


# An Introduction and Guide to the VIMuRe Algorithm and Examining the Influence of Income Bias on Cognitive Social Structures

Christopher Bouma ✉ 

5672988 - Computing Science, Utrecht University, The Netherlands

---

## Abstract

An understudied aspect in the field of Network Science is to account for errors in observed data. This leads to networks being studied that may not be fully correct and to potentially unsupported conclusions. In this thesis we focus on Cognitive Social Structures, which use network data that stems from the reports of the members of the network. These reports are notoriously unreliable, and so taking the proper steps to account for errors is even more important. As a way to counteract such errors, this thesis studies Bayesian inference, particularly Variational Inference, to obtain a probability distribution over the network instead of a single ‘true’ network. We do this using the VIMuRe model presented by De Bacco et al.. We present a comprehensive guide on all stages of the algorithm, and build upon the VIMuRe model with the addition of another parameter, income bias. While this addition does not significantly improve the model, we explain the steps required for others to build upon it further.

**Keywords and phrases** Network Science, Variational Inference, Cognitive Social Structures, VIMuRe, Bayesian Inference

**Acknowledgements** I would like to thank dr. Erik Jan van Leeuwen and dr. Javier Garcia Bernardo for supervising the project and offering invaluable insights. Thanks to Dr. Caterina De Bacco for both their work and their help during this project. Additionally thanks to Imogen, Simon, Sofia, Sam, Catherine, Menno, Noa and Jorik, for everything.

## 1 Introduction

Network Science is an evergrowing field, with many recent developments in the area of perspective based data. This field of research is called Cognitive Social Structures (CSS) and is defined as the study of social networks as perceived by members of the network. For example, a member of a company could be asked to give their interpretation of social circles within the company. These responses are then used to try to reconstruct the original network.

Recently, papers are being released with new models for sampling [13] and new inference methods [5] specifically to analyse this kind of data. People tend to be untrustworthy when it comes to reporting links, and so in the past the field was dismissed by many as being unreliable. Brands released a review paper [2] in 2013 giving a wonderful insight into the progression of the field since its inception in the 1950s. In this paper Brands mentions a harsh critique from Bernard and Killworth in 1979 that “There is no way of knowing whether the data one acquires from asking ‘who do you like or who do you talk to’ have any meaning beyond the trivial statement that they were responses to a question asked by an experimentalist”. In spite of this, the field has come a long way and Brands paper as well as the recent surge in papers [5, 13] show a lot of merit in the field. A large factor in this is the distinction between cognitive and behavioural realms of networks. Behavioural oriented networks consider the actions of the members, where there is hard data that they performed a certain action. Meanwhile, cognitive oriented networks focus on the interpretations of the members about their surroundings and actions. The growth of CSS has seen it focus more on the cognitive perspectives of the network, and how this data can be utilised. It is not being compared to, or used as a replacement for actual behavioural data, but has become its own field and thus can grow and improve.

The origin of this project stemmed from a paper by Peel et al. [11] on some of the more problematic features of the network science world, not just in relation to CSS. Peel et al. describe the issue of accounting for incomplete data, and the possibility that conclusions may be skewed depending on the data that was analysed. In particular, Peel et al. discuss the importance of considering errors, and that the data collected may not be representative of the true network. They comment on how it has been normalised to draw conclusions from data without addressing the possibility of incomplete data. They link this to the growth of Network Science as a field, as it is rapidly spreading to other disciplines, and being used by people that are less familiar with the fundamentals. A result of this is that practices are being normalised which may be harmful to the integrity of papers being published. Peel et al. look to constructively educate others on good practices, taking errors into consideration and accounting for potentially missing data. It is worth mentioning that they do not do this in an aggressive manner, but rather look to provide guidelines and inform on the “rigorous theory and methods” available to network science.

The combination of accounting for errors, and the field of CSS leads us to two prominent methods used in recent years. These are the Markov Chain Monte Carlo (MCMC) method [13], and Variational Inference (VI) [5]. They both are used to approximate posterior densities for Bayesian models, though their approaches differ greatly. MCMC opts for an iterative approach to improve the distribution, and works very well on tailored data which can be helped out with information from the researchers. Conversely, VI is an optimisation algorithm which is a lot less computationally heavy, and can look at many different models, but will give a less accurate result on average. MCMC has been the standard approach over the past few years, see Guimerà and Sales-Pardo [6], Peixoto [12] and Redhead et al. [13], but De Bacco et al.’s [5] paper from this year has opened up an avenue for VI to become much more widely used with their VIMuRe algorithm.

In order to overcome this issue of accounting for errors, we utilise Bayesian Inference, specifically Variational Inference, to estimate how accurate the data is. First we discuss the field of CSS, and popular methods of generating data. We then discuss different models for the sampling of data as well as recent methods for the reconstruction of networks. Then, we provide a comprehensive guide to the VIMuRe algorithm, describing the parameters and mathematical steps taken in detail. Finally, we adapt the VIMuRe algorithm to account for additional parameters, and compare the performance of the models, as well as the runtime of the original VIMuRe algorithm.

## 2 Literature Review

Let us first look at network reconstruction as a general concept. Peel et al.’s paper [11] gives a basic and intuitive introduction into the field of network reconstruction. It describes the processes by which a network can be constructed, the measuring of data as well as the inference involved in the reconstruction. In doing so, they discuss the importance of the quality of data. Noisy data is obviously not ideal, but it is also the most realistic kind. As such, they argue that it is incredibly important to address the possibility of obscured data.

### 2.1 Cognitive Social Structures

First, we consider Cognitive Social Structures. CSS differs from other network science fields due to the nature of the data collected. The data in CSS is based on an individual’s perception of their own social networks. Brands [2] covers the resurgence of the field in the preceding 25 years. The increased attention is due to the innovative ways in which researchers are handling the data. They now utilise models to account for inaccuracies, taking a number of parameters into consideration. For instance, within a company, they might take the hierarchical status of a position into consideration, whereby the CEO of the company will not know all the people at

the bottom, but the bottom will certainly know who the top boss is [15, 16]. Since the data is suddenly based on individual's perceptions, there has to be a shift in how it is collected. Instead of hard stats that can be read from a sheet, there has to be a way for the participants to convey their view in a way that minimises biasing them.

There are three main types as described in Brands paper: roster methods, ego networks, and experimental methods. Take a person  $i$  working in a company. Roster methods involve providing the participant  $i$  with a grid a row and column for everyone in the company. Here they can indicate that there is a link between themselves and person  $j$ , denoted  $X_{ij}$ . This method also allows them to assign links between other members, for instance person  $k$  could claim there is a link between  $i$  and  $j$ , denoted as  $X_{kij}$ . Each individual's perspective is called a 'slice'. Depending on the model, different thresholds may be required for a link to be accepted; for instance 50% of all slices reporting the link.

Ego networks function similarly, but instead of providing the participant with a list of names, they are given a blank paper, and asked questions like who are your closest friends. This means they are restricted to only giving data on links relating to themselves. An interesting aspect of this approach is that it does not impart any bias on the participant, since they will not be given any names to potentially remember links, and instead will be entirely based on their memory. Whether or not this is a positive or negative thing depends on the research question. It is possible that not all of their responses are counted. Past papers such as Marsden [8] describe setting no limit on the number of people the participant can name, but only counting the first five since they are more likely to suggest links they are more sure of first, and also for computational reasons. In De Bacco et al. [5], however, the participants of the Kanataka survey were limited to mentioning four people.

Finally, experimental data involves providing the participant with a made-up social network, and seeing how many attempts it takes them to learn the social structure. While all methods have their place, recent papers that we will focus on such as Redhead et al. [13] work with ego networks, and the VIMuRe algorithm from De Bacco et al. [5] works with both roster methods and ego networks.

## 2.2 Network Generation

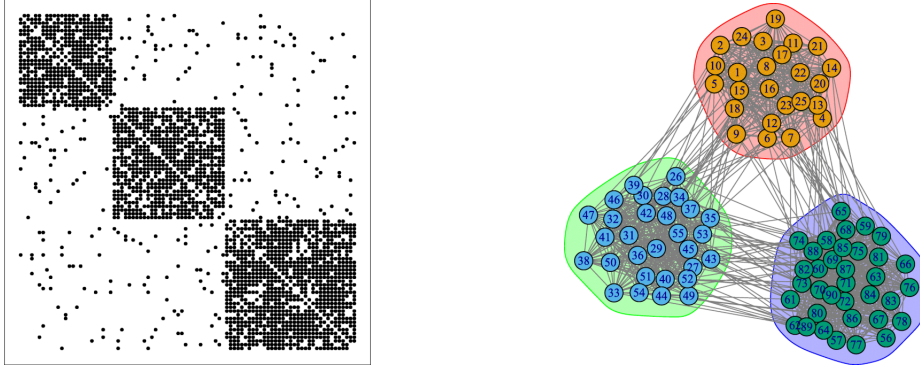
In the network reconstruction process we will consider three main steps. First, we generate a ground truth network that will serve as the unseen network we attempt to reconstruct. Then, we use a specific model to gather reports from the members of this network. Finally, using the reported data, we attempt to infer and reconstruct the original network.

For the purposes of this thesis, we aim to simulate the ground truth network by generating it using modern techniques. Methods for generating data are very well described by Redhead et al. [13]. The basis for data generation stems from Stochastic Block Models (SBM), which act as guidelines for the likelihood of links between nodes. Take three sports clubs; the members of them are more likely to be friends with other people within the clubs than members of other clubs, since they will be spending more time with them on average. As such, we can assign a higher chance that there exists a link between any two members of the same community. This is the premise of SBM, and it can be seen more clearly in Figure 1.

Taking  $Y_{ij}$  as a binary variable indicating a link between nodes  $i$  and  $j$ , and  $B_{[b(i),b(j)]}$  represents the probability of a link between the block communities that  $i$  and  $j$  are in we can represent the formula for whether a link exists as follows:

$$Y_{ij} \sim \text{Bernoulli}(\text{Logistic}(B_{[b(i),b(j)]})). \quad (1)$$

SBM is a useful tool to enable the generation of networks to mimic communities, but we can take



(a) An adjacency matrix, with dots indicating links between two nodes.

(b) The network generated by the matrix shown in (a). The three communities are very apparent.

■ **Figure 1** In Figure 1a, each large block in the matrix represents a community. Members of a community will have a larger chance of being connected to other members of the community so we see the block largely filled. In Figure 1b, we see strongly connected communities, with some connections between them. This is the result of generating a network based on the block model in (a). Images taken from Lee et al. [7]

it one step further by utilising a Social Relations Model as seen in Redhead et al. [13]. The Social Relations Model will allow us to attribute individual characteristics to nodes which will influence their link probabilities not just with other communities, but with every other node. For instance, it can take the social status of a node into consideration, and so a node with high social status will have a higher chance of being connected to other nodes with high social status. We can adjust our above formula this new equation:

$$Y_{ij} \sim \text{Bernoulli}(\text{Logistic}(\phi_{ij})) \quad (2)$$

where:

$$\phi_{ij} = B_{[b(i),b(j)]} + \lambda_i + \pi_j + \delta_{ij} + \dots \quad (3)$$

where  $\lambda$  represents a node's tendency for outgoing links,  $\pi$  the node's tendency for incoming links,  $\delta$  is the matrix denoting the chance for reciprocity between the two nodes. The ellipses imply that there could be more effects, but that is up to the discretion of the researcher.

