing structure in many complex relational systems such as social relationships, information flows, protein interactions, etc. (see Salter-Townshend, White, Gollini, and Murphy 2012 for a recent review of statistical network models).

Social network theory is based on the study of social relations between actors so as to understand the formation of social structures by the analysis of basic local relations. Statistical models have started to play an increasingly important role because they give the possibility to explain the complexity of social behaviour and to investigate issues on how the global features of an observed network may be related to local network structures. The observed network is assumed to be generated by local social processes which depend on the self-organizing dyadic relations between actors. The crucial challenge for statistical models in social network theory is to capture and describe the dependency giving rise to network global topology allowing inference about whether certain local structures are more common than expected.

Exponential random graph models (ERGMs; Frank and Strauss 1986; Wasserman and Pattison 1996; Robins, Pattison, Kalish, and Lusher 2007) are one of the most important family of

models conceived to capture the complex dependence structure of an observed network allowing a reasonable interpretation of the underlying process which is supposed to have produced these structural properties. The dependence hypothesis at the basis of these models is that the connections between actors (edges) self-organize into small structures called configurations or network statistics. These are classical graph-theoretic structures such as degrees, cycles, etc. which can be directly incorporated in ERGMs as sufficient statistics with corresponding parameters measuring their importance in the observed network. The computational intractability of these models is the main barrier to estimation.

The Bayesian approaches for ERGMs developed by Caimo and Friel (2011) and Caimo and Friel (2013) represent one of the first complete Bayesian frameworks for these models. The doubly-intractable posterior is estimated by the use of an approximate exchange algorithm with adaptive direction sampling. This approach has proven to be effective to improve mixing and local moves on the typically thin and high posterior density region. In fact fast convergence occurs even when the algorithm starts from degenerate parameter values. This approach exhibits better performance and convergence properties compared to classical non-Bayesian methods.

The recent progress made in the framework of network analysis has been possible thanks to the development and implementation of software able to perform computational intensive tasks. For this reason, the development of software has always represented an essential aspect of the research activity in this area.

The **Bergm** package (Caimo and Friel 2014) for R (R Core Team 2014) implements Bayesian analysis for ERGMs using the methods described by Caimo and Friel (2011) and Caimo and Friel (2013). The package provides a comprehensive framework for Bayesian inference and model selection using Markov chain Monte Carlo (MCMC) algorithms. It can also supply graphical Bayesian goodness-of-fit procedures that address the issue of model adequacy. Although computationally intensive, the package is simple to use and represents an attractive way of analyzing network data as it offers the advantange of a complete probabilistic treatment of uncertainty. **Bergm** is based on the **ergm** package (Hunter, Handcock, Butts, Goodreau, and Morris 2008b) which is part of the **statnet** suite of packages (Handcock, Hunter, Butts, Goodreau, and Morris 2007) and therefore it makes use of the same model set-up and network simulation algorithms. The **ergm** and **Bergm** packages complement each other in the sense that **ergm** implements maximum likelihood-based inference whereas **Bergm** implements Bayesian inference. The **Bergm** package has been continually improved in terms of speed performance over the last two years and one of the purposes of this paper is to highlight these improvements. We feel that this package now offers the end-user a feasible option for carrying out Bayesian inference for ERGMs.

Three network datasets will be used throughout this tutorial for illustrative purposes: the first is the Kapferer tailor shop dataset (Kapferer 1972) whose directed edges represent work interactions in a tailor shop in Zambia (then Northern Rhodesia) and nodal attributes refer to the job status. The second network is Zachary's karate club (Zachary 1977) which represents the undirected social network graph of friendships between 34 members of a karate club at an US university in the 1970s. The third is an excerpt of 50 girls from the Teenage Friends and Lifestyle Study dataset. Figure 2 displays the graphs of the first two networks, Figure 10 displays the graphs of the Teenage Friends and Lifestyle Study network. The exact R code used to produce these plots is given in Appendix A.

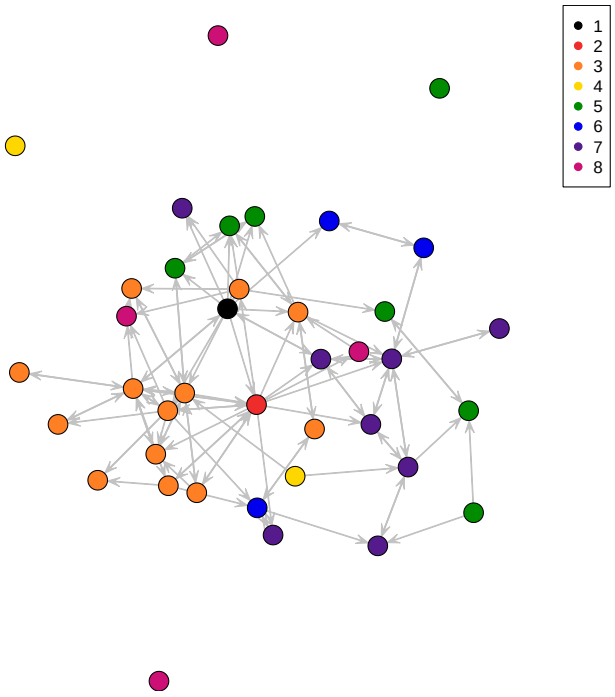


Figure 1: Kapferer tailor shop directed graph. Nodal covariates consist of employees' occupational categories: 1 = head tailor, 2 = cutter, 3 = line 1 tailor, 4 = button machined, 5 = line 3 tailor, 6 = ironer, 7 = cotton boy, 8 = line 2 tailor.

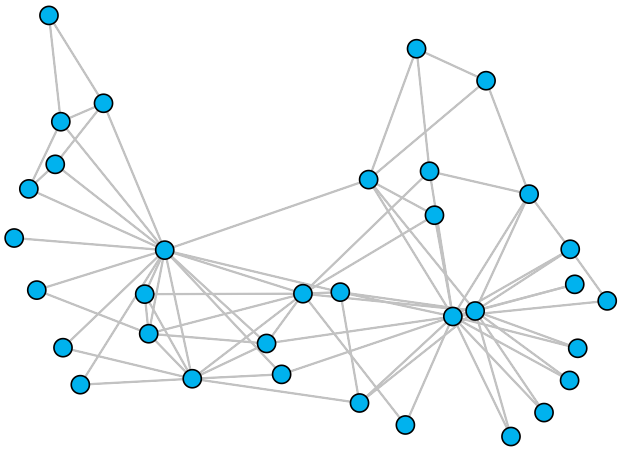

Figure 2: Zachary's karate club undirected graph.

In this paper we describe how to install and load **Bergm** (Section 2) providing a brief summary of what Bayesian ERGMs are (Section 3). Sections 4, 5, and 6 overview the algorithms and the functions used to produce posterior estimates for the parameters, Bayesian goodness-of-fit procedures and model selection respectively. Two examples are developed in Sections 4.3 and 6.1. This paper does not provide an exhaustive description of all the functionality and options available, and more information about the commands and methods mentioned are available through the R help system within the package.

## 2. Getting Bergm

The **Bergm** package can be obtained and loaded in R using the following commands:

```
R> install.packages("Bergm")
R> library("Bergm")
```

Since **Bergm** depends on **ergm** (Hunter *et al.* 2008b) (which in turn depends on **network**; Butts 2008), **coda** (Plummer, Best, Cowles, and Vines 2006), and **mvtnorm** (Genz, Bretz, Miwa, Mi, Leisch, Scheipl, and Hothorn 2014). Loading the package will automatically load all the dependencies. All of these packages are available on the Comprehensive R Archive Network (CRAN) at http://CRAN.R-project.org/.

The results presented in this paper have been obtained using R version 3.1.1 on a Mac using **Bergm** version 3.0; **ergm** version 3.1.3; **network** version 1.10.2; **coda** version 0.16-1; and **mvtnorm** version 1.0-0.

## 3. Bayesian exponential random graphs

The Bayesian approach to statistical problems is probabilistic. Inference is based on the posterior distribution which is the conditional probability of the unknown quantities given the observed ones. The posterior distribution extracts the information in the data and provides a complete summary of the uncertainty about the unknowns.

In the ERGM context (see Wasserman and Pattison 1996 and Robins *et al.* 2007), the purpose of Bayesian inference is to learn about the posterior distribution of the model parameters $\boldsymbol{\theta}$ of an observed graph $\boldsymbol{y}$ on $n$ nodes:

$$p(\boldsymbol{\theta}|\boldsymbol{y}) = \frac{p(\boldsymbol{y}|\boldsymbol{\theta})\, p(\boldsymbol{\theta})}{p(\boldsymbol{y})} = \frac{\exp\{\boldsymbol{\theta}^\top s(\boldsymbol{y})\}}{z(\boldsymbol{\theta})}\, \frac{p(\boldsymbol{\theta})}{p(\boldsymbol{y})}, \tag{1}$$

where $s(\boldsymbol{y})$ is a known vector of sufficient network statistics (Figure 3; Morris, Handcock, and Hunter 2008), $p(\boldsymbol{\theta})$ is a prior distribution placed on $\boldsymbol{\theta}$, $z(\boldsymbol{\theta})$ is the likelihood normalizing constant, and $p(\boldsymbol{y})$ is the model evidence. Equation 1 provides a probabilistic statement about how likely parameter values are after observing the data $\boldsymbol{y}$. The likelihood $p(\boldsymbol{y}|\boldsymbol{\theta})$ is translated into a proper probability distribution that can be summarized by computing expected values, standard deviations, quantiles, etc.

