Exponential random graph models with R

Alberto Caimo and Isabella Gollini

2017-04-10

Contents

Preface

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.

This manual is written for students in statistical network analysis courses as well as for people already engaged in these fields. Our aim is to offer the reader a balanced presentation of the fundamental statistical concepts and methods and practical advice on their effective implementation using R.

About the authors

Alberto Caimo

Alberto Caimo is a lecturer in Statistics in the School of Mathematical Sciences at the Dublin Institute of Technology, Ireland. His current research activity concerns the development and implementation of statistical modelling and computational approaches for complex relational data in a wide variety of applications involving interdisciplinary collaborations. Alberto is the author of the R package `Bergm` for the analysis of Bayesian exponential random graph models.

More info: https://sites.google.com/site/albertocaimo

Isabella Gollini

Isabella Gollini is a lecturer in Statistics in the Department of Economics, Mathematics and Statistics at the Birkbeck, University of London, UK. Her research activity focuses on cutting-edge statistical topics in the area of statistical modelling and computational statistics. Isabella is the leader of the teaching team of Forwards: the R Foundation taskforce on women and under-represented groups, and she is the author of several R packages, including `lvm4net` for the analysis of latent variable models for network data using fast inferential procedures.

More info: https://sites.google.com/site/isabellagollini

Chapter 1

Introduction

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) (Holland and Leinhardt, 1981) are an 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 ERGMs is the main barrier to estimation. Fortunately, recent theoretical developments and advanced computational procedures have given the possibility to make important progress to overcome statistical inference problems.

Chapter 2

R basics

R is an open source programming language and software environment for statistical computing and graphics that is supported by the R Foundation for Statistical Computing (R Core Team, 2016).

2.1 Fundamentals

- R uses a command-like environment
- R is highly extendable
- You can write your own custom functions
- There are more than 10,000 free packages
- Generally good at reading in/writing out other file formats
- Everything in R is an **object**

2.2 R and RStudio

To install R on your computer, go to the home of the R website at https://www.r-project.org/.

We recommend to use the RStudio interface. To install RStudio, go to: http://www.rstudio.org/.

2.3 Package repositories

In R, the fundamental unit of shareable code is the package. A package bundles together code, data, documentation, and tests, and is easy to share with others. Most packages are stored, in an organized way, in online repositories from which they can be easily retrieved and installed on your computer.

There are two main R repositories:

- The Comprehensive R Archive Network (CRAN): https://cran.r-project.org/.
- Bioconductor (open source software for bioinformatics): https://www.bioconductor.org/.

2.4 R packages for statistical network analysis

2.4.1 `statnet`

`statnet` (Handcock et al., 2016) is a suite of software packages providing a comprehensive framework based on exponential random graph models, including tools for model estimation, model evaluation, model-based network simulation, and network visualization.

More info: https://CRAN.R-project.org/package=statnet.

2.4.2 `Bergm`

The `Bergm` package (Caimo and Friel, 2014) provides a comprehensive framework for Bayesian estimation of exponential random graph models. It can also supply graphical Bayesian goodness-of-fit procedures that address the issue of model adequacy.

More info: https://CRAN.R-project.org/package=Bergm.

Chapter 3

The random graph model

3.1 Networks as random graphs

We describe as the observed network the network data the researcher has collected and is interested in modeling. The observed network is regarded as one realization from a set of possible networks with similar important features, that is, as the outcome of some (unknown) stochastic process. In other words, the observed network is seen as one particular pattern of edges out of a large set of possible patterns.

Network data are generally represented by graphs of nodes (actors) and edges (relations). Social network analysis focuses on the relations among actors, and not individual actors and their attributes. In fact, for a given model, the node set is regarded as fixed. The range of possible networks, and their probability of occurrence under the model, is represented by a probability distribution on the set of all possible graphs with this number of nodes.

Once we have defined a probability distribution on the set of all graphs with a fixed number of nodes, we can also draw graphs at random from the distribution according to their assigned probabilities, and we can compare the sampled graphs to the observed one on any other characteristic of interest. If the model is a good one for the data, then the sampled graphs will resemble the observed one in many different respects.

3.1.1 Notation

- N number of nodes (**fixed**).
- D number of dyads in a N-node network (**fixed**).
- Y **random** $N \times N$ adjacency matrix $\in \mathcal{Y}$ where:
  - $Y_{ij} = 0$, if i and j are not connected;
  - $Y_{ij} = 1$, if i and j are connected;
  - $Y_{ii} = 0$, (self-loops are not allowed).
- y realisation of Y (observed adjacency matrix).

Y may be directed so Y_{ij} and Y_{ji} are two different random variables observed between the same pair of nodes.

Let's create a random network on 13 nodes using the `network` function and have a look at the corresponding adjacency matrix:

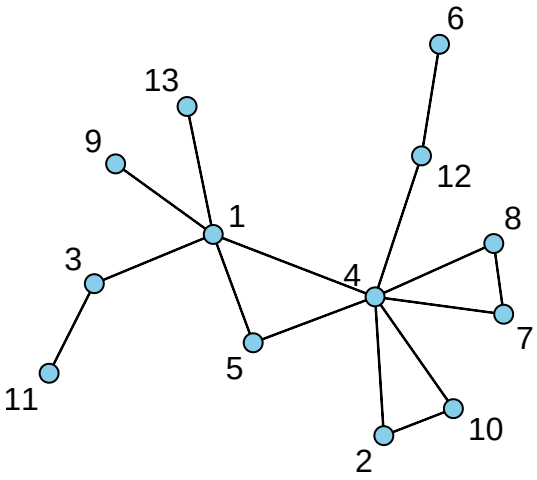
```
library(statnet)
set.seed(11)
N <- 13
y <- network(N, directed = FALSE)
y[,]
```

```
##    1 2 3 4 5 6 7 8 9 10 11 12 13
## 1  0 0 1 1 1 0 0 0 1  0  0  0  1
## 2  0 0 0 1 0 0 0 0 0  1  0  0  0
## 3  1 0 0 0 0 0 0 0 0  0  1  0  0
## 4  1 1 0 0 1 0 1 1 0  1  0  1  0
## 5  1 0 0 1 0 0 0 0 0  0  0  0  0
## 6  0 0 0 0 0 0 0 0 0  0  0  1  0
## 7  0 0 0 1 0 0 0 1 0  0  0  0  0
## 8  0 0 0 1 0 0 1 0 0  0  0  0  0
## 9  1 0 0 0 0 0 0 0 0  0  0  0  0
## 10 0 1 0 1 0 0 0 0 0  0  0  0  0
## 11 0 0 1 0 0 0 0 0 0  0  0  0  0
## 12 0 0 0 1 0 1 0 0 0  0  0  0  0
## 13 1 0 0 0 0 0 0 0 0  0  0  0  0
```

3.1.2 Visualisation

Let's have a look at the network graph:

```
set.seed(11)
plot(y, vertex.cex = 2, vertex.col = "skyblue", label = seq(1, N))
```



3.1.3 Number of dyads

In an undirected graph, we have:

$$D = \frac{N^2 - N}{2}$$

```
D <- (N^2 - N) / 2
D
```

```
## [1] 78
```

3.1.4 Number of edges

The number of edges is the sum of the 1's in the observed adjacency matrix y:

$$E = s_1(y) = \sum_{i<j} y_{ij}.$$

For an undirected adjacency matrix (like y), we have:

```
s1 <- sum(y[,]) / 2; s1
```

```
## [1] 15
```

or using the `summary` function included in the `ergm` package:

```
s1 <- summary(y ~ edges); s1
```

```
## edges
##    15
```

The `summary` function can be used to get general information about the network:

```
summary(y)
```

```
## Network attributes:
##   vertices = 13
##   directed = FALSE
##   hyper = FALSE
##   loops = FALSE
##   multiple = FALSE
##   bipartite = FALSE
##  total edges = 15
##    missing edges = 0
##    non-missing edges = 15
##  density = 0.1923077
##
## Vertex attributes:
##   vertex.names:
##    character valued attribute
##    13 valid vertex names
##
## No edge attributes
##
## Network adjacency matrix:
##    1 2 3 4 5 6 7 8 9 10 11 12 13
## 1  0 0 1 1 1 0 0 0 1  0  0  0  1
## 2  0 0 0 1 0 0 0 0 0  1  0  0  0
## 3  1 0 0 0 0 0 0 0 0  0  1  0  0
## 4  1 1 0 0 1 0 1 1 0  1  0  1  0
## 5  1 0 0 1 0 0 0 0 0  0  0  0  0
## 6  0 0 0 0 0 0 0 0 0  0  0  1  0
## 7  0 0 0 1 0 0 0 1 0  0  0  0  0
## 8  0 0 0 1 0 0 1 0 0  0  0  0  0
## 9  1 0 0 0 0 0 0 0 0  0  0  0  0
## 10 0 1 0 1 0 0 0 0 0  0  0  0  0
## 11 0 0 1 0 0 0 0 0 0  0  0  0  0
## 12 0 0 0 1 0 1 0 0 0  0  0  0  0
## 13 1 0 0 0 0 0 0 0 0  0  0  0  0
```

3.2 Random graph model

Definition - The likelihood of a random graph model represents the probability distribution of a network graph y given a parameter η:

$$p(y \mid \eta) = \prod_{i<j} \eta^{y_{ij}}(1-\eta)^{1-y_{ij}} = \eta^{s_1(y)}(1-\eta)^{D-s_1(y)}$$

where $0 \leq \eta \leq 1$.

3.2.1 Estimation

To estimate η we can just calculate the **density** of the network, i.e., the proportion of edges:

$$\hat{\eta} = \frac{s_1(y)}{D} :$$

```
eta <- s1 / D; eta
```

```
##     edges
## 0.1923077
```

this is equivalent to:

```
eta <- summary(y ~ density); eta
```

```
##   density
## 0.1923077
```

3.2.2 Natural parametrisation

Now consider $\theta = \text{logit}(\eta)$:

$$\eta = \frac{\exp\{\theta\}}{1+\exp\{\theta\}},$$

then the random graph model can be written as an exponential family distribution:

$$p(y \mid \theta) = \prod_{i<j} \frac{\exp\{\theta\}^{y_{ij}}}{1+\exp\{\theta\}} = \frac{\exp\{\theta^t\ s_1(y)\}}{c(\theta)}$$

where $-\infty < \theta < +\infty$ and

$$c(\theta) = \sum_{y' \in \mathcal{Y}} \exp\left\{\theta^t\ s_1(y')\right\}$$

is a normalising constant.