These models will allow many options for the generated network and thus allow many different avenues of research. Unfortunately, the Social Relations Model is not yet supported by the VIMuRe algorithm by De Bacco et al. [5], but it is an interesting option for future research.

### 2.3 Models for Sampling Data

The next step is to sample from the generated ground truth network  $Y$ . We will do this by pulling a number of data points from the generated network with a chance that the data is incorrect. These are taken in the form of surveys, asking the members questions about the network. We can define a base sampling formula slightly altered from the Redhead et al. paper where  $X_{i,j,q}$  represents a data point where person  $i$  has indicated a link between themselves and person  $j$  in question  $q$ :

$$X_{i,j,q} \sim \text{Bernoulli}(\psi_{i,j,q}) \quad (4)$$

where  $\psi$  represents the factors considered when determining if the report is accurate:

$$\psi_{i,j,q} = \alpha_{i,q}(1 - Y_{ij}) + \chi_{i,q}Y_{ij} \quad (5)$$

Here,  $\alpha_{[i,q]}$  represents the false positive rate, that is, participant  $i$  reports a link in question  $q$  when no such link exists in the real network  $Y$ . Conversely,  $\chi_{[i,q]}$  represents the true-tie recall rate, where the participant  $i$  reports a link at layer  $q$  that does exist in the true network  $Y$ . The key to the model is how these  $\alpha$  and  $\chi$  values are calculated.

An example of a more detailed model is the Attribute Related model from the Redhead et al. paper, where they consider the social standing of a person and incorporate it by constructing  $\alpha$  and  $\chi$  as functions of covariates specific to individuals, blocks, or network types. If we take  $S_{[i]}$  as the social status of person  $i$ , we can write the formula:

$$\text{logit}(\alpha_{i,q}) = \text{logit}(\mu_{\alpha_q}) + \sigma_{\alpha_q} \hat{\alpha}_{i,q} + \eta_{\alpha_q} S_i + \dots \quad (6)$$

$$\text{logit}(\chi_{i,q}) = \text{logit}(\mu_{\chi_q}) + \sigma_{\chi_q} \hat{\chi}_{i,q} + \eta_{\chi_q} S_i + \dots \quad (7)$$

where:  $\mu_{\alpha_q}$  and  $\mu_{\chi_q}$  reflect the average rates of all people in question  $q$ ,  $\sigma_{\alpha_q}$  and  $\sigma_{\chi_q}$  scale the variation from individual random effects for question  $q$  and  $\eta_{\alpha_q}$ , and  $\eta_{\chi_q}$  represent the effect of some covariate, in this case social status  $S_i$ . Again, it is left open to incorporate additional features. In this paper we will focus on De Bacco et al.'s models related to reliability and mutuality that we will touch upon shortly.

## 2.4 Network Reconstruction

Once sampled, the next step is to attempt to infer the true network using the samples we have taken. Recent papers such as Guimerà and Sales-Pardo [6] and Clauset [4] have delved into an approach to account for how accurate the data is by using a Bayesian approach. Peixoto's paper [12] describes in detail the process of inferring a network from noisy data. They use the Bayesian Posterior Distribution which tells us how likely any given data point is to be sampled from a given distribution. Bayes' theorem allows us to predict the probability of an event given some data that may or may not be relevant. As described in Peixoto's paper [12], if we take the true network to be  $Y$ , and our noisy measurement as  $X$ , which is related to but not exactly  $Y$ , our goal is to recreate  $Y$  as best we can using  $X$ . Let the network generation be modelled with probability  $P(Y|\sigma)$  where  $\sigma$  are arbitrary model parameters. Let the noisy measurement of the data be modelled using probability  $P(X|Y, \epsilon)$  from generated network  $Y$  and some further parameters  $\epsilon$ . Bayes' rule then allows us to estimate  $Y$  from the posterior distribution as:

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)} \quad (8)$$

where

$$P(X|Y) = \int P(X|Y, \epsilon)P(\epsilon)d\epsilon \quad (9)$$

is the marginal probability of the measurements  $X$  and

$$P(Y) = \int P(Y|\sigma)P(\sigma)d\sigma \quad (10)$$

is the prior probability for  $Y$ . Both of these integrals can be calculated without too much hassle.  $P(X)$  is a normalisation constant based on the probability of the observed measurement. This  $P(X)$  is often called the evidence. The problem is that it is very expensive to compute. This stems from computing the evidence which is either unavailable in closed form or is exponential in the number of variables [1]. Approaches such as Markov Chain Monte Carlo [12], or Variational Inference [1] are clever tactics used so that we do not have to calculate this tricky posterior

distribution, but can approximate it. Markov Chain Monte Carlo (MCMC) goes through many iterations to improve a joint posterior distribution until it can generate sufficiently many samples. Meanwhile, Variational Inference (VI) [10] shows how we can use optimisation to approximate a distribution close to the posterior distribution. Our research will be based on VI, specifically the Variational Inference for Multiply Reported data (VIMuRe) algorithm [5], since it is a brand new method which utilises good error related practices and still has much room to grow.

As discussed at length in Blei et al. [1], VI is a method to approximate posterior densities for Bayesian models. Bayesian models take prior information and use probabilities to infer an outcome, or in our case, a distribution of a network. It takes observations from a network, and uses them to rebuild the original network without being able to access the original. The reason we aim for a distribution as opposed to the exact network, is that the inference process works by altering the appropriate variables over time. By tweaking these variables of the distribution, we can iteratively improve in small steps. Variational Inference differs from other approaches for Bayesian models such as the Markov Chain Monte Carlo method, used by Redhead et al. [13], by sacrificing some accuracy in order to be faster and take more variables into consideration. So, MCMC is capable of providing more precise distributions at the cost of a higher processing time, so when there is a smaller data set that we think we have a good model for, it can benefit from taking the extra time to get a very accurate distribution. VI by contrast allows us to explore many different models with lots of data which may or may not be relevant, it allows us to broaden our search but we are more likely to be less accurate. Blei et al. [1] describe MCMC useful for data collected for 20 years with an appropriate model, and VI useful for sorting one billion text documents which will be suggested to users based on search queries. Blei et al. also comment on how sometimes the geometry of the posterior distribution can be unfavourable for MCMC. For instance, Gibbs sampling (a MCMC algorithm) can efficiently sample from a distribution if it can focus on a single model, but if there are multiple, for instance in a mixture model, VI can outperform even on small data sets. For an in-depth comparison and description of VI as a concept we recommend reading the Blei et al. paper [1].

### 3 An in-depth breakdown of VIMuRe

In order to account for errors, and determine the probability of misrepresented data, De Bacco et al. [5] present the VIMuRe algorithm. VIMuRe uses VI specifically on Cognitive Social Structures, when there are multiple reports about events which may or may not represent the true network. To do this, they observe both reliability of the nodes, as well as mutuality, as variables that influence the reported network. This section expands upon the work done by De Bacco et al. by breaking down the different parts of their VIMuRe algorithm and making them more accessible.

First, let us consider VI as a concept. Blei et al. [1] describe VI as an algorithm that uses optimisation to approximate the conditional density (also called posterior distribution) of latent variables when given observed variables. Latent variables are variables that we cannot directly observe, but we infer based on the other variables available to us. So, we have a number of latent variables which we are going to alter in order to make our distribution as similar as we can to our target distribution. Again, let  $X$  be the set of observed variables, or reports in our case, and  $\epsilon$  be the set of latent variables, with joint density  $p(\epsilon, X)$ . Thus, we define the conditional density as:

$$p(\epsilon|X) = \frac{p(\epsilon, X)}{p(X)}. \quad (11)$$

Note the similarity to Equation 8, where  $p(X)$  here is the evidence that we struggle to calculate. In VI, Blei et al. describe it as specifying a family  $\mathcal{Q}$  of densities over the latent variables. Now,

each  $q(\epsilon) \in \mathcal{Q}$  is a candidate approximation to the exact conditional. As such, we want to find the best candidate, the one with the smallest possible Kullback-Leibler divergence to the ground truth's posterior distribution. The KL divergence tells us how different two given distributions are, so our goal is to minimise it.

$$q^*(\epsilon) = \arg \min_{q(\epsilon) \in \mathcal{Q}} \text{KL}(q(\epsilon) || p(\epsilon|X)) \quad (12)$$

Here  $q^*(\epsilon)$  represents the best approximation of the conditional, within the family  $\mathcal{Q}$ . The problem with this is that it requires calculating the evidence  $\log p(x)$ , which is very well explained in Blei et al. [1]. To avoid this, we can use the evidence lower bound method (ELBO). The ELBO is equivalent to the negative KL divergence plus  $\log p(x)$ , which is a constant with respect to  $q(z)$  [1]. As such, as we maximise the ELBO, we minimise the KL divergence. Additionally, it will act as a lower bound of the evidence  $\log p(x)$ , hence the name. Now that we have an optimisation problem we can adjust the variables in order to get as close as possible to the posterior distribution of the ground truth. We will discuss the next steps later, but for now, know that we are dealing with an optimisation problem.

There are different forms of VI as discussed by Blei et al. [1], a common implementation of which is called Coordinate Ascent Variational Inference (CAVI) [1], where they find a local optimum similar to hill climbing, adjusting different variables over many iterations. VIMuRe utilises this CAVI approach, which we will explore in depth shortly. First, let us consider the variables and notation that the algorithm works with.

### 3.1 The Model

The number of nodes is represented by  $N$  in a directed network, and the number of reporters by  $M$ .  $Y$  represents the ground truth network as an  $N \times N$ -dimensional adjacency matrix, where  $Y_{ij}$  indicates a link between nodes  $i$  and  $j$ .  $Y_{ij} \in \{0, 1, \dots, K-1\}$  gives the weight of the link in question.  $K$  represents the highest weight that can be given, but is usually represented in binary form where 1 indicates a link, and 0 no link.

During inference, we will never see this network  $Y$ , but we will attempt to recreate it as accurately as possible. In order to do this, we will utilise the data from the reported network  $X$ . These observations are based on the ground truth  $Y$ , but may be incorrect due to the nature of perspective data. Reports are presented as  $X_{ijm} \in \{0, 1, \dots\}$ , that there is a link between nodes  $i$  and  $j$  according to reporter  $m$ . This is represented in a  $N \times N \times M$ -dimensional tensor.

The tensor  $R_{ijm} \in \{0, 1\}$  serves as a mask for which links a reporter can observe, where a 1 indicates that reporter  $m$  can report on the link  $i \rightarrow j$ . This is also a  $N \times N \times M$ -dimensional tensor. Currently, VIMuRe only supports networks where reporters can report on ties in which they are involved, but the ability to report on any network is a planned future work.

The way that the reported network  $X$  will differ from the true network  $Y$  is dependent on three parameters:  $\theta$ ,  $\lambda$  and  $\eta$  which we will discuss now.

Theta ( $\theta_m \in \mathbb{R}_{\geq 0}$ ) represents the reliability of a reporter  $m$ .  $\theta < 1$  implies that the reporter is unreliable and under-reports links.  $\theta_m = 1$  implies that they are always reliable in their reports, neither under or over-reporting, and  $\theta_m > 1$  means they over-report the existence of links. This parameter will influence how we view the reported data. In report generation, higher theta values will mean a reporter will report more links. Within inference, a predicted high theta value may lead the algorithm to not trust the links claimed by the reporter.