Unfortunately the posterior distribution (Equation 1) is doubly-intractable as both $z(\boldsymbol{\theta})$ and $p(\boldsymbol{y})$ cannot be evaluated analytically (Koskinen 2004). This makes the use of standard MCMC procedures infeasible.

In order to carry out Bayesian inference for ERGMs, the **Bergm** package makes use of a combination of Bayesian algorithms and MCMC techniques. The exchange algorithm circumvents the problem of computing the normalizing constants of the ERGM likelihoods, while the use of multiple chains interacting with each others (population MCMC approach) by means of adaptive direction sampling is able to speed up the computations and improve chain mixing quite significantly.

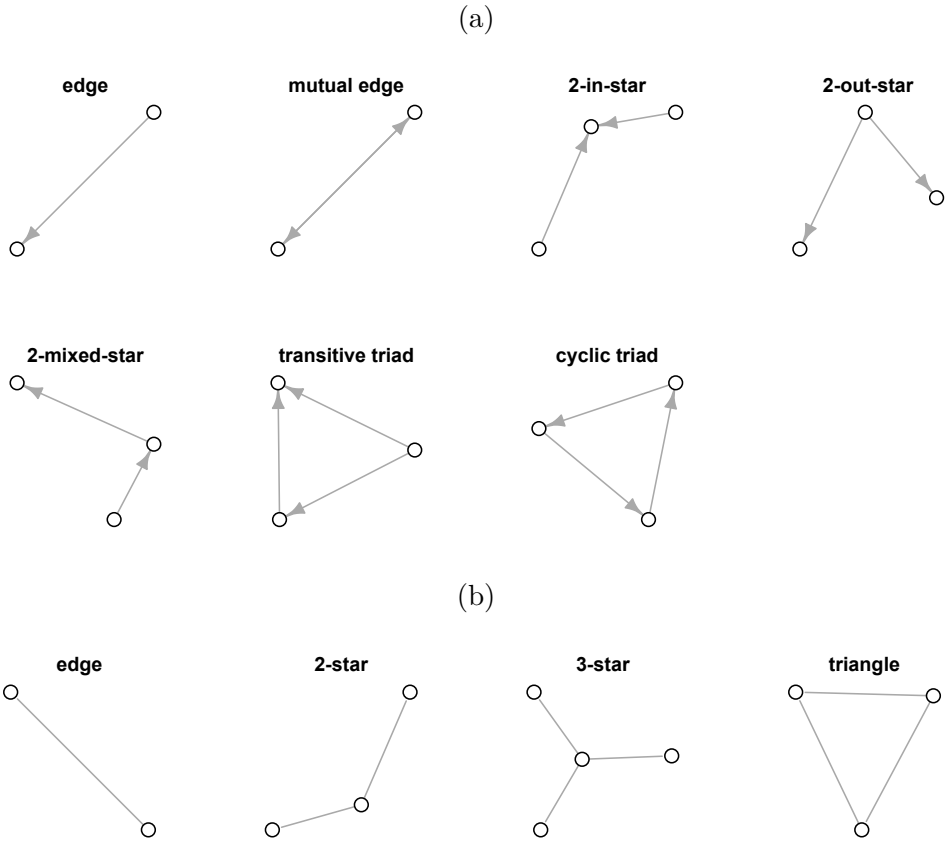


Figure 3: Some of the most used configurations for directed (a) and undirected (b) graphs.

## 4. Bayesian parameter estimation

In order to approximate the posterior distribution $p(\boldsymbol{\theta}|\boldsymbol{y})$, the **Bergm** package uses the exchange algorithm described in Section 4.1 of Caimo and Friel (2011) to sample from the following distribution:

$$p(\boldsymbol{\theta}', \boldsymbol{y}', \boldsymbol{\theta}|\boldsymbol{y}) \propto p(\boldsymbol{y}|\boldsymbol{\theta})p(\boldsymbol{\theta})\epsilon(\boldsymbol{\theta}'|\boldsymbol{\theta})p(\boldsymbol{y}'|\boldsymbol{\theta}'),$$

where $p(\boldsymbol{y}'|\boldsymbol{\theta}')$ is the likelihood on which the simulated data $\boldsymbol{y}'$ are defined and belongs to the same exponential family of densities as $p(\boldsymbol{y}|\boldsymbol{\theta})$, $\epsilon(\boldsymbol{\theta}'|\boldsymbol{\theta})$ is any arbitrary proposal distribution for the augmented variable $\boldsymbol{\theta}'$. As we will see in the next section, this proposal distribution is set to be a normal centered at $\boldsymbol{\theta}$.

At each MCMC iteration, the exchange algorithm consists of a Gibbs update of $\boldsymbol{\theta}'$ followed by a Gibbs update of $\boldsymbol{y}'$, which is drawn from $p(\cdot|\boldsymbol{\theta}')$ via an MCMC algorithm (Hunter *et al.* 2008b). Then a deterministic exchange or swap from the current state $\boldsymbol{\theta}$ to the proposed new parameter $\boldsymbol{\theta}'$ is performed. This deterministic proposal is accepted with probability:

$$\min\left(1, \frac{q_{\boldsymbol{\theta}}(\boldsymbol{y}')p(\boldsymbol{\theta}')\epsilon(\boldsymbol{\theta}|\boldsymbol{\theta}')q_{\boldsymbol{\theta}'}(\boldsymbol{y})}{q_{\boldsymbol{\theta}}(\boldsymbol{y})p(\boldsymbol{\theta})\epsilon(\boldsymbol{\theta}'|\boldsymbol{\theta})q_{\boldsymbol{\theta}'}(\boldsymbol{y}')} \times \frac{z(\boldsymbol{\theta})z(\boldsymbol{\theta}')}{z(\boldsymbol{\theta})z(\boldsymbol{\theta}')}\right),$$

where $q_{\boldsymbol{\theta}}$ and $q_{\boldsymbol{\theta}'}$ indicates the unnormalized likelihoods with parameters $\boldsymbol{\theta}$ and $\boldsymbol{\theta}'$, respectively. Notice that all the normalizing constants cancel above and below in the fraction above, in this way avoiding the need to calculate the intractable normalizing constant.

The exchange algorithm is implemented by the `bergm` function in the following way:

`for` $i = 1, \ldots, N$

1. `generate` $\boldsymbol{\theta}'$ `from` $\epsilon(\cdot|\boldsymbol{\theta})$
2. `simulate` $\boldsymbol{y}'$ `from` $p(\cdot|\boldsymbol{\theta}')$
3. `update` $\boldsymbol{\theta} \rightarrow \boldsymbol{\theta}'$ `with the log of the probability`

$$\min\left(0,\ \left[\boldsymbol{\theta}-\boldsymbol{\theta}'\right]^{\top}\left[s(\boldsymbol{y}')-s(\boldsymbol{y})\right]+\log\left[\frac{p(\boldsymbol{\theta}')}{p(\boldsymbol{\theta})}\right]\right)$$

`end for`

where $s(\boldsymbol{y})$ is the observed vector of network statistics and $s(\boldsymbol{y}')$ is the simulated vector of network statistics.

### 4.1. Block-update sampler

Step 1 of the algorithm consists in generating $\boldsymbol{\theta}'$ from some proposal distribution within each iteration. **Bergm** uses a block-update sampler with normal proposal to simultaneously update the parameter values in the MCMC chain:

$$\epsilon(\boldsymbol{\theta}'|\boldsymbol{\theta}) \sim \mathcal{N}(\boldsymbol{\theta}, \boldsymbol{\Sigma}_{\epsilon}). \tag{2}$$

Typically, tuning the parameter $\boldsymbol{\Sigma}_{\epsilon}$ of the proposal distribution $\epsilon$ from which $\boldsymbol{\theta}'$ is drawn represents the crucial part of the algorithm since a poor tuning of the proposal parameter $\boldsymbol{\Sigma}_{\epsilon}$ can slow down the chain's mixing rate and therefore the algorithm can take a very long time to converge to the stationary posterior density. By default $\boldsymbol{\Sigma}_{\epsilon}$ is set to a diagonal matrix with every diagonal entry equal to 0.0025.

### 4.2. Parallel adaptive direction sampler

In order to improve mixing a parallel adaptive direction sampler (ADS; Gilks, Roberts, and George 1994; Roberts and Gilks 1994) is considered: at the $i$th iteration of the algorithm we have a collection of $H$ different chains interacting with one another. By construction, the state space consists of $\{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_H\}$ with target distribution $p(\boldsymbol{\theta}_1|\boldsymbol{y}) \otimes \cdots \otimes p(\boldsymbol{\theta}_H|\boldsymbol{y})$. A parallel ADS move consists of generating a new value $\boldsymbol{\theta}'_h$ from the difference of two parameters $\boldsymbol{\theta}_{h_1}$ and $\boldsymbol{\theta}_{h_2}$ (randomly selected from other chains) multiplied by a scalar term $\gamma$ which is called parallel ADS move factor plus a random term $\epsilon$ called parallel ADS move parameter (Figure 4) which is equivalent to the block-update sampler defined in Equation 2. The algorithm can be summarized as follows:

`for` $i = 1, \ldots, N$

`for` $h = 1, \ldots, H$

1. `generate` $h_1$ `and` $h_2$ `such that` $h_1 \neq h_2 \neq h$

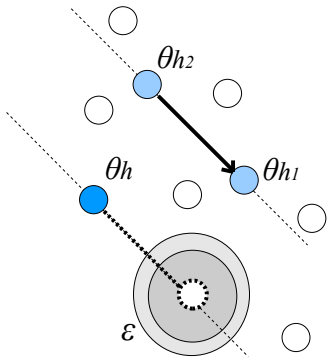


Figure 4: The parallel ADS move of the current state (darker blue dot) consists of generating a new parameter value along the direction made by the difference of two randomly sampled parameter states (light blue dots) belonging to different chains plus a random term.