Let's get an estimate of θ using the `ergm` function:

```
theta <- ergm(y ~ edges); theta$coef
```

```
## Evaluating log-likelihood at the estimate.
```

```
##     edges
## -1.435085
```

Let's check whether $\frac{\exp\{\hat{\theta}\}}{1+\exp\{\hat{\theta}\}} = \hat{\eta}$:

```
round(exp(theta$coef) / (1 + exp(theta$coef)), 4) == round(eta, 4)
```

```
## edges
##  TRUE
```

3.2.3 Parameter interpretation

The parameter estimate θ provide insights about the contribution of the edge statistic $s_1(y)$ to edge formation.

When $\theta = 0$, probability of observing an edge between two nodes i and j is:

$$\Pr(Y_{ij} = 1 \mid \theta = 0) = \frac{\exp\{0\}}{1 + \exp\{0\}} = \eta = 0.5.$$

where Y_{ij} is an node pair in Y and Y_{-ij} is the rest of the network.

A negative (positive) value of θ means that the probability of observing a network with a higher number of edges is lower (higher) than the probability of observing a network whose probability of observing an edge between two nodes is 0.5:

$$\Pr(Y_{ij} = 1 \mid \theta < 0) = \eta < 0.5,$$
$$\Pr(Y_{ij} = 1 \mid \theta > 0) = \eta > 0.5.$$

Let's take a look at the summary of model fit using the `summary` function:

```
summary(theta)
```

```
##
## ==========================
## Summary of model fit
## ==========================
##
## Formula:   y ~ edges
##
## Iterations:  5 out of 20
##
## Monte Carlo MLE Results:
##       Estimate Std. Error MCMC % p-value
## edges  -1.4351     0.2873      0  <1e-04 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##      Null Deviance: 108.13  on 78  degrees of freedom
##  Residual Deviance:  76.37  on 77  degrees of freedom
##
## AIC: 78.37    BIC: 80.73    (Smaller is better.)
```

In this example, $\theta \approx -1.4351$ corresponding to $\eta \approx 0.1923$. So we can conclude that the network is *sparse* as the probability to observe an edge connecting any pair of nodes is about 20%.

3.3 Network simulation

The estimate $\hat{\eta}$ is the probability of observing an edge between any pair of nodes in our N-node network.

To simulate from the estimated random graph model we can simply simulate $N \times N - N$ Bernoulli trials and arrange them into a $N \times N$ adjacency matrix:

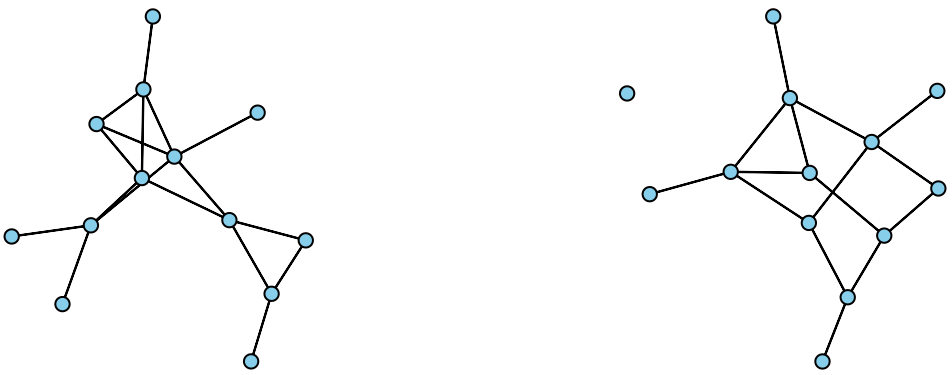
```
set.seed(11)
adj.sim <- matrix(rbinom(N * N, size = 1, prob = eta), N, N)
diag(adj.sim) <- 0
y.sim <- network(adj.sim, directed = FALSE)
```

Alternatevely, you can use the `simulate` function (more info: `?simulate.ergm`) included in the `ergm` package to simulate a realisation from the estimated model using $\hat{\theta}$:

```
set.seed(11)
y.sim.1 <- simulate(network(N, directed = FALSE) ~ edges,
                    coef = theta$coef)
```

Let's have a look at the two network graphs simulated:

```
set.seed(11)
par(mfrow = c(1, 2))
plot(y.sim, vertex.cex = 2, vertex.col = 'skyblue', main = 'y.sim')
plot(y.sim.1, vertex.cex = 2, vertex.col = 'skyblue', main = 'y.sim.1')
```



Now, let's simulate 50 networks from the estimated random graph model:

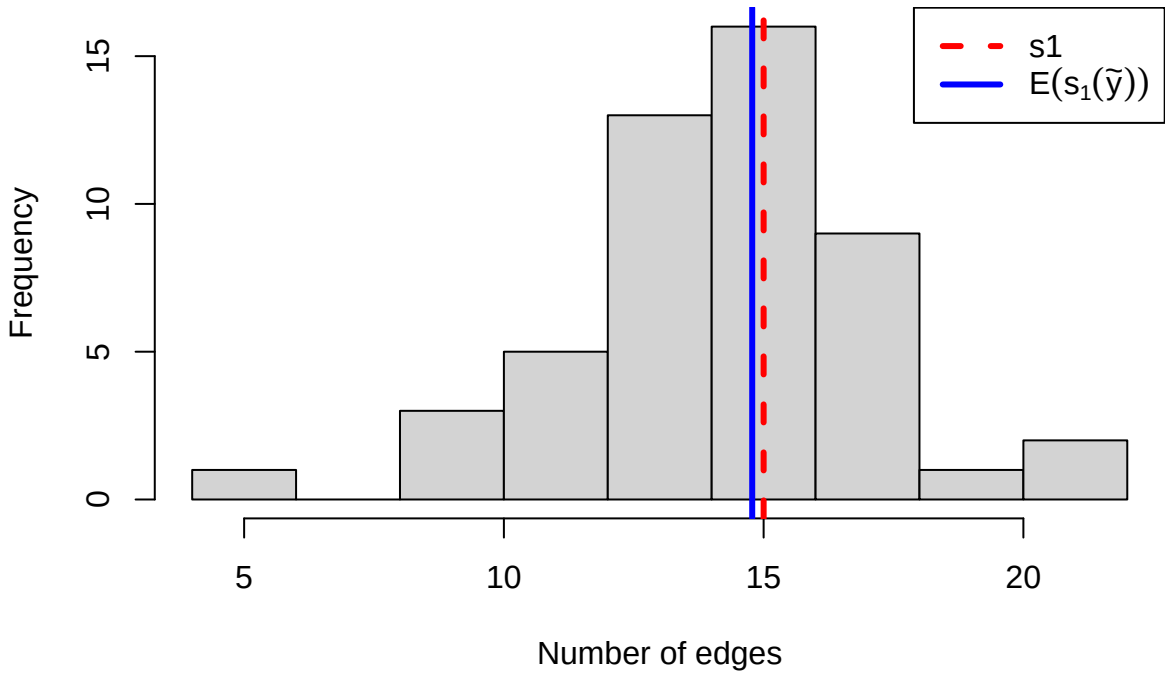
```
set.seed(11)
y.sim.1_50 <- simulate(network(N, directed = FALSE) ~ edges,
                   coef = theta$coef,
                   statsonly = TRUE, # returns only the values of the network statistics
                   nsim = 50) # number of network simulated
summary(y.sim.1_50)
```

```
##      edges
##  Min.   : 5.00
##  1st Qu.:13.25
##  Median :15.00
##  Mean   :14.78
##  3rd Qu.:16.00
##  Max.   :22.00
```

3.3.1 Histogram

We can plot a histogram to visualise the distribution of the number of edges of the 50 simulated networks:

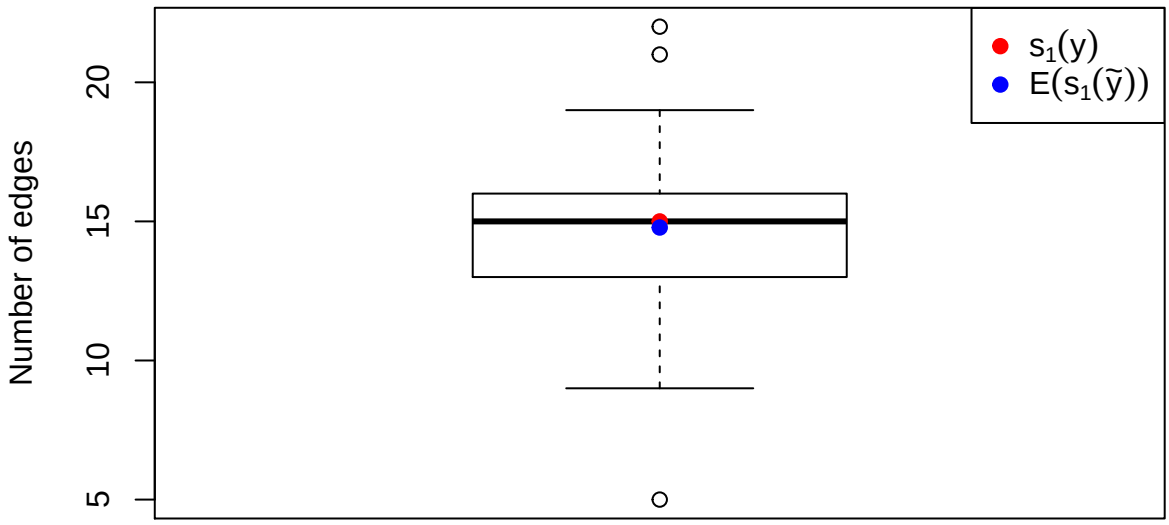
```
hist(y.sim.1_50, xlab = "Number of edges", main = "", col = "lightgray")
abline(v = s1, col = "red", lwd = 3, lty = 2)
abline(v = mean(y.sim.1_50), col = "blue", lwd = 3, lty = 1)
legend("topright",
       c(expression(s1), expression(E(s[1](tilde(y))))),
       lty = c(2, 1),
       col = c("red", "blue"), lwd = 3)
```



3.3.2 Boxplot

Alternatively, we can plot a boxplot:

```
boxplot(y.sim.1_50, ylab = "Number of edges")
points(s1, col = "red", pch = 19)
points(mean(y.sim.1_50), col = "blue", pch = 19)
legend("topright",
       c(expression(s[1](y)), expression(E(s[1](tilde(y))))),
       pch = c(19, 19),
       col = c("red", "blue"))
```



3.4 Network statistics

Due to the complexity of networks, it is necessary to reduce the information to describe essential properties of the network.

