Computational Economics (2019) 54:845–875
https://doi.org/10.1007/s10614-018-9853-2

Introduction to Network Modeling Using Exponential Random Graph Models (ERGM): Theory and an Application Using R-Project

Johannes van der Pol[1]

Accepted: 3 September 2018 / Published online: 17 September 2018

Abstract
Exponential family random graph models (ERGM) are increasingly used in the study of social networks. These models are build to explain the global structure of a network while allowing inference on tie prediction on a micro level. The number of papers within economics is however limited. Possible applications for economics are however abundant. The aim of this document is to provide an explanation of the basic mechanics behind the models and provide a sample code (using R and the packages statnet and ERGM) to operationalize and interpret results and analyse goodness of fit. After reading this paper the reader should be able to start their own analysis.

Keywords Exponential random graph model (ERGM) · Statnet · Networks · Tie formation · Innovation networks · p-Star (p*)

1 Introduction

Networks are representations of relational data. Nodes represent entities while the links connecting them represent any form of interaction or connection between the entities. A large diversity of networks exists ranging from networks of social contacts between individuals to inventor networks, collaboration networks, financial networks and so on.

These networks attract the interest of researchers who want to explain the structure of these networks. In other words, the would like to know why certain agents are

Financial support from IdEx Bordeaux and ITEMM-Lab as well as valuable comments and suggestions from Murat Yildizoglu and several anonymous reviewers are gratefully acknowledged.

✉ Johannes van der Pol
johannes.van-der-pol@u-bordeaux.fr

[1] VIA Inno, GREThA UMR-CNRS 5113, University of Bordeaux, Avenue Léon Duguit, 33608 Pessac, France

central in a network and why other are at the periphery, why some networks are densely interconnected and why others are sparse. In essence, we want to know why firms *a* is connected to firm *b*, or what it the probability that firm *a* will connect to firm *c* and how does this impact the overall structure of the network. This allows us to identify if technological proximity between firms has a significant impact on the choice of collaborator between firms and if this strategy has a significant impact on the structure of the network.

The motivations for link creation cannot be observed directly from a visual representation of interactions, nor are they clear from glimpsing at a database containing relational data. In order to identify the motivations for entities to create links and identify the global network structure, a more in-depth analysis is required.

Methods such as, Interpretive Structural Modeling, Total Interpretive Structural Modeling, and Graph Theory Matrix allow for an analysis of the interactions between variables but do not allow to for conclusions on the structure of the networks we aim at analysing. In other words, the methods allow for identifying a correlation between technological proximity and social proximity but do not explain the structure of the network that allows this conclusion. Econometric analysis could shed more light on the motivations behind an observed link through logistic regressions. The probability of a link could be explained by a number of variables. There is one important limitation to this method. Due to the hypothesis of independence of the observations the probability of a link between two nodes can never be explained by the presence of another link inside the network. It is feasible that a link between two nodes exists only because of the presence of other links in the network. Take for instance the idea that John and Mary are connected solely because they have a common contact: Paul. Methods such as Block-models cannot account for the impact of the structure of the network on the probability of a link.

ERGM models are modified logistic regressions that allow for the probability of a link to depend upon the presence of other links inside the network (amongst other variables). ERGMs are able to take into account directed interactions as well as weighted interactions between nodes. The latter is an important point, since it allows an analysis beyond a simple binary relation between nodes approaching the idea of fuzzy logic.

An ERGM identifies the probability distribution of a network so that it can generate large samples of networks. The samples are then used to infer on the odds of a particular link inside a network. Applications for this method are numerous in many fields of research as shown by the increasing trend in the number of publications using ERGM models (see Fig. 1). In economics the number of published papers appears to be relatively low when compared to the other social sciences. Only 19 published papers could be found in the Scopus database (and even less in the web of science database). The topics are however quite diverse: knowledge sharing in organizations (Caimo and Lomi 2015), alliance networks (Cranmer et al. 2012; Lomi and Pallotti 2012; Lomi and Fonti 2012; Broekel and Hartog 2013; van der Pol 2018) and geographic proximity (Ter Wal 2013).

Since ERGMs allow for hypothesis testing, they can be put into use rather quickly within existing theoretical frameworks adding the possibility to analyse relational data in addition to normal data.

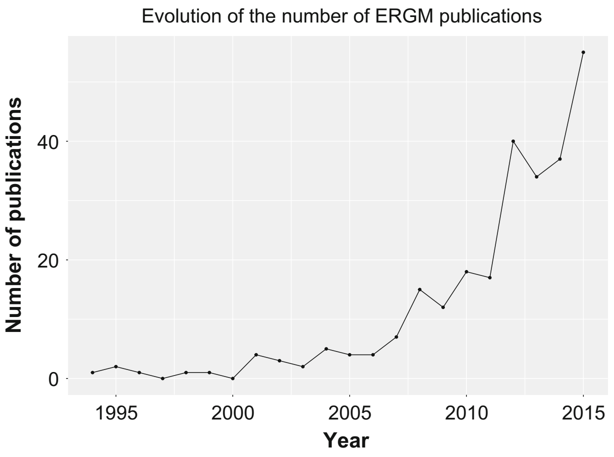

Fig. 1 Evolution of the number of publications involving ERG models for all disciplines (statistics included). (*Source*: Scopus)

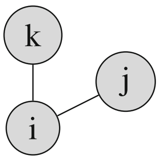

Fig. 2 An example of a sub-graph. This particular structure is called a two-star

The growing interest, and development of a theory of economic networks, provides a fertile ground for the use of ERGM models from the geography of innovation to venture capital investments. The aim of this paper is to provide an overview of the basic statistical theory behind ERGM models, which will be dealt with in the first section. Section 3 discusses the concept of dependence and the explanatory variables that can be included in the models. Section 4 discusses estimation methods while Sect. 5 provides the R scripts and the interpretation of an example using data for the French aerospace sector alliance network.

2 Theory

2.1 The Canonical form of ERGM Models

The aim of an ERGM is to identify the processes that influence link creation. The researcher includes variables in the model that are hypothesised to explain the observed network, the ERGM will provide information relative to the statistical significance of the included variable much like a standard linear regression. In addition, an ERGM provides a probability distribution for the network allowing the creation of a large set of networks which can be used for analysis.

It is useful at this point to explain that sub-structures of a network can (and are predominantly) used as explanatory variables. Substructures are small graphs contained inside the network. An example can be found in Fig. 2. The presence of some of these structures reflects certain link creation phenomena. A random network, i.e a network in which links are created at random, shows a low number of triangles. A triangle is an interconnection of three nodes, the smallest possible complete subgraph. Randomly generated networks have a low level of clustering because they have only a small number of triangles. The presence of triangles in an empiric network bares witness that there is a process that generates a significant number of triangles that is not the result of random link creation e.g, a tendency to create a link between common friends. A network with a small number of large stars and a large number of small stars can be the results of having a small number of very popular nodes. This is found in citation networks as well as lexicographical networks. Including sub-structures allows the modeling of certain processes as would any other variable.

In an ERGM we can find two types of explanatory variables: structural and node or edge-level variables. The latter come from other data sources and can be for example age, size of a firm, proximity, gender and so forth. The structural variables can contain indicators such as triadic closure, degree distribution and subgraphs.

2.2 The Odds of a Link

We are interested in assessing the probability of a link between two nodes, given a certain number of variables. We note G a graph, and i and j two nodes that exist in the graph. If i is linked to j in graph G this is written $G_{ij} = 1$ while the absence of a link between these two nodes is written $G_{ij} = 0$.[1] In addition, we note θ is a vector of parameters and X a vector containing variables.

We suppose that the probability of a link is influenced by different variables. We are interested in identifying the extend to which each variable influences the probability of a link. A widespread method for the study of probabilities is the logistic regression. The advantage of this method is that the logistic curve ensures that the computed values remain between 0 and 1 and allow for an easier analysis of the results, since they provide the marginal probability of a tie.

$$log(odds(G_{ij} = 1)) = log\left(\frac{p(G_{ij} = 1)}{p(G_{ij} = 0)}\right) = \frac{exp\left\{\theta_{ij=1} \cdot X_{ij=1}\right\}}{exp\left\{\theta_{ij=0} \cdot X_{ij=0}\right\}} \tag{1}$$

where we add $ij = 1$ for the parameters and variables when a link between i and j exists and $ij = 0$ when the link is absent. Using these equations the probability of a tie between i and j would be independent from the probability between i and k. Indeed, a logistic regression works under the hypothesis of independence of observations. In the case of networks, observations are however not independent. For instance, common friends tend to connect more in social networks, common collaborators have a higher tendency towards collaboration. A model that aims at explaining a network structure should be able to include these tie formation processes.

[1] The values 0 and 1 refer to values found in an adjacency matrix, 1 indicating the presence of a link, 0 the absence.

We hence modify the initial equations to include the network structure as observed before the link. This modification is introduced by Strauss and Ikeda (1990). We note G^c_{ij} the network without link ij:

$$odds(G_{ij} = 1) = \frac{p(G_{ij} = 1|G^c_{ij})}{1 - p(G_{ij} = 1|G^c_{ij})} = \frac{p(G_{ij} = 1|G^c_{ij})}{p(G_{ij} = 0|G^c_{ij})} \tag{2}$$

In Eq. 2 the odds of a link between nodes i and j now depend on the structure of the network before a link between i and j is created (noted by $|G^c_{ij}$). The probabilities are now conditional.