Lambda ( $\lambda_k \in \mathbb{R}_{\geq 0}$ ) is the influence of the edge's existence in the true network  $Y$  on the reporting model. Generally,  $K = 2$  meaning  $k \in \{0, 1\}$ , where 1 indicates a link exists and 0 indicates it does not. If the link does exist in the ground truth,  $\lambda_k$  will increase and the reporter

will be more likely to report it, or less likely to report it if the link does not exist. If  $K = 3$ , then  $k$  could be 0, 1, or another positive integer, which would represent a higher likelihood of the link.  $\lambda_k$  takes this value into consideration, growing with  $k$  at a set rate. During inference, when we look at our prediction of the link  $i \rightarrow j$ ,  $\rho_{ij,k}$ , we get an array of size  $K$ , showing the probability of each  $k$  value.

Mutuality ( $\eta \in \mathbb{R}_{\geq 0}$ ) works on top of  $\theta$  and  $\lambda$ , and takes effect if there is a link in the opposite direction. As such, if we are looking at  $X_{ijm}$ , the link  $i \rightarrow j$ , we check if the reporter  $m$  also claimed  $X_{jim}$ , the link  $j \rightarrow i$  exists. If so, it is more likely that the link from  $i \rightarrow j$  will exist since a lot of relations are reciprocal in nature. This is factored in by the global constant  $\eta$  being added on to the probability of the link existing. It is worth noting that  $X_{jim}$  takes the reporter into consideration, so in order for the mutuality to influence the model, the reporter must themselves report a link in the opposite direction.

During the Variational Inference process, we will be taking a randomly generated distribution  $\rho$  and improving it over many iterations. It is our estimation of the true unseen network. It can be described as  $\rho_{ij,k}$  to give the probability that there is a link  $i \rightarrow j$  with expected link weight  $k$ . While it is not a physical network, if one were to keep pulling from this distribution, one would hope to create a network as close to the original network  $Y$  as possible.

With these variables in mind, De Bacco et al. posit the expected value of any given link to be reported as:

$$\mathbb{E}[X_{ijm}|Y_{ij} = k] = \theta_m \lambda_k + \eta X_{jim}. \quad (13)$$

An observation made by reporter  $m$  about the link  $i \rightarrow j$ , assuming that the link  $i \rightarrow j$  has value  $k$ , is equal to the reliability of the reporter  $\theta_m$  times the average influence  $\lambda_k$  plus the mutuality factor  $\eta$  assuming that link  $X_{jim}$  is reported.

The model can be extended by creating layers, which function as another set of reports to be inferred upon. For instance, on a survey, the first question will be the first layer of reports, the second question the second layer and so forth. While it would be possible to incorporate layers influencing the inference of other layers, the tests done by De Bacco et al. involve running the model individually on each layer and so we will not be considering this more complex model. As such, these descriptions and calculations are all being made for a single layer.

### 3.1.1 Data Generation and Reporting

In the absence of real data, we will generate our ground truth using a variation of a Stochastic Block Model (SBM), which we have described at length in Subsection 2.2. In VIMuRe, there are three methods available: Standard SBM, Degree Corrected SBM, and Multitensor. Standard SBM will only take the community a node belongs to into account when generating ties. Degree Corrected SBM assigns a new parameter to each node which will influence the number of links a node has. This helps to add variance to the network, allowing some nodes to become hubs. Multitensor, which is adapted from CRep [14], additionally takes the reciprocity of nodes into consideration. This means that a node will be more likely to be linked to another if that node is already linked to it.

So, the first step in the VIMuRe algorithm is to generate the ground truth network to be tested. Multiple rounds of inference will be performed on the same ground truth network in order to improve the robustness of the algorithm. It is possible to input a specific seed to be constant throughout multiple experiments. This can be done to ensure a well-connected network for the algorithm to work on.

Now we generate the reported network using the ground truth. That is, for every reporter  $m$  and every link  $i, j$ , we will generate whether or not a link has been reported. First, a reliability



score is assigned to every reporter,  $\theta_m$ . The reliability value each reporter receives is based on a given input, or if no input is given, the algorithm will assign each member a random  $\theta$  value based on a Gamma distribution using a default shape of 2.0 and rate of 0.5. The reporter  $\theta$  vector is then multiplied by the  $\lambda_k$  matrix. This tensor represents the reliability and average influence of the true network,  $\theta_m \lambda_k$ . Then, if mutuality is included in the input, the mutuality  $\eta X_{jim}$  is added on. Since we must report on both directions of a link,  $i \rightarrow j$  and  $j \rightarrow i$ , and since the other's existence will effect the mutuality factor, one direction is chosen at random first. Say we take  $i \rightarrow j$ , then the mutuality factor for the link would be equal to  $\eta$  times the expected value of the marginal distribution as given in the Supplementary Information [5, SI, Equation S45], changing the probability of the link to:

$$\mu_{ijm} = \frac{\theta_m \lambda_k |_{Y_{ij}=k} + \eta \theta_m \lambda_k |_{Y_{ji}=k}}{(1 - \eta^2)}, \quad (14)$$

where  $\mu$  is our conditional expectation of the link. Now that we have our expectation for one direction, the other direction  $j \rightarrow i$  can be calculated using this prediction for the mutuality factor. To determine if the link exists, the conditional expectation is used as input for a Poisson distribution, and the output is the value of  $X_{ijm}$  or  $X_{jim}$ . This is repeated for every possible link. As previously mentioned, currently the VIMuRe algorithm only supports when the reporter is one of the two possible members  $i$  or  $j$ .

## 3.2 CAVI and the ELBO method

The VIMuRe algorithm [5] utilises a specific form of Variational Inference known as Coordinate Ascent Variational Inference (CAVI) [1]. This is an iterative approach similar to a hill climbing algorithm. We have a base distribution of latent variables which we alter with every iteration. The metric used to identify if the changes are in the right direction is the ELBO method that we will touch upon shortly. Basically, the algorithm will loop until the algorithm has converged according to the ELBO method, or a given number of iterations have occurred.

In order to adjust the variables, we take one latent variable,  $j$ , and fix every other latent variable which we will call the set  $-j$ . With  $\propto$  representing proportionality, we set:

$$q_j^*(\epsilon_j) \propto \exp\{\mathbb{E}_{-j}[\log p(\epsilon_j | \epsilon_{-j}, X)]\} \quad (15)$$

where  $q$  is the distribution we are adjusting, with latent parameter set  $\epsilon$ , and  $p$  is the distribution we are attempting to replicate, using the reported data set  $X$ . Here we adjust  $j$  while taking all non- $j$  latent variables as fixed. We repeat this for all latent variables, pull from the new adjusted distribution and then compare it to our goal distribution using the ELBO method. This is outlined in the pseudocode shown in Algorithm 1, which is altered from Blei et al. [1].

The VIMuRe algorithm uses the mean-field variational family, which means it is assumed that all these variables are considered conditionally independent. According to De Bacco et al., this is a common assumption made in network models.

A conditional distribution gives the probability of an event occurring assuming other variables are fixed. As such, given these parameters, this allows us to estimate the likelihood of a given link. This formulae is based on the Probability Mass Function (PMF) of a Poisson distribution. This is because it leads to the expected value that was posited in Equation 13. So, if we have all the variables, the probability can be defined as:

$$P(X_{ijm} | X_{jim}, Y_{ji} = k, \lambda_k, \theta_m, \eta) = \frac{(\theta_m \lambda_k + \eta X_{jim})^{X_{ijm}}}{X_{ijm}!} e^{-(\theta_m \lambda_k + \eta X_{jim})} \quad (16)$$

---

**Algorithm 1** *Coordinate ascent variational inference (CAVI)*


---

**Input:** A model  $p(X, \epsilon)$ , a data set  $X$   
**Output:** A variational density  $q(\epsilon) = \prod_{j=1}^m q_j(\epsilon_j)$   
**Initialise:** Variational factors  $q_j(\epsilon_j)$   
**while** the ELBO has not converged **do**  
  **for**  $j \in \{1, \dots, m\}$  **do**  
    Set  $q_j(\epsilon_j) \propto \exp\{\mathbb{E}_{-j}[\log p(\epsilon_j | \epsilon_{-j}, X)]\}$   
  **end for**  
  Compute  $\text{ELBO}(q) = \mathbb{E}[\log p(\epsilon, X)] - \mathbb{E}[\log q(\epsilon)]$   
**end while**  
**return**  $q(\epsilon)$

---

Determining the optimal parameters could be done via derivation, but this is costly. As such, we calculate conditional distributions for each parameter as described by Blei et al. [1]. For this, we need to need to adjust some parameters.

In order to make certain calculations easier, auxiliary variables are used. These are derived from a Poisson distribution, which are then used as prior for a Gamma distribution. Since the Gamma and Poisson distributions are members of the exponential family, the product will also be a member of the exponential family and can be calculated cleanly in closed form. This is called making the model conditionally conjugate.

Using  $\sim$  to represent similarity, De Bacco et al. [5, SI, Equation S3] present these variables as

$$z_{mk}^1 \sim \text{Pois}(\theta_m \lambda_k), \quad z_{ijm}^2 \sim \text{Pois}(\eta X_{jim}), \quad \text{s.t.} \quad z_{mk}^1 + z_{ijm}^2 = X_{ijm} | Y_{ij} = k \quad (17)$$

Here  $z_{mk}^1$  represents the reliability aspect and  $z_{mk}^2$  represents the mutuality aspect. We can now use these two  $z$  variables to simplify inference. The conditional distribution is now adjusted to account for the auxiliary variables as follows:

$$P(X_{ijm}, Z | X_{jim}, Y_{ij} = k, \lambda_k, \theta_m, \eta) \quad (18)$$

It is worth noting that this is constructed such that we can recover the original conditional distribution using the marginal:

$$P(X_{ijm} | X_{jim}, Y_{ij} = k, \lambda_k, \theta_m, \eta) = \sum_{z_{mk}^1, z_{ijm}^2} P(X_{ij}, Z | X_{jim}, Y_{ij} = k, \lambda_k, \theta_m, \eta) \quad (19)$$

Note that the variational distribution for  $Z$  is a Multinomial distribution:

$$q((z_{mk}^1, z_{ijm}^2); X_{ijm}, [\hat{z}_{mk}^1, \hat{z}_{ijm}^2]) \quad (20)$$

Since our variables are assumed to be independent, we aim to infer the parameters of their variational distributions. As described in the Supplementary Information [5, SI, Equation S9], using Blei et al.'s [1] findings we can write the full posterior distribution of our system as:

$$\begin{aligned} & P(Y, \lambda, \theta, \eta | X, Z) \\ &= \prod_{i,j,m,k} \left[ \left( \text{Pois}(z_{mk}^1; \theta_m \lambda_k) \text{Pois}(z_{ijm}^2; \eta X_{jim}) \right)^{Y_{ij,k}} \delta(z_{mk}^1 + z_{ijm}^2 - X_{ijm})^{Y_{ij,k}} \right] \\ & \cdot \text{Gam}(\eta; c, d) \prod_k \text{Gam}(\lambda_k; a_k, b_k) \prod_m \text{Gam}(\theta_m; \alpha_m, \beta_m) \prod_{i,j} \text{Cat}(Y_{ij}; p_{ij}) \end{aligned} \quad (21)$$

Note  $\delta$  represents the dirac delta distribution meaning it can only be a 0 or a 1. It acts as a check for whether we think the link exists or not.