Usually this is done via *network statistics* a series of counts of sub-graph configurations, catching the relevant information.

A network statistic:

- should **describe essential properties** of the network. A certain property should be described in a compact and handy form. We would like to forget the exact structure of the underlying graph and concentrate on a restricted set of statistics.
- should **differentiate between certain classes** of networks. A quite common question in network analysis regards the type of the measured network and how to generate models for it. This requires the decision about whether a generated network is similar to another one. In many situations, this may be done by identifying several statistics, which are invariant in the class of networks of interest.
- may be **useful in algorithms and applications**. Some network statistics may be used for algorithms or calculations on the graph. Or they might indicate which graph elements have certain properties regarding the application.

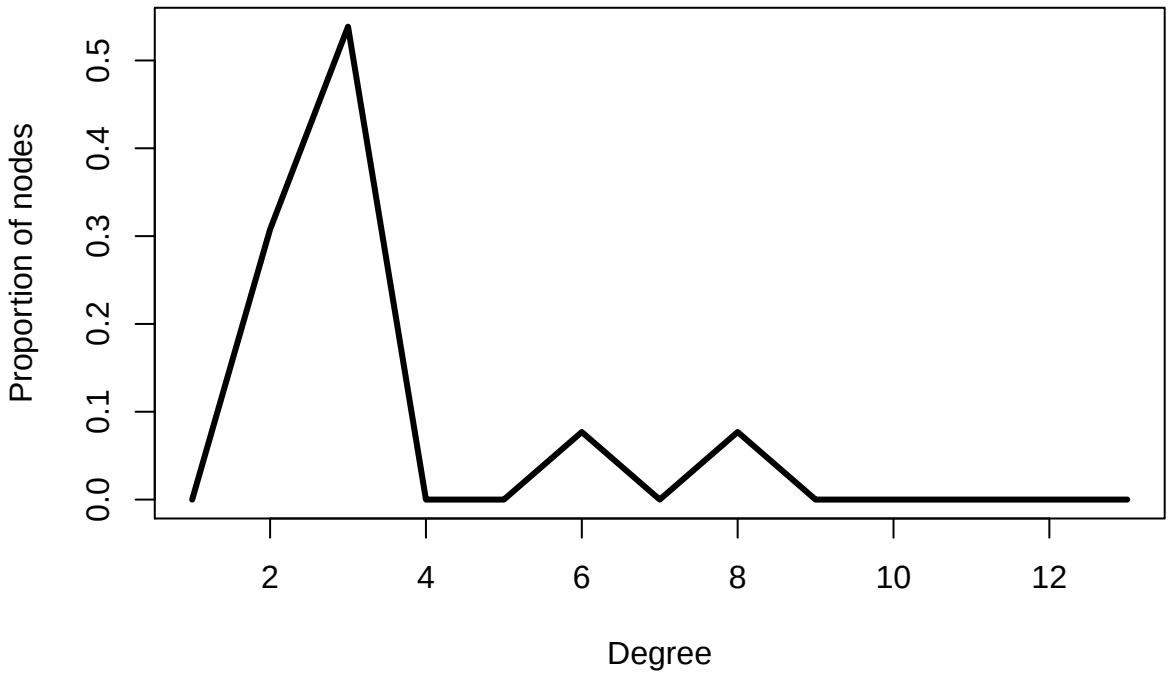
3.4.1 Degree distribution

The degree distribution for a network consists of the values

$$\frac{D_0}{N}, \frac{D_1}{N}, \ldots, \frac{D_{N-1}}{N}$$

where $\frac{D_k}{N}$ equals the proportion of nodes that share edges with exactly k other nodes.

```
Deg.dist <- summary(y ~ degree(0:(N-1)))
plot(Deg.dist / N, type = "l", lwd = 3,
     xlab = "Degree",
     ylab = "Proportion of nodes")
```



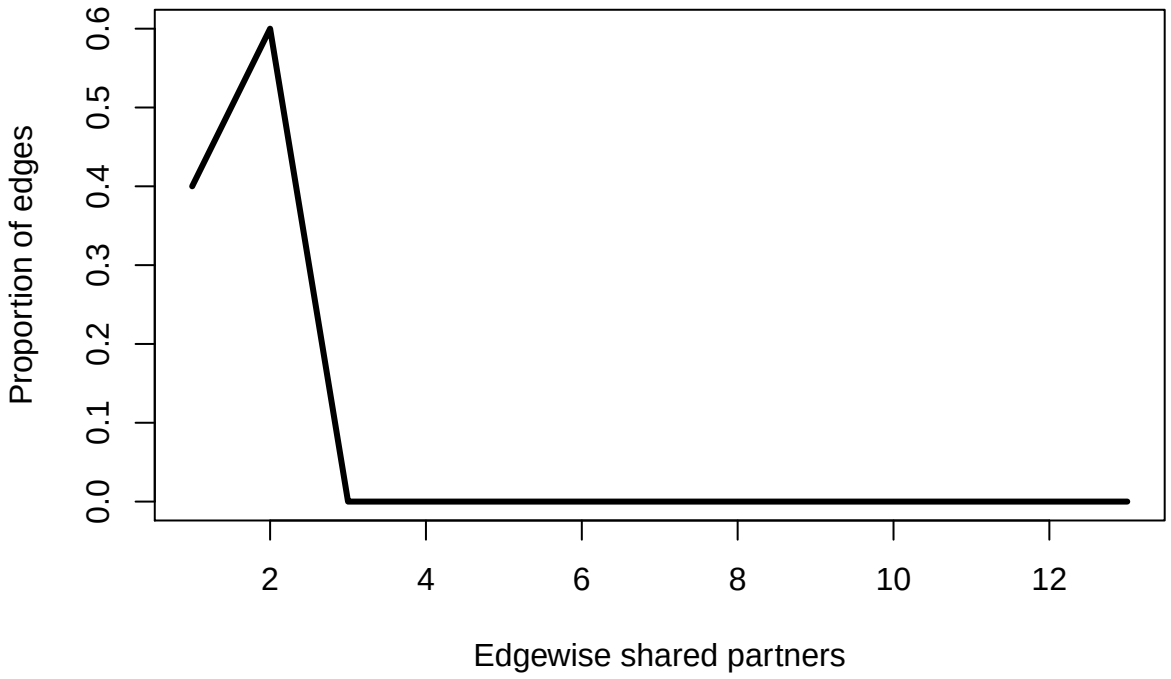
3.4.2 Edgewise shared partners distribution

The edgewise shared partner distribution consists of the values

$$\frac{EP_0}{s_1(y)}, \frac{EP_1}{s_1(y)} \ldots, \frac{EP_{N2}}{s_1(y)}$$

where $s_1(y)$ denotes the number of edges and EP_k equals the number of edges whose endpoints both share edges with exactly k other nodes.

```
Ep.dist <- summary(y ~ esp(0:(N - 1)))
plot(Ep.dist / s1, type = "l", lwd = 3,
     xlab = "Edgewise shared partners",
     ylab = "Proportion of edges")
```

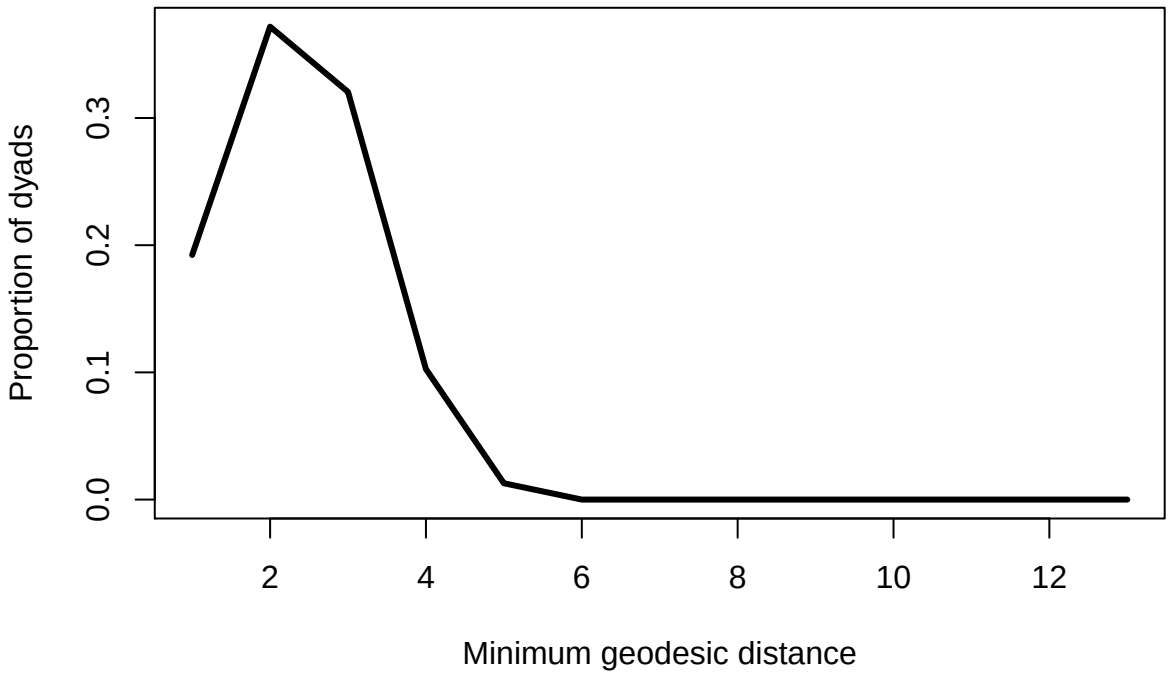


3.4.3 Geodesic distance distribution

The geodesic distance distribution consists of the relative frequencies of the possible values of geodesic distance between two nodes, where the geodesic distance between two nodes equals the length of the shortest path joining those two nodes (or infinity if there is no such path).