We discussed previously that some of the variables in the model can be subgraphs. The manner in which these are included in the model is simply by the count of these sub-structures. In other words, the value of the variable triangles is the number of triangles in the network. The same is true for stars, circuits and shared partners. This has as a consequence that the counts of these variables are not the same when a link between two nodes is present or absent. For instance the number of edges changes by one. This means that we need to differentiate between the value of the variables when a link is present and when it is absent. We hence note the vector of variables $v(G^+_{ij})$ when a link between i and j is added (hence the "+") and $v(G^-_{ij})$ when the link is absent. By including this differentiation we can rewrite Eq. 2 using the result in Eq. 1:

$$odds(G_{ij} = 1) = \frac{p(G_{ij} = 1|G^c_{ij})}{p(G_{ij} = 0|G^c_{ij})} = \frac{exp\left\{\theta' \cdot v\left(G^+_{ij}\right)\right\}}{exp\left\{\theta' \cdot v\left(G^-_{ij}\right)\right\}} \tag{3}$$

where $v(G^+_{ij})$ represents the vector of variables in the network with the link between i and j present and $v(G^-_{ij})$ the vector of variables with no link between i and j.

With some basic algebra we can develop the previous equation a bit further:

$$\frac{exp\left\{\theta' \cdot v\left(G^+_{ij}\right)\right\}}{exp\left\{\theta' \cdot v\left(G^-_{ij}\right)\right\}} = exp\left\{\theta' \cdot v(G^+_{ij})\right\} \cdot exp\left\{-\theta' \cdot v\left(G^-_{ij}\right)\right\} \tag{4}$$

$$= exp\left\{\theta'\left(v\left(G^+_{ij}\right) - v\left(G^-_{ij}\right)\right)\right\} \tag{5}$$

When developing the vector of variables we have:

$$= exp\left\{\theta'_1 \cdot \left(v_1\left(G^+_{ij}\right) - v_1\left(G^-_{ij}\right)\right) + \cdots + \theta'_n \cdot \left(v_n\left(G^+_{ij}\right) - v_n\left(G^-_{ij}\right)\right)\right\} \tag{6}$$

Equation 6 shows that each parameter of the model is associated not with the counts of sub-structure but with the difference in counts. The difference from having an extra link, and the absence of said link. In essence $(v_1(G^+_{ij}) - v_1(G^-_{ij}))$ represents the variation in the number of counts of network statistic 1 that result from the additional link. The variables are hence referred to as "change statistics". In order to remove the exponential from the right hand side of the equation we apply the logarithm:

$$log\left(\frac{exp\left\{\theta \cdot v\left(G_{ij}^{+}\right)\right\}}{exp\left\{\theta \cdot v\left(G_{ij}^{-}\right)\right\}}\right) = \theta_1' \cdot \left(v_1\left(G_{ij}^{+}\right) - v_1\left(G_{ij}^{-}\right)\right) + \cdots + \theta_n' \cdot \left(v_n\left(G_{ij}^{+}\right) - v_n\left(G_{ij}^{-}\right)\right) \quad (7)$$

So we can rewrite Eq. 7 noting $v_1(\Delta_1 G_{ij})$ the change statistic for a link between i and j for variable 1 as follows:

$$log\left(\frac{exp\left\{\theta \cdot v\left(G_{ij}^{+}\right)\right\}}{exp\left\{\theta \cdot v\left(G_{ij}^{-}\right)\right\}}\right) = \theta_1' \cdot v_1(\Delta_1 G_{ij}) + \cdots + \theta_n' \cdot v_n(\Delta_n G_{ij}) \quad (8)$$

Each variable now accounts for the change in counts of network statistics. It is important to remind us that Eq. 8 accounts for the odds of one edge in the network while we are interested in the probability for the whole network. Following Besag (1972) we can invoke here the Hammersley–Clifford theorem. Since this theorem is based on concepts out of the reach and the purpose of this document we will not detail the theorem. For a detailed explanation please refer to Hammersley and Clifford (1971).

The theorem states that the probability of a network can be defined solely by the counts of subgraphs. This is important because it tells us that all we have to do is identify the correct subgraphs to ensure that a model of the network structure can be found. The more accurate the subgraphs to more reliable the inference of additional covariates.

2.3 The Probability Distribution of a Network

The Hammersley–Clifford theorem states that the probability of a graph can be identified solely by counts of subgraphs. As such, we know that the probability is proportional to these variables. Since we have an observed network that we wish to replicate we look for the probability that the network generated by the model (X) is identical to the observed network (x). The logarithm is applied to bound the probability:

$$log(p(X = x)) \propto \theta \cdot v(G) \quad (9)$$

$$p(X = x) \propto exp\{\theta \cdot v(G)\} \quad (10)$$

The right-hand side of the equation now needs to be normalized in order to obtain a proper probability distribution. The normalization is not straightforward, indeed, in order to normalize the probability of a network one needs to normalize by all possible networks with the same number of nodes:

$$p(X = x) = \frac{exp\{\theta \cdot v(G)\}}{\sum_{y \in Y} exp\{\theta \cdot v(G)\}} \quad (11)$$

With y a possible network structure in the set of all possible networks Y. The numerator is normalized by the sum of the parameters over all possible network structures.

Note that this number is large. For a network with n nodes the number of possible graphs is $2^{\frac{n(n-1)}{2}}$. So even for a graph with 10 nodes there are 35,184,372,088,832 possible graphs. The major problem to overcome with ERGMs is exactly this normalizing constant.

With some simple algebra we find a general form for this model. Using θ as a vector of parameters and $v(G)$ a vector of variables for network G:

$$p(X = x) = \frac{exp\left\{\theta \cdot v(G)\right\}}{exp\left\{log\left(\sum_{y \in Y} exp\left\{\theta \cdot v(G)\right\}\right)\right\}} \tag{12}$$

$$p(X = x) = \frac{1}{\psi(\theta)} \cdot exp\left\{\theta \cdot v(G)\right\} \tag{13}$$

Equation 13 is the most general and commonly used form of the model (Lusher et al. 2012). Equation 13 also gives the canonical form of an ERGM model. Since the density of the random variable (the network structure) has the particular form in Eq. 13 it is referred to as an exponential family. In addition, since the structures are represented by a random variable, they are random graphs.

Putting both elements together and this results in an Exponential Family Random Graph. Since ERGM is easier to pronounce than EFRGM, the models are referred to as ERGM.[2]

The canonical form gives the equation we wish to estimate. However, before we tackle the question of estimation we need to explore in more detail the variables that we would want to include in the model. We have stated previously that counts of subgraphs can be used as explanatory variables. The following section will explain which particular subgraphs are to be included in a model.

3 The Dependence Assumption

The previous section has shown that ERGMs are capable of providing conditional probabilities for links. This dependence assumption is important because it allows the researcher to study different phenomena that rule the formation of networks. This section will show how the hypothesis of dependence of links is connected to the choice of subgraphs that may be included in an analysis.

3.1 The Markovian Hypothesis

Links between nodes rarely appear at random, agents have specific reasons to connect with one agent rather than another. The motivations behind interactions are numerous and complex and have been subject to scrutiny from researchers in different strands of

[2] This type of model is also referred to as the P^* family of models (Anderson et al. 1999; Lusher et al. 2012).

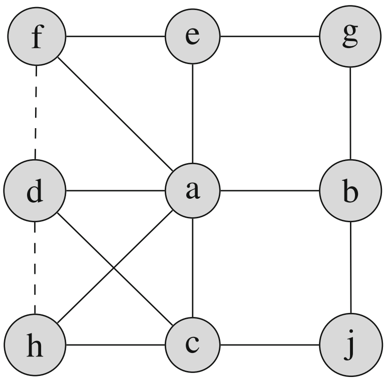

Fig. 3 Node level dependence illustration: the Markovian neighborhood

the social sciences. This research shows that people or firms with common acquaintances or collaborators will have a higher tendency to cooperate for example. This makes sense for two reasons, first, having common partners means common working practices which have a positive impact on collaboration. Second, when searching for collaborations firms tend to rely on referrals. A collaboration already in place allows firms to observe in detail the efficiency of other firms, referrals that result from cooperation should hence be trustworthy. In addition cooperators of a firms have a higher probability to be in contact with each other since they are more likely to meet during social or professional events.

The odds of a link depend on the neighborhood of the node and not on the entire rest of the graph. In more formal terms: two potential links X_{ij} and X_{kl} are in the same neighborhood if $\{i, j\} \cap \{k, l\} \neq 0$. For instance, the dotted lines in Fig. 3 represent potential links. The odds of this link will depend upon the nodes f and d have in common, in this case node a. This shows that not all subgraphs are compatible with analyzing this phenomenon. Any substructure that cannot account for a common node should not be included.

Since it is hypothesized that only neighboring nodes impact the odds of a link, we seek node-level dependence. This level of dependence is also referred to as a nearest-neighbors level of dependence[3] or dyadic dependence (Harris 2013).

For the purpose of the identification of relevant structures we need to find all structures that are subject to a Markov (or nearest-neighbor) level of dependence.

Suppose we have the social network depicted in Fig. 4. The graph shows social interactions between four agents, a, b, c and d. Markovian dependence suggests that a link between a and b depends on the connections between common nodes. The nodes a and b have in common are d and c. A link between a and b hence depends upon connections between $a{-}c$, $a{-}d$, $b{-}c$ and $b{-}d$.