During inference, we will be taking specific parts of this full posterior distribution and updating them individually while we fix the others. Once we have updated them all we then continue on to the ELBO method.

### 3.2.1 Updating Theta

We will now discuss the steps needed to update the latent variables. While the Supplementary Information provided by De Bacco et al. [5, SI] shows some steps, it is not always trivial to see the paths taken. As such, we will describe the steps where relevant and give more insight into what is happening within the formulae. At some points we will refer to De Bacco et al. [5], Safdari et al. [14] or Blei et al. [1] for their explanations of certain topics since they discuss certain things at length. Unless specified otherwise, any reference to the Supplementary Information (SI) will be based on De Bacco et al.'s paper.

Let us first consider the steps needed to update  $\theta_m$ . We are trying to find  $P(\theta_m)$  given the other set parameters. The relevant parts of Equation 21 are the Poisson and Gamma distributions, so we can say it is proportional to the Probability Density Function (PDF) of the Gamma times the PMF of the Poisson.

$$\begin{aligned} P(\theta_m | \theta_{\setminus \theta_m}, X, Z, Y, \lambda, \eta) &\propto \text{Gam}(\theta_m; \alpha_m, \beta_m) \prod_{i,j,k} \text{Pois}(z_{mk}^1; \theta_m \lambda_k)^{Y_{ij,k}} \\ &\propto (\theta_m)^{\alpha_m - 1} e^{-\beta_m \theta_m} \prod_{i,j,k} [(\theta_m \lambda_k)^{z_{mk}^1} e^{-\theta_m \lambda_k}]^{Y_{ij,k}} \end{aligned} \quad (22)$$

Note that the  $\frac{\beta_m^{\alpha_m}}{\Gamma(\alpha_m)}$  term is removed from the Gamma PDF formula and the division by  $z_{mk}^1!$  term is removed from the Poisson PMF formula because they are constants. Assuming  $k \in \{0, 1\}$  which is common in the testing done, the exponent  $Y_{ij,k}$  acts as an indicator to whether the link exists. It will either be to the power of 0 or 1.

From here, we can break it down as follows:

$$\theta_m^{\alpha_m - 1} \prod_{i,j,k} \theta_m^{z_{mk}^1 Y_{ij,k}} \propto \theta_m^{(\alpha_m + \sum_{i,j,k} Y_{ij,k} z_{mk}^1)} \quad (23)$$

We pull out the theta parts of the equation, and using product and exponential rules can rewrite it as such. The product becomes an exponential summation, and the  $Y_{ij,k}$  drops down since an exponent to an exponent is the same as multiplying the exponents. We are ignoring the  $-1$  from the alpha since we are dealing with a proportionality.

Next, we take the parts to the base  $e$ :

$$e^{-\beta_m \theta_m} \prod_{i,j,k} e^{-\theta_m \lambda_k Y_{ij,k}} = e^{-(\beta_m + \sum_{i,j,k} Y_{ij,k} \lambda_k) \theta_m} \quad (24)$$

Using similar product and exponential rules, we get this equation to the base  $e$ . We are left with:

$$\underbrace{\prod_{i,j,k} \lambda_k^{z_{mk}^1 Y_{ij,k}}}_{\text{constant}} \quad (25)$$

This product over lambda is not dependent on theta, and so will be fixed. This means we can consider it a constant and since this is a proportionality, we can ignore it.

Now, we can say Equation 22 is proportional to:

$$P(\theta_m | \theta_{\setminus \theta_m}, X, Z, Y, \lambda, \eta) \propto (\theta_m)^{\alpha_m - 1} e^{-\beta_m \theta_m} \prod_{i,j,k} [(\theta_m \lambda_k)^{z_{mk}^1} e^{-\theta_m \lambda_k}]^{Y_{ij,k}}$$

$$\sim \text{Gam} \left( \alpha_m + \sum_{i,j,k} Y_{ij,k} z_{mk}^1, \beta_m + \sum_{i,j,k} Y_{ij,k} \lambda_k \right) \quad (26)$$

We have taken the exponents of the bases  $\theta$  and  $e$  and used them as the shape and rate of the Gamma distribution. Thus, for the updating of theta, we can set the  $\gamma$  shape and rate to the following:

$$\gamma_m^{shape} = \alpha_m + \sum_{i,j,k} \rho_{ij,k} X_{ijm} \hat{z}_{mk}^1 \quad (27)$$

$$\gamma_m^{rate} = \beta_m + \sum_{i,j,k} \rho_{ij,k} \frac{\phi_k^{shape}}{\phi_k^{rate}} \quad (28)$$

We convert the  $Y_{ij,k}$  to our expectation of the link  $\rho_{ij,k}$ . Since  $Z$  is a Multinomial distribution,  $\hat{z}_{mk}^1$  represents the event probabilities. So multiplying it by  $X_{ijm}$  gives us the mean of  $z_{mk}^1$ . Also, note that De Bacco et al. assign  $\gamma$  to represent the shape and rate values for the prediction of  $\theta$ . Similarly,  $\phi$  represents  $\lambda$  as a parameter, and  $\nu$  represents  $\eta$ . The same steps can be done for updating  $\lambda$  and  $\eta$ . This leaves us with the following update formulae for  $\lambda$ :

$$\phi_k^{shape} = a_k + \sum_{i,j,m} \rho_{ij,k} X_{ijm} \hat{z}_{mk}^1 \quad (29)$$

$$\phi_k^{rate} = b_k + \sum_{i,j,m} \rho_{ij,k} \frac{\gamma_m^{shape}}{\gamma_m^{rate}} \quad (30)$$

and  $\eta$ :

$$\nu^{shape} = c + \sum_{i,j,m,k} \rho_{ij,k} X_{ijm} \hat{z}_{ijm}^2 \quad (31)$$

$$\nu^{rate} = d + \sum_{i,j,m,k} \rho_{ij,k} X_{jim} \quad (32)$$

### 3.2.2 Updating of Rho

Here we will discuss the steps needed to update Rho. For the updating of  $\rho$ , our expected distribution, we base it on our prediction of every tie  $Y_{ij}$  based on the full posterior distribution:

$$P(Y_{ij} | Y_{\setminus Y_{ij}}, X, Z, \lambda, \theta, \eta) \propto$$

$$\propto P(Y_{ij}) \prod_k \left[ \prod_m \left( \text{Pois}(z_{mk}^1; \theta_m \lambda_k) \text{Pois}(z_{ijm}^2; \eta X_{jim}) \right)^{Y_{ij,k}} \delta(z_{mk}^1 + z_{ijm}^2 - X_{ijm})^{Y_{ij,k}} \right] \quad (33)$$

We transform the categorical prior  $P(Y_{ij})$  into  $p_{ij,k}$ , the prior expectation of link  $ij$  assuming  $k$ , as described in De Bacco et al. [5, Equation 4].

$$\propto \prod_k \left[ p_{ij,k} \prod_m \left( \text{Pois}(z_{mk}^1; \theta_m \lambda_k) \text{Pois}(z_{ijm}^2; \eta X_{jim}) \right)^{Y_{ij,k}} \delta(z_{mk}^1 + z_{ijm}^2 - X_{ijm}) \right]^{Y_{ij,k}} \quad (34)$$

Thus our expectation of the link  $ij$  assuming  $k$  in our distribution is:

$$\rho_{ij,k} = p_{ij,k} \prod_m \text{Pois}(z_{mk}^1; \theta_m \lambda_k) \text{Pois}(z_{ijm}^2; \eta X_{jim}) \delta(z_{mk}^1 + z_{ijm}^2 - X_{ijm}) \quad (35)$$

Note this is similar to a Categorical distribution using  $Y_{ij}$  values as possible categories, and the probabilities as given by  $\rho_{ij}$ :

$$\sim \text{Cat}(Y_{ij}; \rho_{ij}) \quad (36)$$

Now we want to break Equation 35 from distributions into estimations. We do this by using the PMF of a Poisson distribution to predict. The PMF of our  $\text{Pois}(z_{mk}^1; \theta_m \lambda_k)$  is:

$$\frac{(\theta_m \lambda_k)^{z_{mk}^1} \cdot e^{-\theta_m \lambda_k}}{z_{mk}^1!} \quad (37)$$

Based on Blei et al.'s [1] work, we know that while the other parameters are fixed, we can approximate  $\rho_{ij,k}$  as proportional to the exponent of the log as we saw in Equation 15. So if we take the log of the PMF formula from Equation 37, we get:

$$z_{mk}^1 \cdot \log(\theta_m \lambda_k) - \theta_m \lambda_k - \log(z_{mk}^1!) \quad (38)$$

From the Supplementary Information [5, SI, Equation S13], we can rewrite the log probabilities of  $z_{mk}^1$  as:

$$\log \hat{z}_{mk}^1 = \mathbb{E}_{q(\theta_m)}[\log \theta_m] + \mathbb{E}_{q(\lambda_k)}[\log \lambda_k] \quad (39)$$

Recall that the mean of  $z_{mk}^1$  is found from  $X_{ijm} \hat{z}_{mk}^1$ . Transforming the product into a summation because of the exponent, and taking the log of Poisson distributions in the same manner as Equation 38, we rewrite  $\text{Pois}(z_{mk}^1; \theta_m \lambda_k)$  from Equation 35 as:

$$\propto \exp \left( \sum_m \left( X_{ijm} \hat{z}_{mk}^1 (\mathbb{E}_{q(\theta_m)}[\log \theta_m] + \mathbb{E}_{q(\lambda_k)}[\log \lambda_k]) \right) - \frac{\phi_k^{shape}}{\phi_k^{rate}} \sum_m \frac{\gamma_m^{shape}}{\gamma_m^{rate}} - \sum_m \mathbb{E}_{q(z_{mk}^1)}[\log z_{mk}^1!] \right) \quad (40)$$

Note the similarity of each part of this formula to Equation 38.

Now we will combine this with the mutuality  $\eta$  part of the equation. Using the same approach as Equation 40 and since the dirac delta drops away due to it being 0 or 1, we can define our expected distribution as:

$$\rho_{ij,k} \propto \exp \left\{ \begin{aligned} & \log p_{ij,k} + \sum_m \left( X_{ijm} \hat{z}_{mk}^1 (\mathbb{E}_{q(\theta_m)}[\log \theta_m] + \mathbb{E}_{q(\lambda_k)}[\log \lambda_k]) \right) \\ & + \sum_m X_{ijm} \hat{z}_{ijm}^2 \mathbb{E}_{q(\eta)}[\log \eta X_{jim}] \\ & - \frac{\phi_k^{shape}}{\phi_k^{rate}} \sum_m \frac{\gamma_m^{shape}}{\gamma_m^{rate}} - \frac{\nu^{shape}}{\nu^{rate}} \sum_m X_{jim} \\ & - \sum_m \mathbb{E}_{q(z_{mk}^1)}[\log z_{mk}^1!] - \sum_m \mathbb{E}_{q(z_{ijm}^2)}[\log z_{ijm}^2!] \end{aligned} \right\} \quad (41)$$

We have now reached the described equation in De Bacco et al. [5, SI, Equation S30]. It is taken one step further by normalising around  $k$ , and so we only need to consider terms dependent on  $k$ . This is described in the Supplementary Information [2, SI], and leaves us with the equation that we see in the actual paper, and the equation that we will use to update rho during inference:

$$\rho_{ij,k} \propto \exp \left\{ \log p_{ij,k} + \sum_m \left( X_{ijm} \hat{z}_{mk}^1 \mathbb{E}_{q(\lambda_k)}[\log \lambda_k] \right) - \frac{\phi_k^{shape}}{\phi_k^{rate}} \sum_m \frac{\gamma_m^{shape}}{\gamma_m^{rate}} \right\} \quad (42)$$

Note that De Bacco et al. include a multiplication by  $R_{ijm}$  which represents the check that reporter  $m$  reported on the link  $ij$ . However, within the paper, the SI, and the code, reporters report only on links that include themselves, and no others, so this is redundant and not shown in the SI.