For instance, because two nodes are at geodesic distance 1 if and only if they are connected by an edge, and because there are $\binom{N}{2}$ possible pairs of nodes, the first value of the geodesic distance distribution equals $\frac{s_1(y)}{\binom{N}{2}}$. The last value, the fraction of dyads with infinite geodesics, is also called the fraction "unreachable" (NR).

```
Geo.dist <- ergm.geodistdist(y)
plot(Geo.dist / D, type = "l", lwd = 3,
     xlab = "Minimum geodesic distance",
     ylab = "Proportion of dyads")
```



3.5 Goodness of fit diagnostics

A pragmatic way to examine the fit of the data to the estimated model the output obtained is to implement a goodness-of-fit procedure.

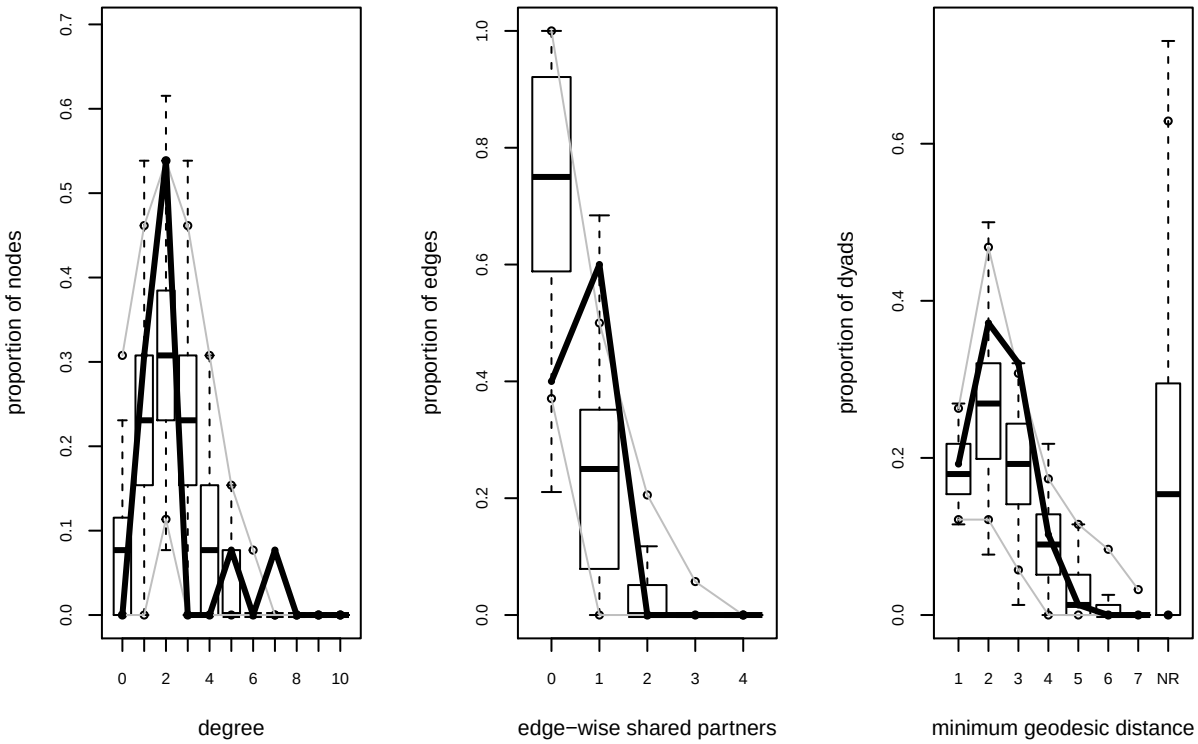
To do this, a set of network graphs are simulated from the maximum likelihood estimate of the parameter $\hat{\theta}$ and compared to the observed graph in terms of high-level network statistics which are not modelled explicitly.

We can use the `gof` function:

```
gof.theta <- gof(theta ~ degree + esp + distance,
                 control.gof.formula(nsim = 100)) # number of simulated networks
```

3.5.1 Graphical diagnostics

```
par(mfrow = c(1, 3))
plot(gof.theta, main = '')
```



3.5.2 Summaries

```
summary(gof.theta)
```

```
##
## Goodness-of-fit for degree
##
##   obs min mean max MC p-value
## 0   0   0 1.00   4       0.74
## 1   4   0 2.86   9       0.58
## 2   7   1 3.87   8       0.08
## 3   0   0 3.15   7       0.08
## 4   0   0 1.45   6       0.54
## 5   1   0 0.50   3       0.76
## 6   0   0 0.15   2       1.00
## 7   1   0 0.01   1       0.02
## 8   0   0 0.01   1       1.00
##
## Goodness-of-fit for edgewise shared partner
##
##      obs min  mean max MC p-value
## esp0   6   4 10.42  18       0.12
## esp1   9   0  3.71  13       0.14
## esp2   0   0  0.50   7       1.00
## esp3   0   0  0.07   1       1.00
##
## Goodness-of-fit for minimum geodesic distance
##
##     obs min  mean max MC p-value
```

```
## 1    15   9 14.70  21       0.94
## 2    29   6 21.04  39       0.32
## 3    25   1 15.19  25       0.02
## 4     8   0  6.92  17       0.90
## 5     1   0  2.40   9       1.00
## 6     0   0  0.77   8       1.00
## 7     0   0  0.16   3       1.00
## 8     0   0  0.01   1       1.00
## Inf   0   0 16.81  58       0.54
```

Chapter 4

Exponential random graph models (ERGMs)

4.1 Introduction

The basci assumption behind exponential random graph models (ERGMs) is that the observed network is generated by a stochastic process in which relational edges come into being in ways that may be shaped by the presence or absence of other edges (and possibly node-level attributes).

In other words, the network is conceptualized as a self-organizing system of relational edges. Substantively, the claim is that there are local processes that generate dyadic relations, and that these processes may depend on the surrounding social environment (i.e. on existing relations).

For example, we can assume that actors with similar attributes are more likely to form friendship edges (homophily), or that if two unconnected actors were connected to a third actor, at some point they are likely to form a friendship tie between them (transitivity). Note that in addition to the assumption of stochasticity, this description is also implicitly temporal and dynamic.

Definition - The likelihood of an ERGM belongs to the exponential family of distributions and represents the probability distribution of a network graph y given a vector of parameters θ:

$$f(y \mid \theta) = \frac{\exp\{\theta^T s(y)\}}{c(\theta)}.$$

This equation states that the probability of observing a given network graph y is equal to the exponent of the observed graph statistics $s(y)$ which is a function of the network data y and covariate information x multiplied by parameter vector θ divided by a normalising constant term $c(\theta)$.

```
require('statnet')
```

4.2 Dependence assumptions

Network statistics $s(y)$ imply different assumptions regarding the network dyadic dependence structure.

4.2.1 Dyadic independence

Dyad independent network statistics imply independence between dyads: the presence or absence of a edge does not depend the connectivity structure of the overall network. The random graph model is a dyadic

independence model as all possible distinct edges in the network are independent of one another:

$$Y_{ij} \mid \eta \sim Bernoulli(\eta),\ \forall i \neq j.$$

4.2.2 Dyadic dependence

Dyadic dependence is an unrealistic assumption in many circumstances.

Dyad dependent network statistics imply dependence between dyads: an edge between i and j is assumed to be dependent on other edges.

For example:

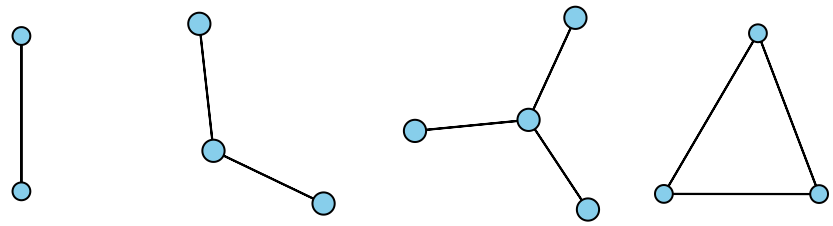
- **stars** statistics assume that an edge between i and j is contingent on any possible edge involving node i and j.
- **triadic** statistics assume that an edge between i and j is contingent on any possible edge involving any node of the network connected to both i and j.

4.3 Network statistics

Common network statistics (undirected networks):

- *edge statistic*: the number of edges in the network. This statistic is commonly density effect
- *k-star statistics*: the number of k-star structures (where generally $k = 2$ or $k = 3$)
- *cyclic statistics*: the number of connected k-cycles (where generally $k = 3 \rightarrow$ triangles)

edge **2−star** **3−star** **triangle**

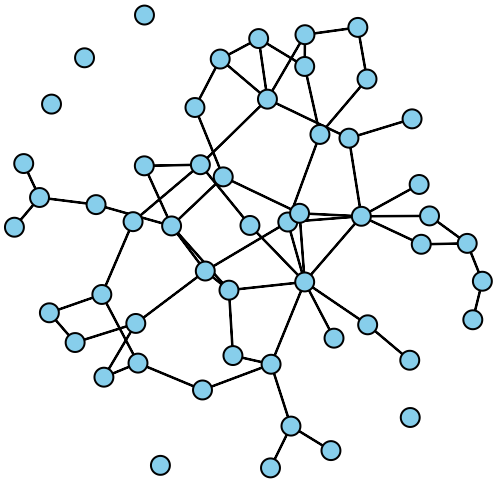


To get a list of network statistics implemented in the `ergm` package:

```
help('ergm-terms', 'ergm')
```

Let's consider a random 53-node undirected network:

```
set.seed(17)
N <- 53
y <- network(N, directed = FALSE)
plot(y, vertex.cex = 2, vertex.col = "skyblue")
```



Let's consider an ERGM with the following network statistics:

- edges: $s_1(y) = \sum_{i<j} y_{ij}$;
- 2-stars: $s_2(y) = \sum_{i<j<k} y_{ij}y_{ik}$;
- triangles: $s_3(y) = \sum_{i<j<k} y_{ij}y_{ik}y_{jk}$.