When one identifies all the possible dependencies one can generate a dependence graph. The dependence graph for the complete Markov graph in our example can be found in Fig. 5. In red we find the dependence links for link $a{-}b$, it shows the links on which $a{-}b$ depends.

[3] Nearest-neighbor systems have been studied by Besag (1972).

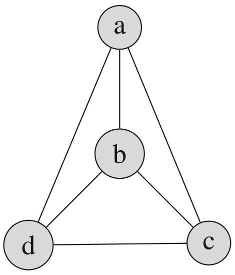

Fig. 4 Markov graph

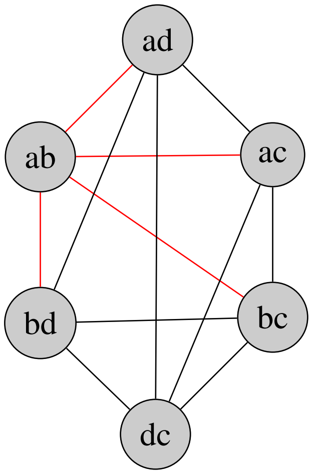

Fig. 5 Dependence graph

From this graph one can identify the substructures that comply with Markovian dependence. All subgraphs in the dependence graph can be included in a model to add Markovian dependence. For example, the dotted line between *ab* and *ad* in Fig. 6 represents a 2-star centered on agent *a* (Fig. 7).

Using the same method as for the 2-star one can also identify a triangle between the three agents on the links *bd*, *bc* and *dc*. The Markov model hence includes three configurations: edges, 2-stars and triangles. With the inclusion of these configurations the Markovian model takes the form in Fig. 6. In more complex graphs one could also identify 3-stars, 4-stars, some example can be found in Figs. 8–11.

In more complex graphs one could also identify 3-stars, 4-stars etc.

It should be obvious here that the number of distinct 2-stars is large and it is near impossible to add a parameter for each distinct 2-star in the dependence graph. To reduce the number of variables a hypothesis is made that each type of configuration has the same probability of appearance, this allows for the inclusion of one parameter per substructure. In the case of Markov dependence the ERGM model would have the following form:

$$p(x = X|\theta) = \frac{1}{\psi(\theta)} exp\left\{\theta_E \cdot v_E(x) + \theta_{S2} \cdot v_{S_2} + \cdots + \theta_{S_{n-1}} \cdot v_{S_{n-1}} + \theta_\Delta \cdot v_\Delta\right\} \tag{14}$$

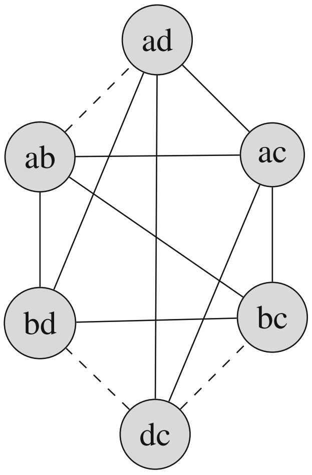

Fig. 6 Dependence graph and configuration identification

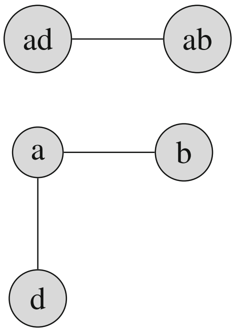

Fig. 7 2-Star identification in the dependence graph

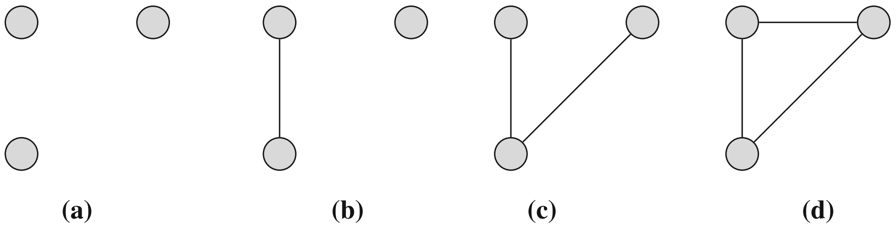

Fig. 8 Triads. **a** Empty. **b** Dyad. **c** 2-Star. **d** Triad

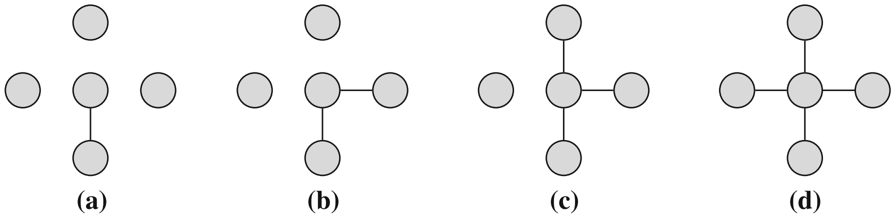

Fig. 9 K-stars. **a** Empty. **b** Dyad. **c** 2-Star. **d** Triad

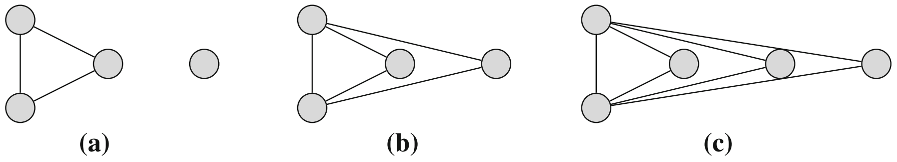

Fig. 10 Shared partners. **a** 1 partner, **b** 2 partners, **c** 3 partners

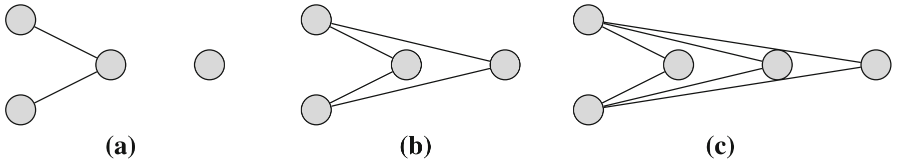

Fig. 11 Shared partners (dyadic). **a** 1 partner, **b** 2 partners, **c** 3 partners

where θ_E is the parameter for the number of edges, θ_{S2} the parameter for the number of 2-stars and θ_Δ the parameter for the number of triangles. Note here that the model does not include simultaneously a 1-star and an edge parameter since they would be the same variable. With this model one is able to study if common nodes have a positive impact on the odds of link creation.

In addition the combination of the 2-star parameter and the triangle parameter account for triadic closure effects. In other words, are triangles created because 3 nodes are connected at the same time, or are triangles formed by the closing of a 2-star.

Of course, Markovian dependence is only one of the possible levels of dependence. One can imagine higher levels of dependence, or even any level of dependence to be empirically relevant. One could suggest that firms evolving on the periphery of a network to have a higher probability to connect with firms in the center of the network than between them. The previous model was hence extended by Wasserman and Pattison (1996) to allow for a general level of conditional dependence giving the researcher a total liberty in the theories to test. Whatever the level of dependence chosen, the dependence graph gives the substructure that may be included (Frank and Strauss 1986).

3.1.1 Higher Levels of Dependence

It is possible to assume that the Markovian level of dependence is not adequate or does not capture the full complexity of mechanisms of link creation. Links can be dependent without there being a common node involved. For instance, consider a case in which people work on the same floor in a company. The probability of a social link does not depend upon a common node but simply on the fact that they are geographically close, belong to the same community or have common cultural aspects (White 1992). In order to be able to model more complex aspects of social interactions and indeed even strategic interactions, one needs to be able to account for more structural aspects than stars and triangles (however potent in explanatory power these might be). The latter implies that the links are only dependent on each-other if nodes are part of a same neighborhood (neighborhood takes a broad definition here, it can be social, geographical or cultural). Due to the inclusion of general dependence the model is transformed to take the form:

$$p(x = X) = \frac{1}{\psi(\theta)} exp \left\{ \sum_{A \in M} \lambda_A \cdot z_A(x) \right\} \tag{15}$$

where A indicates the neighborhood as part of the ensemble M of all possible neighborhoods. The parameter λ_A will take a positive value when the probability of observing network x is increased. With the broad definition of "neighborhood" this model is able is almost limitless. The latter results in a problem, the model is too general.

We have seen in the previous subsection that the structures that can included in the ERGM model are defined by the dependency graph. In the case of a generalization of the dependency assumption, i.e all ties may be dependent upon all other ties, the dependency graph is a complete graph and all possible subgraphs can be considered as variables. This leaves a tremendous amount of parameters to be estimated.

Pattison and Robins (2002) and Carrington et al. (2005) offer two solutions to this problem. Their aim is to find a way to reduce the number of subgraphs to be included in the model. The only way to achieve this is to reduce the level of dependency from general to a more restricted level. A first step is to simply fix a level of dependency which will automatically switch all other parameters to 0. This means that once one defines a condition under which links are dependent upon each other a *setting* is defined. Defining a level of dependency can be simply supporting the hypothesis that links between firms depend upon a common region, or sector, size or any other group. s is a setting, s being a subset of the set of nodes M: $s \in M$. The restriction gives a new dependence graph which will contain a restricted number of subgraphs to include. All parameters for substructures that are not part of the dependency graph are equal to 0. Obviously, defining the settings oneself required extensive knowledge about the network at hand. The inclusion of these settings results in what Pattison and Robins (2002) refer to as *partial conditional dependence*.