### 3.2.3 Evidence Lower Bound Method (ELBO)

In order to determine whether the changes we make to the latent variables are improvements, we require a metric. As described we can compare two distributions with the Kullback-Leibler divergence [1], which takes a sample from a distribution  $P$ , and sees how different it is to what we would expect if we took a sample from distribution  $Q$ . While it is not officially a ‘metric’ by definition, it gives an adequate idea of how different two distributions are. The calculation for determining the KL divergence between distribution  $q(\epsilon)$  and  $p(\epsilon|X)$  is:

$$\text{KL}(q(\epsilon)||p(\epsilon|X)) = \mathbb{E}[\log q(\epsilon)] - \mathbb{E}[\log p(\epsilon|X)] \quad (43)$$

Thus we define the difference of the expectations of the logs of the two distributions as our metric. However, this is very costly to compute, so with steps described in Blei et al. we transform it into:

$$\text{ELBO}(q) = \mathbb{E}[\log p(\epsilon, X)] - \mathbb{E}[\log q(\epsilon)] \quad (44)$$

ELBO( $q$ ) functions as a negative KL divergence, and so by maximising the ELBO, it is equivalent to minimising the KL divergence. While we will not be describing these steps conceptually, instead we will be break down how De Bacco et al. reached their final formulae. The actual implementation of the ELBO method is rather straightforward and is basically a direct representation of the formulae in code form. As such, we will not describe the code itself, but rather how the equations were reached in the next section.

Effectively, what we are doing with the ELBO method is comparing our distribution  $q$  that we are updating against our expectation of the target distribution  $p$ . If there is a ‘large’ difference between this iteration and the last, it means we still have a lot to improve. However, if we do not change sufficiently we increase the ‘convergence’ counter. If this counter reaches an arbitrary number, the algorithm considers it to have converged and stops looking for a better result. However, if a significant change is made, the convergence counter is reset to 0 and we continue. This is to ensure that the algorithm only stops when the algorithm repeatedly fails to change the distribution. There is a set number of iterations that after which the algorithm will terminate even if there has been no convergence. Since there is variation in the starting parameters, the algorithm runs multiple times with different starting parameters to increase robustness.

## 3.3 ELBO equations

Here we will show the workings regarding the ELBO method, specifically from the SI [5, SI, Equation S32] through (S42). As described in Equation 44, the ELBO is defined as:

$$\text{ELBO}(q) = \mathbb{E}_q[\log \mathcal{L}(\lambda, \theta, \eta, Y)] - \mathbb{E}_q[\log q(\lambda, \theta, \eta, Y)] \quad (45)$$

Here, we are checking the progress of our inferred distribution  $q$  by subtracting the expectation of our inferred distribution  $q$  from the expectation of the original distribution  $\mathcal{L}$  which relied on the reports  $X$ . The first part  $\mathcal{L}(\lambda, \theta, \eta, Y]$  represents the full posterior distribution which is determined by our prior information as:

$$\begin{aligned} \mathcal{L}(\lambda, \theta, \eta, Y) = & \prod_{i,j} P(\{X_{ijm}\}_m | \{X_{jim}\}_m, Y_{ij}, \lambda, \{\theta_m\}_m, \eta) \cdot P(Y_{ij}; p_{ij}) \\ & \cdot \prod_k P(\lambda_k; a_k, b_k) \prod_m P(\theta_m; \alpha_m, \beta_m) P(\eta; c, d) \end{aligned} \quad (46)$$

The second part  $q(\lambda, \theta, \eta, Y)$  represents the variational parameters, which will be set to the priors at the beginning and adjusted throughout the algorithm. They are defined as:

$$q(\lambda, \theta, \eta, Y) = q(\eta; \nu^{shape}, \nu^{rate}) \prod_k q(\lambda_k; \phi_k^{shape}, \phi_k^{rate}) \prod_m q(\theta_m; \gamma^{shape}, \gamma^{rate}) \prod_{i,j} q(Y_{ij}; \rho_{ij}) \quad (47)$$

First, let us break down the posterior distribution  $\mathcal{L}$ . We use the inference with expectation maximisation technique from Safdari et al. [14], using their Equation 5 to find the log pseudolikelihood. This allows us to say that Equation 46 is proportional to:

$$\begin{aligned} \mathcal{L}(\lambda, \theta, \eta, Y) \propto & \mathbb{E}_q \left[ \sum_{i,j,m,k} Y_{ij,k} \left( X_{ijm} \log(\theta_m \lambda_k + \eta X_{jim}) - (\theta_m \lambda_k + \eta X_{jim}) \right) \right] \\ & + \mathbb{E}_q \left[ \sum_{i,j,k} Y_{ij,k} \log(p_{ij,k}) \right] \\ & + \mathbb{E}_q \left[ c \log(\eta) - d\eta + \sum_k (a_k \log(\lambda_k) - b_k \lambda_k) + \sum_m (\alpha_m \log(\theta_m) - \beta_m \theta_m) \right] \end{aligned} \quad (48)$$

For the latent parameters we update in Equation 47, we take the log of the PDF of the Gamma distributions. A PDF of  $q(\theta_m; \gamma^{shape}, \gamma^{rate})$  would look like:

$$f(\theta_m; \gamma^{shape}, \gamma^{rate}) = \frac{\gamma^{rate} \gamma^{shape}}{\Gamma(\gamma^{shape})} \cdot \theta_m^{\gamma^{shape}-1} \cdot e^{-\gamma^{rate} \theta_m} \quad (49)$$

Getting the log of this leads us to rewrite it as:

$$\gamma^{shape} \log \gamma^{rate} - \log(\Gamma(\gamma^{shape})) + \gamma^{shape} \log \theta_m - \gamma^{rate} \theta_m \quad (50)$$

Note the  $-1$  is again ignored. Doing this for each part of Equation 47 gives us:

$$\begin{aligned} q(\lambda, \theta, \eta, Y) \propto & \mathbb{E}_q \left[ \sum_{i,j,k} (Y_{ij,k} \log(\rho_{ij,k})) + \nu^{shape} \log(\nu^{rate}) - \log(\Gamma(\nu^{shape})) + \nu^{shape} \log \eta - \nu^{rate} \eta \right] \\ & + \mathbb{E}_q \left[ \sum_k (\phi_k^{shape} \log(\phi_k^{rate})) - \log(\Gamma(\phi_k^{shape})) + \phi_k^{shape} \log \lambda_k - \phi_k^{rate} \lambda_k \right] \\ & + \mathbb{E}_q \left[ \sum_m (\gamma_m^{shape} \log(\gamma_m^{rate})) - \log(\Gamma(\gamma_m^{shape})) + \gamma_m^{shape} \log \theta_m - \gamma_m^{rate} \theta_m \right] \end{aligned} \quad (51)$$

Subtracting Equation 51 from Equation 48 will leave us with the equation given in the SI [5, SI, Equation S33]:

$$\begin{aligned}
 \text{ELBO}(q) \propto & \\
 & \mathbb{E}_q \left[ \sum_{i,j,m,k} Y_{ij,k} \left( X_{ijm} \log(\theta_m \lambda_k + \eta X_{jim}) - (\theta_m \lambda_k + \eta X_{jim}) \right) \right] \\
 & + \mathbb{E}_q \left[ \sum_{i,j,k} Y_{ij,k} \log(p_{ij,k}) \right] \\
 & + \mathbb{E}_q \left[ c \log(\eta) - d\eta + \sum_{k,l} (a_k \log(\lambda_k) - b_k \lambda_k) + \sum_m (\alpha_{ml} \log(\theta_m) - \beta_{ml} \theta_m) \right] \\
 & - \mathbb{E}_q \left[ \sum_{i,j,k} (Y_{ij,k} \log(\rho_{ij,k})) + \nu^{shape} \log(\nu^{rate}) - \log(\Gamma(\nu^{shape})) + \nu^{shape} \log \eta - \nu^{rate} \eta \right] \\
 & - \mathbb{E}_q \left[ \sum_k (\phi_k^{shape} \log(\phi_k^{rate})) - \log(\Gamma(\phi_k^{shape})) + \phi_k^{shape} \log \lambda_k - \phi_k^{rate} \lambda_k \right] \\
 & - \mathbb{E}_q \left[ \sum_m (\gamma_m^{shape} \log(\gamma_m^{rate})) - \log(\Gamma(\gamma_m^{shape})) + \gamma_m^{shape} \log \theta_m - \gamma_m^{rate} \theta_m \right]
 \end{aligned} \tag{52}$$

Next, De Bacco et al. continue to break this equation down to remove potential constants. For these rules, refer to the SI [5, SI, Equation S34] through (S41). Taking the first line from Equation 52, we see that:

$$\begin{aligned}
 & \mathbb{E}_q \left[ \sum_{i,j,m,k} Y_{ij,k} \left( X_{ijm} \log(\theta_m \lambda_k + \eta X_{jim}) - (\theta_m \lambda_k + \eta X_{jim}) \right) \right] \\
 & = \sum_{i,j,m,k} \left[ \rho_{ij,k} \left( X_{ijm} \left( \mathbb{E}_q [\log(\theta_m \lambda_k)] + \mathbb{E}_q [\log(\eta X_{jim})] \right) - \left( \frac{\gamma_m^{shape}}{\gamma_m^{rate}} \frac{\phi_k^{shape}}{\phi_k^{rate}} + \frac{\nu^{shape}}{\nu^{rate}} X_{jim} \right) \right) \right]
 \end{aligned} \tag{53}$$

From lines 2 and 4 of Equation 52:

$$\begin{aligned}
 & \mathbb{E}_q \left[ \sum_{i,j,k} Y_{ij,k} \log(p_{ij,k}) \right] - \mathbb{E}_q \left[ \sum_{i,j,k} (Y_{ij,k} \log(\rho_{ij,k})) \right] \\
 & = \sum_{i,j,k} \left[ \rho_{ij,k} (\log(p_{ij,k}) - \log(\rho_{ij,k})) \right]
 \end{aligned} \tag{54}$$



