Extending the p2 model using triadic social network data: The estimation of dyadic probabilities using the individuals' perception on the social structure

Kamstra, Nelleke

Universiteit
Leiden
The Netherlands

Extending the p_2 model using triadic social network data

The estimation of dyadic probabilities using the individuals' perception on the social structure

Nelleke E. Kamstra

Thesis advisor 1: Saskia Le Cessie, Professor Statistical Methods in Observational (Clinical) Epidemiological Research
Thesis advisor 2: Marijtje van Duijn, Professor Statistics in Social Network Analysis

Defended on May 28, 2025

MASTER THESIS
STATISTICS AND DATA SCIENCE
UNIVERSITEIT LEIDEN

Acknowledgement

First, I would like to thank Marijtje van Duijn (external supervisor) for the idea of this project when looking for a thesis that incorporated sociology into statistics. Also for all the help (especially with understanding the theory and programming the C++ code when I got stuck), her ideas, all the meetups (online and in Groningen), her enthusiasm in academics and the good conversations we had. Especially many thanks for her patience over the years when I was not able to work on my thesis and for every time I made an attempt to continue my thesis. Fortunately last year I was able to do so again. I really enjoyed our collaboration and also enjoyed this project.

I also want to thank Henk Kelderman. He started as my supervisor from Leiden. He always gave me very good feedback and was interested in the field of statistics in social networks. He was very patient with me over the years when I could not work on my thesis and when I was focused on personal matters. Saskia Le Cessie took over the supervisor role for Leiden. I want to thank her for her enthusiasm about social networks and learning more about them. She helped me a lot (in Leiden and online) and gave me very good and critical feedback, listened and was patient in the process of me getting back to study besides working. Our meetings helped with gaining confidence and give structure to the thesis. Sometimes I was (and can be) stubborn, but all my supervisors showed respect and I learned a lot from all of them.

Of course, a lot of appreciation for my family (mum, dad and sister) and friends for their emotional support last years. They stood by me when I needed it the most. Thanks for all the calls and moments to relax after work and writing my thesis. Because of this I am now able to finish this thesis and I am in a stronger and healthier place.

Contents

Abstract

Two-dimensional (self-reported) social network data is commonly used in social networks analysis to investigate the self-reported social relationships between network members. However, Krackhardt (1987) proposed to use three-dimensional (triadic) data based on the theory of cognitive social structures. Actors are not only asked to report their own relations within a social context, but are also asked to report on the relationships between other actors in the same network. In this thesis we will extend the p_2 model by using the triadic data (instead of self-reported data) to estimate dyadic probabilities given the perception of the third actor in a triad. The p_2 model is used, because it includes multiple dependencies found in social networks (e.g., reciprocity, describes a relationship where both individuals see each other as friends) which makes the model more realistic to other models (Van Duijn et al, 2004). We build on the application of Bellio and Soriani (2020) who used Maximum Likelihood Estimation with Laplace approximation to estimate the parameters of the p_2 model. They provided the C++ and the R code, which we use as a basis for our triadic extension of the model.

We programmed the model in C++ and R. We used the triadic high-tech managers data (Krackhardt, 1987) and an aggregated version (as example for self-reported data) to show an application of the triadic model and the p_2 model side-by-side. We encountered the problem of multicollinearity when adding dyadic characteristics to the parameter μ of the triadic model. It was possible to converge the model when reducing the number of dyadic-specific comparisons between actors within the density parameter and only include the comparison between actor i and j.

For reciprocity and the homophily parameters the standard errors were lower in the triadic model than in the p_2 model, indicating more precision. The characteristics of the perceiver added to this triadic model give more information about the effect of the perception of a third actor on the directed ties of other actors. Multiple suggestions for future research are made for overcoming the problem of multicollinearity, such as looking into the correlations between variables and the comparisons for homophily that can be added to our model or not. Besides this, a next step is to perform multiple goodness-of-fit tests. We conclude that this triadic model is a more realistic way to look at the role of the perceiver on the dyadic probability in a social network in combination with the effects of actor characteristics of the sender, receiver and the perceiver.

1 Introduction

In our lives we are part of different social contexts. These can be family, a group of friends, work, school, even the neighborhood you live in and the sports club. All these different environments generate social networks. Sociologists are interested in these social networks, because they initiate influence, dependence, power, trust, opinions and of course friendships.

The complexity of working with social networks and extracting information from social networks lies not only in computational and measuring problems, but also in the vast variety of questions and perspectives one can have (Snijders, 2011). In social networks the terms actors and ties are used. Actors are the individuals in a social network. Ties are possible (friendship) relations between these individuals. In this thesis we focus on social networks with directed ties, ties that can be outgoing/sent or ingoing/received. An example in Figure 1 of an outgoing or a sent tie is from actor A to actor E, meaning that actor A indicates E as a friend. Ingoing or received ties occur when actor A receives a tie from actor C, meaning that actor A receives the status as friend from actor C. We will focus on dichotomous dyadic ties in this thesis. A dyad is a pair of actors with two possible directed ties. Dichotomous ties signify ties which exist (1) or not (0). Reciprocal ties are commonly found in social networks and means that both actors indicate that the other actor is a friend, they reciprocate the relationship. An example of reciprocity in Figure 1 is between actor C and actor F. A non-reciprocal tie occurs when only one actor perceives the other as their friend, but not vice versa. A null dyad is a dyad where there are no ties between two actors. In Figure 1, we see an example of this between actor E and actor F.

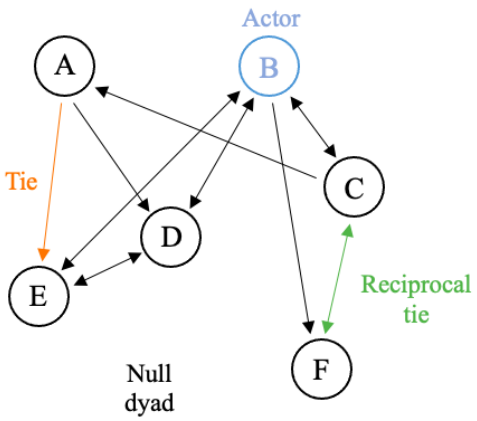


Figure 1. Theoretical visualization of a network with ties and actors

It is possible to measure ties and actors in ego- and socio-centered networks (Kadushin, 2012, Chapter 2). In ego-centered networks a specific individual is the center of the analysis. In socio-centered networks a social network in a specified social group with a specific number of actors is identified, such as a class in a school or a department in an organization. These socio-centered networks can answer questions related to the whole social network and its characteristics. In this thesis we focus on the socio-centered friendship networks.

When analyzing social networks, it is sometimes hard to measure all the relations between the N actors and include all the actors in the social network (as it is in regular data collection). Also in larger social networks the researcher does not want to exhaust the time of the respondent if they need to answer for every other actor whether they are connected with them or not (Stark, 2017). The most common way to collect social network data is asking every actor in the network to list who their friends are in the predetermined social group. Because of this, the sent and received relations per actor are self-reported. With this information it is possible to create an adjacency matrix as in Table 1 indicating the senders (rows) and the receivers (columns).

	A	B	C	D	E	F
A	-	0	0	1	1	0
B	0	-	1	1	1	1
C	1	1	-	0	0	1
D	0	1	0	-	1	0
E	0	1	0	1	-	0
F	0	0	1	0	0	-

Table 1. Adjacency matrix for graph visualization of the social network in Figure 1

Social networks are complicated, because of dependencies present between actors and within dyads and the influence on one another. In social networks the influence of a third actor on the relationship between two other actors is not overlooked. Concepts and theories as triadic closure and transitivity (Kadushin, 2012, Chapter 2) assume that the relationship of actor i and j can be determined by whether they have a relationship with a third actor k. For example, the probability of a reciprocal relationship between actor i and j could be partly determined by others in the complete social network.

So a third individual in the network can influence this outcome of the dyadic relationship between i and j. Instead of only looking at the directed ties between two actors,

we can look at the possible relations between three actors (also called a triad) and the influence of the perception of the third actor on the directed ties between the other two actors in the triad. Instead of only looking at the self-reported data, it is also possible to collect data which focuses on these triads. To collect this data we should not only ask the individual who they report as their friends, but they also are asked how they perceive the (non-)existence of relationships of other dyads which they are not a part of. So for each actor in the social network a complete social network is collected. This is called triadic data.

Existing models using these data are based on the Social Relation Model (Kenny & La Voie, 1984; Snijders & Kenny, 1999). Some focus on estimating continuous valued ties, others on estimating the probability of the dichotomous directed ties in a dyad (e.g., Bond et al, 1997; Swartz et al 2015). This research integrate important dependencies in a social network by adding random effects for sender and receiver effects using all the N layers without aggregating them and with different ways to approximate the estimation. However, their model is not realistic enough as it does not include the important element of reciprocal ties which are common in social networks. A model that includes both random effects for sender and receiver and a parameter that estimates reciprocity, is the p_2 model (Van Duijn et al, 2004). However, it does not use the triadic data and the perception of the third actor. In this thesis, we want to produce a model that not only includes the influence of a third actor in estimating probabilities of dyadic relationships in a complete social network but also includes the important dependencies within a social network. Because of this, we will extend the p_2 model and its elements.

1.1 Outline thesis

We will model the probability of directed ties (existing or not) in a dyad perceived by a third actor to take into account the influence of a third actor on the relationships within a dyad. Including the dependencies that make it more realistic to the elements of a social network. In the situation of the dyadic data, the p_2 model is seen as a realistic model, because it models the dependencies found in social networks by (correlated) random effects and by the addition of a fixed reciprocity parameter (Van Duijn et al., 2004). It can also include actor covariates which could have an influence on the outcome.

In Chapter 2, we first expand on the random effects found in social networks to capture the dependencies commonly found. A more detailed description is given of the p_1 and p_2 model and its components as well as a short explanation of their background and similar existing models to the p_2 model. In Chapter 3 we expand on the specifics of the triadic

extension based on the p_2 model and the triadic data that will be used. Subsequently, in Chapter 4 we elaborate more on the estimation method in this thesis and here we present the likelihood function of the extension with the adjusted estimation.

In Chapter 5 we present the application of this triadic extension. First of all, we adjusted the code of Bellio and Soriani (2020) to our triadic extension and triadic data. An existing triadic dataset is used to show the model in a practical situation. The data is described and the results of the models are given. We will compare the results of triadic model with the results of the p_2 model. In Chapter 6 we discuss the conclusions, recommendations for future research and possible shortcomings.

2 Background

First it is important to discuss the significance of using random effects in estimation models of complete social networks. We give a description of how random effects models work and how this relates to social networks. Then we give an explanation and reasoning behind the p_1 model and the p_2 model.

2.1 Random effects model and social networks

2.1.1 Random effects

When describing the real world, observations can be dependent on certain situations and environments. For example, the performance of children in school can depend on the class they are in. But this dependence is also common in other fields. For example, experiments in plant sciences where background noise such as sunlight can disturb the independence of the different experimental fields with the same crops. Normal regression analysis assumes that the residuals in the models are independent to make the model not too complicated and more applicable in more situations. However, in social settings the assumption of independence is not realistic. It is possible to account for dependence by adding random effects to the model. We can make a distinction between nested and crossed random effects. Nested random effects assume hierarchy in observations, such as students in classes in schools. Crossed random effects indicate a random combination of different levels of multiple conditions which occur multiple times.

As an example we can investigate the effect of parental support on the exam results of students in different classes. In this example we can observe a nested structure where the students' exam results are at level 1. The students' exam scores might however also vary between classes, for example because of differences between teachers. We can observe the hierarchical structure where the students (level 1) are nested in the classes (level 2). Instead of calculating one fixed intercept for the model in a regular linear model over all the students for the effect of parental support on exam results, we expect different modelled intercepts per class. These different intercepts can be grasped by adding a random effect for classes to the intercept of the linear model:

$$Y_{ij} = (b_0 + u_{0j}) + b_1 X_{1ij} + \epsilon_{ij} \tag{1}$$

Where b_0 is the fixed intercept and u_{0j} is the random effect for classes where j represents

the class and i the students. By adding this random effect it is possible to account for the dependency between the students' results and the class they are in. This represents a nested random effect. However, there is also a possibility of the parental support (X_1) depending on the class, where maybe the teacher encourages the children to ask the parents for help in different levels of encouragement. And that we expect the slopes of the model also to be randomly different for each class. In the example of parental support it means that the effect of parental support on the results is randomly different for each class. To achieve this we cross the level 1 students and the level 2 classes by adding a random effect to the estimate of parental support (also including the random intercepts):

$$Y_{ij} = (b_0 + u_{0j}) + (b_1 + u_{1j})X_{1ij} + \epsilon_{ij} \tag{2}$$

Where u_{0j} is the nested random effect that creates the random intercepts and u_{1j} is the crossed random effect creating the different random slopes for the classes. By adding these random effects in the normal linear regression model, dependence can be accounted for that is created by these nested and crossed random effects in the data.

2.1.2 Random effects in social networks

In social networks, we can also identify multiple forms of dependencies which we can relate to random effects. Relations are dependent on a lot of factors and especially other actors in the network. Dyads are not independent in a complete social network, because they can share the same actors. Thereby, relationships influence each other by whether their sent relationship is reciprocated by the other actor or not. Even ties with other actors can influence the (non-)existence of ties between different actors. Instead of only nested structures, social networks also have a crossed structure.

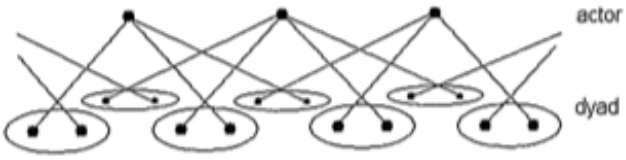


Figure 2. Schematic representation crossed and nested social network structure (taken from Van Duijn, 2013)

The actor is placed at the highest level, here level 3. The dyads are on level 2 and the ties are at level 1. In Figure 2 taken from Van Duijn (2013), we can see the actors on level 3 where the crossed lines represent the sent and received ties from both actors forming a dyad on level 2. A crossed random effect is introduced between the actors and the dyads, because every dyad contains two actors and the actors will be involved in multiple dyads (Van Duijn, 2013). Random effects are introduced to models that make use of complete social network data to model the variances. However, the introduction of random effects makes the estimation of the models more difficult. Solutions for these estimation problems are available.

2.2 p_1 and p_2 model

Complete social networks bring the violation of independency between dyads. There are multiple different dependencies present in social networks (Snijders, 2011). For example within dyad dependencies, where we talk about the probability of reciprocating ties. If actor i sends a tie to actor j, it is more likely that actor j also sends a tie to actor i. An actor is also part of multiple dyads which introduces dependencies between dyads. Also there is a dependency between the actor's sent ties and received ties. If an actor sends more ties, it can impact how many ties they will receive.

A solution can be found in the p_1 model of Holland and Leinhardt (1981). It introduces fixed parameters for sender and receiver and it adds a parameter for reciprocity as an interaction effect. It captures the dependency for reciprocity, which made it more realistic than previous models as a loglinear model. However, the problem of the p_1 model is that for every actor in the network an estimate for the sender and receiver effect is calculated. This makes the model complex with unnecessary information, creates overfitting and generates a model that is more difficult to interpret.