Of course, one can also include other types of variables to a model, such as age of the firm, geographic location, amount of public funds received etc. These variables are referred to as node variates or node attributes The addition of these attributes is introduced by Robins et al. (2001). The idea here is that links depend upon the attributes of the nodes they are connecting. In other words the probability that two nodes are connected depends upon the attribute of the node. These models are also called *social selection models* and take the following form:

$$p(Y = y \mid X = x) = \frac{1}{\psi(\theta)} \cdot exp \left\{ \sum_i \theta z(x) + \theta^a \cdot z^a(x, y) \right\} \tag{16}$$

where the exponent a indicates parameters and structures relative to the inclusion of dyadic dependence for the purpose of social selection.

In the same paper Robins et al. also described a *social influence model* in which the attributes of the nodes are a result of the structure of the network (nodes take a particular attribute according to the nodes in the neighborhood for example). In other words, the probability that a node variables takes a particular value depends upon the structure of the network and the values of this (or indeed any other) node-level variable.

We hence need to add a node variables to the model. Suppose we note Y_i the value of a node variable for node i. This variable can be anything from a geographic region to the amount of public investment received by firms to the number of publications or patents. When this variable is included in the ERGM the model is written:

$$p(Y = y \mid X = x) = \frac{1}{\psi(\theta_i)} \cdot exp\left\{\sum_i \theta z(y, x)\right\} \tag{17}$$

Just as it is possible to put values on nodes it is also possible to values on dyads. The question then is to know if the value of the dyad increases the probability of nodes being connected. Think of situations where we would like to know if the amount of investment between firms is related to cooperation or if technological proximity between firms induces collaboration. Note here that the difference between dyadic covariates and actor attributes resides in the value on the link between two nodes. In the case of proximity is refers to the proximity of both firms, it is hence not a firm-level variable. The value only makes sense when we consider firms two-by-two.

All the extensions made to the ERGM framework allow researchers to answer a large variety of questions about social and economic processes. Many other extensions which are beyond the scope of this document, but worth noting, are multivariate relations in networks (Pattison and Wasserman 1999) and dynamic networks in which new ties depend upon the structure of the network at time $t - 1$. It is also possible to model multivariate networks using ERGM. The idea is then that each link can exist in different matrixes, each corresponding to a different type of link (social, work, geography etc.). This extension allows researchers to study interplay between different networks and how each network is affected by the other networks.

Before looking at estimation methods for ERGM models one problem needs to be addressed: the degeneracy problem.

3.2 Solving Degeneracy: Curved ERGMs

ERGM models are prone to degeneracy issues. When estimating the model the change statistics can behave in such a way that the large majority of the probability distribution is placed on either an empty or a full graph. As we will discuss in more detail a bit later, a simulation is performed to identify the probability distribution of a graph. This is done on a step by step basis, starting with an empty network and adding links one by one until a maximum likelihood is achieved. This probability distribution is a function of the change statistics and is thus impacted by the change in log-odds for an additional tie (for a given effect). In other words if an additional edge would create two new 2-stars, then the log-odds of observing that tie would increase by two multiplied by the parameter of the 2-star variable. A random graph model is considered stable if small changes in the parameter values result in small changes in the probabilistic structure of the model (Handcock et al. 2003). When a new edge is added to the graph this not only increases the number of edges but might also increase the number of other sub-configurations that might be included in the model. The parameters of the model control the number of sub-graphs of a particular type that are expected in the graph. For instance a 2-star might be transformed into a triangle by the addition of an edge which also adds two 2-stars. A 2-star can become at 3-star and so on. This cascading effect will result in the simulation procedure jumping over the MLE and converge to a full graph. Lusher et al. (2012) [chapter 6] show that, in the case of the Markov (or triad model), the number of expected triangles increases as the parameter for triangles increases. They highlight a phase transition for certain values of the parameter where the number of expected triangles increases more than exponentially. This transition explain that the probability density distribution has high weights either on a (close to) empty graph or on a complete graph.[4] This problem is increasingly present as the number of nodes increases. The larger the network the higher the number of substructures one can have.

In order to avoid the model to put too much weight on the full graphs, Snijders et al. (2006) propose to add several variables based on their concept of *partial conditional dependence*. The idea is to include a weighted degree distribution to the model, giving a high weight to low density while decreasing the weights as the degree increases. This reduces the impact of the high density variables responsible for the degeneracy of the initial model. Mathematically we can then write (using the notations of the initial paper):

$$u_{\alpha}^{(d)} = \sum_{k=0}^{n-1} e^{-\alpha k} d_k(y) \tag{18}$$

where $d_k(y)$ is the number of nodes with degree k and α the parameter of the weights. This is referred to as the *geometrically weighted degree distribution*. The degree distribution can also be written as a function of the stars in the network. After all, a degree distribution is nothing more than a distribution of stars. Nodes with a degree of five are 5-stars, degree two are 2-stars and so forth. We can hence formulate the distribution as follows:

$$u_{\lambda}^{s} = S_2 - \frac{S_3}{\lambda} + \frac{S_4}{\lambda^2} - \cdots + (-1)^{n-2} \cdot \frac{S_{n-1}}{\lambda^{n-3}} = \sum_{k=2}^{n-1} (-1)^k \cdot \frac{S_k}{\lambda^{k-2}} \tag{19}$$

where S_k is a the number of stars of degree k and $lambda$ the parameter. This method is referred to as *Alternating k-stars*. The difference between the geometrically weighted degree distribution and the K-stars is resides in the alternating signs. A large value of 3-stars is counterbalanced by a negative value for 4-stars due to the inverse sign of the parameter. The addition of the weights ensures that the change in change statistics stays small. Indeed Snijders et al. (2006) show that the change statistics can be written:

$$z_{ij} = -(1 - e^{-\alpha}) \left(e^{-\alpha y_{\tilde{i}+}} + e^{-\alpha y_{\tilde{j}+}} \right) \tag{20}$$

[4] In addition, the degeneracy of the model can result in problems with the estimation procedures.

where $\tilde{y_{i+}}$ is the density of firm i when the link between i and j is added. Equation 20 shows that the value of the change statistic is reduced by the factor $-(1 - e^{-\alpha})$. This factor hence ensures that the change statistics do not take too high values and result in a nested probability distribution. The inclusion of either the alternating k-stars or the geometrically weighted degrees transform the ERGM model into a Curved Exponential Random Graph Model (Efron 1975).

All that is left to do now is estimate the model.

4 Estimation

Estimation allows for the identification of the parameters that maximize the likelihood of a graph. Since we only have one observation (the observed graph) a set of graphs from the same distribution is required. The set of graphs that may be generated by this procedure should have the observed graph as a central element to ensure a valid sample.

4.1 Markov Chain Monte Carlo

In the first section we identified the general form of an ERGM model (see Eq. 13). The odds of a graph were normalized by the sum of the parameters of all possible graphs. This leaves us with a constant to estimate which is near impossible. A workaround has to be found for ERGMs to be useful. A first development by Besag (1975), Strauss and Ikeda (1990) was to estimate the model using pseudo-likelihood estimation. The properties of this method are however not clear (Snijders et al. 2006; Robins et al. 2007) we shall hence focus here on more recent methods that are better understood.

A method for estimating the parameters of ERGMs using a sample is developed by Anderson et al. (1999), Snijders (2002), Geyer and Thompson (1992). They estimate the model by Markov Chain Monte Carlo (MCMC) to find the maximum likelihood estimates (MLE). The idea is to extract a sample from a distribution that follows Eq. 13 asymptotically, not requiring the direct computation of the normalizing constant. Their paper points out that almost any maximum likelihood can be accomplished by a MCMC.

A Markov chain is a sequence of random variables such that the value taken by the random variable only depends upon the value taken by the previous variable. We can hence consider a network in the form of an adjacency matrix in which each entry is a random variable. By switching the values of these variables to 0 or to 1 (adding or removing a link from the network) one can generate a sequence of graphs such that each graph only depends upon the previous graph. This would be a Markov chain. The hypothesis is then that if the value at step t is drawn from the correct distribution than so will the value at step $t + 1$. Unlike regular Monte-Carlo methods, the observations that are sampled are close to each-other since they vary by a single link. However, one would need a method for selecting which variable should change state in order to get closer to the MLE, this is done using the Metropolis–Hastings algorithm or the Gibbs sampler.

4.2 Metropolis–Hastings Algorithm and the Gibbs Sampler

The Metropolis–Hastings algorithm picks a variable at random and changes it's state. This results in either a new edge in the network or in the disappearance of an edge. The probability of the graph is then computed and only if the probability of the altered graph is higher than the previous one is the new graph retained for the next step. In other words the new graph is retained as long as the likelihood is increased:

$$min\left\{1, \frac{p_{\theta(x^*)}}{p_\theta(x^{m-1})}\right\} \tag{21}$$

This decision rule is called the *Hastings ratio*. The advantage of this ratio is that it does not include the normalizing constant $\psi(\theta)$.

Since the Markov chain starts at 0, a burn-in in needed to remove part of the chain to identify if the chain has converged or not (the burn-in can be parameterised in most software).