2. `generate` $\boldsymbol{\theta}'_h$ `from` $\gamma(\boldsymbol{\theta}_{h_1} - \boldsymbol{\theta}_{h_2}) + \epsilon(\cdot|\boldsymbol{\theta}_h)$
3. `simulate` $\boldsymbol{y}'$ `from` $p(\cdot|\boldsymbol{\theta}'_h)$
4. `update` $\boldsymbol{\theta}_h \rightarrow \boldsymbol{\theta}'_h$ `with the log of the probability`

$$\min\left(0, \left[\boldsymbol{\theta}_h - \boldsymbol{\theta}'_h\right]^\top \left[s(\boldsymbol{y}') - s(\boldsymbol{y})\right] + \log\left[\frac{p(\boldsymbol{\theta}'_h)}{p(\boldsymbol{\theta}_h)}\right]\right)$$

```
    end for
end for
```

### 4.3. Kapferer tailor shop network

Consider the Kapferer tailor shop network and a 3-dimensional model including the following network statistics: edges (`edges`), mutual edges (`mutual`) and cyclic triples (`ctriple`) involving nodes with the same job status, where the job status is represented by a categorical nodal attribute variable consisting of 8 levels described in Figure 1.

| | |
|---|---|
| `edges` | $\sum_{i \neq j} y_{ij}$ |
| `mutual` | $\sum_{i \neq j} y_{ij} y_{ji}$ |
| `ctriple("job")` | $\sum_{i \neq j \neq k} y_{ij} y_{jk} y_{ki}$ where $i, j, k$ have the same job status |

The format of the model specification is the same as of an `ergm` formula:

```
R> formula <- y ~ edges + mutual + ctriple("job")
```

Then we can use the `bergm` function to sample from the posterior distribution using the MCMC algorithm described above. In this example we use the parallel ADS procedure described in Section 4.2. By default, the number of chains in the population is set as twice the number of dimensions of the model. It is possible to choose a different number of chains by using the argument `nchains`. In order to perform the block-site update described in Section 4.1 it is necessary to set `nchains` = 1. For each chain, we can then set the number of

burn-in iterations (`burn.in`) and the number of iterations after the burn-in (`main.iters`). The number of iterations used to simulate a network $\boldsymbol{y}'$ at each iteration is defined by the argument `aux.iters`.

```
R> post.est  <- bergm(formula, burn.in = 500, gamma = 0.7, main.iters = 1500,
+    aux.iters = 25000)
```

The population MCMC with parallel ADS move is the default procedure of the `bergm` function. The total number of iterations, e.g., the size of the posterior sample, is `nchains` $\times$ `main.iters`. The proposal covariance structure $\boldsymbol{\Sigma}_\epsilon$ is defined by the argument `sigma.epsilon` which is set to be a diagonal matrix with every diagonal entry equal to a small number. In many cases, good mixing of the chain is ensured by a sensible tuning of the parallel ADS move factor `gamma` and therefore the argument `sigma.epsilon` can be generally left at its default value. The parameter `gamma` can be easily tuned to achieve a suitable acceptance rate by starting from its default value (0.5). The range of values that `gamma` can take depends on the size of network and the kind of network statistics included in the model.

As said above, parallel ADS is adopted as the default procedure but it is automatically disabled in the case of uni-dimensional models where the block-update sampler is applied and the argument `gamma` is used to tune the variance of the normal proposal distribution $\epsilon$.

It is possible to visualize the results of the MCMC estimation by using the `bergm.output` function which is based on the **coda** package (Plummer *et al.* 2006) which is an R package for MCMC output analysis and diagnostics.

```
R> bergm.output(post.est, lag.max = 50)

 MCMC results for Model: y ~ edges + mutual + ctriple("job")

 Posterior mean:
         theta1 (edges) theta2 (mutual) theta3 (ctriple.job)
Chain 1       -3.4959346       3.6878307            0.9911443
Chain 2       -3.5206027       3.7528209            1.0021785
Chain 3       -3.4500632       3.5582041            0.9674991
Chain 4       -3.4675361       3.6290626            0.9669324
Chain 5       -3.5121016       3.7160281            0.9960164
Chain 6       -3.4930906       3.6577777            0.9825008

 Posterior sd:
         theta1 (edges) theta2 (mutual) theta3 (ctriple.job)
Chain 1        0.1528820       0.3595036            0.1808921
Chain 2        0.2138147       0.4328422            0.2055869
Chain 3        0.1768300       0.4407702            0.2090132
Chain 4        0.1499667       0.3609036            0.2171879
Chain 5        0.1668627       0.3703079            0.2024424
Chain 6        0.1832811       0.3892008            0.2065293

         Acceptance rate:
Chain 1          0.2686667
```

```
Chain 2          0.2660000
Chain 3          0.2673333
Chain 4          0.2306667
Chain 5          0.2693333
Chain 6          0.2900000


 Overall posterior density estimate:
           theta1 (edges) theta2 (mutual) theta3 (ctriple.job)
Post. mean     -3.4898881        3.666954             0.9843786
Post. sd        0.1768949        0.398493             0.2043015

 Overall acceptance rate: 0.27
```

The output above shows the results of the MCMC estimation: posterior means and standard deviations, and acceptance rates for every chain in the population and for the overall chain. Notice that the posterior summaries of each chain are consistent with each other. Figure 5 displays the MCMC diagnostic plots produced by the `bergm.output` function. In this example, we observe a low density effect expressed by the negative value of the posterior mean of the edge effect parameter ($\theta_1$) combined with the positive mutuality and transitivity within people having the same job status expressed by the mutual edge parameter ($\theta_2$) and cyclic triple parameter ($\theta_3$) respectively. The overall acceptance rate is around 27% and the autocorrelation is negligible after lag 50.

## 5. Bayesian goodness-of-fit diagnostics

An important contribution of this article is to propose a Bayesian procedure to establish whether the estimated parameter posterior of the model achieves a good fit to the key topological features of the observed network.

The `bgof` function provides a useful tool for assessing Bayesian goodness-of-fit so as to examine the fit of the data to the posterior model obtained by the `bergm` function. The observed network data $\boldsymbol{y}$ are compared with a set of networks $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_S$ simulated from $S$ independent realizations $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_S$ of the posterior density estimate. This comparison is made in terms of high-level characteristics $g(\cdot)$ such as higher degree distributions, etc. (see Hunter, Goodreau, and Handcock 2008a). The algorithm can be summarized as follows:

for $i = 1, \ldots, S$

1. sample $\boldsymbol{\theta}_i$ from the estimate of $p(\boldsymbol{\theta}|\boldsymbol{y})$
3. simulate $\boldsymbol{y}_i$ from $p(\cdot|\boldsymbol{\theta}_i)$
4. calculate $g(\boldsymbol{y}_i)$

end for

For example, the code below is used to compare the Kapferer tailor shop network with a series of networks simulated from $S = 100$ random realizations (`sample.size`) of the estimated posterior distribution `post.est` using $10,000$ iterations (`aux.iters`) for the network simulation