We can also look at the Social Relation Model (SRM; Kenny & La Voie, 1984) and especially its extension (Snijders & Kenny, 1999). The SRM models the sender and receiver effect with an addition of a relationship effect. In the extension of Snijders and Kenny (1999) they treat the sender and receiver effects as random, which allows for actor and dyadic characteristics in the model. However, SRM considers a continuous valued variable as outcome, not a dyadic outcome such as a (non-)existing tie which is especially common in social network analysis and it also does not include the reciprocity found commonly in social networks.

Exponential Random Graph Models (ERGMs) (Amati et al, 2018; Robins et al., 2007; Robins & Lusher, 2013) includes these dependencies through local configurations in the

form of certain network structures (e.g. triangles, star structures). In these models the probability of ties given the rest of the complete social network is calculated with also possibilities for different levels of flexibility for dependence in the network. For example it makes the assumption that if actors do not share any direct friends, they will be further away from each other and the probability of forming ties will be smaller. It is a very flexible model that can include multilevel structures of networks and also actor attributes (Robins & Lusher, 2013, Chapter 2). However, computation is more difficult and interpretation is harder for ERGMs.

The p_2 model combines the random effects for sender and receiver parameters from the SRM (Snijders & Kenny, 1999) and the reciprocity parameter and dyadic outcome from the p_1 model (Van Duijn, 2004). By using the p_2 model we overcome the problem of the nested and crossed structure in social networks and the fact that we want to look at a dyadic outcome with four possible combinations in one dyad (Van Duijn, 2013).

2.2.1 p_1 model

Holland and Leinhardt (1981) introduced the p_1 model to account for reciprocal ties commonly found in social networks, as a first step to a more realistic model:

$$\begin{gathered} P(Y_{ij} = y_1, Y_{ji} = y_2 \mid A_i, B_i, A_j, B_j) = \\ \frac{1}{h_{ij}} \exp\{y_1(\mu + \alpha_i + \beta_j) + y_2(\mu + \alpha_j + \beta_i) + y_1 y_2 \rho_{ij}\} \\ y_1, y_2 = 0, 1; i, j = 1, ..., n; i \neq j \end{gathered} \tag{3}$$

where

$$\begin{gathered} h_{ij} = 1 + \exp(\mu_{ij} + \alpha_i + \beta_j) + \exp(\mu_{ji} + \alpha_j + \beta_i) + \\ \exp(2\mu + \alpha_i + \beta_j + \alpha_j + \beta_i + \rho_{ij}) \end{gathered} \tag{4}$$

The model calculates the probability of a dyad occuring in a social network with actor i as the sender and actor j as the receiver. Instead of having two outcomes (0 and 1), a dyad has four possible outcomes. The combination $y_1 = 1$ and $y_2 = 1$ captures a reciprocal relationship. When $y_1 = 1$ and $y_2 = 0$ or $y_1 = 0$ and $y_2 = 1$ this captures a one-sided relationship. When $y_1 = 0$ and $y_2 = 0$ there is no relationship between actors in the dyad (also a null dyad). In this model the reciprocity is accounted for by including fixed parameters for sender and receiver effects α and β respectively and by including an explicit reciprocity parameter ρ. For each actor a value for the α and β effect is estimated.

When there is a null dyad, ρ will not be taken into account to predict this probability. The μ parameter can be seen as the overall mean or density of the network and is thus primarily used as the intercept of the model. This parameter is assumed to be the same for all dyads.

2.2.2 p_2 model

Van Duijn et al. (2004) extended the p_1 model to the p_2 model by treating the sender (α) and receiver (β) effects as random effects instead of as fixed. This makes it also possible to include actor characteristics X as covariates related to the sender and receiver:

$$\alpha_i = X_{1i}\gamma_1 + A_i, \tag{5}$$

$$\beta_i = X_{2i}^T\gamma_2 + B_i \tag{6}$$

With γ_1 and γ_2 being the regression parameters, and the parameters A and B being random sender and receiver effects respectively. In this way, it is possible to take into account the dependence structure by making the dyads become conditionally independent. If $u_i = (A_i, B_i)$ the random effects are assumed to be normal distributed random variables $u_i = N_2(0, \Sigma)$ with:

$$\Sigma = \begin{bmatrix} \sigma_A^2 & \sigma_{AB} \\ \sigma_{AB} & \sigma_B^2 \end{bmatrix} \tag{7}$$

A correlation is added between the sender and receiver effects to account for the crossed structure in a network. Each actor will be the sender as well as the receiver of directed ties. Independence is assumed between the random parameters of different actors:

$$cov(A_i, A_j) = cov(B_i, B_j) = cov(A_i, B_j) = 0$$

Another addition of the p_2 model is that the parameter μ can contain dyad-specific covariates which represent characteristics to account for homophily. Homophily implies that actors with similar characteristics (e.g., sex, age) are more likely to become friends. It is modelled as this, where δ_1 is the parameter for homophily:

$$\mu_{ij} = \mu + Z_{1ij}\delta_1 \tag{8}$$

The parameter ρ measures reciprocity in the model, as it does in the p_1 model. It can be modelled as this:

$$\rho_{ij} = \rho + Z_{2ij}\delta_2 \tag{9}$$

With the possibility to also add dyad-specific characteristics. However, in this thesis we will not use this and only will model ρ without the dyad-specific characteristics. The addition of the random effects and the reciprocity parameter makes the outcome more realistic, better interpretable, and also still manageable to estimate.

The p_2 model uses two dimensions for the sender and the receiver. To estimate the likelihood for the p_2 model it is necessary to multiply the two likelihoods together, resulting in the likelihood function given the random effects A and B (Bellio & Soriani, 2020):

$$p(y|u;\theta) = \prod_{i=1}^{n-1} \prod_{j=i+1}^{n} p(y_{ij}, y_{ji}|u_i, u_j) \tag{10}$$

With $p(y_{ij}, y_{ji}|u_i, u_j)$ corresponding to (3). The parameters that are estimated in the p_2 model are:

$$\theta = (\gamma_1, \gamma_2, \mu, \delta_1, \rho, \sigma_A^2, \sigma_B^2, \sigma_{AB}) \tag{11}$$

3 Triadic extension

3.1 Triadic data

Traditional social network methods are focused mainly on self-reported data and the actors within the dyad. The researcher works with one two-dimensional adjacency matrix including the actors sending ties and receiving ties from other actors (as in Table 1). However, Bond et al. (1997) and Krackhardt (1987) considered the use of triadic data. Here a third person is involved in judging the social relations in a complete social network. Instead of asking participants to report (a fixed number of) their relations within a social context, they have to report every possible relation in the complete network. The researcher will end up with as many complete social networks as there are actors in the network. In Figure 3 taken from Card, Rodkin and Garandeau (2010) we can see that each perceiver has produced an adjacency matrix of the ties between senders ('Actor') and receivers ('Partner').

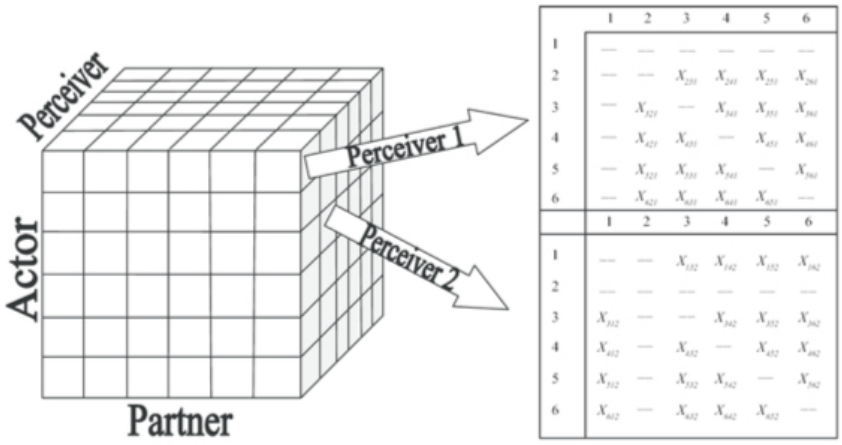


Figure 3. Visualization of the layers in triadic data (Card, Rodkin & Garandeau, 2010)

Krackhardt (1987) discusses the use of triadic social network data based on cognitive social structures (CSS) more in theory. The basis of these CSS lies in how an actor perceives relations with and between their friends, and adjusts their relations according to these perceptions. According to balance theory (Heider, 1958), if an actor k is friends with actor i and j, they may believe that i and j are also friends and reciprocate their ties. It does not matter whether actor i and j are really friends and really reciprocate the friendship to k. The idea behind these triadic data is also that the precision and consensus of self-reported relations within a social network can be investigated. Krackhardt (1987)

describes three ways of using these triadic data: one could aggregate the different layers, take slices from it or use all layers in the data. In this thesis we will make use of all the layers in the data as visualized in Figure 3. In this way it is possible to take into account the effect of actor characteristics on the probability of a relationship between two actors.

3.2 Triadic model

A triad consist of three actors in a complete social network with six possible ties in three dyads. Figure 4 visualizes a triad with actor i, j and k and the six ties between the three actors. If we look at the dyad between i and j, it consists of the ties Y_{ijk} and Y_{jik} with i and j taking both the role as sender and receiver and k perceiving the ties in this dyad. When looking at the other two dyads the roles change. The dyad between actor i and k consists of the ties Y_{ikj} and Y_{kij}, but here actor j perceives this dyad; the third dyad between actor j and k consists of the ties Y_{jki} and Y_{kji} with actor i perceiving those ties. Corresponding to the p_1 and p_2 model, the tie variables Y_{ijk} and Y_{jik} can take the values 1 (relationship exists) or 0 (relationship does not exist). Just as in the p_2 model there are four possible outcomes for one dyad. So in a triad there are 12 possible outcomes.

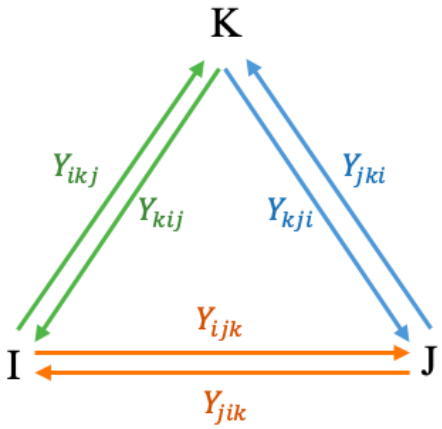


Figure 4. Visualization of a triad

Every actor in a complete social network is part of multiple triads and exchanges the role of perceiver with the role of the sender and receiver depending on the triad looked at. With every perceiver a layer of the triadic data is selected (see Figure 3). Each layer is a two-dimensional adjacency matrix of a social network perceived by the selected actor. In our model we will select and use every layer instead of only using one two-dimensional layer. Because of this, we can use the p_2 model for the extension as it

models the probability of all dyads in a two-dimensional social network. Instead of only using the information of the sender and receiver effects from one layer, we extend the model with a third random effect for the perceiver. By including this parameter, it will take into account the possible effect of the perceiver on the probability of ties existing in a dyad. Because of the addition of this extra effect, our model for one side of the triad is formulated as this:

$$\begin{gathered} P(Y_{ijk} = y_1, Y_{jik} = y_2 \mid A_i, B_i, A_j, B_j, C_k) = \\ \frac{1}{h_{ijk}} \exp\{y_1(\mu_{ijk} + \alpha_i + \beta_j + \eta_k) + y_2(\mu_{jik} + \alpha_j + \beta_i + \eta_k) + y_1 y_2 \rho_{ijk}\} \\ y_1, y_2 = 0, 1; i, j, k = 1, ..., n; i \neq j, k; j \neq k \end{gathered} \tag{12}$$

where h_{ijk} becomes

$$\begin{gathered} h_{ij} = 1 + \exp(\mu_{ijk} + \alpha_i + \beta_j + \eta_k) + \exp(\mu_{jik} + \alpha_j + \beta_i + \eta_k) + \\ \exp(\mu_{ijk} + \mu_{jik} + \alpha_i + \beta_j + \alpha_j + \beta_i + 2\eta_k + \rho_{ijk}) \end{gathered} \tag{13}$$

This model captures the actor i as sender and actor j as receiver in a dyad, but with the addition of the third actor k as the perceiver. In comparison with the p_2 model, the addition of the random effect η for the perceiver effect is the only adjustment to connect every perceived social network in our model. This can be explained by the following; in Figure 4 the three different actors have different roles of sender, receiver or perceiver, depending on which dyad is looked at. So each actor will adopt the other roles too. This ties together the three dyads and also all the perceived layers in the triadic data. It introduces overlap and similarity as the same three actors in a triad all occur in each of the three dyads. Conditional on the random effects, the triads in a complete social network are assumed to be independent. The three dyads within one triad are also assumed to be independent given the random effects, however they still relate because of each actor having each role of the sender, receiver and perceiver dependent on which dyad is looked at. Thereby, the random perceiver effect (η) is correlated with the random sender and receiver effects for the same actor. The sender (α) and receiver (β) random effects are equal to the random effects in the p_2 model.

We will model the fixed effects μ and ρ and the random effects α, β and η. α, β and η are the random effects in the model and represent the sender, receiver and perceiver effect respectively. The random effects can be regressed on the actor covariates X with A, B and C representing the residual variance as in Van Duijn et al (2004) and γ representing

the vector of parameters which needs to be estimated. A, B and C are also assumed to be normally distributed. Taking the example of the dyad in (12) focusing on actor i:

$$\alpha_i = X_{1i}\gamma_1 + A_i \tag{14}$$

$$\beta_i = X_{2i}\gamma_2 + B_i \tag{15}$$

$$\eta_i = X_{3i}\gamma_3 + C_i \tag{16}$$

The three residual terms (A_i, B_i, C_i) are normally distributed with mean 0 and variance Σ. It results in the covariance matrix in which six variance parameters are estimated:

$$\Sigma = \begin{bmatrix} \sigma_A^2 & \sigma_{AB} & \sigma_{AC} \\ \sigma_{AB} & \sigma_B^2 & \sigma_{BC} \\ \sigma_{AC} & \sigma_{BC} & \sigma_C^2 \end{bmatrix} \tag{17}$$

Independence is assumed between the residuals of different actors:

$$\text{cov}(A_i, A_j) = \text{cov}(B_i, B_j) = \text{cov}(C_i, C_j) = \text{cov}(A_i, B_j) = \text{cov}(A_i, C_j) = \text{cov}(B_i, C_j) = 0$$

for $i \neq j$. In the p_2 model it is assumed that the random sender and receiver effects are dependent for the same actor (Van Duijn et al, 2004). In this extension it is also assumed that the random perceiver effect is dependent on the sender and receiver effects for the same actor. Consequently, covariance is introduced between these three random effects: $\text{cov}(A_i, B_i) = \sigma_{AB}$ for all i; $\text{cov}(A_i, C_i) = \sigma_{AC}$ for all i; and $\text{cov}(B_i, C_i) = \sigma_{BC}$ for all i. This represents an actor being part of multiple dyads and triads, but also an actor adopting each role in a triad.