The steps taken by the Metropolis–Hastings algorithm are quite small. These small steps are implemented in order to avoid overstepping the global optimum which can easily happen in the case of larger parameter spaces. Other methods allow for bigger steps and as such converge faster and need a lower burn-in. The risk of larger steps is however overstepping the global optimum and convergence towards other local optima. The Metropolis–Hastings algorithm may be slower than others but is more precise in it's estimation.

Some programs use the Gibbs sampler, which is a special case of the Metropolis–Hastings algorithm (Hunter and Handcock 2006). The difference between Gibbs and Metropolis–Hastings resides in the chosen step. In the case of the Gibbs sampler, the state of each element in the vector of parameters is chosen and updated conditionally on the state of the other parameters. This means that if this decision rule was implemented in the Metropolis–Hastings algorithm the probability that the change is retained is equal to one. This makes the Gibbs sampler a relatively fast method. This sampling method is used by the different algorithms that are used to estimate ERGM models.

Both methods allow for the generation of a sample of graphs that can be used for inference. The sample of graphs is obtained by varying not the parameters but the variables of the model until it is centered around the observed graph. Now that a sample of graphs has been found we need to estimate the parameters of the model. Two of the most widely used estimation algorithms, the "Stepping" and "Robbins–Monro" algorithm will now be reviewed.

4.3 The "Stepping Algorithm"

This method introduced by Hummel et al. (2012). It has the advantage of approaching the MLE directly while "Robbins–Monro" does not. ERGM models are indeed estimated using the maximum likelihood method. Starting from the canonical ERGM form we define the log likelihood function as:

$$L(\theta) = \theta \cdot v(G) - log(\psi(\theta)) \tag{22}$$

The problem here is the presence of the normalizing constant which cannot to computed. The improvements of this method over the previously one resides in the use of a log-normal approximation. The algorithm proposed here will converge towards the log-likelihood using a step-by-step method. The sampler used in with this estimation procedure is the Metropolis–Hastings sampler discussed previously. Once a sample of graphs has been identified the estimation algorithm is launched. Since the normalizing constant in Eq. 22 cannot be compute a workaround has to be found. The idea is to give starting parameters (θ_0).The log-likelihood ratio can then be written (Hummel et al. 2012):

$$L(\theta) - L(\theta_0) = (\theta - \theta_0)^T V(G) - log E_{\theta_0}\left[exp(\theta - \theta_0)^T V(g)\right] \tag{23}$$

Geyer and Thompson (1992) point out that maximizing this ration by the means of a sample distribution of graphs generated with θ_0 only behaves well when θ is close to θ_0. In other words one has to choose the correct starting point for the algorithm to find the MLE. The MLE solves the equation:

$$E_{\hat{\theta}} v(G) = v(G^{obs}) \tag{24}$$

The idea is to suppose that the MLE is not the observed value of the parameters but some point between the mean value parameterization and the observed value. A parameter γ defines the steps taken:

$$\hat{\omega_t} = \gamma_t \cdot v(G) + (1 - \gamma_t)\bar{\omega} \tag{25}$$

where ω_t represents the estimate in the mean parameter space.[5] Ideally then, we would want $\gamma = 1$ so that the expected value of the parameters is the observed value. If this is the case the algorithm is considered to converge, this is shown in Fig. 12 which is the output of the R code. Once convergence is detected a large sample based on the parameters is computed and the MLE are estimated and gives as the final values.

Step 1: Set the iteration number equal to 0 and choose initial parameters for vector η_0.
Step 2: Use MCMC to simulate graphs from the probability function for parameter vector η_0.
Step 3: Compute the mean of the sample.
Step 4: Define a pseudo observation that is a convex combination of the mean of the sample and the observed value.
Step 5: Replace the observed value by the pseudo observation.

4.4 Stepping Algorithm Output

Robbins–Monro The Robbins–Monro algorithme is a stochastic approximation method introduced by Robbins and Monro (1951) which is used by Snijders (2001)

[5] In the algorithm the initial values are chosen to be the MPLE.

```
Iteration # 1 .   Trying gamma= 0.17
Iteration # 2 .   Trying gamma= 0.14
Iteration # 3 .   Trying gamma= 0.16
Iteration # 4 .   Trying gamma= 0.22
Iteration # 5 .   Trying gamma= 0.25
Iteration # 6 .   Trying gamma= 0.34
Iteration # 7 .   Trying gamma= 0.31
Iteration # 8 .   Trying gamma= 0.46
Iteration # 9 .   Trying gamma= 0.74
Iteration # 10 .   Trying gamma= 0.97
Iteration # 11 .   Trying gamma= 1
Iteration # 12 .   Trying gamma= 1
Now ending with one large sample for MLE.
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 20 .
```

Fig. 12 R output for the stepping algorithm

and Snijders (2002) to estimate ERGM models. Typically the method estimates:

$$E\{Z_\theta\} = 0 \tag{26}$$

where Z_θ is a vector of parameters equal to $u(Y) - u_0$ where u_0 is the observed value of the statistics. This allows us to rewrite the equation as a moment equation. The algorithm gives starting parameter values equal to the average of the parameter values. The initial parameters that will launch phase two are defined by (Lusher et al. 2012):

$$\theta^{t+1} = \theta^t - a_t \cdot D^{-1}\left(z\left(x^m\right) - z(x_{obs})\right) \tag{27}$$

where D is the co-variance matrix, the diagonal of this matrix will be used as the scaling matrix. a defines the convergence, it is set to $a_t = \frac{a_{t-1}}{2}$. The idea is that each step brings the values closer to the MLE. Hence large steps might result in the exceeding the MLE and divergence. The fact that the a_r reduces in value with each step allows a smooth path to the MLE. As we move closer to the observed values of the statistics $z(x^m) - z(x_{obs})$ tends towards zero. The R output in Fig. 12 shows how the steps (a) start at a value of 0.1 and tend towards 0 with each iteration of the second phase of the algorithm. At the start of each step the starting parameters are considered to be the average values of the previous step. The number of iterations varies from model to model. The iterations stop once the trajectories of the generated statistics cross those of the observed ones (Lusher et al. 2012).

The burn-in represents the number of simulations that are removed from the MCMC in order to make the chain “forget” the starting point. In other words it is to make sure the starting values do not impact the final result.

Finally the algorithm checks for convergence using a convergence statistic. Just as in the case of the stepping algorithm one supposes that the MLE is reached when the distance between the observed values and the average of the simulated ones is close to 0. If there is no convergence than one can relaunch the estimation with as starting parameters the results of the previous simulation (Lusher et al. 2012).

The largest difference between this method and the stepping method resides in two factors. First this method approaches an estimate of the MLE and does not evaluate the

```
Robbins−Monro algorithm with theta_0 equal to:
    edges  triangle
−4.676219  1.456380
Phase 1:  13 iterations (interval=1024)
Phase 1 complete; estimated variances are:
   edges triangle
3676.692 1175.308
Phase 2, subphase 1 : a= 0.1 , 9 iterations (burnin=16384)
theta new: −4.66068075119643
 theta new: 1.42768924899568
Phase 2, subphase 2 : a= 0.05 , 23 iterations (burnin=16384)
theta new: −4.64740903958273
 theta new: 1.4235669242362
Phase 2, subphase 3 : a= 0.025 , 58 iterations (burnin=16384)
theta new: −4.62881856406474
 theta new: 1.41405593966042
Phase 2, subphase 4 : a= 0.0125 , 146 iterations (burnin=16384)
theta new: −4.60985096388914
 theta new: 1.39932813390954
Phase 3:  20 iterations (interval=1024)
Evaluating log−likelihood at the estimate.
```

Fig. 13 R output for the Robbins–Monro algorithm

MLE function directly. Second, the steps are of a higher magnitude and can exceed the MLE if the starting values are close to the MLE. The use of either of the algorithms purely depends upon the model to be estimated. One algorithm might have better convergence in one case while the opposite can be true in another case. Note however that both use the Metropolis–Hastings method for the simulation of the MC.

Now that we have discussed which variables can be included and how to estimate the parameters we will turn to an application using R (Fig. 13).

5 Code R and Example

We use here different R packages (Hunter et al. 2008; Handcock et al. 2008; Butts 2008).

The data (and hence the results of the estimations) are from the French aerospace collaboration network. Using patent data, collaborations were identified which were used to create a network. Previous studies have focused on the factors that influence collaboration (Cantner and Meder 2007; van der Pol and Rameshkoumar 2018). The aim of the study is to understand if technological proximity played a significant role in the structuring of the collaboration network. We hence used a dyadic covariate called “proximity”. The network contains 176 firms.

```
#Import data.
Network <− read.table("ADJ_MATRIX.csv", sep=";")
```

The data used here was already in the form of an adjacency matrix and hence could be used directly. It is however also possible to use edgelists. Since the data needs to be transformed into a network object the network package will be needed. The latter is able to transform edgelists into adjacency matrices.

```
#Import the dyadic covariates
proximity <- read.table("Proximity_matrix.csv", sep=";")
proximity.e <- read.table("Proximity_matrix_exp.csv", sep=";", dec=",")
citation <- read.table("Citation_matrix.csv", sep=";")
```

Since I'm using two measures of proximity I have two matrices and a matrix that includes the number of citations between firms. These need to be imported in the same manner as the network itself.

```
#We now transform the imported data into network objects with the package 'network'
Network <- as.matrix(Network)
proximity <- as.matrix(proximity)
proximity.e <- as.matrix(proximity.e)

#Transform to network format
Network <-  as.network(Network, directed = FALSE)
proximity <- as.network(proximity, directed=FALSE)
exp_proximity <- as.network(exp_proximity, directed=FALSE)
citation <- as.network(citation, directed=TRUE)
```

We now have different objects to work with: the network and dyadic covariates in the form of networks. Note that the model constructed here is for un undirected network as shown by the option $directed = FALSE$ in the $as.network$ function.