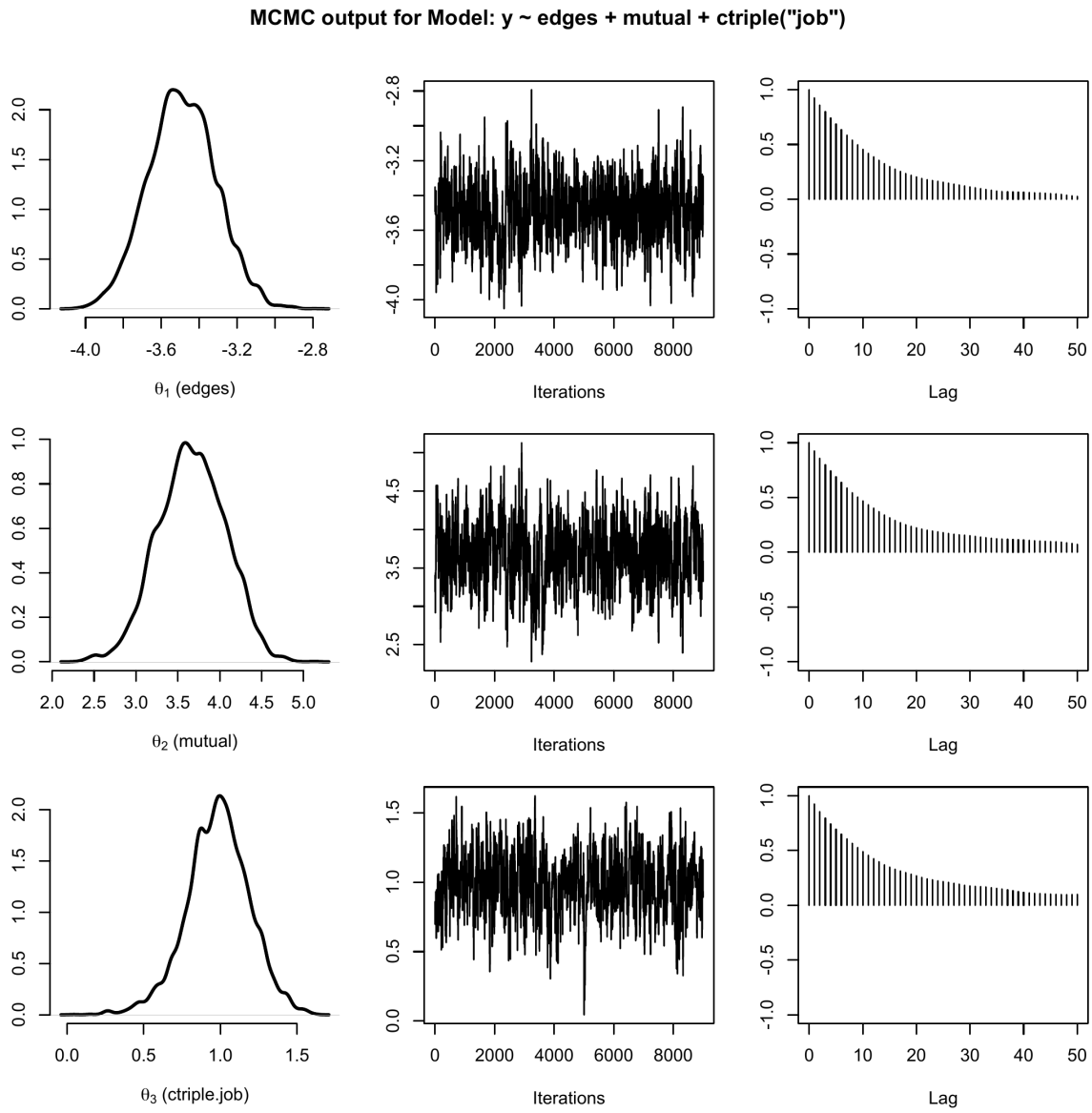


Figure 5: MCMC diagnostics for the overall chain. The three plot columns, from left to right, are: estimated marginal posterior densities, trace plots, and autocorrelation plots.

step. The `bgof` function may take a few seconds to run and, at the end of the execution, it will automatically plot the results as shown in Figure 6.

```
R> bgof(post.est, sample.size = 100, aux.iters = 10000, directed = TRUE,
+    n.ideg = 20, n.odeg = 20, n.dist = 10, n.esp = 15)
```

The set of statistics used for the comparison of directed networks includes the in-degree distribution, the out-degree distribution, the minimum geodesic distance distribution and the edgewise shared partner distribution. The arguments `n.ideg`, `n.odeg`, `n.dist`, and `n.esp` indicates the number of boxplots to plot for each distribution respectively.

In Figure 6 we see, based on the various goodness-of-fit statistics, that the networks simulated

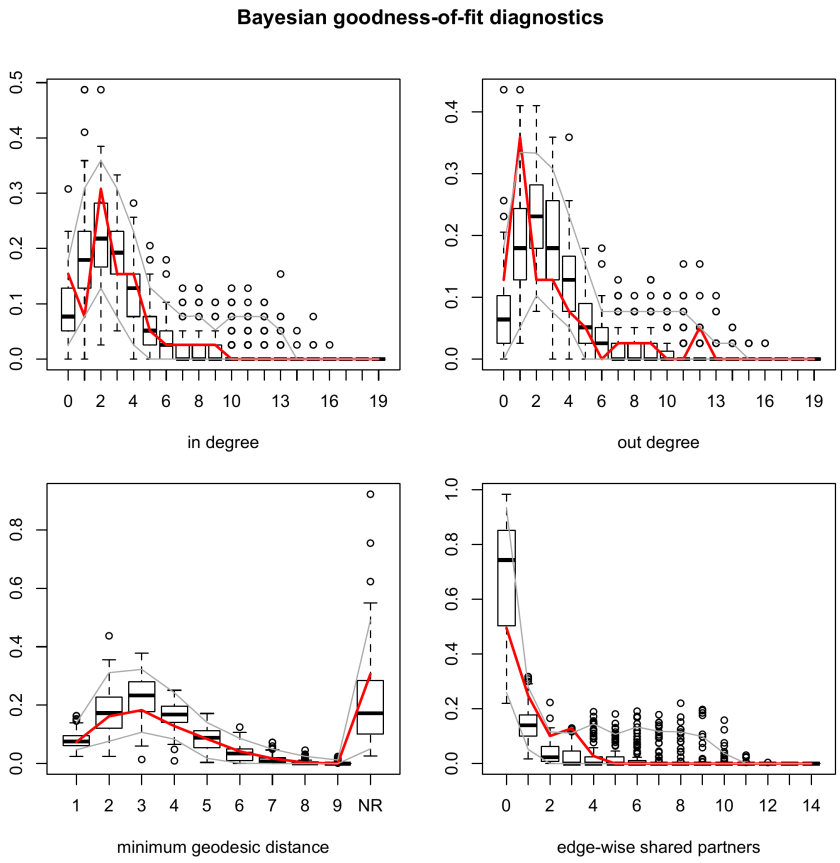


Figure 6: Bayesian goodness-of-fit diagnostics. The red line displays the goodness-of-fit statistics for the observed data together with boxplots of goodness-of-fit statistics based on 100 simulated networks from the posterior distribution.

from the posterior distribution are in reasonable agreement with the observed network. We can therefore conclude that the model fits reasonably to the data, despite its simplicity.

## 6. Bayesian model selection

An important problem in statistical analysis is the choice of an optimal model from a set of *a priori* competing models. In the ERGM context, this task translates into the choice of which subset of network statistics should be included into the model.

Let $m$ indicate a particular model from a set of competing models with corresponding parameters $\boldsymbol{\theta}_m$. Following the Bayesian paradigm, interest focuses on exploring the posterior distribution,

$$p(m, \boldsymbol{\theta}_m|\boldsymbol{y}) \propto p(\boldsymbol{y}|m, \boldsymbol{\theta}_m)\, p(\boldsymbol{\theta}_m|m)\, p(m)$$

where $p(\boldsymbol{\theta}_m|m)$ and $p(m)$ are prior distributions within model $m$, and on model $m$, respectively. The reversible jump Markov chain Monte Carlo (RJMCMC) algorithm (Green 1995) was designed to explore this type of posterior distribution across the joint model and parameter space. It is therefore a type of MCMC algorithm that allows one to jointly explore the uncertainty between and within models.

This approach is very appealing since it relies exclusively on probabilistic considerations but is very challenging from a computational viewpoint. As stated above, the intractability of the likelihood normalizing constant $z(\boldsymbol{\theta}_m)$ in Equation 1 renders standard RJMCMC techniques infeasible. However, the exchange algorithm used for parameter estimation can be easily generalized so as to include model indicators.

The auto-RJ exchange algorithm described in Caimo and Friel (2013) represents a trans-dimensional RJMCMC extension of the exchange algorithm involving an independence sampler based on a distribution fitting a parametric density approximation to the within-model posterior. This approach overcomes the issue of the likelihood intractability sampling from:

$$p(\boldsymbol{\theta}'_{m'}, \boldsymbol{\theta}_m, m', m, \boldsymbol{y}'|\boldsymbol{y}) \propto p(\boldsymbol{y}|\boldsymbol{\theta}_m, m)p(\boldsymbol{\theta}_m|m)p(m)w(\boldsymbol{\theta}'_{m'}|m')h(m'|m)p(\boldsymbol{y}'|\boldsymbol{\theta}'_{m'}, m'), \qquad (3)$$

where $m$ and $m'$ are two competing models, $p(\boldsymbol{y}|\boldsymbol{\theta}_m, m)$ and $p(\boldsymbol{y}'|\boldsymbol{\theta}'_{m'}, m')$ are the two likelihoods for the data $\boldsymbol{y}$ under model $m$ and the simulated data $\boldsymbol{y}'$ under model $m'$ respectively, $p(\boldsymbol{\theta}_m|m)$ and $p(m)$ are the priors for the parameter and the respective model $m$, $w(\cdot)$ is a within-model proposal (independence sampler) which fits a parametric density approximation to the model posteriors and $h(\cdot)$ is a between-model proposal. Notice that the marginal distribution for $\boldsymbol{\theta}'_{m'}$ and $m'$ in Equation 3 is the target distribution of interest $p(\boldsymbol{\theta}_m, m|\boldsymbol{y})$.

The `bergmS` function implements the auto-RJ exchange algorithm which consists of two parts: an offline step and an online step. In the online step, samples from the posterior $p(\boldsymbol{\theta}_m|\boldsymbol{y}, m)$ are gathered from each competing model using the `bergm` function and then approximated by normal distributions $\mathcal{N}(\hat{\boldsymbol{\mu}}_m, \hat{\boldsymbol{\Sigma}}_m)$ determined by the first and second moments from a sample from the model.