The dyad-specific parameter μ is extended in comparison with the p_2 model. The μ parameter can be seen as the overall mean or density of the network and thus primarily used as the intercept of the model. We assume that this parameter can contain dyad-specific covariates which represent characteristics (Z) of the actors as a test for homophily as it did in the p_2 model. However, here we compare all three actors (sender, receiver and perceiver) for homophily. δ_1, δ_2 and δ_3 are the parameters for the three possible comparisons for homophily between actor i, j and k.

$$\mu_{ijk} = \mu + Z_{1ij}\delta_1 + Z_{2ik}\delta_2 + Z_{3jk}\delta_3 \tag{18}$$

As in the p_2 model, for this thesis we will only estimate the reciprocity parameter ρ without dyad-specific characteristics. However, for future research it could be a possibility. So in this thesis ρ is estimated as this:

$$\rho_{ijk} = \rho \tag{19}$$

4 Estimation

The parameters that need to be estimated for the triadic model are

$$\Theta = (\gamma_1, \gamma_2, \gamma_3, \mu, \delta_1, \delta_2, \delta_3, \rho, \sigma_A^2, \sigma_B^2, \sigma_C^2, \sigma_{AB}, \sigma_{AC}, \sigma_{BC}) \tag{20}$$

Because we are using the p_2 model as the basis for our model and are extending it with the perceiver random effect, we can also use the same estimation method. Because this model is a generalized linear mixed model (GLMM), it is not possible to find a closed-form solution of the likelihood. With estimation methods and calculation-ability of computers improving, estimations of parameters become more precise. Originally, iterative generalized least squares was used for the p_2 model (Van Duijn et al, 2004) and later Bayesian estimation proved to be more accurate (Zijlstra et al, 2009). However, Bellio and Soriani (2020) estimated the p_2 model with maximum likelihood estimation (MLE) based on Laplace approximation and concluded that it performs well. Since the Laplace approximation performs well and is easier to understand, better interpretable and faster to execute compared to the other estimation methods, we will use this approximation for the triadic model.

4.1 Likelihood function

In (10) we formulated the likelihood function for the p_2 model which uses two dimensions for only the sender and the receiver. We integrate out the random effects for the p_2 model (Bellio & Soriani, 2020):

$$L(\theta) = \int_{R^2} p(y|u;\theta) \left\{ \prod_{i=1}^{n} \phi_2(u_i; 0, \Sigma) \right\} du \tag{21}$$

With $u_i = (A_i, B_i)$ and ϕ_2 following a bivariate normal density.

Because of the N layers of perceived networks in the triadic model, we need to add another dimension to estimate the likelihood of the triadic model. To do this we need to add another multiplication to (10) to capture and estimate the perceiver random effect and extra variance parameters. The likelihood function that follows is

$$p(y|u;\theta) = \prod_{k=1}^{n} \prod_{\substack{i=1 \\ i \neq k}}^{n-1} \prod_{j=i+1}^{n} p(y_{ijk}, y_{jik} | A_i, B_i, A_j, B_j, C_k) \tag{22}$$

With $p(y_{ijk}, y_{jik}|A_i, B_i, A_j, B_j, C_k)$ corresponding to (12)

Because of the addition of the third multiplication in (20), we need to integrate out more random effects than in the p_2 model to obtain Σ in (17). We can still follow the logic of the p_2 model, so it follows that

$$L(\theta) = \int_{R^3} p(y|u;\theta)\left\{\prod_{i=1}^{n} \phi_3(u_i; 0, \Sigma)\right\} du \tag{23}$$

With $u_i = (A_i, B_i, C_i)$ and ϕ_3 following a trivariate normal density.

4.2 Laplace approximation

Laplace approximation is a method that can be used in situations where estimation is harder because of the complexity of the model and the data. It uses Taylor series to approximate the integral of a function (Shun & McCullagh, 1995). In this way it is possible to assume a normal distribution of fixed and random parameters using at least first order derivatives. Making it possible to use maximum likelihood estimation (MLE) for this model because of this approximation method and to find convergence for the parameters estimated. It is a fast method to approximate the estimates, especially in comparison with an approximation method such as Monte Carlo simulations which can be computational expensive. The parameter estimation of Laplace approximation is also accurate in comparison with other methods (Bellio & Soriani, 2020; Azevedo-Filho & Shachter, 1994).

5 Application

5.1 Implementation software

To apply the triadic model to real data, the model is programmed in C++ and R. Bellio and Soriani (2020) shared their code on Github (Bellio & Soriani 2019a) of the application for the p_2 model with Laplace approximation. It includes C++ code and code in R (to create a function to estimate the model and for their R package). Because our model is an expansion of the p_2 model with the same estimation method as Bellio and Soriani (2020), we used this code as a base to program the triadic model.

The R package Template Model Builder (TMB) is used which can implement complex random effect models using Laplace approximation to estimate the negative log likelihood (Kristensen et al, 2025). To use this in R, a separate C++ file is programmed where the exact model is formulated to use for estimation. This includes the declaration of all parameters, the data, the calculation of the model formula in (12) of the triadic model as well as the calculation of the covariance matrix Σ in (17). The C++ is compiled and used in R within the TMB package and the TMB function `MakeAdFun`.

Together with an optimization function in R (we used the optimization function `nlminb`), the TMB package and the C++ file, the parameters are estimated. This R code and the C++ code can be found in Appendix D and G.

5.1.1 C++ code

In the C++ code the triadic model is defined. It includes parameters, the data format, the calculation of the covariance matrix and the calculation of the fixed effects. We concluded that it is possible to adjust the code of Bellio and Soriani (2019a) to implement the changes found in the triadic model compared to the p_2 model.

We made three changes to the data format. First, we include actor characteristics for the perceiver (a separate dataframe). Secondly, the dimensions for the triadic data were changed, because it has a dimension more than in the p_2 model. The last change is that we added a dimension for reciprocity (see Appendix D). The data format for sender and receiver characteristics and the density (homophily) effects were not changed from the p_2 model. The actor characteristics are dataframes and the input for density effects is a three dimensional array.

The parameters that we added to the model found in (20) are also included in the code (see Appendix D). γ_3 and c are added for the addition of the perceiver, δ_2 (dyadic comparison between actor i and actor k) and δ_3 (dyadic comparison between actor j and

actor k) are also added for the addition for the calculation of μ_{ijk}. Because of this δ_4 becomes the parameter that is estimated for ρ. The parameter *alpha* in the parameter section contains the variances, and consists of 6 elements that we use to calculate the covariance matrix.

A vital part of the code that is changed in the C++ file are the calculations of the correct values for the covariances matrix Σ for the triadic model, which is a 3 x 3 matrix. For the estimation, we need Σ^{-1}. Bellio and Soriani (2020) used the Cholesky decomposition to calculate the inverse of the covariance matrix. We also use the Cholesky decomposition to calculate Σ^{-1} of the triadic model. The calculations are found in Appendix A and B.

For the estimation of the fixed parameters, we changed the code to capture the third dimension of each layer of each perceiver. We also adjusted some indices so the correct values are taken into account to calculate (12). We added the parameter η (perceiver) for calculating the estimated negative log likelihood of the model. In Appendix C and Appendix D we added the C++ code for the p_2 model and our code for the triadic model respectively so it is more clear what we changed.

5.1.2 R code

In the R code we also made changes. In this phase we chose to focus on step by step code and not a function or package. The only package we used for estimation is `TMB`. The first step in R after loading this package is compiling the C++ code. After compiling we can load the compiled file into R.

Before running TMB, we added the random and fixed parameters that will be estimated, as well as the input of the function and other changes were made to obtain the correct output. These are the same parameters added in the C++ file. We made a separate dataframe for the actor characteristics of the perceiver and we adjusted the input of reciprocity to a four dimensional array. For the density (homophily), matrices were created for the differences in actor characteristics and put in one array with the first array containing 1's. For example we can measure differences between actors to create a binary matrix by comparing all combinations of two actors whether they have the same sex (1) or not (0).

After creating the correct data input, the compiled C++ file is loaded into R. The TMB function `MakeAdFun` will return a list with functions to calculate the objective function and its gradient (partial derivatives). This objective function and its gradient are put in the optimization function `nlminb` to estimate the parameters. The starting values for the parameters come from the output of this objective function. We set the

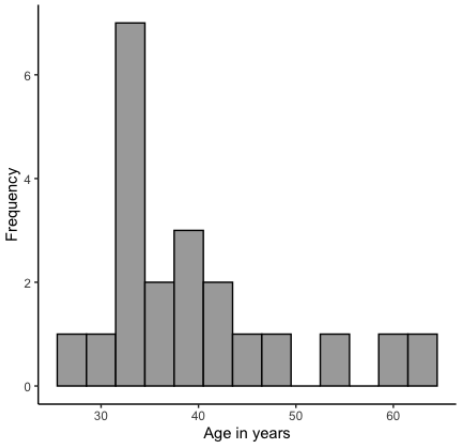
lower bounds at $-\infty$ for the fixed parameters and the variances and at 0 for the random covariances. After estimation of the parameters, the function `sdreport` calculates the standard errors of all the parameters with the optimized parameters as input (Kristensen et al, 2025).

We will compare the triadic model with the p_2 model. For the results of the p_2 model the package `p2model` made by Bellio and Soriani (2019a) is used with the packages `remotes` and `NetData`.

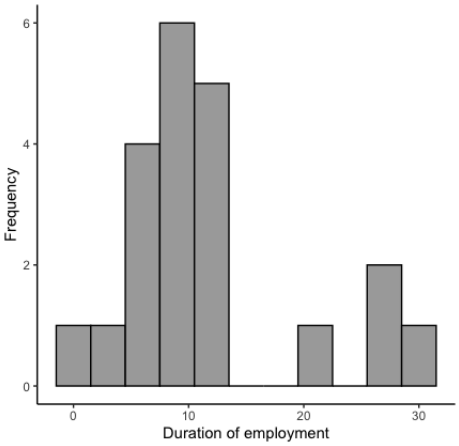
5.2 Description data

We will use the high-tech managers data of Krackhardt (1987). The data describes the friendship relations of 21 managers (actors) in an organization. Every manager in this network was asked to look at each combination of the actors and was asked to indictate whether they had a friendship or not. Because of this, our data consists of 21 complete social networks for each of the 21 managers. For each manager we also have information about their characteristics **age** (in years), department of the manager in the organization (**department**), duration of employment in the organization (**tenure**) and level of seniority in the organization hierarchy (**level**) where 1 is the highest level and 3 the lowest level of seniority. The triadic data can be found in the package `cssTools` under 'highTechManagers'.

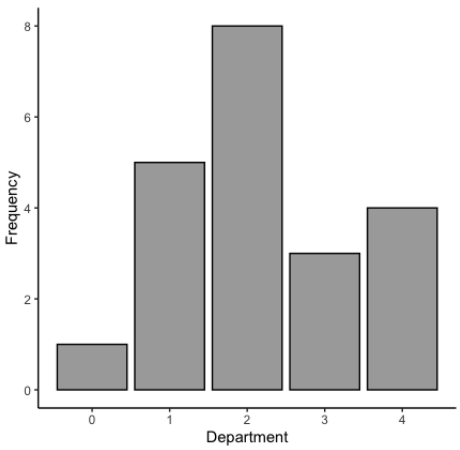
We will use all the layers of the triadic data collected by Krackhardt (1987). We will use the age and tenure as actor characteristics for the sender, receiver and the perceiver. The mean of the age of the 21 managers is 39.71 with standard deviation 9.56 with a minimum age of 27 and maximum age of 62. The average duration of employment is 11.74 years with standard deviation 8.04. The correlation between age and tenure is $r = .489$. To measure potential homophily in this social network, we will add three matrix variables to our model which indicate; the absolute differences in age between the sender, receiver and perceiver; whether each actor in the network works in the same department (1) or not (0); and whether they have the same level of seniority (1) or not (0). In Figure 5c we visualized the distribution of managers over each department. In Figure 5d we visualized the distribution of seniority of the 21 actor, which is not evenly distributed.



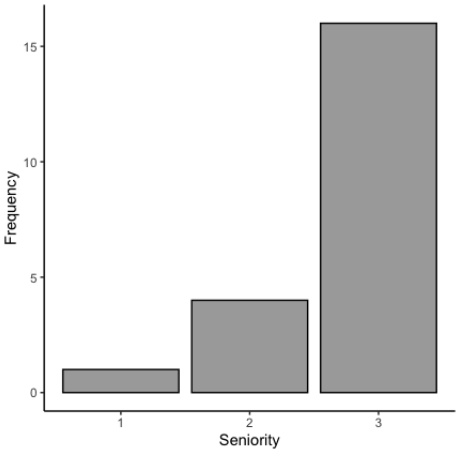
(a) Age



(b) Tenure



(c) Department



(d) Seniority

Figure 5. Distribution of age, tenure, department and seniority

Bellio and Soriani (2020) also used this high-tech managers data, but they used the aggregated data of all the layers (two-dimensional dataset) for the p_2 model. We use this data as example for self-reported data, so it is possible to compare the p_2 model with the triadic model. For the comparison with the p_2 model we will follow the vignette made by Bellio and Soriani (2019b). It uses the same actor characteristics for the sender and receiver and the same homophily variables. The attributes of these data (for both models) can be found in the package `NetData` under 'kracknets'. Also the aggregated data is stored here.