The vector of observed network statistics $s(y)$ can be calculated using the `summary` function:

```
model.1 <- y ~ edges + kstar(2) + triangle
summary(model.1)
```

```
##    edges   kstar2 triangle
##       64      157        4
```

4.3.1 Higher-order network statistics

Simulation-based inferential procedures have brought to light model degeneracy problems related to ERGMs defined by traditional specification (Schweinberger, 2011). In order to overcome these issues, a new specification of network statistics based on *geometrically weighted* functions of extra-triadic network statistics distributions such as degree distributions have been proposed by Snijders et al. (2006).

Some of the geometrically weighted undirected network statistics that are common in empirical research are:

- *Geometrically weighted degree statistic* (`gwdegree`) is a function of the degree counts $D_d(y)$ defined as the number of nodes in y with degree d:

$$gwdegree(y, \alpha) = \sum_d g_d(\alpha) D_d(y),$$

  where $g_d(y)$ is an exponential weight function defined as:

$$g_d(\alpha) = e^{\alpha} \left\{ 1 - \left(1 - e^{-\alpha}\right)^d \right\}.$$

This statistic expresses a version of preferential attachment (Albert and Barabási, 2002) with edges from low degree to high degree nodes being more probable than edges among low degree nodes.

- *Geometrically weighted edgewise shared partner statistic* (`gwesp`) is a function of the edgewise shared partner statistics $EP_d(y)$ defined as the number of unordered connected pairs (i, j) (partners) that are both connected to exactly d other nodes:

$$gwesp(y, \alpha) = \sum_d g_d(\alpha)\ EP_d(y) = \sum_d g_d(\alpha) \sum_{i<j} y_{ij} \mathbb{I}_{\{\sum_k y_{ik}y_{jk}=d\}},$$

where $\mathbb{I}_{\{\cdot\}}$ is the indicator function. For directed networks the geometric weighting is over homogeneous shared partners only (i.e., only partners on a directed two-path connecting the nodes in the edge and in the same direction).

This statistic provides a suitable representation of transitivity in social networks by taking into account high-order k-triangles.

For example, let's consider the following ERGM:

```
model.2 <- y ~ edges + gwesp(decay = 1, fixed = TRUE)
summary(model.2)
```

```
##         edges gwesp.fixed.1
##      64.00000      11.63212
```

4.3.2 Statistics with nodal attributes variables

There are various ways of introducing node-level effects (node attributes) into ERGMs. We assume an vector X of attribute variables is given.

In *social selection* models, attributes are assumed to be exogenous predictors of network edges (Robins et al., 2001): edges tend to develop between actors with certain attribute values. This means that the interest is in the probability of the network graph y given the observations of attributes x: $\Pr(Y = y|X = x)$.

4.4 Parameter interpretation

The parameter estimates associated with the network effects expressed by the network statistics provide insights about the contribution of each network statistic to edge formation. ERGMs allow to establish a relationship between a binary outcome variable (presence/absence of an edge between nodes) and a group of predictor variables (network statistics). It models the logit-transformed probability as a linear relationship with the predictor variables.

The conditional log-odds of an edge between i and j is:

$$\text{logit}\left(\Pr(Y_{ij} = 1 \mid Y_{-ij}, \theta)\right) = \theta^T \ \Delta(s(y))_{ij},$$

where $\Delta(s(y))_{ij}$ is the change in $s(y)$ when the value of Y_{ij} switch value from 0 to 1. So the coefficient θ can be interpreted as the log-odds of an individual edge conditional on all others.

4.5 Network simulation

The ERGM normalising constant $c(\theta)$ is calculated over the sum of all possible graphs on N nodes and it is therefore extremely difficult to evaluate in presence of extra-dyadic network statistics for all but trivially small graphs.

The simulation of graphs from the parameter values of the model can be generally provided by MCMC algorithms. One of the convenient ways to generate random graphs is by the Metropolis algorithm applied to an initial adjacency matrix $Y^{(0)}$ whose elements dyads are stochastically updated so that at each iteration t, the procedure implies that $Y^{(t-1)}$ and $Y^{(t)}$ differs in only one dyad. This mechanism cycles through the whole matrix so as to produce a distribution $Y^{(T)}$ tends asymptotically to the desired random graph distribution.

In practice the algorithm compares the probability of a proposed graph y' to the current one y_c where y' is selected at each step by proposing a change in the current state of a single dyad (i, j), i.e. creating a

new edge or dropping an old edge and then decides whether or not accept the proposed network with the following acceptance probability:

$$\alpha = \min\left\{1, \frac{p(y' \mid \theta_0)\ q(y_c \to y')}{p(y_c \mid \theta_0)\ q(y' \to y_c)}\right\} = \exp\left\{\theta_0^t\left[s(y') - s(y_c)\right]\right\}$$

where $q(y_c \to y')$ and $q(y' \to y_c)$ denote the probability of y' given y_c and of y_c given y' respectively.

The behaviour of MCMC algorithm is dependent on the choice of the proposal density $q(\cdot)$ and on the network statistics $s(\cdot)$ although it is proven that in principle the Metropolis yields convergence to the target distribution.

Suppose we want to simulate undirected networks on 53 nodes from the same ERGM defined above with parameter values:

- $\theta_1 = -1.9$, parameter related to the edge statistic $s_1(y)$;
- $\theta_2 = -0.1$, parameter related to the 2-star statistic $s_2(y)$;
- $\theta_3 = 0.4$, parameter related to the triangle statistics $s_3(y)$.

We can use the `simulate` function to do that:

```
set.seed(11)
y.sim <- simulate(network(N, directed = FALSE) ~ edges + kstar(2) + triangle,
                  coef = c(-1.9, -0.1, 0.4),
                  statsonly = TRUE,
                  nsim = 1000,
                  control = control.simulate.formula(MCMC.burnin = 1000,    # burnin
                                                     MCMC.interval = 100)) # mcmc thinning interval
summary(y.sim)
```

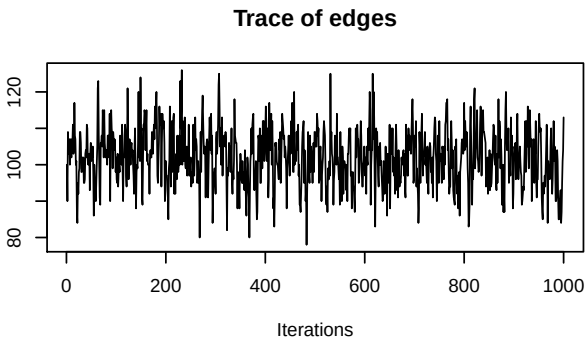
```
##      edges          kstar2         triangle
##  Min.   : 78   Min.   :214.0   Min.   : 0.00
##  1st Qu.: 97   1st Qu.:326.8   1st Qu.: 8.00
##  Median :102   Median :365.0   Median :11.00
##  Mean   :102   Mean   :367.1   Mean   :11.18
##  3rd Qu.:107   3rd Qu.:405.0   3rd Qu.:14.00
##  Max.   :126   Max.   :597.0   Max.   :31.00
```

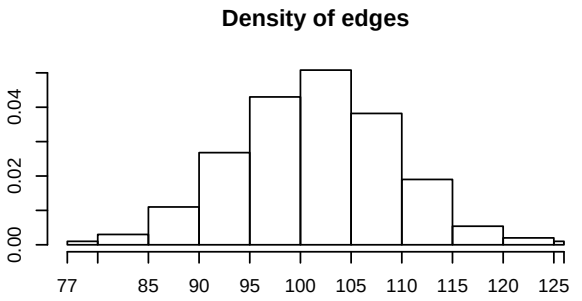
We can analyse the MCMC output diagnostics using the `coda` package:

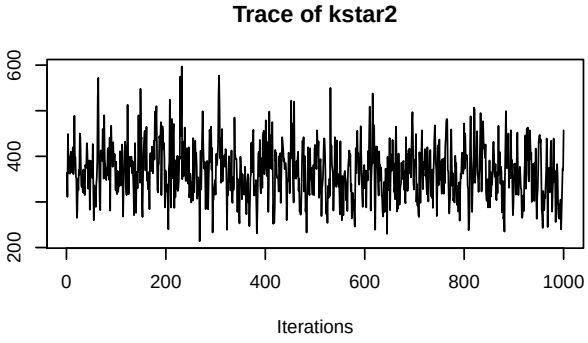
```
require(coda)
```

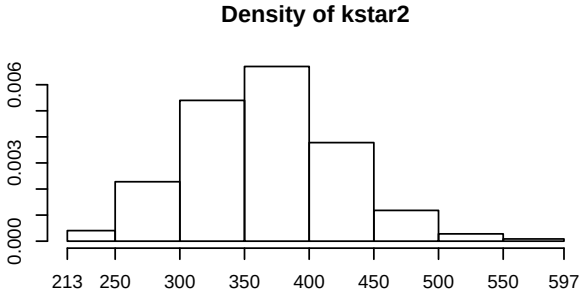
4.5.1 Trace and density plots

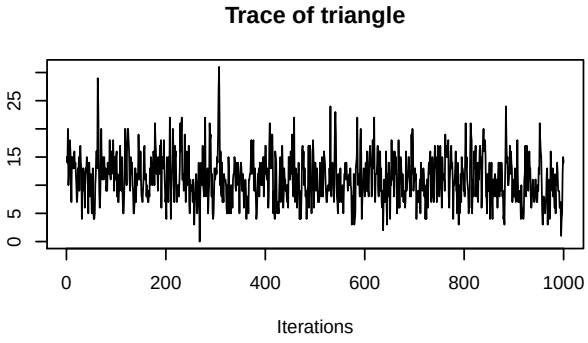
```
plot(mcmc(y.sim))
```

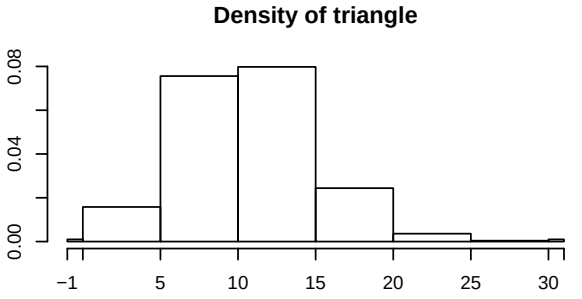






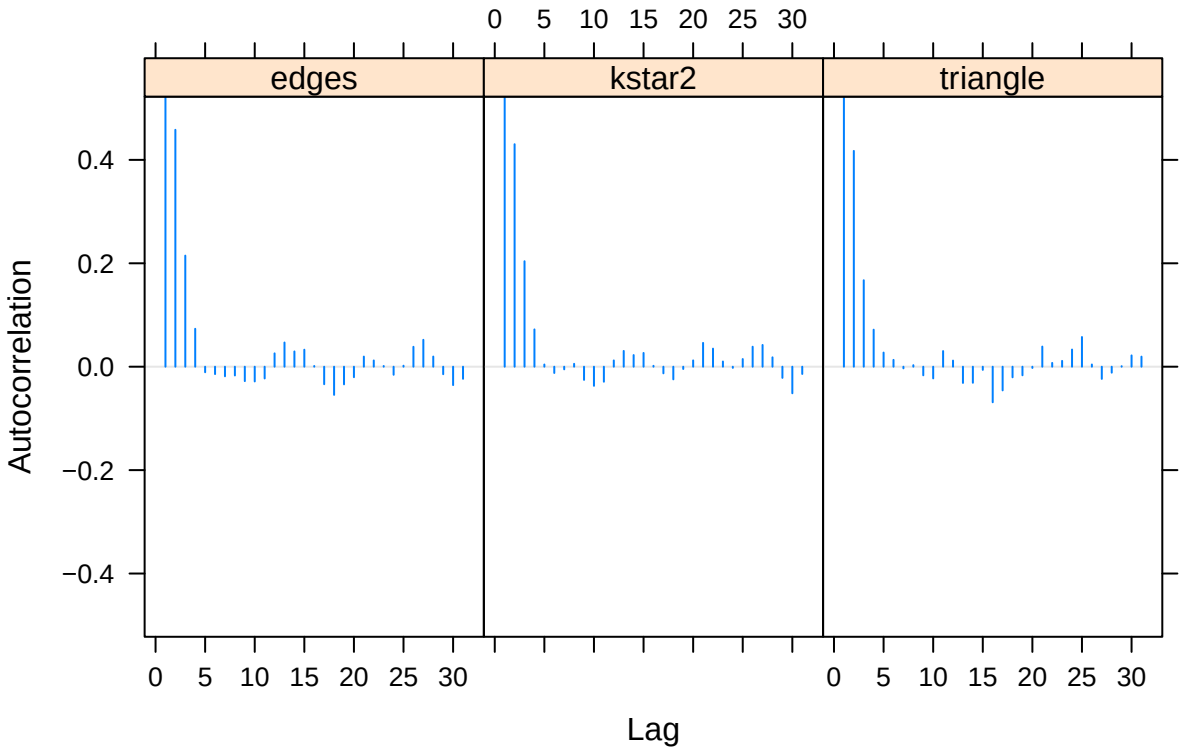






4.5.2 Auto-correlation plots

```
acfplot(mcmc(y.sim))
```


edges
kstar2
triangle
Autocorrelation
0.4
0.2
0.0
−0.2
−0.4
0 5 10 15 20 25 30
Lag

Chapter 5

Maximum likelihood estimation for ERGMs

The estimation procedure starts assuming that the observed network y is just one realization of a set of many possible networks that could have been formed by the same underlying structural processes. Each network statistic $s(y)$ is conceptualised as having a particular probability of occurring which is incorporated into the model as a parameter vector θ.

The aim of the estimation is to find accurately a parameter value so that the observed network has the highest possible likelihood of being simulated by the given model. In other words, when attempting to generate an ERGM we find the set of parameters which maximise the probability that any random network graph generated by simulating from the ERGM will be identical to the observed network in terms of network statistics:

$$\mathbb{E}_{y \mid \theta}\left[s(y)\right] = s(y)$$

where $\mathbb{E}_{y \mid \theta}$ is the expectation under the likelihood distribution $p(y \mid \theta)$ so that it ensures that the probability mass of the ERGM is centred at $s(y)$.

In practice, an exact solution for the ERGM is actually impossible to calculate directly due to the **intractability** of the normalising constant $c(\theta)$.

5.1 Monte Carlo maximum likelihood estimation (MC-MLE)

The most advanced classical estimation method used is the Monte Carlo Maximum Likelihood Estimation (MC-MLE) procedure proposed by Geyer and Thompson (1992).

This involves the simulation of a set of random networks from a starting set of estimated parameter values θ_0, and then the progressive refinement (until convergence) of the parameter values by measuring how closely these networks match the observed network.

Let's denote with $q_\theta(y) = \exp\{\theta^t s(y)\}$ the unnormalised likelihood. Mathematically, the key identity of the MC-MLE method is the following:

$$\frac{c(\theta)}{c(\theta_0)} = \mathbb{E}_{y \mid \theta_0}\left[\frac{q_\theta(y)}{q_{\theta_0}(y)}\right] = \sum_y \frac{q_\theta(y)}{q_{\theta_0}(y)}\frac{q_{\theta_0}(y)}{z(\theta_0)} \tag{5.1}$$
$$\approx \frac{1}{m}\sum_{i=1}^{m} \exp\left\{(\theta-\theta_0)^t s(y_i)\right\}$$

where $q(\cdot)$ is the unnormalised likelihood, θ_0 is fixed set of parameter values, and $\mathbb{E}_{y \mid \theta_0}$ denotes an expectation taken with respect to the distribution $p(y \mid \theta_0)$. In practice this ratio of normalising constants is approximated using graphs $y_1, \ldots, y_m$ sampled via MCMC from $p(y \mid \theta_0)$ and importance sampling.

This yields the following approximated log likelihood ratio:

$$w_{\theta_0}(\theta) = \ell(\theta) - \ell(\theta_0) \approx (\theta-\theta_0)^t s(y) - \log\left[\frac{1}{m}\sum_{i=1}^{m}\exp\left\{(\theta-\theta_0)^t s(y_i)\right\}\right] \tag{5.2}$$

where $\ell(\cdot)$ is the log-likelihood. w_{θ_0} is a function of θ, and its maximum value serves as a Monte Carlo estimate of the MLE.

A crucial aspect of this algorithm is the choice of θ_0. Ideally θ_0 should be very close to the maximum likelihood estimator of θ. Viewed as a function of θ, $w_{\theta_0}(\theta)$ is very sensitive to the choice of θ_0. A poorly chosen value of θ_0 may lead to an objective function that cannot even be maximised.

The result of the MC-MLE procedure is a set of estimated parameter values and relative standard errors which measure the degree of accuracy and significance of the estimates.

To implement the MC-MLE procedure we can use the `ergm` function:

```
set.seed(11)
model.2 <- y ~ edges +
               gwesp(decay = 1, fixed = TRUE) +
               gwnsp(decay = 1, fixed = TRUE)
MLE.2 <- ergm(model.2)
```

```
## Starting maximum likelihood estimation via MCMLE:
## Iteration 1 of at most 20:
## The log-likelihood improved by 0.004112
## Step length converged once. Increasing MCMC sample size.
## Iteration 2 of at most 20:
## The log-likelihood improved by 0.0082
## Step length converged twice. Stopping.
## Evaluating log-likelihood at the estimate. Using 20 bridges: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 2
##
## This model was fit using MCMC.  To examine model diagnostics and check for degeneracy, use the mcmc.diagnos
```

```
summary(MLE.2)
```