The second step (online run) of the algorithm consists of a Gibbs update of $m'$ followed by a Gibbs update of $\boldsymbol{\theta}'_{m'}$ which is generated via the independence sampler $w(\boldsymbol{\theta}'_{m'}|m') \sim \mathcal{N}(\hat{\boldsymbol{\mu}}_{m'}, \hat{\boldsymbol{\Sigma}}_{m'})$. This is followed by a Gibbs update of $\boldsymbol{y}'$ which is generated from $p(\cdot|\boldsymbol{\theta}'_{m'}, m')$. Then a deterministic exchange move from a current state $(\boldsymbol{\theta}_m, m)$ to the proposed new state $(\boldsymbol{\theta}'_{m'}, m')$ is accepted with probability:

$$\min\left\{1, \frac{q_{\boldsymbol{\theta}_m,m}(\boldsymbol{y}')}{q_{\boldsymbol{\theta}_m,m}(\boldsymbol{y})} \frac{q_{\boldsymbol{\theta}'_{m'},m'}(\boldsymbol{y})}{q_{\boldsymbol{\theta}'_{m'},m'}(\boldsymbol{y}')} \frac{p(\boldsymbol{\theta}'_{m'}|m')}{p(\boldsymbol{\theta}_m|m)} \frac{p(m')}{p(m)} \frac{w(\boldsymbol{\theta}_m|\hat{\boldsymbol{\mu}}_m, \hat{\boldsymbol{\Sigma}}_m)}{w(\boldsymbol{\theta}'_{m'}|\hat{\boldsymbol{\mu}}_{m'}, \hat{\boldsymbol{\Sigma}}_{m'})} \frac{h(m|m')}{h(m'|m)}\right\},$$

where $q_{\boldsymbol{\theta}_m,m}$ and $q_{\boldsymbol{\theta}'_{m'},m'}$ indicate the unnormalized likelihoods under model $m$ with parameter $\boldsymbol{\theta}_m$ and under model $m'$ with parameter $\boldsymbol{\theta}'_{m'}$ respectively.

The structure of the `bergmS` function can be described in the following way:

`for` $i = 1, \ldots, N$

1. `generate` $m'$ from $h(\cdot|m)$
2. `generate` $\boldsymbol{\theta}'_{m'} \sim \mathcal{N}(\hat{\boldsymbol{\mu}}_{m'}, \hat{\boldsymbol{\Sigma}}_{m'})$
3. `simulate` $\boldsymbol{y}'$ `from` $p(\cdot|\boldsymbol{\theta}'_{m'}, m')$
4. `update` $(\boldsymbol{\theta}_m, m) \rightarrow (\boldsymbol{\theta}'_{m'}, m')$ `with the log of the probability:`

$$\min\left(0, \boldsymbol{\theta}_m^\top[s_m(\boldsymbol{y}') - s_m(\boldsymbol{y})] + \boldsymbol{\theta}'^\top_{m'}[s_{m'}(\boldsymbol{y}) - s_{m'}(\boldsymbol{y}')] + \log\left[\frac{p(\boldsymbol{\theta}'_{m'})}{p(\boldsymbol{\theta}_m)} \frac{w(\boldsymbol{\theta}_m|\hat{\boldsymbol{\mu}}_m, \hat{\boldsymbol{\Sigma}}_m)}{w(\boldsymbol{\theta}'_{m'}|\hat{\boldsymbol{\mu}}_{m'}, \hat{\boldsymbol{\Sigma}}_{m'})}\right]\right)$$

`end for`

where $s_m(\boldsymbol{y})$ and $s_{m'}(\boldsymbol{y})$ are the observed vectors of network statistics under model $m$ and $m'$ respectively, and $s_m(\boldsymbol{y}')$ and $s_{m'}(\boldsymbol{y}')$ are the simulated vectors of network statistics under model $m$ and $m'$ respectively. Here, we have assumed the model priors and the between model proposals to be uniform. The reader is referred to Caimo and Friel (2013) for more details on this algorithm.

### 6.1. Karate club network

In this example, we consider the karate club network and we propose three competing models to fit the data using a set of new specification statistics introduced by Hunter and Handcock (2006) and Hunter (2007): geometrically weighted edgewise shared partners (`gwesp`) and geometrically weighted degrees (`gwdegree`):

$$\begin{aligned} \texttt{gwesp} \quad & e^{\phi_v} \sum_{k=1}^{n-2} \left\{1 - \left(1 - e^{-\phi_v}\right)^k\right\} EP_k(\boldsymbol{y}) \\ \texttt{gwdegree} \quad & e^{\phi_u} \sum_{k=1}^{n-1} \left\{1 - \left(1 - e^{-\phi_u}\right)^k\right\} D_k(\boldsymbol{y}) \end{aligned}$$

where the scale parameters $\phi_v = 0.2$ and $\phi_u = 0.8$. $D_k(\boldsymbol{y})$ is the number of pairs that have exactly $k$ common neighbours and $EP_k(\boldsymbol{y})$ is the number of connected pairs with exactly $k$ common neighbours.

The specification of these models requires the creation of a list of formulas:

```
R> formulae <- c(y ~ edges + gwesp(0.2, fixed = TRUE),
+    y ~ edges + gwdegree(0.8, fixed = TRUE),
+    y ~ edges + gwesp(0.2, fixed = TRUE) + gwdegree(0.8, fixed = TRUE))
```

The `bergmS` command is then used to carry out the algorithm. To do this we have to specify several arguments for both the offline and online step.

The offline run consists of running the `bergm` function for each of the models proposed. Therefore we set some arguments `main.iters`, `burn.ins`, `gammas` which are vectors containing values for `bergm` arguments: `main.iters`, `burn.in`, `gamma` for each competing model.

The argument `iters` refers to the number of iterations used for the online run. The number of MCMC steps used for network simulation is specified as usual by the argument `aux.iters` and this will be used in both the offline and the online step.

```
R> mod.sel <- bergmS(formulae, iters = 25000, aux.iters = 2000,
+    main.iters = rep(700, 3), burn.ins = rep(100, 3), gammas = c(1, 1, 0.8))
```

The `bergmS.output` function produces the MCMC diagnostics for each competing model explored by the MCMC algorithm. Figures 7 and 8 display the plots regarding the posterior model and parameter density estimate for the best model, respectively.

```
R> best.mod <- bergmS.output(mod.sel, lag.max = 100)

 BEST MODEL
 ----------
Model 1: y ~ edges + gwesp(0.2, fixed = TRUE)
```

```
 Posterior parameter estimate:
                          Post. mean:  Post. sd:
theta1 (edges)             -3.2655591  0.3280836
theta2 (gwesp.fixed.0.2)    1.1048745  0.2473057

 Within-model acceptance rate: 0.25


Model 3: y ~ edges + gwesp(0.2, fixed = TRUE) + gwdegree(0.8, fixed = TRUE)

 Posterior parameter estimate:
                          Post. mean:  Post. sd:
theta1 (edges)             -3.4900339  0.5002413
theta2 (gwesp.fixed.0.2)    1.1717390  0.2743921
theta3 (gwdegree)           0.4537180  0.5895084

 Within-model acceptance rate: 0.18

BF_13 = 11.84936


Model 2: y ~ edges + gwdegree(0.8, fixed = TRUE)

 Posterior parameter estimate:
                  Post. mean: Post. sd:
theta1 (edges)      -1.332852  0.000000
theta2 (gwdegree)   -1.331943  0.000000

 Within-model acceptance rate: 0

BF_12 = 2880.875


 Between-model acceptance rate: 0.03
```

In the results above we have the posterior parameter estimates for two of the competing models (Model 2 has not been visited enough times through the MCMC runs to have reliable posterior parameter estimates) with respective within-model acceptance rates and an estimate of the Bayes Factor (about 12) for the comparison between Model 1 and Model 3 which makes clear that there is evidence that Model 1 is the best model of the set. This implies that the observed network is not enhanced by the effect captured by the geometrically weighted degree network statistic.

After running the command `bergmS.output`, it is possible to perform a Bayesian goodness-of-fit test. In this case, since the observed network is undirected, the set of high-level statistics include the degree distribution in place of the in-degree and out-degree distributions.

```
R> bgof(best.mod, aux.iters = 10000, n.deg = 20, n.dist = 10, n.esp = 15)
```

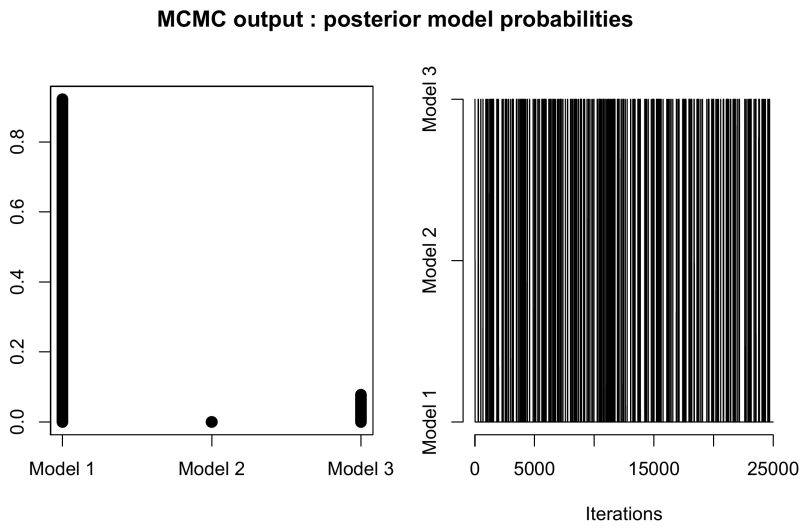


Figure 7: MCMC diagnostics: posterior model probabilities.