We want to describe the self-reported social network data and two of the layers from the triadic data. We randomly selected the perceived social network of actor 2 and actor 19. First we will take a look at the adjacency matrices. In Table 2 we see the first five rows of the adjacency matrix of the aggregated data which represents the self-reported data. We can see for example that actor 1 sends ties to actor 2, 4, 8, 12 and 16. Actor 21 receives (of the first five rows displayed) ties from actor 2 and 5. And we can also see reciprocal ties between actor 1 and 2.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
1	-	1	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	0	0
2	1	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1
3	0	0	-	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0
4	1	1	0	-	0	0	0	1	0	0	0	1	0	0	0	1	1	0	0	0	0
5	0	1	0	0	-	0	0	0	1	0	1	0	0	1	0	0	1	0	1	0	1

Table 2. Adjacency matrix for self-reported data (first five rows)

In Table 3 we see the adjacency matrix perceived by actor 2. Actor 1 still sends a tie to actor 2, 12 and 16 according to the perceiver. However not to actor 4 and actor 8 compared to the self-reported data. We can still see the reciprocal ties between actor 1 and actor 2. According to the perception of actor 2, actor 21 only receives a tie from actor 2 of these five rows displayed.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
1	-	1	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0
2	1	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1
3	0	0	-	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
4	0	0	0	-	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0
5	0	1	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

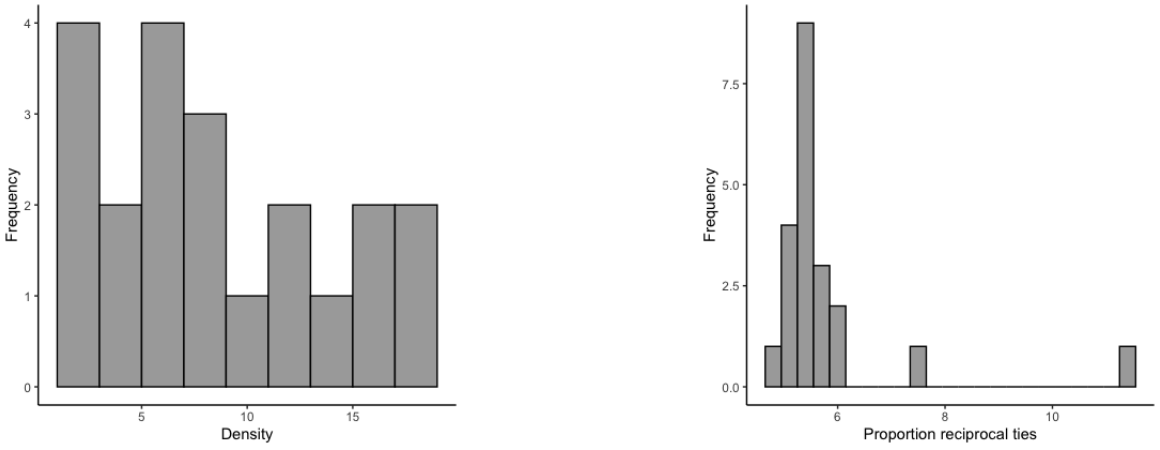
Table 3. Adjacency matrix perceived by actor 2 (first five rows)

The adjacency matrix in Table 4 is perceived by actor 19. In comparison with Table 2 (the self-reported data) and with the social network perceived by actor 2 in Table 3, actor 19 perceives actor 1 sending ties to actor 2, 4 and 16. But not to actor 8 and actor 12. According to actor 19 there is no reciprocal relationship between actor 1 and actor 2. And actor 21 receives only a tie from actor 2 of the five actors displayed in these tables.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
1	-	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
2	0	-	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1
3	0	0	-	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	1	0	0
4	0	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	-	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0

Table 4. Adjacency matrix perceived by actor 19 (first five rows)

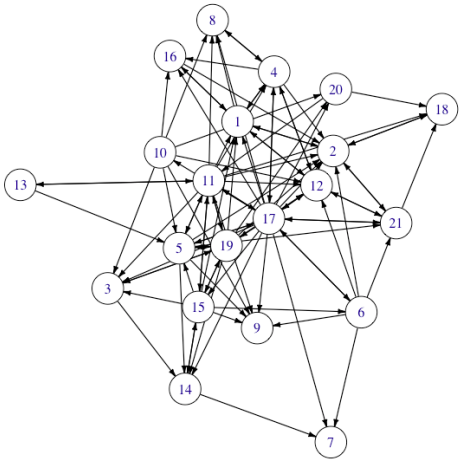
The characteristics density (the proportion of existing ties of the potential ties in the social network) and reciprocity (the proportion of reciprocal dyads of all dyads possible in this social network) give us another description of the social network data. We calculated the density and proportion of reciprocal dyads for each perceived network of the triadic data and the self-reported data. The density of the self-reported data is 24.29%. In Figure 6a we can see that the distribution of proportion of number of ties perceived is skewed. With the density being less than 10% for just more than half of the 21 perceived networks. The proportion of reciprocal dyads for the self-reported data is 10.95%. In Figure 6b we can see that the proportion of reciprocal ties for a lot of layers is below 6%. With an outlier towards the 12%.



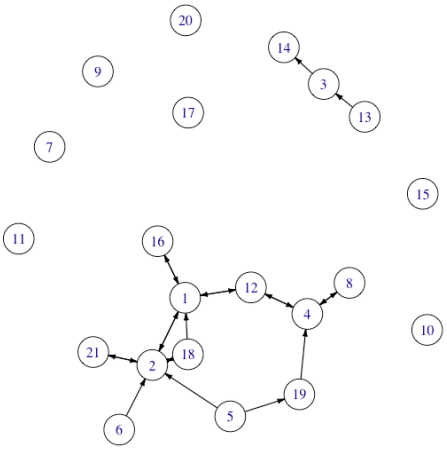
(a) Density **(b)** Reciprocal ties

Figure 6. Distribution of density and proportion of reciprocal ties of 21 layers

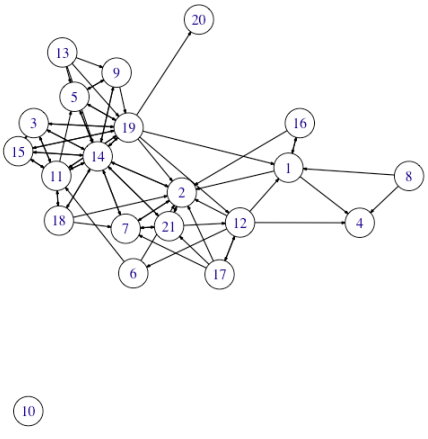
The sociograms of the three social networks can give us more information about the structure of our social networks (the self-reported data, the layer perceived by actor 2 and by actor 19). They are visualized in Figure 7. We visualized all actors in the social network. We can see that in Figure 7a all actors are connected. For example, we can see that actor 3 receives five ties and sends 1 tie to actor 14. Between actor 4 and actor 8 we can see an example of reciprocal ties. In Figure 7b the social network is visualized as perceived by actor 2. We can see that less ties are perceived and there are a lot of isolated actors. These are actors without ties. According to perceiver 2 there is a clique of actors in the network. When looking at the social network perceived by actor 19 (Figure 7c), only actor 10 is isolated from the rest of the actors. Actor 1 receives ties from actor 2, 12, 16 and 18 according to perceiver 19.



(a) Self-reported data



(b) Perceiver 2



(c) Perceiver 19

Figure 7. Sociograms of the self-reported data and perceived by actor 2 and 19

5.3 Results

During the process of programming the triadic extension we discovered some problems. First we programmed the model as defined in Chapter 3. We were able to estimate the parameter values, however we could not estimate the standard errors of the parameter values. There could be multiple explanations to this problem. One of them is multicollinearity which could be due to the density parameter that measures homophily. We decided to try to only include the homophily between actor i and actor j, and only estimate δ_1 for μ_{ijk} which also includes the intercept μ. After this modification, it was possible for R to estimate the standard errors. This version of the C++ code can be found in Appendix E.

We will take this version to show the results and compare where possible to the results of the p_2 model. This indicates that we will compare self-reported dyads with dyads perceived by a third actor in the triad. In the discussion we will further explore possible solutions to estimate the full model described by us in Chapter 3. We tested the parameters in the model with the Z-score and a two-tailed test. The results can be found in Table 5. Rstudio version 2024.12.0+467 is used with R version 4.4.2 (R Core Team, 2024).

First we take a look at the individual level effects. These are the sender, receiver and perceiver effects (Table 5). We found a negative effect of age on sending ties in the p_2 model ($\gamma_1 = -.131(.059)$, p = .027). The older the actor, the less friendship ties they will send. This effect does not exist in the triadic model ($\gamma_1 = -.001(.019)$, p = .954). The effect of duration of employment (tenure) on sending ties is positive in the p_2 model ($\gamma_1 = .140(.062)$, p = .024). Meaning that the longer an actor is employed, the more ties they send. However, in the triadic model this effect perceived by a third actor is not present ($\gamma_1 = .037(.022)$, p = .095). If we look at the receiver effects, the effect of age on receiving ties is not significant in the p_2 model as well as in the triadic model (p_2: $\gamma_2 = -.002(.035)$, p = .952; triadic: $\gamma_2 = -.0003(.0369)$, p = .993). The effect of duration of employment on receiving ties increases somewhat. A small positive effect (p_2: $\gamma_2 = .045(.041)$, p = .277; triadic: $\gamma_2 = .076(.044)$, p = .083) indicates the longer someone works at the organization, the more ties they receive. For the triadic model this means that this indication is perceived by other actors. However it is not significant in both models. The added perceiver effects show some small effects. Age of the perceiver has a small negative effect ($\gamma_3 = -.017(.021)$, p = .437) on perceiving ties. The effect of duration of employment of the perceiver on perceiving ties is small and

positive ($\gamma_3 = .022(.025)$, p = .375). If the effects were larger, it could indicate that the older the perceiver is, the less likely they will perceive ties between other actors and the longer the perceiver is employed at the organization, the more likely they will perceive ties.

Results p_2 and Triadic Model					
		p_2 Model		**Triadic Model**	
Effect		*Estimate*	*S.E.*	*Estimate*	*S.E.*
Sender	Age	-.131*	.059	-.001	.019
	Tenure	.140*	.062	.037	.022
Receiver	Age	-.002	.035	-.0003	.037
	Tenure	.045	.041	.076	.044
Perceiver	Age	-	-	-.017	.021
	Tenure	-	-	.022	.025
Reciprocity	ρ	2.117**	.633	2.783***	.186
Density	μ	.039	1.735	-5.719*	2.271
	Department (same = 1)	1.576***	.348	1.723***	.111
	Seniority (same = 1)	1.154**	.405	1.101***	.117
	Age diff.	-.055*	.025	-.035***	.007
(Co)variances					
Sender	σ_A^2	2.054	.950	.413	.173
Receiver	σ_B^2	.998	.565	1.789	.685
Perceiver	σ_C^2	-	-	.556	.207
Sender-Receiver	σ_{AB}	-1.136	.637	.595	.278
Sender-Perceiver	σ_{AC}	-	-	.149	.129
Receiver-Perceiver	σ_{BC}	-	-	.531	.275

Table 5. Estimates and standard errors p_2 model and triadic model;
*p = .05, **p < .01, ***p < .001

We will now look at the dyad-specific level effects of ρ and μ. The effect of reciprocity is high in both the p_2 model ($\rho = 2.117(.633)$, p = .001) and the triadic model ($\rho = 2.783(.186)$, $p < .001$), indicating that there is reciprocity in the social network attributing to the probability of a tie existing between two actors. However, the standard error in the triadic model is lower than in the p_2 model. This is interesting, because it makes the

triadic model more precise. As mentioned in this thesis, a goal of using the perceived layers in the triadic data is improving the precision of self-reported relations. We have more data, so a larger "sample" for making it more precise. The parameter μ for the density, is low in the triadic model ($\mu = -5.719(2.271)$, p = .012) and also in the p_2 model ($\mu = .039(1.735)$, p = .982). This is a common phenomenon when dealing with larger networks. Because the triadic data exist of multiple layers of social networks, a lot more dyadic data is available as in larger networks. It also depends on the number of dyadic covariates added to μ.

The first effect for homophily is being part of the same department. The probability of having a friendship relation increases when both actors are in the same department in comparison with not being in the same department. This effect is present in the p_2 model ($\delta_1 = 1.576(.348)$, p < .001) and we can see a similar effect in the triadic model ($\delta_1 = 1.723(.111)$, p < .001). So the increase in probability of having friendship relations in the same department is also perceived by the third actor in the triad. The probability of a friendship existing is also higher when the two actors have the same level of seniority in comparison with not having the same level of seniority. This is also present when perceived by a third actor. The effect in the p_2 model ($\delta_1 = 1.154(.405)$, p = .004) is also similar to the effect in the triadic model ($\delta_1 = 1.101(.117)$, p < .001). Here the standard error is also lower for the triadic model than in the p_2 model. When we look at differences in ages, we can see a small negative effect in both the p_2 model ($\delta_1 = -.055(.25)$, p = .026) and the triadic model ($\delta_1 = -.035(.007)$, p < .001). The probability of a friendship between two actors decreases when differences in age increases. This effects seems to be especially perceived by a third actor. It is a small effect and also here the standard error is lower for the triadic model.

We will now look at the variances in sender (σ_A^2), receiver (σ_B^2) and perceiver (σ_C^2) activity. For the sender (variation between actors how many ties they send) the variance is higher in the p_2 model ($\sigma_A^2 = 2.054(.950)$) than in the triadic model ($\sigma_A^2 = .413(.173)$). This can be explained by the fact that in the p_2 model the sender is also the perceiver of the self-reported ties. For the receiver (variation between actors how many ties they receive) the variance is in the triadic model ($\sigma_B^2 = 1.789(.685)$) similar as in the p_2 model ($\sigma_B^2 = .998(.565)$). The higher standard error of the triadic model could be explained by the distance of the perceiver to certain actors in the social network and being more informed about actors close by such as friends of friends. Maybe perceiving received ties at more extreme values (having no ties or more ties). For the perceiver (variation between layers how many ties are perceived) the variation is only calculated in the triadic model

and is $\sigma_C^2 = .556(.207)$.

We are also interested in the covariances between sender, receiver and perceiver. The covariance between sender and receiver effect of the same actor (σ_{AB}) is negative in the p_2 model ($\sigma_{AB} = -1.136(.637)$). Meaning the more an actor sends ties, the less they receive ties. However, in the triadic model the covariance is positive (($\sigma_{AB} = .595(.278)$). Meaning the more sent ties are perceived, the more received ties for the same actor are perceived. These two opposite results can be explained by how the data is collected. The negative effect in the self-reported data could be explained by how every actor only needs to indicate who they see as their friends and not whether the friendship is reciprocal. Making it possible that not all those friends will nominate them as friends. In this case many sent ties makes an actor more sociable (and thus not immediately seen as more popular), many received ties represents an actor as more popular. The perceiver however looks at every directed tie in the network. It is possible that the number of received and sent ties by the same actor become almost equal according to the perceiver. The perceiver subconsciously does not differ a more social actor from a more popular actor. It could be possible that the perceiver judges an actor as also sending more ties when receiving more ties and receiving less ties when sending less ties. This also lies in the basis of CSS where a perceiver adjusts their own relation on the basis of what they perceive. The covariance between sender and perceiver (σ_{AC}) is only estimated in the triadic model and is positive but small ($\sigma_{AC} = .149(.129)$). Meaning the more send ties that are perceived by an actor, the more ties are perceived in general. The covariance between receiver and perceiver σ_{BC} is also only estimated in the triadic model and is positive ($\sigma_{BC} = .531(.275)$). The more received ties perceived by an actor, the more ties are perceived in general.

6 Discussion

In this thesis we have built a triadic extension of the p_2 model using Laplace approximation based on Bellio and Soriani (2020). Instead of using self-reported data we used triadic data where every actor in the social network perceives every directed tie in the network. It was possible to program the triadic model, however only the version without the homophily comparisons between actor i and actor k, and actor j and actor k would converge. This version was evaluated in the results and compared to the p_2 model. The results show that the triadic model works. In comparison with the results of the p_2 model some results were more precise in the triadic model. Especially for the parameter ρ and the homophily effects the standard errors are smaller in the triadic model compared to the p2 model. This can be explained by the fact we have more information and network data than the self-reported data. The variance of the sender was higher in the p_2 model than in the triadic model. The variance for receiver was similar for both models, but the standard error was higher for the triadic model. The covariance between sender and receiver was negative for the p_2 model and positive for the triadic model. Which could possibly be explained by how the data is collected.

The addition of the perceiver effect was not significant in our example. However, it is interesting we can now measure with this effect to what extent an actor perceives other ties different from theirs dependent on certain characteristics of the perceiver. The variance of the perceiver now gives information about the variation between the N layers how many ties are perceived. The covariance between sender and perceiver is positive in our example and the covariance between receiver and perceiver is also positive. This indicates that for the same actor, if they send or receive more ties, they are also more likely to perceive more ties in their role as perceiver.

We can link these results to our theory. In the cognitive social structures of Krackhardt (1987) it is described that how an actor perceives the relationships of themselves and their friends, influences how this perceiving actor is adjusting their relationships accordingly. Where the balance theory of Heider (1958) can play a role in this perception. It does not matter whether the perception is correct, the third perceiving actor believes it exist and acts accordingly. Each actor in the triadic data gives us this judgement about whether they perceive relationships or not between all the actors. And we are able to measure certain actor characteristics which can effect the perception of these relations existing or not. We can observe variance of the number of perceived relations between all the actors (the N social network layers in the triadic data), which indicates that perception of the relationships within a social network is different among actors. Whether it is a correctly

identified relationship or not. This model gives us more information about the role of the third actor k perceiving the relationship between actor i and j and gives us more information about the effect of the perceiver, the third actor, on the dyadic probability between the two other actors in the triad.