We now have to decide which variables to include in the model. Let's start with a simple model containing only edges. We invoke here the $ergm()$ function from the $ergm$ package:

```
model <- ergm(Network ~ edges)
```

This gives us the most basic form of an ERGM model, the estimation method defaults to Monte Carlo MLE (Geyer and Thompson 1992).

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

Formula:    Network ~ edges

Iterations:  7 out of 20

Monte Carlo MLE Results:
           Estimate        Std. Error  MCMC      \%p-value
edges -3.85280      0.05649       0            <1e-04 ***
----
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

     Null Deviance: 21349  on 15400  degrees of freedom
Residual Deviance:   3113  on 15399  degrees of freedom

AIC: 3115      BIC: 3122      (Smaller is better.)
```

The parameter for the variable $edges$ has an estimated value of -3.8528. This means that the probability of two ties connecting is:

$$p(i \rightarrow j) = \frac{exp(-3.85)}{1 - exp(-3.85)} = 0.02174241 \tag{28}$$

Recall Eq. 6, this equation stated that the variables were change statistics. The parameter should hence be multiplied by the change in the number of subgraphs. In other words, if an additional edge creates three triangles the parameter should be multiplied by three. Since an additional edge only creates one new edge we do not multiply. Let's try the same but with only triangles as explanatory variable.

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

Formula:    Network ~ triangles

Iterations:  NA

Stepping MLE Results:
          Estimate  Std. Error MCMC \% p-value
triangle  -1.91496     0.01712       0  <1e-04 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

     Null Deviance: 21349  on 15400  degrees of freedom
Residual Deviance:  8935  on 15399  degrees of freedom

AIC: 8937     BIC: 8945     (Smaller is better.)
```

The parameter for the variable triangles has an estimated value of − 1.9146. This means that the log-odds of two nodes connecting is:

$$-1.9146 * \delta \, triangles \tag{29}$$

where $\delta \, triangles$ gives the change in the number of triangles. Hence the log-odds depend upon the number of triangles that will be created by an additional tie:

- If the link creates 1 triangle, the log-odds are 1 * − 1.9146. The probability is then 0.1284
- If the link creates 2 triangles, the log-odds are 2 * − 1.9146. The probability is then 0.0213

Let's see how we interpret estimates when we have more than one variable: the Markov model.

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

Formula:    Network ~ edges + kstar(2) + triangles

Iterations:  NA

Monte Carlo MLE Results:
           Estimate  Std. Error MCMC% p-value
edges     -28.30222     0.22246      0  <1e-04 ***
kstar2      7.68159     0.07356      0  <1e-04 ***
```

```
triangle   57.09972     0.79544       0  <1e-04 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

     Null Deviance:     21349  on 15400  degrees of freedom
Residual Deviance: 35811780  on 15397  degrees of freedom

AIC: 35811786     BIC: 35811809     (Smaller is better.)
```

In this case an additional edge, if it creates x 2-stars and y triangles, has log-odds:

$$-28.30 + x \cdot 7.68 + y \cdot 57.099$$

Since the model includes lower and higher order subgraphs (2-stars are a substructure of triangles) we can conclude here that triadic closure is significant in the network. In other words, 2-stars are closed to form triangles.

Note however the values of the information criteria AIC and BIC, stating these values are high is an understatement. The model must suffer from some kind of problem. Section 5.3 will show how to deal with degeneracy and other problems.

Now we want to see how proximity (technological in the case of this study) influences the structuring of the network. Let's start with a simple model using the edge covariate "proximity". The comment for adding an edge covariate is simple $edgecov$. Similarly node covariates (age, gender, type of firm, country...) are added using the $nodecov$() or $nodefactor$ command. The node factor command is particularly useful since it allows to compare log-odds to a reference point. For example, one could categorise the R&D expenses of firms into low, $average$ and $high$. The ERGM would then give the log-odd of 2 categories as compared to a third. In other words is a link more likely for firm with average expenses than for low? This feature seems not to be available for edge covariates however.

The following model only contains edges and the edge covariate. A first point we can notice is that the AIC and BIC criteria are lower with the addition of the edgecovariate. The model is hence improved with the addition of the proximity parameter. Firms with proximity are 2.13 times more likely to connect in this network (ceteris paribus).[6] The probability of an additional edge is then positively impacted by the technological proximity. More specifically the average degree of the network is positively impacted by technological proximity.

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

Formula:    Network ~ edges + edgecov(proximity)

Iterations:  7 out of 20

Monte Carlo MLE Results:
                      Estimate  Std. Error MCMC%  p-value
edges                  -4.5222      0.1972      0  < 1e-04 ***
edgecov.proximity       0.7575      0.2058      0 0.000234 ***
```

6 $e^{0.7575}$ since this is the odds and not the log-odds.

```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

     Null Deviance: 21349  on 15400  degrees of freedom
Residual Deviance:  3096  on 15398  degrees of freedom

AIC: 3100    BIC: 3115    (Smaller is better.)
```

A network analysis performed on this network showed that the network has a scale-free structure. This information can be helpful in the modeling of the ERGM as we have information on the distribution of the degrees. The same is valid for any other information about the network, small-world properties, level of clustering or centrality distribution. The information provided by SNA allows a first understanding of the structural properties of the network that will allow for a more robust model once edge and nodal covariates are added.

So if the network ha a scale-free structure the structure should be explained by the degree distribution. To check this we can add different degrees to the model. This can be done by using the command *degree*(). On can simple add one statistic for one degree, i.e *degree*(3), for the impact of nodes with degree 3, or add multiple degrees as was done in the following model. Note that the addition of multiple degrees is achieved by writing *degree*(a : b) to add all the degrees between a and b.

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

Formula:   Network ~ edges + degree(2:6) + edgecov(proximity)

Iterations:  7 out of 20

Monte Carlo MLE Results:
                   Estimate Std. Error MCMC% p-value
edges               -4.2417     0.1606     0 < 1e-04 ***
degree2             -0.8945     0.2233     0 < 1e-04 ***
degree3             -1.4608     0.2381     0 < 1e-04 ***
degree4             -1.8040     0.2720     0 < 1e-04 ***
degree5             -1.6668     0.2699     0 < 1e-04 ***
degree6             -2.0959     0.4109     0 < 1e-04 ***
edgecov.proximity    0.5451     0.1726     0 0.00159 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

     Null Deviance: 21349  on 15400  degrees of freedom
Residual Deviance:  2989  on 15393  degrees of freedom

AIC: 3003    BIC: 3056    (Smaller is better.)
```

The addition of these variables to the model once again decreases the AIC and the BIC, the model is hence enhanced. The structure of the network can be explained by a degree distribution.

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

Formula:   Network ~ edges + degree(2:6) + edgecov(proximity2)

Iterations:  7 out of 20

Monte Carlo MLE Results:
                    Estimate  Std. Error MCMC\% p-value
edges                -4.2382      0.1815      0 < 1e-04 ***
degree2              -0.8764      0.2053      0 < 1e-04 ***
degree3              -1.4438      0.2229      0 < 1e-04 ***
degree4              -1.7876      0.2728      0 < 1e-04 ***
degree5              -1.6763      0.2874      0 < 1e-04 ***
degree6              -2.1068      0.4595      0 < 1e-04 ***
edgecov.proximity2    0.5446      0.1941      0 0.00502 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

     Null Deviance: 21349  on 15400  degrees of freedom
 Residual Deviance:  2989  on 15393  degrees of freedom

AIC: 3003    BIC: 3057    (Smaller is better.)
```

These models seem to work qui nicely. We discussed in the previous sections that models were prone to degeneracy and the solutions to this problem. Let’s have a look at how we can model ERGMs with alternating k-stars and a geometrically weighted degree distribution.

5.1 Curved Exponential Random Graph Models

Studies show that the addition of weights on the degree distribution helps to avoid bi-modal distributions in the parameter space, i.e avoids the generated networks from being either full or close to empty. Different forms can be added to the R code. Since we have here an undirected network we can use either the alternating k-stars $altkstar()$ or the geometrically weighted degree distribution $gwdegree$. For directed graph there are additional commands which work in similar manner as what we show here. We include here a statistic for the gwdegree with the option $fixed = TRUE$. The latter means that we do not make an estimation of the scaling parameter, we want it to be equal to 1. The resulting model is hence not a curved ERGM.