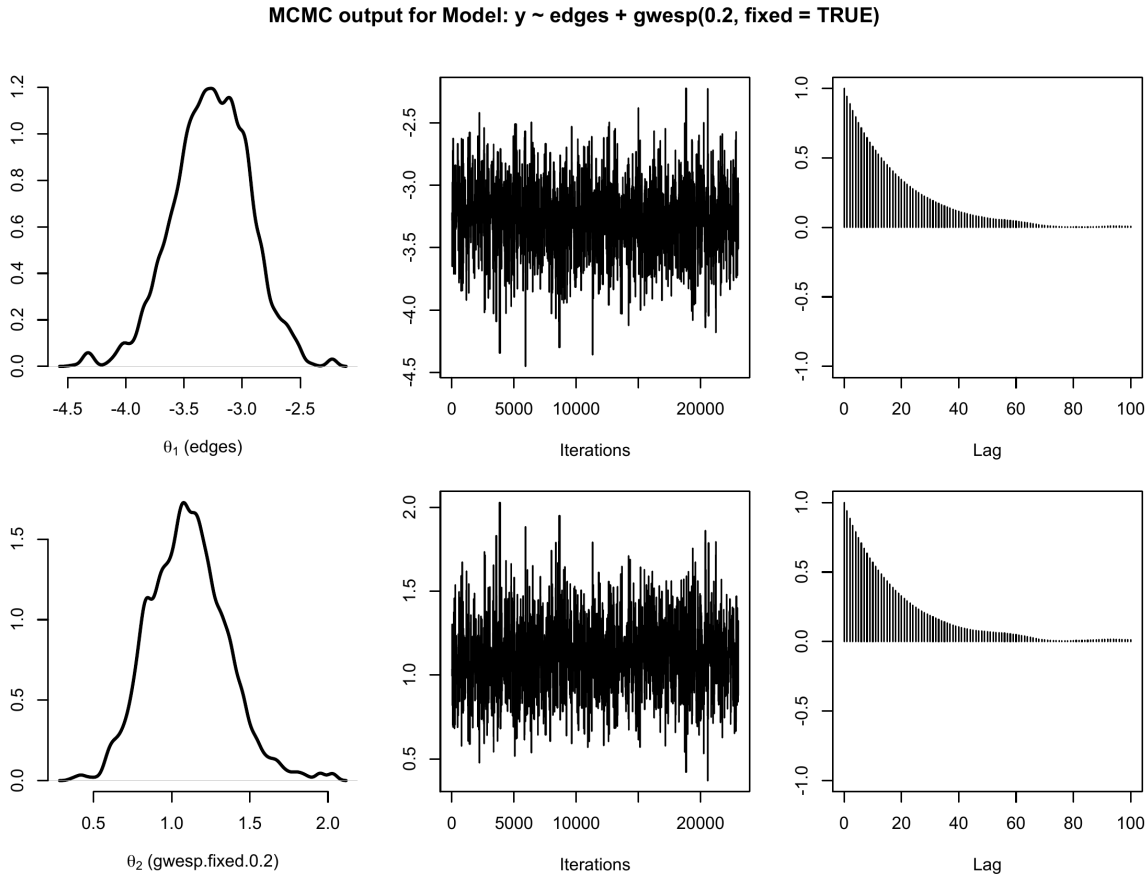


Figure 8: MCMC diagnostics: posterior parameter probabilities.

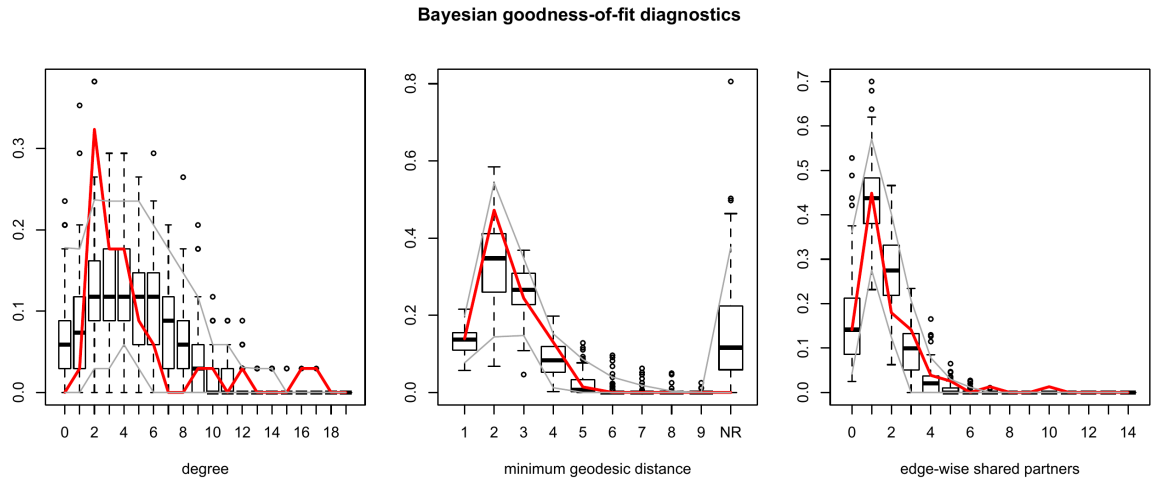


Figure 9: Bayesian goodness-of-fit diagnostics.

In this example, the observed data appears to be a reasonable fit to the posterior distribution of the model selected (Model 1), based on the goodness-of-fit geodesic and the shared partners statistics displayed in Figure 9.

## 7. Example: Teenage Friends and Lifestyle Study

The adolescent friendship network were collected in the Teenage Friends and Lifestyle Study (Pearson and Michell 2000). Friendship network data and substance use were recorded for a cohort of pupils in a secondary school in Glasgow (Scotland). Here we consider 3 actor covariates: drugs consumption (which was binarized in this example), sport activity, and smoking (Figure 10).

In this example we focus on the transitivity effect expreessed by the geometrically weighted edgewise shared partner network statistic and the homophily effect of the drugs consumption (`nodematch("drugs")`) and its relationship to sport activity (`nodematch(c("sport", "drugs")`) and smoking (`nodematch(c("smoke", "drugs"))`).

The homophily effect modeled by `nodematch` counts the number of edges for which two nodes share the same covariate value. When multiple relationships are studied, the `nodematch` statistic counts only those on which all the covariate values match. More information about network statistics and their description can be found by typing `?ergm.terms`.

We propose the following 4 competing models:

```
R> m1 <- y ~ edges + gwesp(log(2), fixed = TRUE) +
+    nodematch(c("sport", "drugs"))
R> m2 <- y ~ edges + gwesp(log(2), fixed = TRUE) +
+    nodematch(c("smoke", "drugs"))
R> m3 <- y ~ edges + gwesp(log(2), fixed = TRUE) + nodematch("drugs")
R> m4 <- y ~ edges + gwesp(log(2), fixed = TRUE)
```

An important advantage of the Bayesian approach includes easily interpretable measures of

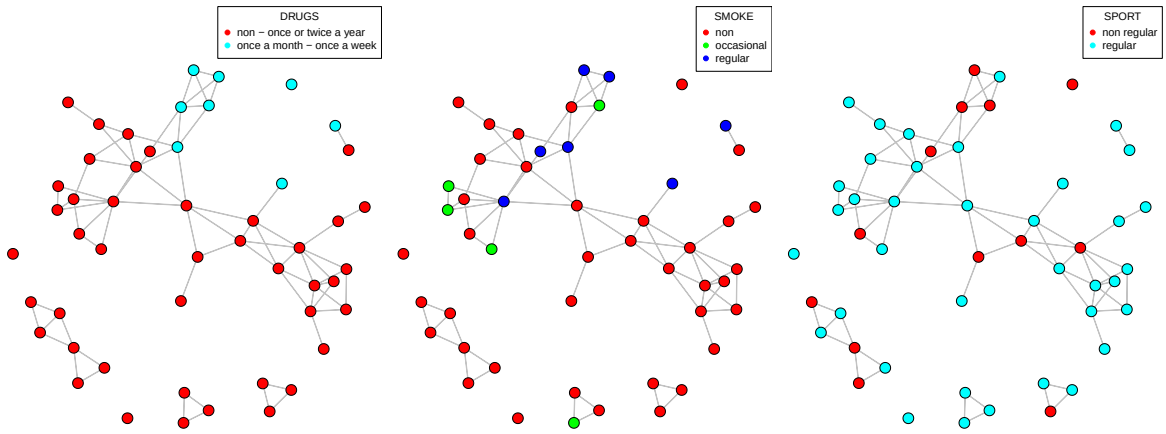


Figure 10: 50 girls from the Teenage Friends and Lifestyle Study dataset.

uncertainty through the use of prior knowledge. In this context, for example, it is known that the network graph is sparse meaning that the density effect (expressed by the `edge` statistic) is likely to be negative. For this reason we can include this prior information by setting the parameter value for the `edge` statistic equal to $-1$.

We can also set up the prior variance/covariance structure. In this case we set the prior covariance matrix of each model to be a diagonal matrix with every entry equal to 5.

```
R> mean.priors <- list(c(-1, 0, 0), c(-1, 0, 0), c(-1, 0, 0), c(-1, 0))
R> sigma <- 5
R> sigma.priors <- list(diag(sigma, 3), diag(sigma, 3), diag(sigma, 3),
+    diag(sigma, 2))
```