We also concluded that in our example in some cases the standard errors were lower in the triadic model compared to the p_2 model and the estimated parameter value was similar to the value in the p_2 model. Krackhardt (1987) stated that these triadic data could be also used to investigate the precision and consensus of self-reported data. The lower standard errors indicate more precise results. However, we cannot fully conclude only with the example in this thesis that the triadic model gives more precise results of these effects. Future research could look into this through for example simulation studies. The variance of the perceiver can be used to investigate the consensus of self-reported data.

The biggest problem in this thesis was the multicollinearity found when programming the homophily parameters δ_1, δ_2 and δ_3. For the comparisons between actor i and j, actor i and actor k, and actor j and actor k. We were able to include the comparison between actor i and j and the model was able to converge, but not with the other pairs. Correlations between variables we added could produce this problem. As we showed, the correlation between tenure and age was already higher. But we also ran into the overlap where all the N actors fill in the role of the sender, receiver and perceiver. If we compare similarities in actor characteristics between actor i and j, there is overlap (or correlation) when comparing actor i and actor k and after that actor j and actor k. A possible solution in future research is looking into a restriction on which variables and characteristics can be added and which characteristics can be compared to measure homophily between actors especially between the three pairs. In theory, we want to include all three comparisons in μ, because all three pairings are important and can give us more information. However in practice, the correlations between variables and the overlap in use of data; every actors being part of each role; and the sparse network data ask for a thorough investigation where computational problems lie and which combination of variables can be used. Also other more common solutions that can solve multicollinearity can be investigated whether they work for this triadic model.

In this thesis we made the assumption that adding the perceiver parameter η in combination with the triadic data is sufficient to argue that we can model triads in a realistic way. Also because every actor will take the role of sender, receiver and perceiver and consequently ties together the three dyads in a triad. We showed that we now

can model the effect of actor characteristics of the perceiver on the dyadic probability. However, one can argue that maybe this assumption is too bold. We can now estimate the probability of each dyad in the social network and separately all three dyads in a triad. But maybe a further step needs to be taking in future research to tie those three dyads together. A possible way could be to add a triadic effect to the model besides the perceiver effect that we added.

For the comparison between the estimates of the p_2 model with the triadic model we used the aggregated data as self-reported data. This is not the same as the usually self-reported data where the actor is asked who their friends are from the predetermined social group. Because of this, the density and proportion of reciprocity can be somewhat higher than normal. However, we used it so we could compare the two models and it still gives a good inside in what the triadic data and model brings. It brings something to consider what happens subconsciously when actors self-report their relations or when they perceive the whole network and the effect of this on evaluating their own relationships. Other recommendations are adding dyad-specific characteristics to ρ, calculating the AIC and the BIC for the triadic model and evaluate their use, and looking at other assumptions that need to be adhered to for a good model fit.

Next, we also want to measure how well the triadic model fits on the triadic data. This can be the triadic high-tech managers dataset, but also other datasets providing triadic social network data where every actor reports every directed tie in the social network. For this we can use goodness-of-fit tests. As Bellio and Soriani (2020 suppl.) did for the p_2 model with Laplace approximation, we also can simulate networks from the fitted model and calculate certain descriptive structures from these models. Examples for these structures are the proportion of triads (triangles), the indegree or outdegree and compare it to the observed social network. However, it needs to be investigated in further research which structures can be used for goodness-of-fit tests for the triadic model.

We described in this thesis an extension of the p_2 model using triadic data. To model the dyadic probabilities including the perception of the third actor in a triad on the relationships of others. It was possible to converge a large part of the model, because we encountered the problem of multicollinearity when programming the whole model. The results provide more information about the role and characteristics of the perceiver on estimating the probability of a dyad. More research needs to be done on the implications of using certain actor characteristics on convergence of the whole model. Recommendations for future research were made to ensure that researchers can eventually use this model to answer research questions concerning cognitive social structures and balance theory.

7 References

[1] Amati, V., Lomi, A., & Mira, A. (2018). Social network modeling. *Annual Review of Statistics and Its Application*, 5(1), 343-369. https://doi.org/10.1146/annurev-statistics-031017-100746.

[2] Azevedo-Filho, A., & Shachter, R. D. (1994, January). Laplace's method approximations for probabilistic inference in belief networks with continuous variables. *Uncertainty in Artificial Intelligence*, 28-36. https://doi.org/10.1016/B978-1-55860-332-5.50009-2.

[3] Bellio, R. & Soriani, N. (2019a). Software for [Maximum likelihood estimation based on the Laplace approximation for p2 network regression models]. GitHub, https://github.com/rugbel/p2model.

[4] Bellio, R. & Soriani, N. (2019b). Vignettes for [Maximum likelihood estimation based on the Laplace approximation for p2 network regression models]. GitHub, https://github.com/rugbel/p2model/tree/master/vignettes.

[5] Bellio, R. & Soriani, N. (2020). Maximum likelihood estimation based on the Laplace approximation for p2 network regression models, *Statistica Neerlandica*, 75(1), 24-41. https://doi.org/10.1111/stan.12223

[6] Bond Jr, C. F., Horn, E. M., & Kenny, D. A. (1997). A model for triadic relations. *Psychological Methods*, 2(1), 79–94. https://doi.org/10.1037/1082-989X.2.1.79

[7] Card, N. A., Rodkin, P. C., & Garandeau, C. F. (2010). A description and illustration of the Triadic Relations Model: Who perceives whom as bullying whom?. *International Journal of Behavioral Development*, 34(4), 374-383. https://doi.org/10.1177/0165025410371418

[8] Heider, F. (1958). The psychology of interpersonal relations. Lawrence Erlbaum Associates.

[9] Holland, P. W., & Leinhardt, S. (1981). An exponential family of probability distributions for directed graphs. *Journal of the American Statistical association*, 76(373), 33-50. https://doi.org/10.1080/01621459.1981.10477598

[10] Kadushin, C. (2012). *Understanding social networks: Theories, concepts, and findings*. Oxford university press.

[11] Kenny, D. A. & La Voie, L. (1984). The social relations model. *Advances in Experimental Social Psychology*, 18, 141-182. https://doi.org/10.1016/S0065-2601(08)60144-6

[12] Krackhardt, D. (1987). Cognitive social structures. *Social networks*, 9(2), 109-134. https://doi.org/10.1016/0378-8733(87)90009-8

[13] Kristensen, K., Bell, B., Skaug, H., Magnusson, A., Berg, C., Nielsen, A., Maechler, M., Michelot, T., Brooks, M., Forrence, A., Albertsen, C.M., Monnahan, C. (2025). TMB: Template Model Builder: A General Random Effect Tool Inspired by 'ADMB', https://CRAN.R-project.org/package=TMB.

[14] R Core Team (2024). R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria. https://www.R-project.org/.

[15] 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. https://doi.org/10.1016/j.socnet.2006.08.002

[16] Robins, G. & Lusher, D. (2013). What are exponential random graph models. In Lusher, D., Koskinen, J.,& Robins, G. (Eds.).*Exponential random graph models for social networks: Theory, methods, and applications* (9-14). Cambridge University Press.

[17] Shun, Z., & McCullagh, P. (1995). Laplace approximation of high dimensional integrals. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 57(4), 749-760. https://doi.org/10.1111/j.2517-6161.1995.tb02060.x

[18] Snijders, T. A., & Kenny, D. A. (1999). The social relations model for family data: A multilevel approach. *Personal Relationships*, 6(4), 471-486. https://doi.org/10.1111/j.1475-6811.1999.tb00204.x

[19] Snijders, T. A. (2011). Statistical models for social networks. *Annual review of sociology*, 37(1), 131-153. https://doi.org/10.1146/annurev.soc.012809.102709

[20] Stark, T. H. (2017). Collecting social network data. *The Palgrave handbook of survey research*, 241-254.

[21] Swartz, T. B., Gill, P. S., & Muthukumarana, S. (2015). A Bayesian approach for the analysis of triadic data in cognitive social structures. *Journal of the Royal Statistical Society Series C: Applied Statistics*, 64(4), 593-610. https://doi.org/10.1111/rssc.12096

[22] Van Duijn, M. A., Snijders, T. A., & Zijlstra, B. J. (2004). p2: A random effects model with covariates for directed graphs. *Statistica Neerlandica*, 58(2), 234-254. https://doi.org/10.1046/j.0039-0402.2003.00258.x

[23] Van Duijn, M. A. (2013). Multilevel modeling of social network and relational data. *The SAGE handbook of multilevel modeling*, 599-618.

[24] Zijlstra, B. J., Van Duijn, M. A., & Snijders, T. A. (2009). MCMC estimation for the p2 network regression model with crossed random effects. *British Journal of Mathematical and Statistical Psychology*, 62(1), 143-166. https://doi.org/10.1348/000711007X255336

8 Appendix

A: Cholesky decomposition

We are looking for the Cholesky decomposition of the covariance matrix for the triadic model. 6 elements (three variances, three covariances) need to be calculated with the use of α which needs to be estimated.

$$\Sigma = \begin{bmatrix} \sigma_A^2 & \sigma_{AB} & \sigma_{AC} \\ \sigma_{AB} & \sigma_B^2 & \sigma_{BC} \\ \sigma_{AC} & \sigma_{BC} & \sigma_C^2 \end{bmatrix}$$

In our C++ code we use the vector S() and alpha() consisting of 6 elements. C++ starts counting at 0. These are the places in the vectors linked to the (co)variances.

$$0 = \sigma_A^2, 1 = \sigma_{AB}, 2 = \sigma_B^2, 3 = \sigma_{AC}, 4 = \sigma_{BC}, 5 = \sigma_C^2$$

$$\begin{bmatrix} 0 & \sigma_{AB} & \sigma_{AC} \\ 1 & 2 & \sigma_{BC} \\ 3 & 4 & 5 \end{bmatrix}$$

The calculation for the Cholesky decomposition is:

$$A = LL^T$$

Where L is the lower triangular matrix and L^T is its transpose. For the Cholesky decomposition we assign also letters to these places to make it easier for ourselves in assigning the correct places for the Cholesky decomposition.

$$L = \begin{bmatrix} a & 0 & 0 \\ b & c & 0 \\ d & e & f \end{bmatrix}$$

And now the Cholesky decomposition:

$$A = \begin{pmatrix} a & 0 & 0 \\ b & c & 0 \\ d & e & f \end{pmatrix} \begin{pmatrix} a & b & d \\ 0 & c & e \\ 0 & 0 & f \end{pmatrix} = \begin{pmatrix} a^2 & ab & ad \\ ab & b^2 + c^2 & bd + ce \\ ad & bd + ce & d^2 + e^2 + f^2 \end{pmatrix} = \begin{pmatrix} \sigma_A^2 & \sigma_{AB} & \sigma_{AC} \\ \sigma_{AB} & \sigma_B^2 & \sigma_{BC} \\ \sigma_{AC} & \sigma_{BC} & \sigma_C^2 \end{pmatrix}$$

So it follows that:

$$\sigma_A^2 = S(0) = alpha(0) + alpha(0)$$
$$\sigma_{AB} = S(1) = alpha(0) + alpha(1)$$
$$\sigma_B^2 = S(2) = alpha(1) * alpha(1) + alpha(2) * alpha(2)$$
$$\sigma_{AC} = S(3) = alpha(0) * alpha(3)$$
$$\sigma_{BC} = S(4) = alpha(1) * alpha(3) + alpha(2) * alpha(4)$$
$$\sigma_C^2 = S(5) = alpha(3) * alpha(3) + alpha(4) * alpha(4) + alpha(5) * alpha(5)$$

B: Calculation determinant and inverse

For our likelihood we need the inverse of the covariance matrix found with the Cholesky decomposition. For this we need the adjugated matrix and the determinant. For this we are still using the place numbers for each element in the covariance matrix to program this correctly in our C++ code. The matrix is symmetrical so transpose is the same.

$$A = \begin{bmatrix} 0 & 1 & 3 \\ 1 & 2 & 4 \\ 3 & 4 & 5 \end{bmatrix}$$

$$A^{adj} = \begin{pmatrix} \begin{vmatrix} 2 & 4 \\ 4 & 5 \end{vmatrix} & -\begin{vmatrix} 1 & 4 \\ 3 & 5 \end{vmatrix} & \begin{vmatrix} 1 & 2 \\ 3 & 4 \end{vmatrix} \\ -\begin{vmatrix} 1 & 3 \\ 4 & 5 \end{vmatrix} & \begin{vmatrix} 0 & 3 \\ 3 & 5 \end{vmatrix} & -\begin{vmatrix} 0 & 1 \\ 3 & 4 \end{vmatrix} \\ \begin{vmatrix} 1 & 3 \\ 2 & 4 \end{vmatrix} & -\begin{vmatrix} 0 & 3 \\ 1 & 4 \end{vmatrix} & \begin{vmatrix} 0 & 1 \\ 1 & 2 \end{vmatrix} \end{pmatrix}$$

All these elements in the matrix A^{adj} needs to be divided by the determinant of A. The numbers again indicated the places in vectors responding to the (co)variance. Which is:

$$det(A) = 0 + \begin{vmatrix} 2 & 4 \\ 4 & 5 \end{vmatrix} - 1 * \begin{vmatrix} 1 & 4 \\ 3 & 5 \end{vmatrix} + 3 * \begin{vmatrix} 1 & 2 \\ 3 & 4 \end{vmatrix}$$