From lines 3 and 4 of Equation 52:

$$\begin{aligned}
& \mathbb{E}_q \left[ c \log(\eta) - d\eta \right] - \mathbb{E}_q \left[ \nu^{shape} \log(\nu^{rate}) - \log(\Gamma(\nu^{shape})) + \nu^{shape} \log \eta - \nu^{rate} \eta \right] \\
&= c(\Psi(\nu^{shape}) - \log(\nu^{rate})) - d \left( \frac{\nu^{shape}}{\nu^{rate}} \right) - \nu^{shape} \log(\nu^{rate}) \\
&\quad + \log(\Gamma(\nu^{shape})) - \nu^{shape} (\Psi(\nu^{shape}) - \log(\nu^{rate})) + \nu^{rate} \left( \frac{\nu^{shape}}{\nu^{rate}} \right) \\
&= \Psi(\nu^{shape})(c - \nu^{shape}) + \log(\Gamma(\nu^{shape})) - c \log(\nu^{rate}) + \nu^{shape} \left( 1 - \frac{d}{\nu^{rate}} \right) \underbrace{- \nu^{shape} \log(\nu^{rate}) + \nu^{rate} \log(\nu^{rate})}_{\text{Cancels out}} \\
&= \Psi(\nu^{shape})(c - \nu^{shape}) + \log(\Gamma(\nu^{shape})) - c \log(\nu^{rate}) + \nu^{shape} \left( 1 - \frac{d}{\nu^{rate}} \right)
\end{aligned} \tag{55}$$

Where  $\Psi(x)$  represents the di-gamma function. We use it here to estimate the log of the variables. Similar to Equation 55, from lines 3 and 5 of Equation 52:

$$\begin{aligned}
& \mathbb{E}_q \left[ \sum_k (a_k \log(\lambda_k) - b_k \lambda_k) \right] - \mathbb{E}_q \left[ \sum_k (\phi_k^{shape} \log(\phi_k^{rate}) - \log(\Gamma(\phi_k^{shape})) + \phi_k^{shape} \log \lambda_k - \phi_k^{rate} \lambda_k) \right] \\
&= \sum_k \left[ \Psi(\phi_k^{shape})(a_k - \phi_k^{shape}) + \log(\Gamma(\phi_k^{shape})) - a_k \log(\phi_k^{rate}) + \phi_k^{rate} \left( 1 - \frac{b_k}{\phi_k^{rate}} \right) \right]
\end{aligned} \tag{56}$$

And finally from lines 3 and 6 of Equation 52:

$$\begin{aligned}
& \mathbb{E}_q \left[ \sum_m (\alpha_m \log(\theta_m) - \beta_m \theta_m) \right] - \mathbb{E}_q \left[ \sum_m (\gamma_m^{shape} \log(\gamma_m^{rate}) - \log(\Gamma(\gamma_m^{shape})) + \gamma_m^{shape} \log \theta_m - \gamma_m^{rate} \theta_m) \right] \\
&= \sum_m \left[ \Psi(\gamma_m^{shape})(\alpha_m - \gamma_m^{shape}) + \log(\Gamma(\gamma_m^{shape})) - \alpha_m \log(\gamma_m^{rate}) + \gamma_m^{rate} \left( 1 - \frac{\beta_m}{\gamma_m^{rate}} \right) \right]
\end{aligned} \tag{57}$$

All of these formulae from Equation 53 through Equation 57 combine to give us the equation as given in the SI [5, SI, Equation S42]:

ELBO( $q$ )  $\propto$

$$\begin{aligned}
& \sum_{i,j,m,k} \left[ \rho_{ij,k} \left( X_{ijm} (\mathbb{E}_q[\log(\theta_m \lambda_k)] + \mathbb{E}_q[\log(\eta X_{jim})]) - \left( \frac{\gamma_m^{shape}}{\gamma_m^{rate}} \frac{\phi_k^{shape}}{\phi_k^{rate}} + \frac{\nu^{shape}}{\nu^{rate}} X_{jim} \right) \right) \right] \\
&+ \Psi(\nu^{shape})(c - \nu^{shape}) + \log(\Gamma(\nu^{shape})) - c \log(\nu^{rate}) + \nu^{shape} \left( 1 - \frac{d}{\nu^{rate}} \right) \\
&+ \sum_k \left[ \Psi(\phi_k^{shape})(a_k - \phi_k^{shape}) + \log(\Gamma(\phi_k^{shape})) - a_k \log(\phi_k^{rate}) + \phi_k^{rate} \left( 1 - \frac{b_k}{\phi_k^{rate}} \right) \right] \\
&+ \sum_m \left[ \Psi(\gamma_m^{shape})(\alpha_m - \gamma_m^{shape}) + \log(\Gamma(\gamma_m^{shape})) - \alpha_m \log(\gamma_m^{rate}) + \gamma_m^{rate} \left( 1 - \frac{\beta_m}{\gamma_m^{rate}} \right) \right] \\
&+ \sum_{i,j,k} [\rho_{ij,k} (\log(p_{ij,k}) - \log(\rho_{ij,k}))]
\end{aligned} \tag{58}$$

After each iteration of inference, this calculation is done and the result will be the difference between the current distribution and the original estimate.

## 4 Methodology

We now alter the VIMuRe algorithm. The aim is to explore the impact of an alternate variable on the efficacy of the algorithm, but also to show the steps required to adapt the algorithm.

We investigate the impact of income bias on social ties. We do this by adding two parameters to VIMuRe, used in the reporting and inference stages. An income parameter for every node, which acts as a prior constant during an experiment that is available to the algorithm at all times. The other parameter is income bias  $\tau$ , which acts as a latent variable. That is, every node has a unique income bias value and we attempt to infer it alongside the other latent variables. The altered formula for link generation becomes:

$$\mathbb{E}[X_{ijm}|Y_{ij} = k] = \theta_m \lambda_k + \eta X_{jim} + \tau_m \chi_{ij} \quad (59)$$

where  $\tau_m$  is the income bias of reporter  $m$ , and  $\chi_{ij}$  represents the difference in income between  $i$  and  $j$ . Every node receives an income which is held constant throughout the test, and is accessible by the algorithm in the inference step.

The new latent variable being introduced is tau,  $\tau_m$ , which represents the bias of a reporter about the income of the nodes they are reporting on. So, if reporter  $m$  has a high bias, and difference in income between  $i$  and  $j$  is relatively low, reporter  $m$  will be more likely to report the link. Since the income matrix was always positive, the income bias was always a positive factor added on to Equation 59, so it would always increase the likelihood of a link being reported. A highly biased reporter discussing a link between people with a low difference in income would receive a boost, whereas a reporter with low bias discussing a highly varied income pair would have virtually no difference.

### 4.1 Network and Reporting Generation

The first step for our system of VIMuRe is to generate the income of every node, and the income matrix, showing the difference between each node. Relative to the Network Generation described in Section 3, there are not too many changes. Tau represents a bias of the reporter, so it does not make sense for it to influence the ground truth network  $Y$ . It would be interesting to have the income of each individual influence the SBM; however, this was considered beyond the scope of this project. As such, we will only alter the generation of the reported network  $X$ . To do this, we have posited the new expected value of any given link as seen in Equation 59. This is straightforward as we already have the necessary parameters  $i, j, m$  from reliability and mutuality.

When incorporating mutuality, as discussed in Section 3, we require a marginal distribution. The marginal distribution had to be updated as follows:

$$\mu_{ijm} = \frac{\theta_m \lambda_k |Y_{ij}=k + \tau_m \chi_{ij} + \eta \theta_m \lambda_k |Y_{ji}=k + \eta \tau_m \chi_{ij}}{(1 - \eta^2)} \quad (60)$$

The steps for this calculation can be found in Appendix Subsection A.2. Recall that this equation is used to predict if there will be a link in the other direction when using mutuality. With the network  $X$  generated, we can proceed to the inference.

### 4.2 Inference

We can take the income matrix as a constant, and  $\tau$  becomes one of our latent variables. We treat it in the same manner as  $\theta$  or  $\nu$ , fixing it while adjusting other variables, and vice versa. In order

to achieve this, some equations need to be rewritten, and some functions have to be created and altered. The full steps taken to reach these equations will be within the Appendix, but they are quite similar to the descriptions within Subsubsection 3.2.1. As such, we will only be showing the final equations here.

As we are adding a latent variable, the addition of a third auxiliary variable  $z^3$  is required.  $z^3$  will represent  $\tau_m \chi_{ij}$ . We now define the discrete latent auxiliary variables as:

$$z_{mk}^1 \sim \text{Pois}(\theta_m \lambda_k), \quad z_{ijm}^2 \sim \text{Pois}(\eta X_{jim}), \quad z_{ijm}^3 \sim \text{Pois}(\tau_m \chi_{ij}), \quad s.t. \quad z_{mk}^1 + z_{mk}^2 + z_{ijm}^3 = X_{ijm} | Y_{ij} = k. \quad (61)$$

The full posterior of the new system is:

$$\begin{aligned} P(Y, \lambda, \theta, \eta | X, Z) &= \prod_{i,j,m,k} \left[ \left( \text{Pois}(z_{mk}^1; \theta_m \lambda_k) \text{Pois}(z_{ijm}^2; \eta X_{jim}) \underbrace{\text{Pois}(z_{ijm}^3; \tau_m \chi_{ij})}_{\text{new}} \right)^{Y_{ij,k}} \delta(z_{mk}^1 + z_{ijm}^2 + z_{ijm}^3 - X_{ijm})^{Y_{ij,k}} \right] \\ &\times \text{Gam}(\eta; c, d) \prod_k \text{Gam}(\lambda_k; a_k, b_k) \prod_m \text{Gam}(\theta_m; \alpha_m, \beta_m) \underbrace{\prod_m \text{Gam}(\tau_m; f_m, g_m)}_{\text{new}} \prod_{i,j} \text{Cat}(Y_{ij}; p_{ij}) \end{aligned} \quad (62)$$

When we want to update  $\tau$ , the complete conditional for  $\tau$  becomes:

$$P(\tau_m | \tau_{\setminus \tau_m}, X, Z, Y, \theta, \eta) \propto \text{Gam}(\tau_m; f_m, g_m) \prod_{i,j,m,k} \text{Pois}(z_{ijm}^3; \tau_m \chi_{ij})^{Y_{ij,k}} \quad (63)$$

As described in Subsection 3.2, we write this conditional as:

$$\sim \text{Gam} \left( f_m + \sum_{i,j,m} Y_{ij,k} z_{ijm}^3, g_m + \sum_{i,j,k} Y_{ij,k} \chi_{ij} \right) \quad (64)$$

As with the other latent variables, we will use another letter,  $\xi$ , to define the shape and rate parameters of the Gamma distribution for  $\tau$ . The updating of these break down to:

$$\xi^{shape} = f_m + \sum_{i,j,m} \rho_{ij,k} X_{ijm} \hat{z}_{ijm}^3 \quad (65)$$

$$\xi^{rate} = g_m + \sum_{i,j,k} \rho_{ij,k} \chi_{ij} \quad (66)$$

Note that since the summation over  $\rho_{ij,k}$  will always sum to 1, and the sum over  $\chi$  is a constant, we do not have to update it every time, but can leave  $\xi^{rate}$  as a constant. For the updating of  $\rho$  as in Equation 42, we normalise around  $k$ , removing everything that is not dependent on it. Since  $\tau_m$  is based on reporter  $m$  and the income difference between  $i \rightarrow j$ , it is not dependent on  $k$ , and we can remove it as a constant. We refer to the Appendix Subsection A.1 for details.