As we have done above, we can use the `bergmS` function to perform Bayesian model selection and get an estimate of the Bayes factors.

```
R> mod.sel <- bergmS(c(m1, m2, m3, m4), iters = 50000,
+    mean.priors = mean.priors, sigma.priors  =  sigma.priors,
+    aux.iters = 5000, main.iters = rep(1000, 4), burn.ins = rep(100, 4),
+    gammas = rep(0.7, 4))
R> best.mod <- bergmS.output(mod.sel,lag.max = 200)

 BEST MODEL
 ----------
Model 3: y ~ edges + gwesp(log(2), fixed = TRUE) +
  nodematch(c("drugs"))

 Posterior parameter estimate:
                                        Post. mean:  Post. sd:
theta1 (edges)                           -4.5805494  0.3395364
theta2 (gwesp.fixed.0.693147180559945)    1.0009564  0.1330043
theta3 (nodematch.drugs)                  0.8180081  0.3348887
```

```
 Within-model acceptance rate: 0.11


Model 4: y ~ edges + gwesp(log(2), fixed = TRUE)

 Posterior parameter estimate:
                                       Post. mean:  Post. sd:
theta1 (edges)                          -4.0061009  0.2135235
theta2 (gwesp.fixed.0.693147180559945)   1.0852981  0.1295703

 Within-model acceptance rate: 0.21

BF_34 = 6.1875


Model 2: y ~ edges + gwesp(log(2), fixed = TRUE) +
  nodematch(c("smoke", "drugs"))

 Posterior parameter estimate:
                                       Post. mean:  Post. sd:
theta1 (edges)                          -4.1570235  0.2314948
theta2 (gwesp.fixed.0.693147180559945)   1.0586371  0.1416855
theta3 (nodematch.smoke.drugs)           0.3143428  0.1885215

 Within-model acceptance rate: 0.16

BF_32 = 12.4805687203791


Model 1: y ~ edges + gwesp(log(2), fixed = TRUE) +
  nodematch(c("sport", "drugs"))

 Posterior parameter estimate:
                                       Post. mean:  Post. sd:
theta1 (edges)                          -4.0878854  0.2355906
theta2 (gwesp.fixed.0.693147180559945)   1.0849115  0.1317221
theta3 (nodematch.sport.drugs)           0.1633458  0.1778837

 Within-model acceptance rate: 0.12

BF_31 = 41.58

Between-model acceptance rate: 0.03
```

In the results above we have the posterior parameter estimates for all the competing models with respective within-model acceptance rates and an estimate of the Bayes Factors for the comparison between the four models which makes clear that there is evidence that Model 3

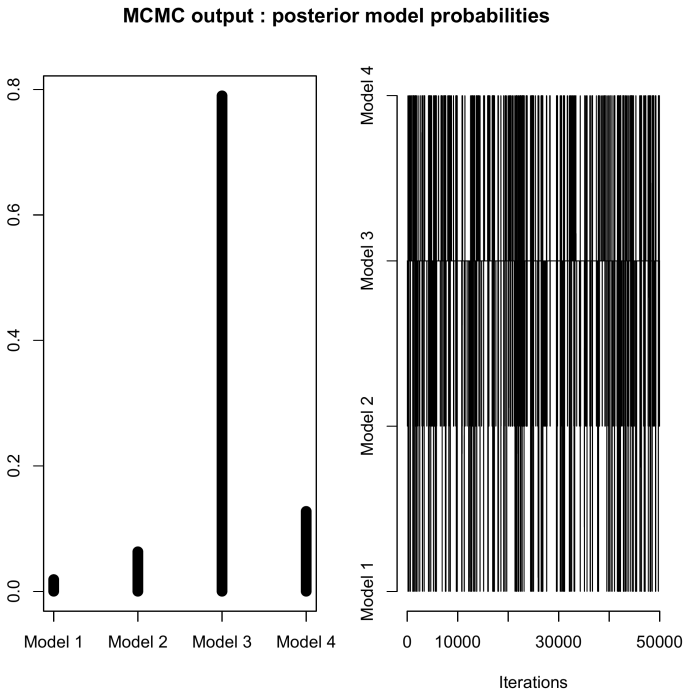


Figure 11: MCMC diagnostics: posterior model probabilities.

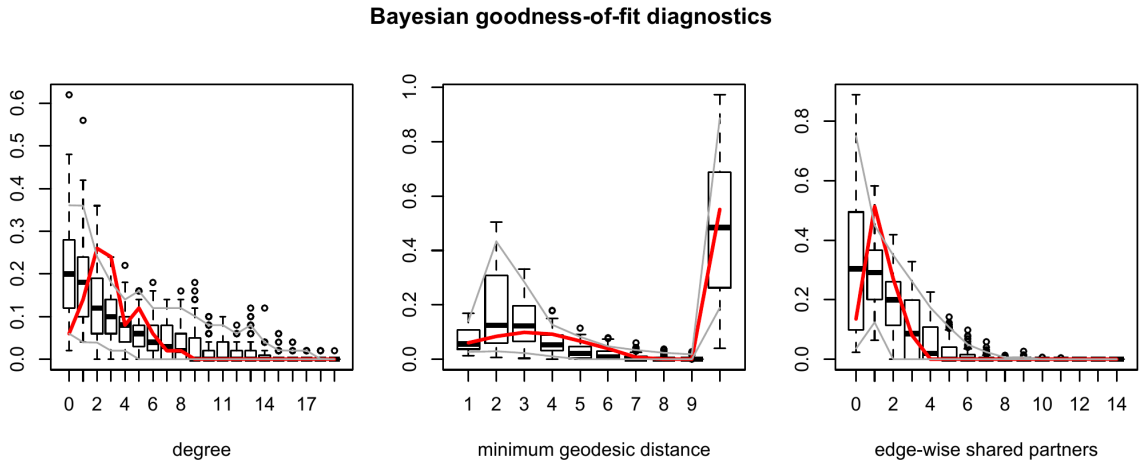


Figure 12: Bayesian goodness-of-fit diagnostics.

is the best model of the set. This implies that the observed network is enhanced by the drugs consumption homophily effect.

```
R> bgof(best.mod, aux.iters = 20000, n.deg = 20, n.dist = 10, n.esp = 15)
```

In this example, the observed data appears to be a reasonable fit to the posterior distribution of the model selected (Model 3), based on the goodness-of-fit geodesic and the shared partners statistics displayed in Figure 12.

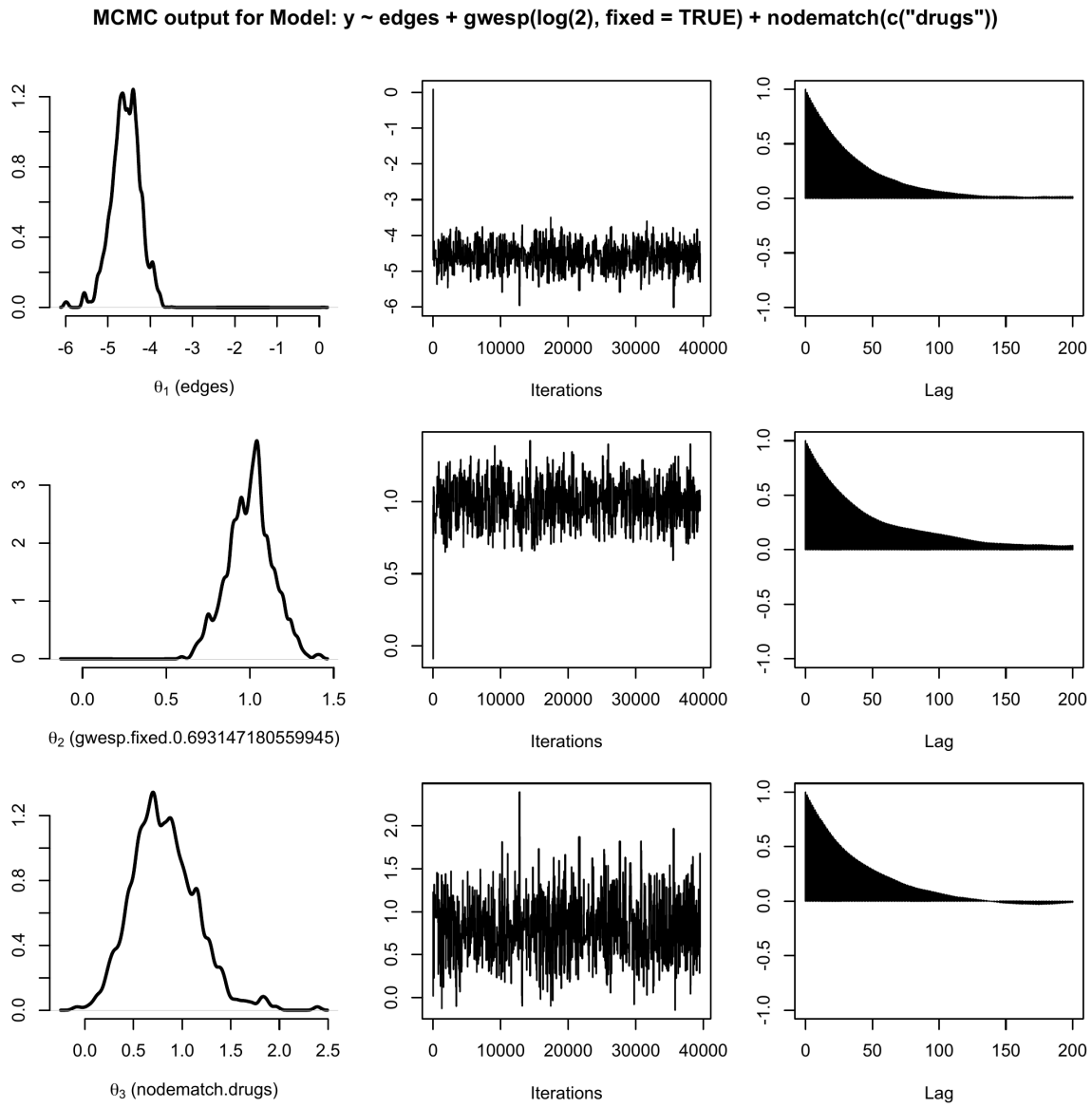


Figure 13: MCMC diagnostics: posterior parameter probabilities.

From this results we can conclude that girls having the same level of drugs consumption tend to be friends. While girls with the same level of drugs consumption and same level of smoking behavior or sport activity do not seem to create a significant number of friendship connections. The transitivity effect is an important feature of the network graph but it is not sufficient to explain the complexity of the observed network data.