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

Formula:   Network ~ edges + triangles + edgecov(proximity2) + gwdegree(1,
    fixed = TRUE)

Iterations:  NA

Stepping MLE Results:
                    Estimate  Std. Error MCMC%  p-value
edges              -5.717e+00  1.829e-01      0  < 1e-04 ***
triangle            1.802e+00  3.026e-05      0  < 1e-04 ***
edgecov.proximity2  6.811e-01  2.159e-01      0 0.001607 **
gwdegree            2.917e-01  8.380e-02      0 0.000502 ***
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

     Null Deviance: 21349  on 15400  degrees of freedom
Residual Deviance:  4170  on 15396  degrees of freedom

AIC: 4178    BIC: 4208    (Smaller is better.)
```

In order to have a curved exponential random graph model, the parameter that defines that we fixes in the previous code has to be estimates as well. In the following code we estimated the model with an edgewise shared partners variable. This variable is used to check for transitivity. By switching the option $fixed = TRUE$ to $fixed = FALSE$ the model becomes curved. The results now include an estimate for the parameter alpha of the model. Note here that the parameter can only be interpreted if the $gwesp$ statistic is significant.

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

Formula:   Network ~ edges + edgecov(proximity) + gwesp(alpha = 1, fixed = FALSE)

Iterations:  NA

Stepping MLE Results:
                   Estimate  Std. Error MCMC% p-value
edges              -5.39343   0.45601      0  <1e-04 ***
edgecov.proximity   0.48255   0.51641      0    0.05 ***
gwesp               1.19503   0.08333      0  <1e-04 ***
gwesp.alpha         0.88784   0.09792      0  <1e-04 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

     Null Deviance: 21349  on 15400  degrees of freedom
Residual Deviance:  2847  on 15396  degrees of freedom

AIC: 2855    BIC: 2885    (Smaller is better.)
```

The interpretation of these estimates are much more complex than previously. The parameters need to be exponentiated to find λ that we saw in the equations (Hunter 2007). We hence find $e^{0.88784} = 2.42$. Since the parameter is positive we can conclude that transitivity is present.

5.2 Goodness of Fit Diagnostics

In order to check if a model is a good fit we use the $mcmc.diagnostics$ command. This gives us a number of outputs, notable the matrix of correlations and p values for both the individual parameters and the model as a whole.

```
Sample statistics cross-correlations:
                      kstar2  edgecov.proximity
kstar2             1.0000000          0.5494967
edgecov.proximity  0.5494967          1.0000000

Individual P-values (lower = worse):
```

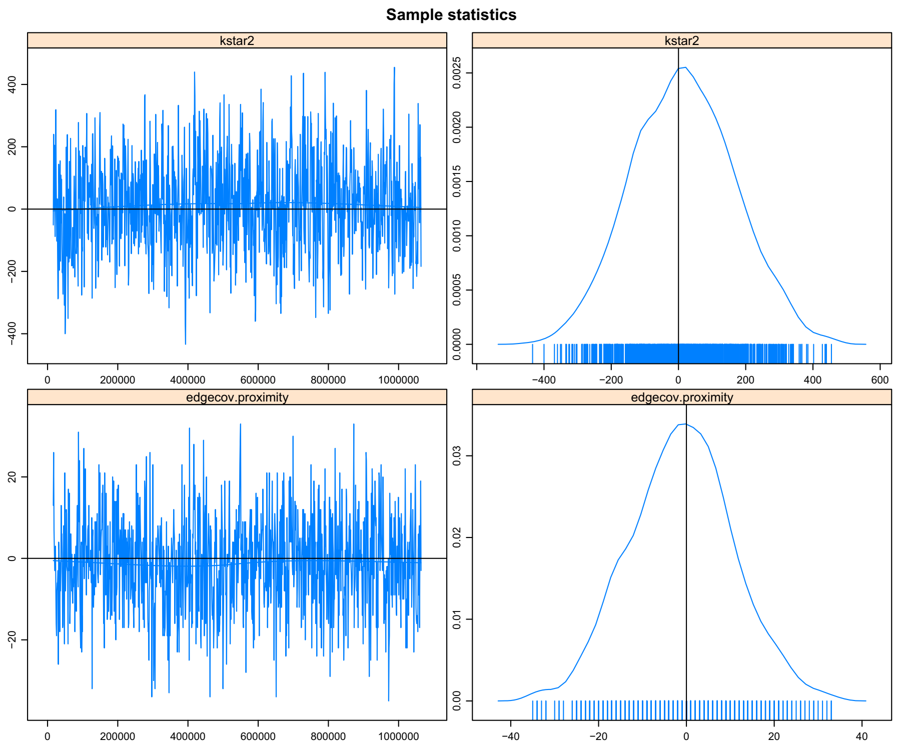

Fig. 14 Goodness of fit diagnostics

```
           kstar2  edgecov.proximity
        0.3126642           0.9294963
Joint P−value (lower = worse):   0.6994233  .
```

The p values are high for the parameters and the model, we can hence conclude that the model is globally significant. In order to go into a bit more detail when it comes to the estimates, the commands also provides us with plots, see Figs. 14 and 15. We stated that an ERGM should fit the observed network perfectly, on average. This means that from the simulated networks we expect the average values to be those of the observed network. If this is not the case then the sample the model is based on does not come from the same distribution as the observed network.

Figure 14 shows an example of what we want to observe. in the first graphs the values oscillate around the mean which is what we want. The graphs on the right hand show a centered distribution of the values, we hence conclude that the model is a good fit. A bad example can be found in Fig. 15. The graphs show that the distribution is not at all centered, and there is no oscillation around the mean. This model is hence a bad fit.

One can also study the goodness of fit using the $gof()$ command. Using a plot command this provides a box plot (see Fig. 16).

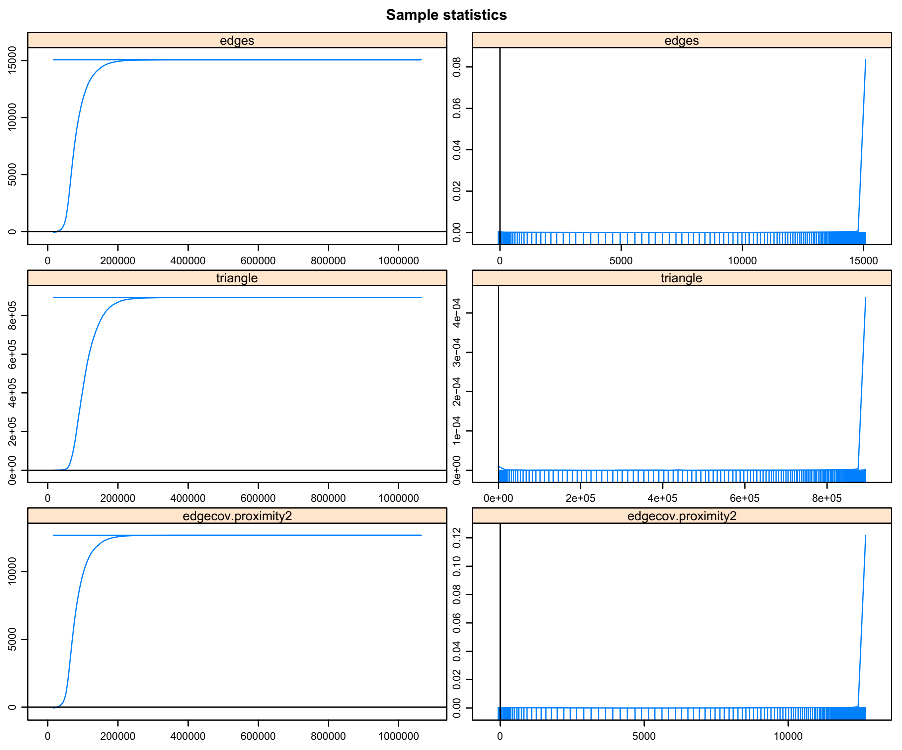

Fig. 15 Goodness of fit diagnostics, bad example

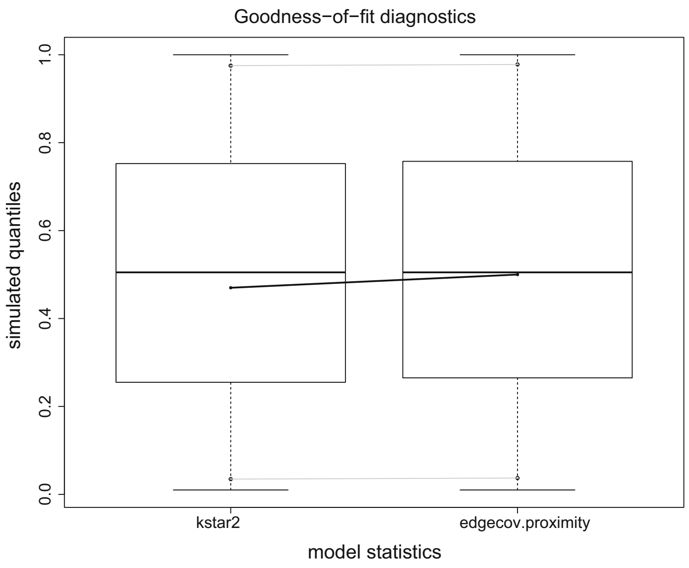

Fig. 16 GOF: boxplot analysis

The gof command can receive different parameters, one can chose to plot any number of variables and decide to increase or decrease the number of simulations to refine the results. The following code provides the GOF of the whole model ($GOF = Model$) using 20 simulations ($nsim = 10$).

```
gof_model <- gof(model, GOF = ~Model, nsim=20)
```

The results give us a box plot per variable and a black line representing the observations on the empirical network. Since we want the mean of the simulations to be equal to the observed network, the dark line should coincide with the center of the boxplots (the vertical line in the boxplot representing the median of the distribution). This is the case here, the model is hence a good fit.

5.3 Improve Bad Models

The fitting of an ERGM is a trial and error procedure. If a model behaves badly there are a couple of parameters to change in order to improve results. Of course one should only start these procedures once the variables chosen are stabilized. Starting with a null model containing only edges and adding on to this model while comparing the AIC and BIC values to find the variables of importance.