### 4.3 ELBO

Adjustments to the ELBO method were relatively straightforward. As with the inference equations, they can be found in the Appendix Subsection A.3, and the final product is as follows:

Algorithm	$\tau_m \chi_{ij}$ used in Reporting	$\tau_m \chi_{ij}$ used in Inference
Origin	×	×
Bias	✓	✓
MixSynth	✓	×
MixInfer	×	✓

■ **Table 1** The different variations of the algorithm that were tested

$$\begin{aligned}
\text{ELBO}(q) \propto & \\
& \sum_{i,j,m,k} \left[ \rho_{ij,k} \left( X_{ijm} (\mathbb{E}_q[\log(\theta_m \lambda_k)] + \underbrace{\mathbb{E}_q[\log(\tau_m \chi_{ij})]}_{\text{new}} + \mathbb{E}_q[\log(\eta X_{jim})]) \right. \right. \\
& \quad \left. \left. - \left( \frac{\gamma_m^{shape}}{\gamma_m^{rate}} \frac{\phi_k^{shape}}{\phi_k^{rate}} + \frac{\nu^{shape}}{\nu^{rate}} X_{jim} + \underbrace{\frac{\xi^{shape}}{\xi^{rate}} \chi_{ij}}_{\text{new}} \right) \right) \right] \\
& + \Psi(\nu^{shape})(c - \nu^{shape}) + \log(\Gamma(\nu^{shape})) - c \log(\nu^{rate}) + \nu^{shape} \left( 1 - \frac{d}{\nu^{rate}} \right) \\
& + \sum_k \left[ \Psi(\phi_k^{shape})(a_{kl} - \phi_k^{shape}) + \log(\Gamma(\phi_k^{shape})) - a_{kl} \log(\phi_k^{rate}) + \phi_k^{rate} \left( 1 - \frac{b_{kl}}{\phi_k^{rate}} \right) \right] \\
& + \sum_m \left[ \Psi(\gamma_m^{shape})(\alpha_{ml} - \gamma_m^{shape}) + \log(\Gamma(\gamma_m^{shape})) - \alpha_{ml} \log(\gamma_m^{rate}) + \gamma_m^{rate} \left( 1 - \frac{\beta_{ml}}{\gamma_m^{rate}} \right) \right] \\
& + \underbrace{\sum_m \left[ \Psi(\xi_m^{shape})(f_{ml} - \xi_m^{shape}) + \log(\Gamma(\xi_m^{shape})) - f_{ml} \log(\xi_m^{rate}) + \xi_m^{rate} \left( 1 - \frac{g_{ml}}{\xi_m^{rate}} \right) \right]}_{\text{new}} \\
& + \sum_{i,j,k} [\rho_{ij,k} (\log(p_{ij,k}) - \log(\rho_{ij,k}))]
\end{aligned} \tag{67}$$

## 5 Experimental Setup

Four separate model variations of the algorithm were tested for this experiment as can be seen in Table 1. Origin, the original algorithm by De Bacco et al. [5], unaltered. The second variation is called Bias, and represents  $\tau$  bias in the Network Generation as well as Inference steps. Thirdly, is MixSynth which mixes both of the above. It uses the  $\tau$  bias to generate the reported network  $X$ , but then the original inference method is used. This is to show if the inclusion of  $\tau$  on the generation has an overly negative impact on the inference. The final variation is MixInfer, in which the network generation will be the same as the original, but the inference of the network will be done using  $\tau$ . This is to see the impact of the bias when no bias is present in the generation.

The first test was comparing the Bias model against the MixSynth model. The goal here is to observe the influence  $\tau$  has on the inference and reconstruction of the network, while having the same reported network. The next test was done comparing the Origin model against the MixInfer model. This comparison had both models not utilise  $\tau$  for the generation of the reported network, but MixInfer would attempt to infer the network using  $\tau$ . From this, we aim to see if there was an overly negative effect from the inclusion of  $\tau$  in the inference phase.

For network generation, we used the Multitensor version of SBM, as described by De Bacco et al. [2]. This was left unchanged since the income bias only affects the reporters. The income values for every member were determined by using a study done in the Netherlands in 2022 by the Central Bureau of Statistics [9]. The data used had incomes per household from 0K to 100K+ per year, with 7912 participants. We fit this data to a lognormal distribution, and then pulled from the distribution to randomly assign each node an income. The distance matrix,  $\chi_{ij}$  for the incomes was done using the following formula:

$$\chi_{ij} = | \log(\text{income}[i]) - \log(\text{income}[j]) | \quad (68)$$

Note that log to the base 10 was used. The reason for the difference of logs instead of the absolute difference is that we wanted there to be a relative difference. So a difference between an income of 10 and 20, is greater than the difference between 90 and 100. Finally, we adjusted the final matrix  $\chi_{ij}^*$  as follows:

$$\chi_{ij}^* = \left( \frac{1}{\chi_{ij} + 0.1} \right) \quad (69)$$

We wanted the matrix at this point to always be a positive value within the range  $\approx 0.5 \rightarrow 10$ . This was done to ensure a minimum difference to affect the model. With the largest possible difference, 1 and 100, this will result in a log difference of 2, so the updated matrix difference will be  $\frac{1}{2.1} \approx 0.476$ . If there is no difference between the incomes, the updated matrix will be  $\frac{1}{0.1} = 10$ .

When using  $\tau$  bias for the generation,  $\tau$  would always increase the probability of reporting a link. We did not wish for  $\tau$  to be overpowering, and instead to act as a slight adjustment to the existing algorithm. To generate the bias of the reporters, we pulled them from a Gamma distribution with a shape of 2.0 and a scale of 0.5. This gives an average value of 1, and virtually all values between  $0 \rightarrow 4$ . It was noted that when the income values were not scaled down, the  $\xi^{rate}$  which was based on the summation over  $\chi$  was far too high, and this led to the impact of  $\tau$  being negligible. This was because the  $\xi^{shape}$ , which begins inference around 0.1, was being divided by a number close to 3,500. Sensitivity testing was done in order to find a range that allowed for  $\tau$  to have some impact without overpowering the model. This led to dividing the  $\chi$  values by one million to obtain more reasonable numbers. It would be interesting to examine how adjusting other parameters could lead to a more fair distribution, but there was insufficient time in this project. Given more time, it would be interesting to experiment with the  $\theta_m$  values being similarly distributed to  $\tau_m$ .

For the experiments ran in this project, we opted to simulate De Bacco et al.'s approach to their Over and Under-Reporter experiment. They take a more extreme approach to reporters, building a custom theta, setting  $\theta$  of over-reporters to 50 and under-reporters to 0.5. Additionally, there are only under-reporters, or over-reporters at one time, so the algorithm is run once with each type. While it may be more realistic for the  $\theta$  values to be determined from a Gamma distribution like  $\tau$ , we felt that simulating the existing work would be more informative. 25 different seeds were used for  $X$  generation from the same  $Y$  ground truth to improve robustness. The same set of tests were ran with mutuality enabled with a  $\eta$  value of 0.2.

For the comparison of the inferred network  $\rho$  against the ground truth  $Y$ , we also used the same method as De Bacco et al. in the over- and under-reporter experiment. That is to use the ski-kit GridSearch class [3]. When attempting to reconstruct the network using our prediction  $\rho$ , we used a threshold value of 0.01. This means that any link  $i \rightarrow j$  with a higher prediction value than 0.01 would be set as existing. However, when we included mutuality, we adopted the approach recommended by De Bacco et al. and added the estimated  $\eta$  value divided by 3 to the threshold. Since mutuality adds to the probability of a link existing, the network will naturally

be more populated. Adjusting the threshold like this acts as a way to make the algorithm more selective. The base value of 0.01 seems very low, but it is the same as used by De Bacco et al. in this experiment that we are imitating. When we adjusted this value to other values suggested, it did not yield significant improvements. Once the predicted network is generated, we compare it with the ground truth using F1-score as a metric. To compare the F1-scores against each other, we used a Mixed Linear Model Regression to see if there was a significant difference.

## 5.1 Time Complexity

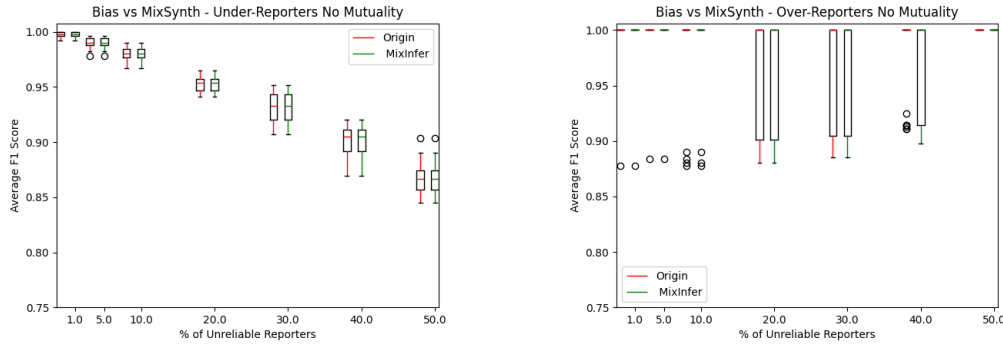
While some experiments were done by De Bacco et al. regarding time complexity, they were only shown in the Supplementary Information and not discussed [5, SI, Figure S5]. As such, we will test the average running time for one cycle of CAVI using different network sizes. We assume that all members of a network report on the network. When examining the code, we determined that the `_update_phi` and `_update_rho` functions were the most expensive. We take  $M$  to be the total number of reporters, and  $W$  to be the total number of reports made. While  $M$  is something that we input,  $W$  is something we don't control, and is dependent on the seed of the reported network, as well as factors such as the amount of over-reporting nodes. Since every reporter is very likely to report on at least one tie, we can assume  $W > M$ , however the extent to which it is greater could vary greatly. As such, it is difficult to make any definite claims without extensive testing. Nevertheless, we have done some testing to observe the total time taken by one round of inference. That is, given a reported network  $X$ , the time required for the algorithm to fully run the CAVI method. Within the CAVI method, `_update_phi` has an einsum with  $W^2$  computations, which we sum over leaving us with  $O(W^2)$  time. `_update_rho` has an einsum with  $M^3$  computations, which we sum over bringing us to  $O(M^3)$  time. There are other constants but we ignore them since they are relatively insignificant.

When testing the algorithm, we used the same seed for the ground truth  $Y$ , but for the reported network  $X$  we used 10 different seeds. We observed  $N$  values of  $\{50, 100, 200, 300, 500\}$ . We observed  $\theta$  generation percentage values of  $\{0.01, 0.02, 0.03, 0.05, 0.08, 0.10, 0.15, 0.20, 0.30, 0.40, 0.50\}$ . The runtime will be determined as the duration of the `_update_CAVI` method. This method is run many times until convergence is reached, and the average time of these runs were taken. This was done 10 times and the average over the 10 runs was used for the results. The computer running the tests was using an AMD Ryzen 7 4800H, 2.9Mhz with 8 cores (16 logical processors).