C: C++ code p_2 model (Bellio & Soriani, 2019a)

```
#define TMB_LIB_INIT R_init_mypkg
#include <TMB.hpp>

template<class Type>
Type objective_function<Type>::operator() ()
{
  /* data section */
  DATA_MATRIX(XnS);
  DATA_MATRIX(XnR);
  DATA_ARRAY(XvD);
  DATA_ARRAY(XvC);
  DATA_MATRIX(y); //network
  DATA_INTEGER(penflag);
  DATA_VECTOR(penSigma);




  /* Parameter section */
  PARAMETER_VECTOR(gamma1);
  PARAMETER_VECTOR(gamma2);
  PARAMETER_VECTOR(delta1);
  PARAMETER_VECTOR(delta2);
  PARAMETER_VECTOR(alpha);
  PARAMETER_VECTOR(a);
  PARAMETER_VECTOR(b);

  int kd = XvD.cols();
  int kc = XvC.cols();
  int g = y.cols();


  Type nll=0.0;       // Negative log likelihood function
```

```
vector<Type> S(3);
vector<Type> S1(3);
S(0) = alpha(0) * alpha(0);
S(1) = alpha(0) * alpha(1);
S(2) = alpha(1) * alpha(1) + alpha(2) * alpha(2);
Type DS =  S(0) * S(2) - S(1) * S(1);
S1(0) = S(2) / DS;
S1(1) = - S(1) / DS;
S1(2) = S(0) / DS;
Type quadu = ((a*a).sum()) * S1(0) +
  ((a*b).sum()) * S1(1) * 2.0 +
  ((b*b).sum()) * S1(2);
nll -= -0.5 * quadu - g * 0.5 * log(DS);



ADREPORT(S);

// nll from y
vector<Type> XSg1 = XnS * gamma1; //alpha_i in the model
vector<Type> XRg2 = XnR * gamma2; //beta_i


for(int i=0;i<(g-1);i++)
  for(int j=i+1;j<g;j++)
  {
   Type y1 = y(i,j);
   Type y2 = y(j,i);
   Type alphai = XSg1(i) + a(i);
   Type alphaj = XSg1(j) + a(j);
   Type betai = XRg2(i) + b(i);
   Type betaj = XRg2(j) + b(j);
   Type muij = 0.0;
   Type muji = 0.0;
```

```
    for(int k=0;k<kd;k++)
      {
       muij += XvD(i,j,k) * delta1(k);
       muji += XvD(j,i,k) * delta1(k);
      }
    Type rhoij = 0.0;
    for(int k=0;k<kc;k++)
         rhoij += XvC(i,j,k) * delta2(k);
    Type xi1 =  muij + alphai + betaj;
    Type xi2 =  muji + alphaj + betai;
    Type xi3 =  muij + muji + alphai + betaj + alphaj + betai + rhoij;
    Type den = 1.0 + exp(xi1) + exp(xi2) + exp(xi3);
    nll -=  y1 * xi1  + y2 * xi2 + rhoij * y1 * y2 - log(den);
}

Type pen = 0.0;
Type rho = S(1) / sqrt(S(0) * S(2));

 if(penflag==1)
   pen = 0.5 * log(DS);

 if(penflag==2)
  {
    Type rhoinfo = penSigma(1) / sqrt(penSigma(0) * penSigma(2));
    pen = 0.5 * log(DS) + Type(2) * log(S(0)) + Type(2) * log(S(2))
     - (Type(2) / sqrt(penSigma(0))) * sqrt(S(0)) -  (Type(2) /
       sqrt(penSigma(2))) * sqrt(S(2))  - pow(rho-rhoinfo, Type(2))
      / 0.125;
  }

 nll -= pen;

 ADREPORT(rho);
 REPORT(pen);
```

```
  return nll;
}
```

D: C++ code complete triadic model

```
#define TMB_LIB_INIT R_init_mypkg
#include <TMB.hpp>

template<class Type>
Type objective_function<Type>::operator() ()
{
  /* data section */
  DATA_MATRIX(XnS);
  DATA_MATRIX(XnR);
  DATA_MATRIX(XnP);
  DATA_ARRAY(XvD); //still will have 3 dimensions, 21,21,N covariates+1
  DATA_ARRAY(XvC); //will have 4 dimensions, 21,21,21,1
  DATA_ARRAY(y); //network, 3 dimensions




  /* Parameter section */
  PARAMETER_VECTOR(gamma1); //sender
  PARAMETER_VECTOR(gamma2); //receiver
  PARAMETER_VECTOR(gamma3); //perceiver
  PARAMETER_VECTOR(delta1); //first parameter mu
  PARAMETER_VECTOR(delta2); //second parameter mu
  PARAMETER_VECTOR(delta3); //third parameter mu
  PARAMETER_VECTOR(delta4); //rho
  PARAMETER_VECTOR(alpha); //vector all (co)variances, in R c(1,0,1,0,0,1)
  PARAMETER_VECTOR(a);
  PARAMETER_VECTOR(b);
  PARAMETER_VECTOR(c);

  int kd = XvD.cols();
  int kc = XvC.cols();
  int g = y.cols();
```

```
Type nll=0.0;      // Negative log likelihood function

/* Cholesky decomposition and inverse covariance matrix */
vector<Type> S(6);
vector<Type> S1(6);
S(0) = alpha(0) * alpha(0);    //a
S(1) = alpha(0) * alpha(1);    //ab
S(2) = alpha(1) * alpha(1) + alpha(2) * alpha(2); //b
S(3) = alpha(0) * alpha(3);    //ac
S(4) = alpha(1) * alpha(3) + alpha(2) * alpha(4); //bc
S(5) = alpha(3) * alpha(3) + alpha(4) * alpha(4) +
  alpha(5) * alpha(5); //c
Type DS =  S(0) * (S(2) * S(5) - S(4) * S(4)) - S(1) *
  (S(1) * S(5) - S(3) * S(4)) + S(3) * (S(1) * S(4) - S(2) * S(3));
S1(0) = (S(2) * S(5) - S(4) * S(4)) / DS;
S1(1) = - (S(1) * S(5) - S(4) * S(3)) / DS;
S1(2) = (S(0) * S(5) - S(3) * S(3)) / DS;
S1(3) = (S(1) * S(4) - S(2) * S(3)) / DS;
S1(4) = - (S(0) * S(4) - S(1) * S(3)) / DS;
S1(5) = (S(0) * S(2) - S(1) * S(1)) / DS;
Type quadu = ((a*a).sum()) * S1(0) + ((a*b).sum()) * S1(1) * 2.0 +
  ((b*b).sum()) * S1(2) + ((a*c).sum()) * S1(3) * 2.0 + ((b*c).sum()) *
  S1(4) * 2.0 + ((c*c).sum()) * S1(5);
nll -= -0.5 * quadu - g * 0.5 * log(DS);


ADREPORT(S);

/* Defining fixed effects triadic model */
// nll from y
vector<Type> XSg1 = XnS * gamma1; //alpha_i in the model
vector<Type> XRg2 = XnR * gamma2; //beta_i
vector<Type> XRg3 = XnP * gamma3; //eta_k
```

```
for (int k=0; k<g; k++)
 for(int i=0; i<(g-1); i++)
   for(int j=i+1; j<g; j++)
   {
     if(i == k || j == k) {
       continue;
     }
     else {
     Type y1 = y(i,j,k); //here we make use of the actual data
     Type y2 = y(j,i,k);
     Type alphai = XSg1(i) + a(i);
     Type alphaj = XSg1(j) + a(j);
     Type betai = XRg2(i) + b(i);
     Type betaj = XRg2(j) + b(j);
     Type etak = XRg3(k) + c(k);
     Type muijk = 0.0;
     Type mujik = 0.0;
     for(int t=0;t<kd;t++)
         muijk += (XvD(i,j,t) * delta1(t) + XvD(i,k,t) * delta2(t));
         mujik += (XvD(j,i,t) * delta1(t) + XvD(j,k,t) * delta3(t));
     }
     Type rhoij = 0.0;
     for(int t=0;t<kc;t++)
          rhoij += XvC(i,j,k,t) * delta4(t);
     Type xi1 =  muijk + alphai + betaj + etak;
     Type xi2 =  mujik + alphaj + betai + etak;
     Type xi3 =  muijk + mujik + alphai + betaj + alphaj + betai +
       etak + rhoij;
     Type den = 1.0 + exp(xi1) + exp(xi2) + exp(xi3);
     nll -=  y1 * xi1  + y2 * xi2 + rhoij * y1 * y2 - log(den);
     }}

  vector<Type> R(3);
 R(0) = S(1) / sqrt(S(0) * S(3));
 R(1) = S(2) / sqrt(S(0) * S(5));
```

```
    R(2) = S(4) / sqrt(S(5) * S(3));

    ADREPORT(R);



  return nll;
}
```

E: C++ code triadic model results

```
#define TMB_LIB_INIT R_init_mypkg
#include <TMB.hpp>

template<class Type>
Type objective_function<Type>::operator() ()
{
  /* data section */
  DATA_MATRIX(XnS);
  DATA_MATRIX(XnR);
  DATA_MATRIX(XnP);
  DATA_ARRAY(XvD); //will have 3 dimensions, 21,21,x alleen i,j similarity
  DATA_ARRAY(XvC); //will have 4 dimensions, 21,21,21,1 (is rho)
  DATA_ARRAY(y); //network, 3 dimensions




  /* Parameter section */
  PARAMETER_VECTOR(gamma1); //sender
  PARAMETER_VECTOR(gamma2); //receiver
  PARAMETER_VECTOR(gamma3); //perceiver
  PARAMETER_VECTOR(delta1); //density
  PARAMETER_VECTOR(delta2); //rho
  PARAMETER_VECTOR(alpha); //vector all (co)variances, in R c(1,0,1,0,0,1)
  PARAMETER_VECTOR(a);
  PARAMETER_VECTOR(b);
  PARAMETER_VECTOR(c);

  int kd = XvD.cols();
  int kc = XvC.cols();
  int g = y.cols();


  Type nll=0.0;      // Negative log likelihood function
```

```
/* Cholesky decomposition and inverse covariance matrix */
vector<Type> S(6);
vector<Type> S1(6);
S(0) = alpha(0) * alpha(0);    //a
S(1) = alpha(0) * alpha(1);    //ab
S(2) = alpha(1) * alpha(1) + alpha(2) * alpha(2); //b
S(3) = alpha(0) * alpha(3);    //ac
S(4) = alpha(1) * alpha(3) + alpha(2) * alpha(4); //bc
S(5) = alpha(3) * alpha(3) + alpha(4) * alpha(4) +
  alpha(5) * alpha(5); //c
Type DS =  S(0) * (S(2) * S(5) - S(4) * S(4)) - S(1) *
  (S(1) * S(5) - S(3) * S(4)) + S(3) * (S(1) * S(4) - S(2) * S(3));
S1(0) = (S(2) * S(5) - S(4) * S(4)) / DS;
S1(1) = - (S(1) * S(5) - S(4) * S(3)) / DS;
S1(2) = (S(0) * S(5) - S(3) * S(3)) / DS;
S1(3) = (S(1) * S(4) - S(2) * S(3)) / DS;
S1(4) = - (S(0) * S(4) - S(1) * S(3)) / DS;
S1(5) = (S(0) * S(2) - S(1) * S(1)) / DS;
Type quadu = ((a*a).sum()) * S1(0) + ((a*b).sum()) * S1(1) * 2.0 +
  ((b*b).sum()) * S1(2) + ((a*c).sum()) * S1(3) * 2.0 +
  ((b*c).sum()) * S1(4) * 2.0 + ((c*c).sum()) * S1(5);
nll -= -0.5 * quadu - g * 0.5 * log(DS);


ADREPORT(S);

/* Defining fixed effects triadic model */
// nll from y
vector<Type> XSg1 = XnS * gamma1; //alpha_i in the model
vector<Type> XRg2 = XnR * gamma2; //beta_i
vector<Type> XRg3 = XnP * gamma3; //eta_k

for (int k=0; k<g; k++)
 for(int i=0; i<(g-1); i++)
```

```
for(int j=i+1; j<g; j++)
{
  if(i == k || j == k) {
    continue;
  }
  else {
 Type y1 = y(i,j,k); //here we make use of the actual data
 Type y2 = y(j,i,k);
 Type alphai = XSg1(i) + a(i);
 Type alphaj = XSg1(j) + a(j);
 Type betai = XRg2(i) + b(i);
 Type betaj = XRg2(j) + b(j);
 Type etak = XRg3(k) + c(k);
 Type muijk = 0.0;
 Type mujik = 0.0;
 for(int t=0;t<kd;t++)
   {
    muijk += (XvD(i,j,t) * delta1(t));
    mujik += (XvD(j,i,t) * delta1(t));
   }
 Type rhoij = 0.0;
 for(int t=0;t<kc;t++)
      rhoij += (XvC(i,j,k,t)*delta2(t));

 Type xi1 =  muijk + alphai + betaj + etak;
 Type xi2 =  mujik + alphaj + betai + etak;
 Type xi3 =  muijk + mujik + alphai + betaj + alphaj +
   betai + 2*etak + rhoij;
 Type den = 1.0 + exp(xi1) + exp(xi2) + exp(xi3);
 nll -=  y1 * xi1  + y2 * xi2 + rhoij * y1 * y2 - log(den);
  }}

Type pen = 0.0;
vector<Type> R(3);
R(0) = S(1) / sqrt(S(0) * S(3));
```

```
    R(1) = S(2) / sqrt(S(0) * S(5));
    R(2) = S(4) / sqrt(S(5) * S(3));

    pen = 0.5 * log(DS);

    nll -= pen;

    REPORT(pen);
    ADREPORT(R);



  return nll;
}
```