Once this is done and degeneracy is still observed one can start by switching estimation methods. One method might work better in one case than the other.

The estimation algorithm can be chosen in the control arguments of the $ergm()$ command.

```
model <- ergm(Network~ edges, control=control.ergm(main.method = "Stepping"))
model <- ergm(Network~ edges, control=control.ergm(main.method = "Stepping",MCMC.
    samplesize=70000, MCMC.interval=5000))
```

The burn-in can also solve problems, the burin represents the number of iterations that are removed from the simulation. It other words, the higher the burnin the more the procedure forgets about it's initial parameters. Increasing this value hence allows for keeping only the latest values which might represent the real values better. This can be achieved by adding an option to the ERGM.

A second method of improving estimation would be to increase the sample size by changing the parameter *MCMC.samplesize*. This increase will result in having more precise estimates by an increase in the number of statistics drawn from the sample. This, of course, increases computation time. The ERGM package includes multicore features that can help reduce computation time drastically. All this requires is the addition of some parameters to the $control.ergm$ argument. Adding $parallel = 4$ notifies the package that the computer has 4 cores, $parallel.type$ sets the type of multicore. For a regular computer this should be fixed to "*PSOCK*". This will distribute the computations over the 4 cores of the computer and hence increase speed.

```
a <- ergm(Network~edgecov(proximity2)+triangles+altkstar(1.812, fixed=FALSE),control=
    control.ergm(main.method=c("Stepping"), parallel=4, parallel.type="PSOCK", MCMC.
    samplesize=20000))
```

6 Conclusion

With the interest for network analysis growing in economics, ERGM are an efficient tool for an in-depth analysis while checking for statistical significance. The complexity of social and economic phenomena are difficult to assess and even though existing methods such as block-models and classic logistic regressions do not allow an assessment of all the intricate rules that are at play. By taking into account the impact of the current structure of the network ERGM distinguish themselves from other methods of network analysis and allow for more accurate hypothesis testing. Being able to add relational data to an analyses allows for increasing the precision of existing models as well as the testing of new hypotheses.

Even though the models are powerful, the tools used for their analysis still need to be improved. The fast expansion of research on the subject of ERGMs is tackling issues improving the quality of ERGMs overall. One important issue is worth mentioning here, which is the size of the network under analysis. In the present paper the network had a size of 176 nodes, large networks can cause significant issues with computation time or the program simply refusing to run. Recent research is aiming at tackling this issue (Schmid and Desmarais 2017; Bouranis et al. 2017).

Many questions in economics are related to the strategies of link creation which are difficult to answer with other methods of analysis. Whether they are used to analyse social networks, collaboration networks, trade networks or financial networks, ERGMs can provide vital insights into the understanding of network dynamics. I should however point out at this point that the hypothesis of modeling networks with an exponential law should not be forgotten. Certain interactions might not be well suited for exponential laws, a notable example is citations networks. Research has shown that citation networks follow a power-law rather than an exponential law. For networks with an underlying power-law one should look at Dynamic Process Modeling which is based on a power-law.

References

Anderson, C. J., Wasserman, S., & Crouch, B. (1999). A p* primer: Logit models for social networks. *Social Networks*, *21*(1), 37–66.

Besag, J. E. (1972). Nearest-neighbour systems and the auto-logistic model for binary data. *Journal of the Royal Statistical Society Series B (Methodological)*, *34*, 75–83.

Besag, J. E. (1975). Statistical analysis of non-lattice data. *The Statistician*, *24*, 179–195.

Bouranis, L., Friel, N., & Maire, F. (2017). Bayesian model selection for exponential random graph models via adjusted pseudolikelihoods. arXiv preprint arXiv:1706.06344.

Broekel, T., & Hartog, M. (2013). Explaining the structure of inter-organizational networks using exponential random graph models. *Industry and Innovation*, *20*(3), 277–295.

Butts, C. T. (2008). Network: A package for managing relational data in R. *Journal of Statistical Software*, *24*(2), 1–36.

Caimo, A., & Lomi, A. (2015). Knowledge sharing in organizations: A Bayesian analysis of the role of reciprocity and formal structure. *Journal of Management*, *41*(2), 665–691.

Cantner, U., & Meder, A. (2007). Technological proximity and the choice of cooperation partner. *Journal of Economic Interaction and Coordination*, *2*(1), 45–65.

Carrington, P. J., Scott, J., & Wasserman, S. (2005). *Models and methods in social network analysis* (Vol. 28). Cambridge: Cambridge University Press.

Cranmer, S., Desmarais, B., & Menninga, E. (2012). Complex dependencies in the alliance network. *Conflict Management and Peace Science*, *29*(3), 279–313.

Efron, B. (1975). Defining the curvature of a statistical problem (with applications to second order efficiency). *The Annals of Statistics*, *3*, 1189–1242.

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

Geyer, C. J., & Thompson, E. A. (1992). Constrained Monte Carlo maximum likelihood for dependent data. *Journal of the Royal Statistical Society Series B (Methodological)*, *64*, 657–699.

Hammersley, J. M., & Clifford, P. (1971). Markov fields on finite graphs and lattices.

Handcock, M. S., Hunter, D. R., Butts, C. T., Goodreau, S. M., Krivitsky, P. N., Bender-deMoll, S., et al. (2008). Statnet: Software tools for statistical analysis of network data. *Journal of Statistical Software*, *24*(1), 1–11.

Handcock, M. S., Robins, G., Snijders, T. A., Moody, J., & Besag, J. (2003). *Assessing degeneracy in statistical models of social networks*. Citeseer: Technical report.

Harris, J. K. (2013). *An introduction to exponential random graph modeling* (Vol. 173). Beverly Hills: Sage Publications.

Hummel, R. M., Hunter, D. R., & Handcock, M. S. (2012). Improving simulation-based algorithms for fitting ergms. *Journal of Computational and Graphical Statistics*, *21*(4), 920–939.

Hunter, D. R. (2007). Curved exponential family models for social networks. *Social Networks*, *29*(2), 216–230.

Hunter, D. R., & Handcock, M. S. (2006). Inference in curved exponential family models for networks. *Journal of Computational and Graphical Statistics*, *15*(3), 565–583.

Hunter, D. R., Handcock, M. S., Butts, C. T., Goodreau, S. M., & Morris, M. (2008). Ergm: A package to fit, simulate and diagnose exponential-family models for networks. *Journal of Statistical Software*, *24*(3), 1–29.

Lomi, A., & Fonti, F. (2012). Networks in markets and the propensity of companies to collaborate: An empirical test of three mechanisms. *Economics Letters*, *114*(2), 216–220.

Lomi, A., & Pallotti, F. (2012). Relational collaboration among spatial multipoint competitors. *Social Networks*, *34*(1), 101–111.

Lusher, D., Koskinen, J., & Robins, G. (2012). *Exponential random graph models for social networks: Theory, methods, and applications*. Cambridge: Cambridge University Press.

Pattison, P., & Robins, G. (2002). Neighborhood-based models for social networks. *Sociological Methodology*, *32*(1), 301–337.

Pattison, P., & Wasserman, S. (1999). Logit models and logistic regressions for social networks: II. Multivariate relations. *British Journal of Mathematical and Statistical Psychology*, *52*(2), 169–194.

Robbins, H., & Monro, S. (1951). A stochastic approximation method. *The Annals of Mathematical Statistics*, *22*, 400–407.

Robins, G., Pattison, P., & Elliott, P. (2001). Network models for social influence processes. *Psychometrika*, *66*(2), 161–189.

Robins, G., Pattison, P., Kalish, Y., & Lusher, D. (2007). An introduction to exponential random graph (p*) models for social networks. *Social Networks*, *29*(2), 173–191.

Schmid, C. S., & Desmarais, B. A. (2017). Exponential random graph models with big networks: Maximum pseudolikelihood estimation and the parametric bootstrap. arXiv preprint arXiv:1708.02598.

Snijders, T. A. (2001). The statistical evaluation of social network dynamics. *Sociological Methodology*, *31*(1), 361–395.

Snijders, T. A. (2002). Markov chain Monte Carlo estimation of exponential random graph models. *Journal of Social Structure*, *3*(2), 1–40.

Snijders, T. A., Pattison, P. E., Robins, G. L., & Handcock, M. S. (2006). New specifications for exponential random graph models. *Sociological Methodology*, *36*(1), 99–153.

Strauss, D., & Ikeda, M. (1990). Pseudolikelihood estimation for social networks. *Journal of the American Statistical Association*, *85*(409), 204–212.

Ter Wal, A. L. (2013). The dynamics of the inventor network in German biotechnology: Geographic proximity versus triadic closure. *Journal of Economic Geography*, *14*(3), 589–620.

van der Pol, J. (2018). Explaining the structure of collaboration networks: From firm-level strategies to global network structure. *Les cahiers du GREThA*.

van der Pol, J., & Rameshkoumar, J.-P. (2018). The co-evolution of knowledge and collaboration networks: The role of the technology life-cycle. *Scientometrics*, *114*(1), 307–323.

Wasserman, S., & Pattison, P. (1996). Logit models and logistic regressions for social networks: I. An introduction to markov graphs and p. *Psychometrika*, *61*(3), 401–425.

White, H. C. (1992). *Identity and control: A structural theory of social action*. Princeton: Princeton University Press.