## 8. Discussion

The software package **Bergm** aims to help researchers and practitioners in two ways. Firstly, it is currently the only package for R that provides a simple and complete range of tools for

conducting Bayesian analysis for ERGMs. Secondly, **Bergm** makes available a platform that can be easily customised, extended, and adapted to address different requirements.

The software package is under continual development and it is far from finished. The main limitation of the software is its computational cost which makes it unsuitable for managing network graphs larger than hundreds of nodes, however it is perfectly suited for networks involving up to a hundred nodes. An important improvement in terms of computational time and efficiency will be done by turning some of the R functions into C functions integrated with the **ergm** package. We expect that will yield further reductions in computational run time.

Future versions of the **Bergm** package will address several issues including Bayesian analysis of curved ERGMs (Hunter and Handcock 2006) and ERGMs with missing data (Koskinen, Robins, and Pattison 2010).

## References


Butts CT (2008). "**network**: A Package for Managing Relational Data in R." *Journal of Statistical Software*, **24**(2), 1–36. URL http://www.jstatsoft.org/v24/i02/.

Caimo A, Friel N (2011). "Bayesian Inference for Exponential Random Graph Models." *Social Networks*, **33**(1), 41–55.

Caimo A, Friel N (2013). "Bayesian Model Selection for Exponential Random Graph Models." *Social Networks*, **35**(1), 11–24.

Caimo A, Friel N (2014). ***Bergm**: Bayesian Analysis for Exponential Random Graph Models*. R package version 3.0, URL http://CRAN.R-project.org/package=Bergm.

Frank O, Strauss D (1986). "Markov Graphs." *Journal of the American Statistical Association*, **81**(395), 832–842.

Genz A, Bretz F, Miwa T, Mi X, Leisch F, Scheipl F, Hothorn T (2014). ***mvtnorm**: Multivariate Normal and t Distributions*. R package version 1.0-0, URL http://CRAN.R-project.org/package=mvtnorm.

Gilks WR, Roberts GO, George EI (1994). "Adaptive Direction Sampling." *Journal of the Royal Statistical Society D*, **43**(1), 179–189.

Green PJ (1995). "Reversible Jump Markov Chain Monte Carlo Computation and Bayesian Model Determination." *Biometrika*, **82**(4), 711–732.

Handcock MS, Hunter DR, Butts CT, Goodreau SM, Morris M (2007). "**statnet**: Software Tools for the Representation, Visualization, Analysis and Simulation of Network Data." *Journal of Statistical Software*, **24**(1), 1–11. URL http://www.jstatsoft.org/v24/i01/.

Hunter DR (2007). "Curved Exponential Family Models for Social Networks." *Social Networks*, **29**(2), 216–230.

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

Hunter DR, Handcock MS (2006). "Inference in Curved Exponential Family Models for Networks." *Journal of Computational and Graphical Statistics*, **15**(3), 565–583.

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. URL http://www.jstatsoft.org/v24/i03/.

Kapferer B (1972). *Strategy and Transaction in an African Factory: African Workers and Indian Management in a Zambian Town*. Manchester University Press, London.

Koskinen JH (2004). "Bayesian Analysis of Exponential Random Graphs – Estimation of Parameters and Model Selection." *Research Report 2004:2*, Department of Statistics, Stockholm University.

Koskinen JH, Robins GL, Pattison PE (2010). "Analysing Exponential Random Graph (p-star) Models with Missing Data Using Bayesian Data Augmentation." *Statistical Methodology*, **7**(3), 366–384.

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. URL http://www.jstatsoft.org/v24/i04/.

Pearson M, Michell L (2000). "Smoke Rings: Social Network Analysis of Friendship Groups, Smoking and Drug-Taking." *Drugs: Education, Prevention and Policy*, **7**(1), 21–37.

Plummer M, Best N, Cowles K, Vines K (2006). "**coda**: Convergence Diagnosis and Output Analysis for MCMC." *R News*, **6**(1), 7–11. URL http://CRAN.R-project.org/doc/Rnews/.

R Core Team (2014). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org/.

Roberts GO, Gilks WR (1994). "Convergence of Adaptive Direction Sampling." *Journal of Multivariate Analysis*, **49**(2), 287–298.

Robins G, Pattison P, Kalish Y, Lusher D (2007). "An Introduction to Exponential Random Graph Models for Social Networks." *Social Networks*, **29**(2), 169–348.

Salter-Townshend M, White A, Gollini I, Murphy TB (2012). "Review of Statistical Network Analysis: Models, Algorithms, and Software." *Statistical Analysis and Data Mining*, **5**(4), 243–264.

Wasserman S, Pattison P (1996). "Logit Models and Logistic Regression for Social Networks: I. An Introduction to Markov Graphs and $p^*$." *Psychometrika*, **61**(3), 401–425.

Zachary W (1977). "An Information Flow Model for Conflict and Fission in Small Groups." *Journal of Anthropological Research*, **33**(4), 452–473.

# A. R code for loading and plotting the network data

Here we give the code to load the datasets presented in this paper and to plot the network graphs displayed in Figure 2. The main functions to load and plot network data are `network` and the `plot` method for '`network`' objects respectively. They are both included in the **network** package which is one of the dependencies of **Bergm** and it is automatically loaded by typing the `library("Bergm")` command (see Section 2). We include the `set.seed` statement in order to produce exactly the same graphs of Figures 2 and 10. The Zachary karate network dataset used in this paper is available in the **statnet** package. The directed network data and nodal attributes for the Kapferer dataset and the Teenage Friends and Lifestyle Study' social network dataset can be downloaded from the Siena datasets repository http://www.stats.ox.ac.uk/~snijders/siena/.

```
R> install.packages("statnet")
R> library("statnet")
R> install.packages("Bergm")
R> library("Bergm")
```

The Kapferer Taylor Shop network and nodal attribute data can be loaded into R by typing:

```
R> y <- network(read.table("kapferer.dat"), directed = TRUE)
R> x <- read.table("kapfa_stat.dat")
R> y %v% "job" <- x
```

The last command is used to attache the nodal covariate "job" represented by the object `x` to the network object `y`.

To plot the network graph as in Figure 1 we used the following code:

```
R> CC <- colors()[c(24, 135, 53, 142, 258, 28, 551, 119)]
R> set.seed(20)
R> par(oma = rep(0, 4), mar = rep(0, 4))
R> plot(y, vertex.col = CC[x[, 1]], edge.col = colors()[229],
+    vertex.cex = 1.5, usearrows = TRUE)
R> legend("topright", legend = seq(1, 8), col = CC, yjust = 0, pch = 19)
```

The `legend` function creates a legend showing the colors associated to the levels of the "job" nodal attribute. For more information about this function type `?legend`.

The following code was used to load Zachary's karate club network and to plot it as in Figure 2:

```
R> data("zach", package = "ergm.count")
R> y <- zach
R> par(oma = rep(0, 4), mar = rep(0, 4))
R> set.seed(20)
R> plot(y, vertex.col = colors()[123], edge.col = colors()[229],
+    vertex.cex = 1.5)
```

The following code was used to load the Teenage Friends and Lifestyle Study' social network and to plot it as in Figure 10:

```
R> y <- read.table("s50-network1.dat")
R> y <- network(as.matrix(y), matrix.type = "adjacency", directed = FALSE)
R> x1 <- read.table("s50-sport.dat")
R> x2 <- read.table("s50-smoke.dat")
R> x4 <- read.table("s50-drugs.dat")
R> y %v% "sport" <- x1[, 1]
R> y %v% "smoke" <- x2[, 1]
R> y %v% "alcohol" <- x3[, 1]
R> x4[, 1] <- ifelse(x4[, 1] < 3, 1, 2)
R> y %v% "drugs" <- x4[, 1]
R> CC <- rainbow(2)

R> set.seed(20)
R> par(mfrow = c(1, 3), oma = rep(0.2, 4), mar = rep(0.2, 4))
R> CC <- rainbow(2)
R> set.seed(20)
R> plot(y, vertex.col = CC[x4[, 1]], edge.col = colors()[229],
+    vertex.cex = 1.5, usearrows = TRUE)
R> legend("topright", legend = c("non - once or twice a year",
+    "once a month - once a week"),
+    col = CC, yjust = 0, pch = 19, title = "DRUGS")
R> CC <- rainbow(3)

R> set.seed(20)
R> plot(y, vertex.col = CC[x2[, 1]], edge.col = colors()[229],
+    vertex.cex = 1.5, usearrows = TRUE)
R> legend("topright", legend = c("non", "occasional", "regular"),
+    col = CC, yjust = 0, pch = 19, title = "SMOKE")
R> CC <- rainbow(2)

R> set.seed(20)
R> plot(y, vertex.col = CC[x1[, 1]], edge.col = colors()[229],
+    vertex.cex = 1.5, usearrows = TRUE)
R> legend("topright", legend = c("non regular", "regular"), col = CC,
+    yjust = 0, pch = 19, title = "SPORT")
```

More details about the commands used in this section can be found in the help files by typing `?network` and `?plot.network`. More information about the **network** package can be found in Butts (2008).

**Affiliation:**

Alberto Caimo
Faculty of Economics
University of Lugano, Switzerland

E-mail: acaimo.stats@gmail.com
URL: https://sites.google.com/site/albertocaimo/

*Journal of Statistical Software* http://www.jstatsoft.org/
published by the American Statistical Association http://www.amstat.org/
Volume 61, Issue 2 *Submitted:* 2012-02-16
October 2014 *Accepted:* 2014-05-27