F: R code p_2 function (Bellio & Soriani, 2019a)

```
fit_p2 <- function(y, XnS, XnR, XvD, XvC, M = 0, seed = NULL, trace = FALSE,
                   init = NULL, penalized = FALSE, penSigma = NULL,
                   opt = nlminb, singular.ok = TRUE, ...)
{
  # Do some argument checking
  if(is.null(y) | is.null(XvD) | is.null(XvC)) stop("y, XvD and XvC must be provided \n")
  if(!is.matrix(y))
    {
     warning("network data should be provided as a matrix\n")
     y <- as.matrix(y)
    }
  if(ncol(y) != nrow(y)) stop("y must be a squared matrix\n")
  g <- ncol(y)
  if(!is.null(XnS) && !is.matrix(XnS)) stop("XnS must be a matrix\n")
  if(!is.null(XnR) && !is.matrix(XnR)) stop("XnR must be a matrix\n")
  if(!is.array(XvD) | length(dim(XvD))!=3) stop("XvD must be a 3-dim array\n")
  if(!is.array(XvC) | length(dim(XvC))!=3) stop("XvC must be a 3-dim array\n")
  if(!is.null(XnS) &&  nrow(XnS)!=g) stop("Wrong dimension of XnS\n")
  if(!is.null(XnR) &&  nrow(XnR)!=g) stop("Wrong dimension of XnR\n")
  if(dim(XvD)[2]!=g | dim(XvD)[1]!=g) stop("Wrong dimension of XvD\n")
  if(dim(XvC)[2]!=g | dim(XvC)[1]!=g) stop("Wrong dimension of XvC\n")
  if(!is.numeric(M) || M<0) M <- 0
  if(!is.matrix(penSigma)) penSigma <- NULL
  if(M>0) warning("IS as implemented by TMB is still experimental -->
                  use the fitIS function instead\n")
  #  Create starting values
  kd <- dim(XvD)[3]
  kc <- dim(XvC)[3]
  map <- list()
  XnS.int <- XnS
  XnR.int <- XnR
  if(is.null(XnS) & !is.null(XnR))
    {
     XnS.int <- matrix(0, nrow = g, ncol = 1)
     map <- list(gamma1 = factor(NA))
   }
  if(!is.null(XnS) & is.null(XnR))
    {
     XnR.int <- matrix(0, nrow = g, ncol = 1)
     map <- list(gamma2 = factor(NA))
    }
  if(is.null(XnS) & is.null(XnR))
    {
     XnS.int <- matrix(0, nrow = g, ncol = 1)
     XnR.int <- matrix(0, nrow=g, ncol = 1)
     map <- list(gamma1 = factor(NA), gamma2 = factor(NA))
   }
  kr <- ncol(XnR.int)
  ks <- ncol(XnS.int)
  model.param <-  list(gamma1 = rep(0, ks) , gamma2 = rep(0, kr), delta1 = rep(0, kd),
```

```
                delta2 = rep(0, kc), alpha=c(1, 0, 1), a=rep(0, g), b = rep(0, g))
#  Create list of model data for optimization
if(!penalized) flagpen <- 0
else
{
  if(is.null(penSigma)) flagpen <- 1 else flagpen <- 2
}
flagSigma <- if(is.null(penSigma)) 0 else  c(penSigma[1,1], penSigma[1,2],
                                             penSigma[2,2])
model.data <- list(XnS = XnS.int, XnR = XnR.int, XvD = XvD, XvC = XvC, y = y,
              penflag = as.integer(flagpen), penSigma = as.vector(flagSigma))
# Create AD function with data and parameters
myseed <- if(is.null(seed)) sample(1:10^5, 1) else seed
obj <- TMB::MakeADFun(data = model.data, parameters = model.param, random = c("a", "b"),
       DLL = "p2model", map = map, silent = !trace,
       MCcontrol = list(doMC = M>0, seed = myseed, n = M))
if(trace) cat("\nfitting the model\n")
start <- if(is.null(init)) obj$par else init
lower.vect <-  if(singular.ok) c(rep(-Inf, length(obj$par)-3), 0, -Inf, 0) else -Inf
mod <- try(opt(start, obj$fn, obj$gr, lower = lower.vect), silent = TRUE)
if(!is.character(mod))
{
  par <- mod$par
  if(trace) cat("\ncomputing Hessian\n")
  sde <- try(TMB::sdreport(obj, par.fixed=par), silent=TRUE)
  if(is.character(sde))
    {
      warning("\nNumerical issues in the computation of the Hessian\n")
      hess <- try(numDeriv::jacobian(obj$gr, par, method="simple"), silent=TRUE)
      if(is.character(hess))
       stop("\nNumerical issues in the computation of standard errors are too serious...
              Change optimizer, try out a better starting point or
            switch to penalized estimation\n")
      else  sde <- TMB::sdreport(obj, par.fixed=par, hessian.fixed=hess)
    }
  theta.vcov <-  sde$cov.fixed
  theta.se <- sqrt(diag(theta.vcov))
  Sigma <- matrix(c(sde$value[1], sde$value[2], sde$value[2], sde$value[3]), 2, 2)
  Sigma.se <- matrix(c(sde$sd[1], sde$sd[2], sde$sd[2], sde$sd[3]), 2, 2)
  rho <- sde$value[4]
  rho.se <- sde$sd[4]
  random <- sde$par.random
  random.se <- sqrt(sde$diag.cov.random)
  lnl <- obj$fn(par) + obj$env$report()$pen     #### eliminate the penalty
  lnl <- lnl + g * log(pi * 2)                  #### this constant is introduced by TMB
  res <- list(theta = par, loglik = -lnl,
            AIC = 2 * lnl + 2 * length(par), BIC = 2 * lnl + log(g) * length(par),
            XnS.null = is.null(XnS), XnR.null = is.null(XnR), seed = myseed,
            theta.vcov = theta.vcov, theta.se = theta.se,
            opt = opt, opt.details = mod, ADobj = obj, model.data = model.data,
            sdrep = sde, ranef = random, ranef.se = random.se,
            Sigma = Sigma, Sigma.se = Sigma.se, rho = rho, rho.se = rho.se, M = M,
            penflag = model.data$penflag, penSigma = model.data$penSigma)
```

```
   }
   else stop("Optimization did not converge. Change optimizer, try out a better
   starting point or switch to penalized estimation")
  # Assign S3 class values and return
  class(res)=c("p2")
  return(res)
}
```

G: R code triadic model and p_2 model

```
rm(list = ls())
```

Loading data and libraries

In this code we are going to set the working directory, loading packages, compiling the .ccp file and loading the C++ file

```
#Set working directory
setwd("/Users/nelleke/Desktop/triadisch model")
```

```
#Load packages
packages <- c("TMB", "NetData", "cssTools", "p2model", "igraph", "ggplot2")
sapply(packages, require, character.only = TRUE)
```

```
#Compile C++ file
compile("triadictest4.cpp")
```

Preparation data

Triadic model

In the preparation of the triadic data, we need to load the data. Prepare the input of the data (actor characteristics, and input for density and reciprocity)

```
#Here we collect the necessary data.
data("highTechManagers", package = "cssTools") #triadic network data
data(kracknets, package = "NetData") #attributes of actors
```

```
#Save data in .RData file
rm(advice_data_frame,
   friendship_data_frame,
   krack_full_data_frame,
   reports_to_data_frame)
save(attributes,highTechManagers, file="triadicdata.RData")
```

```
#Load triadic data including actor attributes
load("triadicdata.RData")
```

```
#Create design matrices for actor covariates for sender, receiver and perceiver
XnS <- model.matrix( ~ AGE + TENURE, attributes)[,-1] #sender
XnR <- model.matrix( ~ AGE + TENURE, attributes)[,-1] #receiver
XnP <- model.matrix( ~ AGE + TENURE, attributes)[,-1] #perceiver
```

```r
#Preparing the comparisons between actors i and j for measuring homophily
g <- 21
XvD <- array(1, dim = c(g, g, 4))

for(i in 1:g) {
  for(j in 1:g){
    XvD[i, j, 2] <- as.numeric(attributes$DEPT[i]==attributes$DEPT[j])
    XvD[i, j, 3] <- as.numeric(attributes$LEVEL[i]==attributes$LEVEL[j])
    XvD[i, j, 4] <- abs(attributes$AGE[i] - attributes$AGE[j])
  }
}
```

```r
#Preparing the data input of reciprocity.
#In future research here covariates can be added
XvC <- array(1, dim = c(g, g, g, 1))
```

```r
#Putting data in y so it is easier to use in following code
y <- highTechManagers
```

p_2 Model

```r
#Load aggregated data for use p2 model
data(kracknets, package = "NetData")
```

```r
#Changing edgelist into adjacency matrix
g <- 21
datap2 <- matrix(0, g, g)
ind <- 1
for(i in 1:nrow(friendship_data_frame)){
  sele <- friendship_data_frame[i,]
  datap2[sele$ego, sele$alter] <- sele$friendship_tie
}
```

```r
#Create design matrices for actor covariates for sender and receiver
Xn <- model.matrix( ~ AGE + TENURE, attributes)[,-1]
```

```r
#Preparing the comparisons between actors i and j for measuring homophily
XvDp2 <- array(1, dim = c(g, g, 4))
for(i in 1:g){
  for(j in 1:g){
    XvDp2[i, j, 2] <- as.numeric(attributes$DEPT[i]==attributes$DEPT[j])
    XvDp2[i, j, 3] <- as.numeric(attributes$LEVEL[i]==attributes$LEVEL[j])
    XvDp2[i, j, 4] <- abs(attributes$AGE[i] - attributes$AGE[j])
  }}
```

```r
#Preparing the data input of reciprocity.
XvCp2 <- array(1, dim = c(g, g, 1))
```

Description data triadic and p_2 model

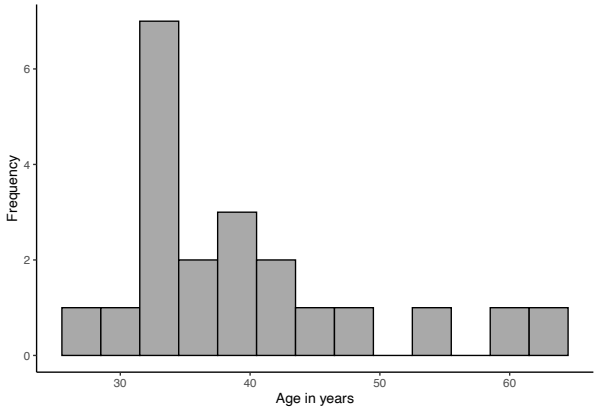
Here we will plot the actor covariates used, show the network data. Measure density and reciprocity and plot sociograms of the network data.

```
triadicnetwerk <- y
```

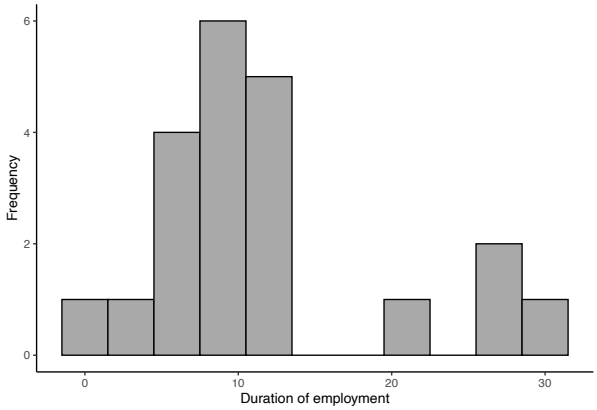
```
#Calculating the mean and standard deviation for every attribute
for (i in 1:4) {
  print(colnames(attributes)[i])
  print(mean(attributes[,i]))
  print(sd(attributes[,i]))
}
```

```
## [1] "AGE"
## [1] 39.71429
## [1] 9.555851
## [1] "TENURE"
## [1] 11.74605
## [1] 8.039362
## [1] "LEVEL"
## [1] 2.714286
## [1] 0.5606119
## [1] "DEPT"
## [1] 2.190476
## [1] 1.167007
```

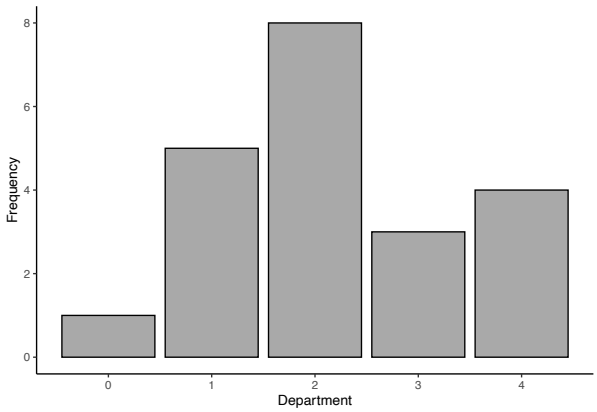
```
#Plot distribution of age of 21 actors
ggplot(attributes, aes(AGE)) +
  geom_histogram(binwidth=3, fill = "darkgrey", color = "black") +
  theme_classic() +
  labs(x = "Age in years", y = "Frequency")
```



```r
#Plot distribution of tenure of 21 actors
ggplot(attributes, aes(TENURE)) +
  geom_histogram(binwidth=3, fill = "darkgrey", color = "black") +
  theme_classic() +
  labs(x = "Duration of employment", y = "Frequency")
```



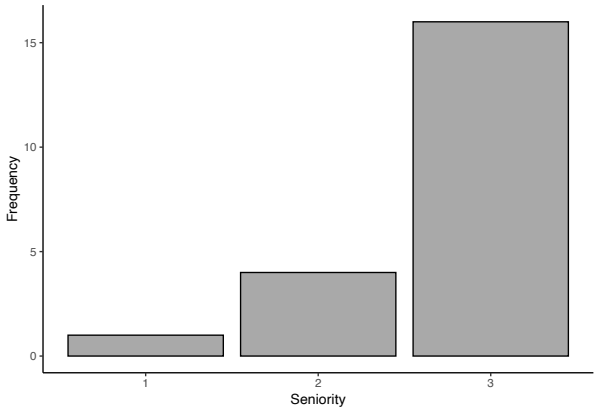

```r
#Plot distribution of 21 actors per department
ggplot(attributes, aes(DEPT)) +
  geom_bar(fill = "darkgrey", color = "black")+
  theme_classic() +
  labs(x = "Department", y = "Frequency")
```




```r
#Correlation between age and tenure
cor(attributes$AGE, attributes$TENURE) #.489
```

```
## [1] 0.4893735
```

```r
#Plot distribution of 21 actors per seniority
ggplot(attributes, aes(LEVEL)) +
  geom_bar(fill = "darkgrey", color = "black")+
  theme_classic() +
  labs(x = "Seniority", y = "Frequency")
```


15
10
5
0
Frequency
1
2
3
Seniority

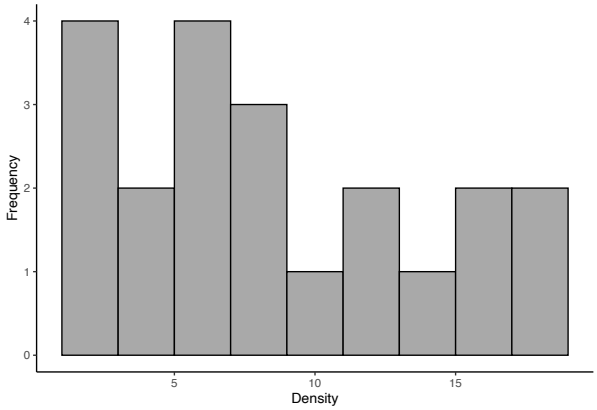
Density

```
#Density p2 model
sum(datap2)/(21*20)*100 #24.29
```

```
## [1] 24.28571
```

```
#Density of each layer in the triadic data
dens <- rep(NA, 21)
for (k in 1:21) {
  dens[k] <- sum(triadicnetwerk[,,k])/(21*20)*100
}
```

```
#Plot of the distribution of the 21 density scores
ggplot(data.frame(dens), aes(dens)) +
  geom_histogram(binwidth=2, fill = "darkgrey", color = "black") +
  theme_classic() +
  labs(x = "Density", y = "Frequency")
```



Reciprocity

```
#Reciprocity p2 model
RECP <- NA
k = 0

for (i in 1:21) {
  for (j in 1:21) {
    if (j <= i)
      next
    else
    k = k + 1
    RECP[k] <- datap2[i,j] && datap2[j,i] == 1
  }

}

sum(RECP)/length(RECP)*100 #10.95
```

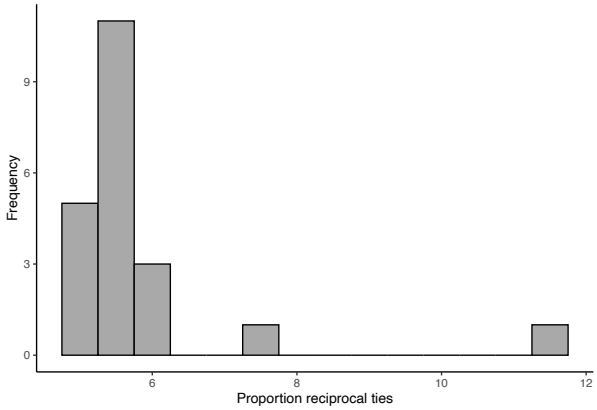
```
## [1] 10.95238
```

```
#Reciprocity triadic data for each layer
REC <- rep(NA, 21)
A <- NA
k = 0

for (t in 1:21) {
  for (i in 1:21) {
    for (j in 1:21) {
      if (j <= i)
        next
      else
        k = k + 1
      A[k] <- triadicnetwerk[i,j,t] && triadicnetwerk[j,i,t] == 1
    }

  }
  REC[t] <- sum(A)/length(A)*100
}
```

```r
#Plot of the distribution of the 21 reciprocity scores
ggplot(data.frame(REC), aes(REC)) +
  geom_histogram(binwidth=.5, fill = "darkgrey", color = "black") +
  theme_classic() +
  labs(x = "Proportion reciprocal ties", y = "Frequency")
```