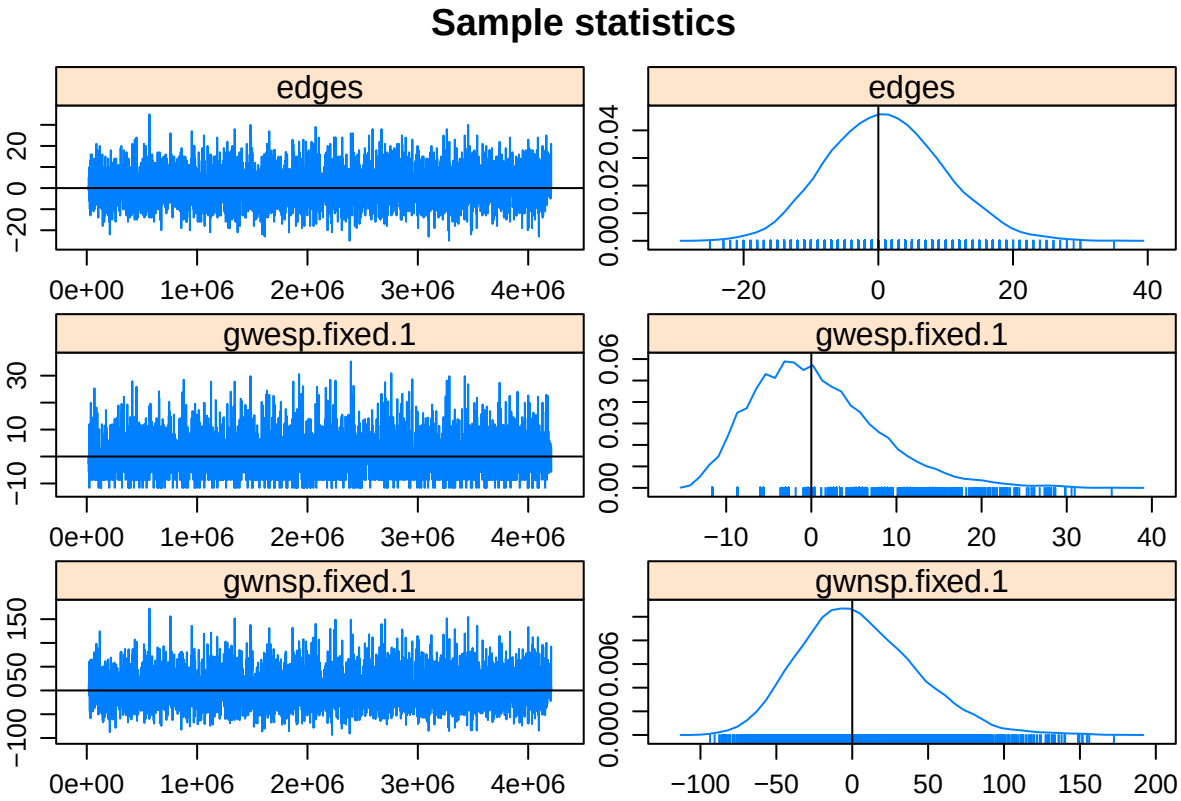
```
##
## ==========================
## Summary of model fit
## ==========================
##
## Formula:   y ~ edges + gwesp(decay = 1, fixed = TRUE) + gwnsp(decay = 1,
##     fixed = TRUE)
##
## Iterations:  2 out of 20
```

```
##
## Monte Carlo MLE Results:
##                Estimate Std. Error MCMC % p-value
## edges         -3.126974   0.414423      0  <1e-04 ***
## gwesp.fixed.1  0.194962   0.179318      0   0.277
## gwnsp.fixed.1  0.004851   0.089213      0   0.957
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##      Null Deviance: 1910  on 1378  degrees of freedom
##  Residual Deviance:  517  on 1375  degrees of freedom
##
## AIC: 523    BIC: 538.6    (Smaller is better.)
```

```
mcmc.diagnostics(MLE.2)
```

```
## Sample statistics summary:
##
## Iterations = 16384:4209664
## Thinning interval = 1024
## Number of chains = 1
## Sample size per chain = 4096
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##                 Mean     SD Naive SE Time-series SE
## edges         1.0830  8.514   0.1330         0.1265
## gwesp.fixed.1 0.5493  7.271   0.1136         0.1136
## gwnsp.fixed.1 4.4974 38.127   0.5957         0.5957
##
## 2. Quantiles for each variable:
##
##                 2.5%     25%       50%    75% 97.5%
## edges         -15.00  -5.000 1.000e+00  7.000 18.00
## gwesp.fixed.1 -11.63  -5.632 3.451e-13  5.264 17.26
## gwnsp.fixed.1 -60.49 -22.240 8.964e-01 28.528 85.71
##
##
## Sample statistics cross-correlations:
##                   edges gwesp.fixed.1 gwnsp.fixed.1
## edges         1.0000000     0.6410327     0.9557442
## gwesp.fixed.1 0.6410327     1.0000000     0.6050813
## gwnsp.fixed.1 0.9557442     0.6050813     1.0000000
##
## Sample statistics auto-correlation:
## Chain 1
##                 edges gwesp.fixed.1 gwnsp.fixed.1
## Lag 0     1.000000000   1.000000000  1.000000e+00
## Lag 1024  0.003546387   0.020586964  3.443736e-05
## Lag 2048 -0.021609688  -0.007190510 -2.622667e-02
## Lag 3072 -0.032834876   0.007026375 -2.331004e-02
## Lag 4096 -0.013443575  -0.009443708 -1.369215e-02
## Lag 5120  0.008690002  -0.010584478  1.835021e-02
##
```

```
## Sample statistics burn-in diagnostic (Geweke):
## Chain 1
##
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
##         edges gwesp.fixed.1 gwnsp.fixed.1
##       -1.7545       -0.9937       -2.2127
##
## Individual P-values (lower = worse):
##         edges gwesp.fixed.1 gwnsp.fixed.1
##    0.07934137    0.32037506    0.02691517
## Joint P-value (lower = worse):  0.1506554 .
```




```
##
## MCMC diagnostics shown here are from the last round of simulation, prior to computation of final parameter
```

5.2 Goodness of fit diagnostics

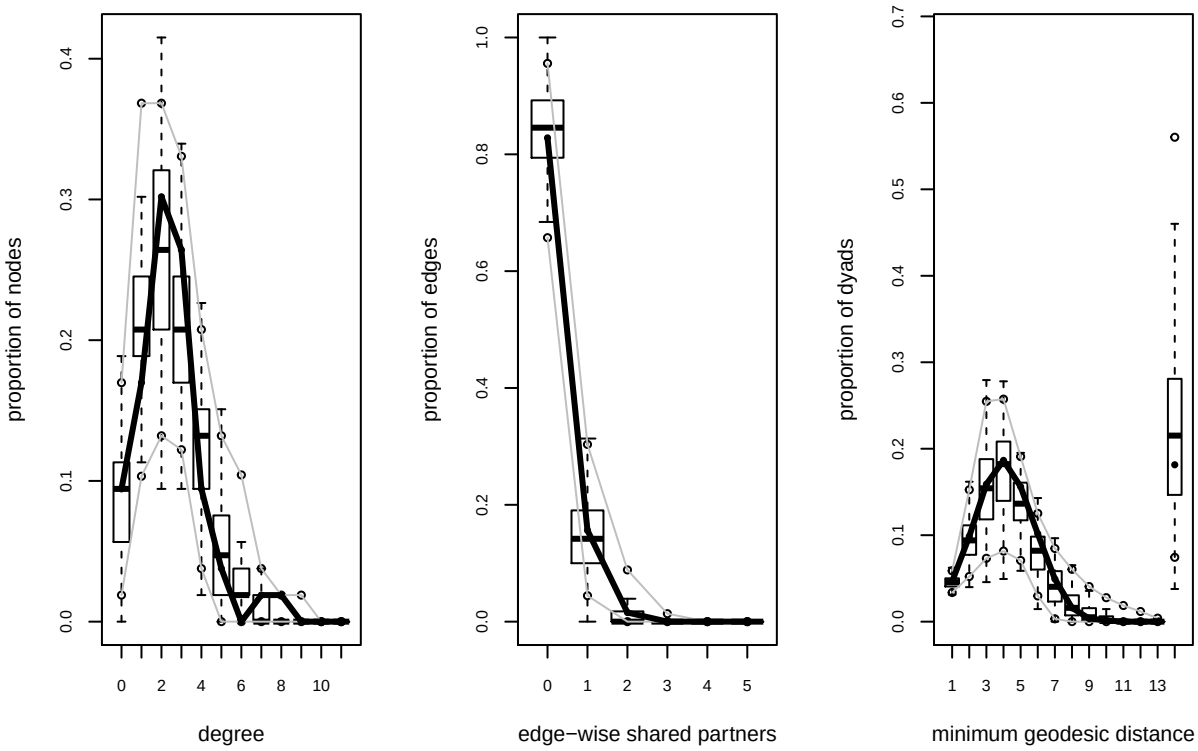
If the estimated ERGM is a good fit to the observed data, then networks simulated $y_1, \ldots, y_m$ from its MLE should resemble the connectivity structure of the observed data y.

To do this, $m = 100$ graphs are simulated from the maximum likelihood estimate of the parameter vector $\hat{\theta}$ and compared to the observed graph in terms of high-level network statistics which are not modelled explicitly:

```
model.2.gof <- gof(MLE.2 ~ degree + esp + distance,
                   control.gof.formula(nsim = 100))
```

5.2.1 Graphical diagnostics

```
par(mfrow = c(1, 3))
plot(model.2.gof, main = '')
```



5.2.2 Summaries

```
summary(model.2.gof)
```

```
##
## Goodness-of-fit for degree
##
##    obs min  mean max MC p-value
## 0    5   0  4.79  10       1.00
## 1    9   4 11.80  22       0.42
## 2   16   5 14.10  22       0.72
## 3   14   5 10.95  20       0.36
## 4    5   1  6.46  12       0.70
## 5    2   0  2.68  10       1.00
## 6    0   0  1.53   7       0.44
## 7    1   0  0.48   2       0.74
## 8    1   0  0.13   2       0.22
## 9    0   0  0.07   1       1.00
## 10   0   0  0.01   1       1.00
##
## Goodness-of-fit for edgewise shared partner
##
##      obs min  mean max MC p-value
```

```
## esp0  53  35 52.57  76       0.96
## esp1  10   0  9.61  29       0.82
## esp2   1   0  0.97   8       0.86
## esp3   0   0  0.05   1       1.00
##
## Goodness-of-fit for minimum geodesic distance
##
##     obs min   mean max MC p-value
## 1    64  43  63.20  86       0.96
## 2   136  55 132.85 234       0.88
## 3   219  63 213.89 385       0.86
## 4   257  68 239.57 383       0.90
## 5   215  59 185.51 269       0.58
## 6   140  20 109.02 197       0.42
## 7    68   0  57.13 133       0.72
## 8    22   0  28.33 109       1.00
## 9     5   0  14.27  86       0.92
## 10    2   0   6.77  68       0.92
## 11    0   0   3.15  44       1.00
## 12    0   0   1.38  22       1.00
## 13    0   0   0.40   9       1.00
## 14    0   0   0.06   2       1.00
## Inf 250  52 322.47 931       0.84
```

Chapter 6

Bayesian inference for ERGMs

6.1 Parameter inference via approximate exchange algorithm

Bayesian inference for ERGMs is based on the posterior distribution of θ given the data y:

$$p(\theta|y) = p(y|\theta)\ \frac{p(\theta)}{p(y)} = \frac{\exp\{\theta^t s(y)\}}{z(\theta)}\ \frac{p(\theta)}{p(y)},$$

where $p(\theta)$ is the prior parameter distribution and $p(y)$ is the evidence or marginal likelihood of y which is typically intractable.

Standard MCMC methods such as the Metropolis-Hastings algorithm, can deal with posterior estimation as long as the target posterior density is known up to the model evidence $p(y)$. Unfortunately in the ERGM context the posterior density $p(\theta|y)$ of non-trivially small ERGMs includes two intractable normalising constants, the model evidence $p(y)$ and $z(\theta)$. For this reason, the ERGM posterior density is **doubly intractable**.

In order to carry out Bayesian inference for ERGMs, the `Bergm` package makes use of MCMC techniques (Caimo and Friel, 2011). The *approximate exchange algorithm* circumvents the problem of computing the normalising constants of the ERGM likelihoods, while the use of multiple chains and efficient adaptive proposal strategies are able to speed up the computations and improve chain mixing quite significantly.

Let's denote $q_\theta(y) = \exp\{\theta^t s(y)\}$ the unnormalised likelihood. The approximate exchange algorithm implemented by the `bergm` function can be summarised in the following way:

1. Gibbs update of (θ', y')

   i) Draw $\theta' \sim h(\cdot|\theta)$

   ii) Approximately draw $y' \sim f(\cdot|\theta')$ via MCMC

2. Accept move from θ to θ' with probability:

$$1 \wedge \frac{q_{\theta'}(y)}{q_\theta(y)} \frac{p(\theta')}{p(\theta)} \frac{h(\theta|\theta')}{h(\theta'|\theta)} \frac{q_\theta(y')}{q_{\theta'}(y')} \times \underbrace{\frac{z(\theta)z(\theta')}{z(\theta')z(\theta)}}_{1}.$$

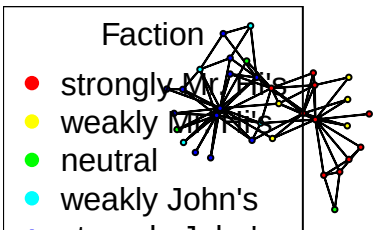
6.1.1 Parallel adaptive direction sampler

In order to improve mixing a parallel adaptive direction sampler (ADS) (Gilks et al., 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 $\{\theta_1, \ldots, \theta_H\}$ with target distribution $p(\theta_1|y) \otimes$

$\cdots \otimes p(\theta_H|y)$. A parallel ADS move consists of generating a new value θ'_h from the difference of two parameters θ_{h_1} and θ_{h_2} (randomly selected from other chains) multiplied by a scalar term γ which is called parallel ADS move factor plus a random term ϵ called parallel ADS move parameter.

Consider the Zachary Karate club social network consisting of social relations in a university karate club involving 34 individuals. The nodal covariate `faction.id` is encoding the faction alignment coded numerically, as -2 (strongly Mr. Hi's), -1 (weakly Mr. Hi's), 0 (neutral), +1 (weakly John's), and +2 (strongly John's):

```
library(Bergm); library(statnet)
data(zach)
y <- zach
y.node.col <- rainbow(6)[y %v% "faction.id" + 3]
set.seed(11)
plot(y, vertex.col = y.node.col, vertex.cex = 1.3)
legend("topleft",
       pch = 16,
       col = rainbow(6)[sort(unique(y %v% "faction.id") + 3)],
       legend = c("strongly Mr. Hi's",
                  "weakly Mr. Hi's",
                  "neutral",
                  "weakly John's",
                  "strongly John's"),
       title = 'Faction')
```



and consider the following model:

```
model.3 <- y ~ edges +
               nodematch("faction.id") +
               gwesp(decay = log(2), fixed = TRUE)
```

6.2 Prior specification

We can also set up the prior mean and variance/covariance structure. The prior distribution used by the `Bergm` package is the normal distribution. In this case, we set the prior mean to be $\bar{\theta} = (-1, 0.5, 0)$ (corresponding to negative density and positive density withing factions and positive transitivity) and the covariance matrix of each model to be a diagonal matrix with every entry equal to 4. So

$$\theta \sim N\left(\begin{bmatrix}-1\\0.5\\0.5\end{bmatrix}, \begin{bmatrix}4 & 0 & 0\\0 & 4 & 0\\0 & 0 & 4\end{bmatrix}\right)$$

```
mean.prior <- c(-1, 0.5, 0.5)
sigma.prior <- diag(4, 3)
```

To implement the Bayesian parameter inference procedure we can use the `bergm` function:

```
post <- bergm(model.3,
              aux.iters = 2000, # number of iterations for step 1.ii
              burn.in = 100, # number of burnin iterations for each chain
              main.iters = 3000, # number of iterations for each chain (after burnin)
              nchains = 5, # number of chains
              gamma = 0.8, # gamma parameter (tuned to get ~20% acceptance probability)
              mean.prior = mean.prior,
              sigma.prior = sigma.prior)
```

It is possible to visualise the results of the MCMC estimation by using the `bergm.output` function which is based on the `coda` package:

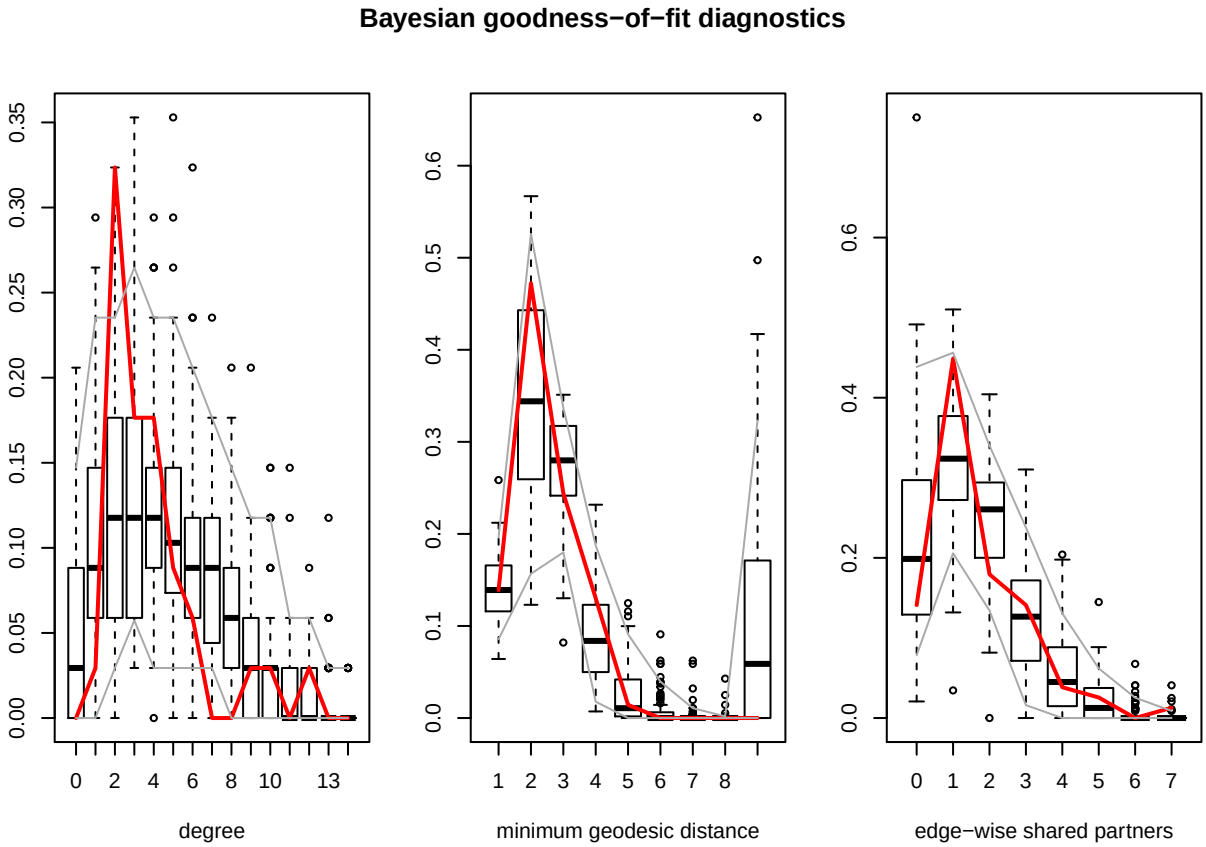
```
bergm.output(post)
```

```
##
##  Posterior Density Estimate for Model: y ~ edges + nodematch("faction.id") + gwesp(decay = log(2), fixed =
##
##                                            Mean        SD    Naive SE
## theta1 (edges)                          -3.2221441 0.2548631 0.002080948
## theta2 (nodematch.faction.id)            1.0492332 0.2318037 0.001892669
## theta3 (gwesp.fixed.0.693147180559945)   0.6070649 0.1359139 0.001109732
##                                         Time-series SE
## theta1 (edges)                             0.012117759
## theta2 (nodematch.faction.id)              0.011372732
## theta3 (gwesp.fixed.0.693147180559945)     0.006532328
##
##                                             2.5%        25%        50%
## theta1 (edges)                          -3.7453945 -3.3850316 -3.2117128
## theta2 (nodematch.faction.id)            0.5937836  0.8970344  1.0494414
## theta3 (gwesp.fixed.0.693147180559945)   0.3594777  0.5150651  0.6046604
##                                              75%      97.5%
## theta1 (edges)                          -3.052811 -2.7196526
## theta2 (nodematch.faction.id)            1.205293  1.5186659
## theta3 (gwesp.fixed.0.693147180559945)   0.695165  0.8978658
##
##  Acceptance rate: 0.1824667
##
##
```

6.3 Bayesian goodness of fit diagnostics

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 y are compared with a set of networks $y_1, y_2, \ldots, y_S$ simulated from S independent realisations $\theta_1, \theta_2, \ldots, \theta_S$ of the posterior density estimate. This comparison is made in terms of high-level characteristics $g(\cdot)$ such as higher degree distributions, etc. (Hunter et al., 2008).

```
bgof(post, aux.iters = 5000, n.deg = 15, n.dist = 9, n.esp = 8)
```



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 estimated posterior distribution.

Bibliography

Albert, R. and Barabási, A. L. (2002). Statistical mechanics of complex networks. *Reviews of modern physics*, 74(1):47.

Caimo, A. and Friel, N. (2011). Bayesian inference for exponential random graph models. *Social Networks*, 33(1):41 – 55.

Caimo, A. and Friel, N. (2014). Bergm: Bayesian exponential random graphs in R. *Journal of Statistical Software*, 61(2):1–25.

Gilks, W. R., Roberts, G. O., and George, E. I. (1994). Adaptive direction sampling. *Statistician*, 43(1):179–189.

Handcock, M. S., Hunter, D. R., Butts, C. T., Goodreau, S. M., Krivitsky, P. N., Bender-deMoll, S., and 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.

Holland, P. W. and Leinhardt, S. (1981). An exponential family of probability distributions for directed graphs (with discussion). *Journal of the American Statistical Association*, 76:33–65.

Hunter, D. R., Goodreau, S. M., and Handcock, M. S. (2008). Goodness of Fit of Social Network Models. *Journal of the American Statistical Association*, 103(481):248–258.

R Core Team (2016). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Roberts, G. O. and Gilks, W. R. (1994). Convergence of adaptive direction sampling. *Journal of Multivariate Analysis*, 49(2):287–298.

Robins, G., Elliott, P., and Pattison, P. (2001). Network models for social selection processes. *Social networks*, 23(1):1–30.

Schweinberger, M. (2011). Instability, sensitivity, and degeneracy of discrete exponential families. *Journal of the American Statistical Association*, 106(496):1361–1370.

Snijders, T. A. B., Pattison, P. E., Robins, G. L., and S., H. M. (2006). New specifications for exponential random graph models. *Sociological Methodology*, 36:99–153.