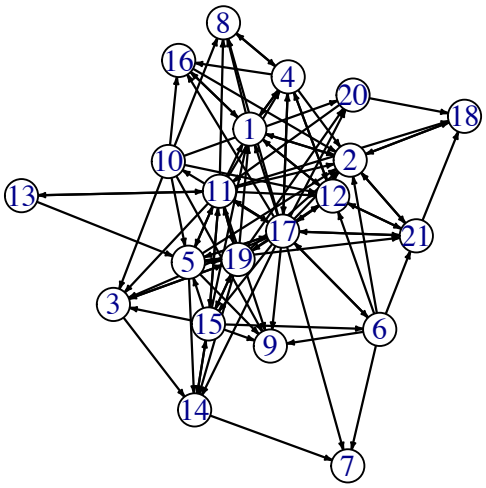


Sociograms

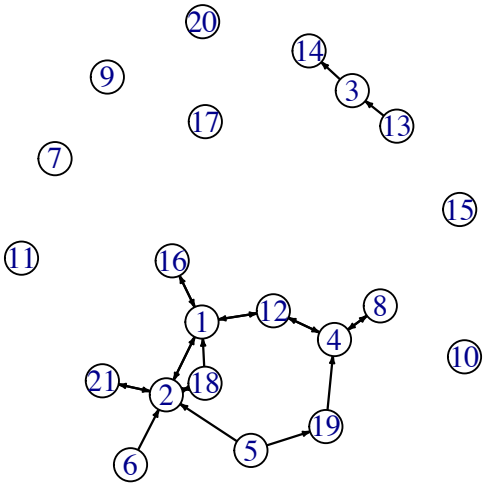
```
#Select two networks of 21 layers
set.seed(042025)
sample(1:21,2) #network 2 and 19
```

```
## [1]  2 19
```

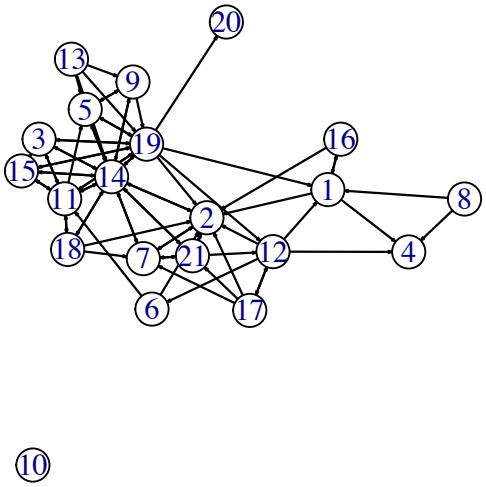
```
#Plotting sociogram of self-reported/aggregated data
g <- graph_from_adjacency_matrix(datap2) #p2
set.seed(042025)
plot(g, edge.arrow.size=0.2,
     vertex.color="white",
     edge.width = 1.2,
     edge.color = "black",
     margin=-.07)
```



```r
#Plotting sociogram of social network data perceived by actor 2
g2 <- graph_from_adjacency_matrix(triadicnetwerk[,,2])
set.seed(042025)
plot(g2, edge.arrow.size=0.2,
     vertex.color="white",
     edge.width = 1.2,
     edge.color = "black",
     margin=-.07)
```




```r
#Plotting sociogram of social network data perceived by actor 19
g19 <- graph_from_adjacency_matrix(triadicnetwerk[,,19])
set.seed(042025)
plot(g19, edge.arrow.size=0.1,
     vertex.color="white",
     edge.width = 1.2,
     edge.color = "black",
     margin=-.07)
```



```r
#First 5 rows of network data self-reported data
head(datap2,5)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14]
## [1,]    0    1    0    1    0    0    0    1    0     0     0     1     0     0
## [2,]    1    0    0    0    0    0    0    0    0     0     0     0     0     0
## [3,]    0    0    0    0    0    0    0    0    0     0     0     0     0     1
## [4,]    1    1    0    0    0    0    0    1    0     0     0     1     0     0
## [5,]    0    1    0    0    0    0    0    0    1     0     1     0     0     1
##      [,15] [,16] [,17] [,18] [,19] [,20] [,21]
## [1,]     0     1     0     0     0     0     0
## [2,]     0     0     0     1     0     0     1
## [3,]     0     0     0     0     1     0     0
## [4,]     0     1     1     0     0     0     0
## [5,]     0     0     1     0     1     0     1
```

```r
#First 5 rows of social network data perceived by actor 2
head(triadicnetwerk[,,2],5)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14]
## [1,]    0    1    0    0    0    0    0    0    0     0     0     1     0     0
## [2,]    1    0    0    0    0    0    0    0    0     0     0     0     0     0
## [3,]    0    0    0    0    0    0    0    0    0     0     0     0     0     1
## [4,]    0    0    0    0    0    0    0    1    0     0     0     1     0     0
## [5,]    0    1    0    0    0    0    0    0    0     0     0     0     0     0
##      [,15] [,16] [,17] [,18] [,19] [,20] [,21]
## [1,]     0     1     0     0     0     0     0
## [2,]     0     0     0     1     0     0     1
## [3,]     0     0     0     0     0     0     0
## [4,]     0     0     0     0     0     0     0
## [5,]     0     0     0     0     1     0     0
```

```r
#First 5 rows of social network data perceived by actor 19
head(triadicnetwerk[,,19],5)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14]
## [1,]    0    1    0    1    0    0    0    0    0     0     0     0     0     0
## [2,]    0    0    0    0    0    0    1    0    0     0     0     0     0     1
## [3,]    0    0    0    0    0    0    0    0    0     0     1     0     0     1
## [4,]    0    0    0    0    0    0    0    0    0     0     0     0     0     0
## [5,]    0    0    0    0    0    0    0    0    1     0     0     0     0     1
##      [,15] [,16] [,17] [,18] [,19] [,20] [,21]
## [1,]     0     1     0     0     0     0     0
## [2,]     0     0     0     0     0     0     1
## [3,]     0     0     0     0     1     0     0
## [4,]     0     0     0     0     0     0     0
## [5,]     0     0     0     0     1     0     0
```

Analyses

Triadic model

```
#Load the compiled file of the C++ code in R
dyn.load(dynlib("triadictest4"))
```

```
#Set certain values necessary for our model
M = 0 #is needed for using multiple imputation in TMB or not
trace = FALSE #needed for TMB
opt = nlminb #using nlminb as optimization
```

```
#Determinen number of actors
g <- ncol(y)
```

```
#Determine value of number of dyad-specific covariates for  mu and rho
kd <- dim(XvD)[3]
kc <- dim(XvC)[4]
```

```
#Needed for the TMB function where output TMB is stored
map <- list()
```

```
#Determine value of number of actor-specific covariates
ks <- ncol(XnS) #sender
kr <- ncol(XnR) #receiver
kp <- ncol(XnP) #perceiver
```

```
#List of parameters which need to be estimated
model.param <-  list(gamma1 = rep(0, ks) , gamma2 = rep(0, kr),
                     gamma3 = rep(0,kp), delta1 = rep(0, kd),
                     delta2 = rep(0, kc), alpha=c(1, 0, 1, 0, 0, 1),
                     a=rep(0, g), b = rep(0, g), c = rep(0,g))
```

```
#List of data input we are using
model.data <- list(XnS = XnS, XnR = XnR, XnP = XnP,
                   XvD = XvD, XvC = XvC, y = y)
```

```
#Set seed
myseed <- 042025
```

```
#Determine the objective and gradient function from TMB
obj <- TMB::MakeADFun(data = model.data, parameters = model.param,
                      random = c("a", "b", "c"),DLL = "triadictest4",
                      map = map, silent = !trace,
                      MCcontrol = list(doMC = M>0, seed = myseed, n = M))
```

```
#Determine starting parameters from TMB function
start <-  obj$par

#Determine lower bounds for optimization
lower.vect <- c(rep(-Inf, length(obj$par)-6),0,-Inf,0,-Inf,-Inf,0)

#Optimization of parameter estimates
mod <- try(opt(start, obj$fn, obj$gr, lower = lower.vect), silent = TRUE)

#Estimated parameter values in vector
par <- mod$par

#Estimation of standard errors with parameter estimates as input
sde <- try(TMB::sdreport(obj, par.fixed=par), silent=TRUE)

#Estimated standard errors of the parameter estimates in vector
theta.vcov <-  sde$cov.fixed

theta.se <- sqrt(diag(theta.vcov))

#Covariance matrix with standard errors
Sigma <- matrix(c(sde$value[1], sde$value[2], sde$value[4],
                  sde$value[2],sde$value[3], sde$value[5], sde$value[4],
                  sde$value[5], sde$value[6]), 3, 3)

Sigma.se <- matrix(c(sde$sd[1], sde$sd[2], sde$sd[4],
                     sde$sd[2], sde$sd[3], sde$sd[5], sde$sd[4],
                     sde$sd[5], sde$sd[6]), 3, 3)

#Correlations and standard errors
rho <- c(sde$value[7],sde$value[8],sde$value[9])

rho.se <- c(sde$sd[7],sde$sd[8],sde$sd[9])

#Used for calculation of AIC and BIC
lnl <- obj$fn(par) + obj$env$report()$pen
lnl <- lnl + g * log(pi * 2)

#List with all the estimated results
res <- list(theta = par, loglik = -lnl,
            AIC = 2 * lnl + 2 * length(par), BIC = 2 * lnl + log(g) * length(par),
            XnS.null = is.null(XnS), XnR.null = is.null(XnR), seed = myseed,
            theta.vcov = theta.vcov, theta.se = theta.se,
            opt = opt, opt.details = mod, ADobj = obj, model.data = model.data,
            sdrep = sde,
            Sigma = Sigma, Sigma.se = Sigma.se, rho = rho, rho.se = rho.se, M = M,
            penflag = model.data$penflag, penSigma = model.data$penSigma)

#Report results to prevent scientific notation
options(scipen = 999)
```

```r
#Zscore and probability two-sided triadic model

Zscore <- res$theta/res$theta.se #zscore
Prob <- 2*pnorm(abs(Zscore),lower.tail = F) #p-value
```

```r
#Bind results (parameter estimates and standard errors with zscore and pvalue)
Results <- cbind(effect=res$theta,
                 SE=res$theta.se,
                 Zscore,
                 prob=round(Prob,3))

rownames(Results) <- c("Sender Age", "Sender Tenure", "Receiver Age",
                       "Receiver Tenure","Perceiver Age", "Perceiver Tenure",
                       "mu", "Department","Seniority", "Age diff", "rho",
                       "alpha", "alpha", "alpha", "alpha", "alpha", "alpha")

Results
```

```
##                         effect          SE       Zscore  prob
## Sender Age       -0.001090961 0.018721504 -0.058273139 0.954
## Sender Tenure     0.037228460 0.022307610  1.668868170 0.095
## Receiver Age     -0.000336743 0.036919642 -0.009120972 0.993
## Receiver Tenure   0.075949570 0.043753331  1.735858007 0.083
## Perceiver Age    -0.016665405 0.021418305 -0.778091659 0.437
## Perceiver Tenure  0.022132117 0.024926238  0.887904406 0.375
## mu               -5.718917519 2.271476715 -2.517709066 0.012
## Department        1.722566888 0.110631286 15.570341423 0.000
## Seniority         1.101040575 0.117003797  9.410297842 0.000
## Age diff         -0.034678002 0.007241154 -4.789015655 0.000
## rho               2.783224269 0.186491180 14.924160325 0.000
## alpha             0.642998563 0.134253272  4.789444260 0.000
## alpha             0.924899289 0.319692378  2.893091457 0.004
## alpha             0.966634995 0.209939244  4.604355903 0.000
## alpha             0.231804847 0.189637968  1.222354621 0.222
## alpha             0.328031129 0.186729769  1.756715765 0.079
## alpha             0.628561828 0.126132948  4.983327832 0.000
```

```r
#Covariance matrix with standard errors
res$Sigma
```

```
##           [,1]      [,2]      [,3]
## [1,] 0.4134472 0.5947089 0.1490502
## [2,] 0.5947089 1.7898219 0.5314825
## [3,] 0.1490502 0.5314825 0.5564279
```

```r
res$Sigma.se
```

```
##           [,1]      [,2]      [,3]
## [1,] 0.1726493 0.2784966 0.1292736
## [2,] 0.2784966 0.6845010 0.2749033
## [3,] 0.1292736 0.2749033 0.2067371
```

p_2 model

```
#Fit the p2 model and summary of results
fit <- fit_p2(datap2, Xn, Xn, XvDp2, XvCp2)
summary(fit)
```

```
## ------------------------------------------
## Approximate maximum likelihood estimation of p2 model
## Log-likelihood at maximum -169.4082
## Model selection criteria
## AIC =  362.8164   BIC = 375.3507
## ------------------------------------------
## Density coefficients
##           Estimate Std. error
## delta1  0.03885399 1.73521103
## delta1  1.57645521 0.34814083
## delta1  1.15354386 0.40545038
## delta1 -0.05488179 0.02458207
## ------------------------------------------
## Reciprocity coefficients
##        Estimate Std. error
## delta2 2.117182  0.6329099
## ------------------------------------------
## Sender coefficients
##          Estimate Std. error
## gamma1 -0.1309324 0.05911481
## gamma1  0.1403820 0.06213760
## ------------------------------------------
## Receiver coefficients
##            Estimate Std. error
## gamma2 -0.002099014 0.03503430
## gamma2  0.045075545 0.04146455
## ------------------------------------------
## Variance matrix of random effects
##           [,1]       [,2]
## [1,]  2.054007 -1.1362520
## [2,] -1.136252  0.9976877
## ------------------------------------------
```

```
#Covariance matrix with standard error
fit$Sigma
```

```
##           [,1]       [,2]
## [1,]  2.054007 -1.1362520
## [2,] -1.136252  0.9976877
```

```
fit$Sigma.se
```

```
##           [,1]      [,2]
## [1,] 0.9504914 0.6369785
## [2,] 0.6369785 0.5646846
```

```r
#Zscore and probability two-sided p2 model

Zscorep2 <- fit$theta/fit$theta.se #zscore
Probp2 <- 2*pnorm(abs(Zscorep2),lower.tail = F) #p2 model
```

```r
#Bind results (parameter estimates and standard errors with zscore and pvalue)
Resultsp2 <- cbind(effect=fit$theta,
                   SE=fit$theta.se,
                   Zscore=Zscorep2,
                   Prob = round(Probp2,3))

rownames(Resultsp2) <- c("Sender Age", "Sender Tenure", "Receiver Age",
                         "Receiver Tenure", "mu", "Department","Seniority",
                         "Age diff", "rho", "alpha", "alpha", "alpha")

Resultsp2
```

```
##                       effect         SE      Zscore  Prob
## Sender Age      -0.130932378 0.05911481 -2.21488278 0.027
## Sender Tenure    0.140382010 0.06213760  2.25921209 0.024
## Receiver Age    -0.002099014 0.03503430 -0.05991312 0.952
## Receiver Tenure  0.045075545 0.04146455  1.08708628 0.277
## mu               0.038853993 1.73521103  0.02239151 0.982
## Department       1.576455206 0.34814083  4.52821114 0.000
## Seniority        1.153543857 0.40545038  2.84509259 0.004
## Age diff        -0.054881787 0.02458207 -2.23259377 0.026
## rho              2.117182055 0.63290987  3.34515571 0.001
## alpha            1.433180559 0.33160213  4.32198843 0.000
## alpha           -0.792818437 0.32068537 -2.47226260 0.013
## alpha            0.607557946 0.28245028  2.15102614 0.031
```