## 6 Results

### 6.1 Bias Versus MixSynth

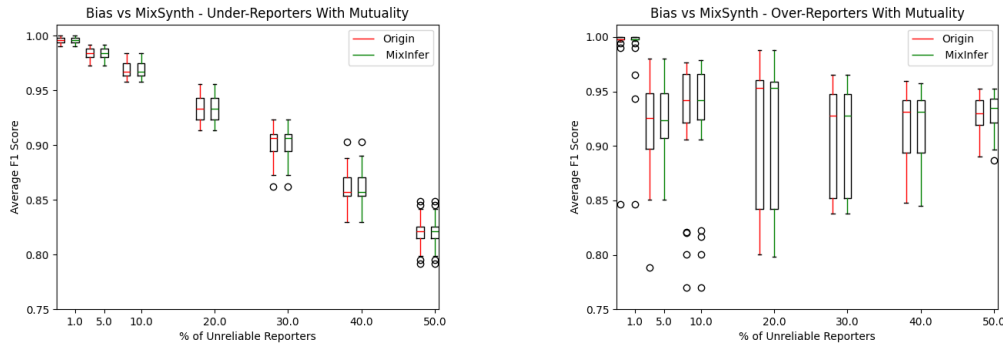
We use F1-scores to determine how well the algorithm was able to reconstruct the hidden network  $Y$  using  $X$ . These figures display boxplots showing the median, the upper and lower interquartile ranges. The highest and lowest values within 1.5 times the interquartile ranges are displayed by the caps. Any results outside this range is presented as a circle and is considered an outlier. In the under-reporter setting we can see in Figure 2a and Figure 3a, there is little difference between the Bias model and the MixSynth model. The difference was confirmed not significant using a linear mixed model. In the case of the over-reporters, there is again not a significant difference. However, in Figure 2b and Figure 3b we can see that in the case of extreme over-reporting there is a much higher variance in the range of responses. Nevertheless, the model still performs well in reconstructing the network. This is likely due to the nature of the theta generation, with there being a much greater difference between the  $\theta_m$  values of 50 for over-reporters relative to the  $\theta_m$  values of 0.5 for under-reporters. Since there is no significant difference in these comparisons,



(a) Under-reporters with no mutuality.

(b) Over-reporters with no mutuality.

■ **Figure 2** A comparison between the Bias method and the MixSynth method. The same ground truth network of  $N = 100$  was used, with all members reporting on only their own links. 25 different reported networks used for inference for each F1-score, from which the average was taken.



(a) Under-reporters with mutuality.

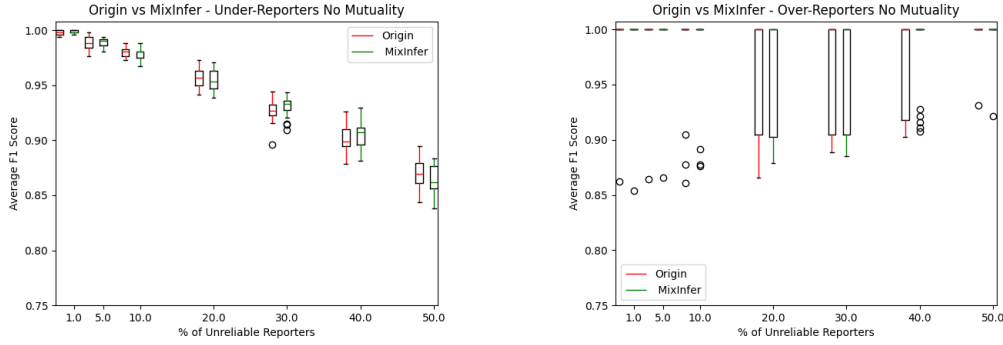
(b) Over-reporters with mutuality.

■ **Figure 3** A comparison between the Bias method and the MixSynth method. The same ground truth network of  $N = 100$  was used, with all members reporting on only their own links. 25 different reported networks used for inference for each F1-score, from which the average was taken. This model included mutuality, using a base  $\eta$  value of 0.2.

perhaps the influence of  $\tau_m$  on the report generation is too low. The inference of MixSynth is done without using  $\tau$  and it was still able to infer the network to the same level. It is interesting that in the case of over-reporters, the model does not seem to strongly depend on the percentage of unreliable reporters. It is likely due to the fact that the algorithm can more freely mark nodes as over-reporters, leading it to good results with lower variance at 50%.

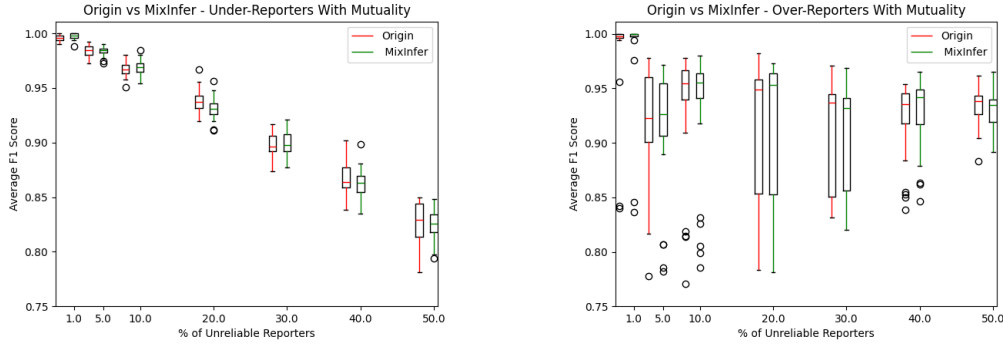
## 6.2 Origin Versus MixInfer

In Figure 4 and Figure 5, we again use F1-scores to judge the success of the network reconstruction. Neither under nor over-reporting instances showed significant differences using a linear mixed model. We do see the same pattern as in the other comparison, where the over-reporter model seems to be less strongly influenced by the percentage of reporters. In contrast to Figure 4a and Figure 5a, we see a particularly large variance in Figure 5b when compared to Figure 4b. It is worth noting that every model performs significantly worse when mutuality was included according to the linear mixed model. While the previous experiments tells us that  $\tau$  does not



(a) Under-reporters with no mutuality. (b) Over-reporters with no mutuality.

■ **Figure 4** A comparison between the Origin method and the MixInfer method. The same ground truth network of  $N = 100$  was used, with all members reporting on only their own links. 25 different reported networks used for inference for each F1-score, from which the average was taken.



(a) Under-reporters with mutuality. (b) Over-reporters with mutuality.

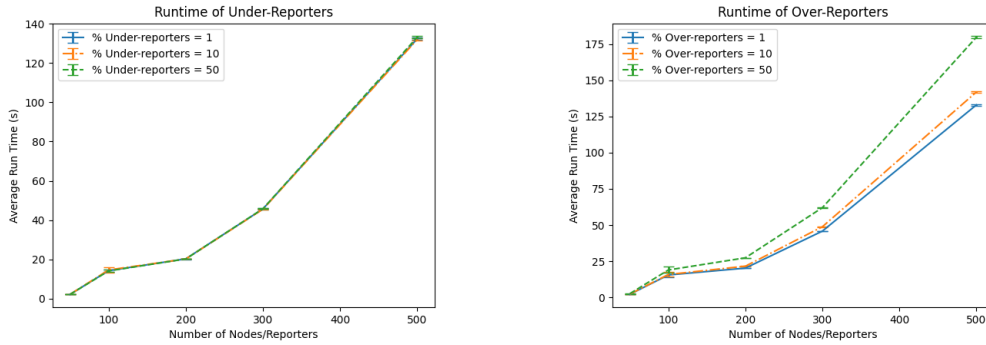
■ **Figure 5** A comparison between the Origin method and the MixInfer method. The same ground truth network of  $N = 100$  was used, with all members reporting on only their own links. 25 different reported networks used for inference for each F1-score, from which the average was taken.

seem to have a large impact on the generation, here we see that it does not seem to negatively impact the inference stage. The reported networks of both models were built without  $\tau$ , but the MixInfer model still performs as well as the Origin. This speaks to the strength of the original algorithm created by De Bacco et al..

### 6.3 Runtime Tests

For the runtime tests, only three theta generation percentages are shown in Figure 6 for clarity. We show the lowest, highest and the median. It is not surprising that as the number of over-reporters grows in Figure 6b, the longer the algorithm takes. This means that there are more reports, and the algorithm scales with  $O(W^2)$  as discussed previously. In contrast, Figure 6a shows virtually no difference in time taken. This is likely due to the fact that over-reporters report 50 times more than an average reporter, while under-reporters only report half as much. So, as the network grows, there is a large increase in time required for the over-reporting model.





(a) Theta parameters set to under-reporting.

(b) Theta parameters set to over-reporting.

**Figure 6** The runtime of the `_update_CAVI` method over a number of network sizes. The average over 10 runs is shown with standard deviations. The three lines indicate the % of reporters that exaggerate, taking the lowest, highest, and the median.

## 7 Discussion

The fact that the addition of another parameter, notably one that relies on reliable information, was not immediately beneficial is a testament to the original VIMuRe algorithm. While we do believe that with adjustments and time, there is certainly potential to improve upon our variation, it is important to appreciate the core algorithm's efficacy. It is apparent that the model does not take  $\tau$  into enough consideration in the report generation or the inference phases. One hypothesis we have is that since  $\tau_m$  is only dependent on the reporter, and not on the actual expected network, it is less influential. For example, in the reporting phase,  $\theta_m$  is multiplied by  $\lambda_k$ , which means that if the link exists in the real network,  $\theta$  will be multiplied by 1, and if it does not exist in the real network, it is multiplied by 0.01. There is a similar effect in the inference process but taking the expected network into consideration. So when  $\tau$  increases the probability of a link, it has a much larger effect on links that we do not think exist. This means  $\tau$  may be doing more damage than good, and the algorithm is thus doing its best to ignore it. It would be interesting to explore a variation of the model where  $\tau$  factors in the expectation of the network.

It would also be interesting to investigate a  $\tau$  generation approach similar to De Bacco's et al.'s way of handling reliability. That is, to set a variable for the percentage of over- or under-biased reporters, and then give each reporter of those category the same value. Alternatively, to have the  $\theta$  values distributed in a similar manner to  $\tau$ , from a Gamma distribution. While mutuality did reduce the efficacy of a model, it did so to every version, and no significant differences were found between the models when they had the same mutuality parameters. We hypothesise that the mutuality performing worse in the case of over-reporters is linked to the increased number of reports causing the mutuality to play a bigger role than in the ground truth.

In both comparisons, we see that as the percentage of over-reporters grows, the algorithm does not seem to perform much worse. This is interesting as the algorithm seems to have an easier time determining over-reporters when there are more of them. It realises that a significant portion of the reporters are giving false information and can use this to do a better job of reconstructing the network. If nothing else, this speaks to the potential of VIMuRe.

In the runtime analysis, it is hard to pinpoint exactly how much the number of reports versus the number of reporters ties into the time required. It would be very interesting in future to track the number of reports made and use them to more clearly assign values for the projected runtime.

## 8 Conclusion

While the experiments were inconclusive with regard to improving the VIMuRe algorithm, the goal of this thesis was to break down the VIMuRe algorithm and to show how it could be altered. We hope that the work done will make this process easier for others who are looking to use Variational Inference. Network Science is growing rapidly, so it is the most important time to make sure that documentation is sufficient, and that other faculties of science can easily join and learn the intricacies of the field. In particular, Cognitive Social Structures are on the rise with Redhead et al. and De Bacco et al.'s work. It is worth noting that as this thesis is being finished, De Bacco et al. are still updating VIMuRe and their tutorials for it. It is easier than ever to use their algorithm.

Moving forward, the addition of any information, for example covariates on nodes, would be an interesting way to build upon VIMuRe. If each node had additional parameters, it would be possible to incorporate other version of SBM, such as one utilising a Social Relations model as in Redhead et al. [13]. With these models, the variance in link generation can be come very interesting. The VIMuRe model can also be made a lot more complex if reporters could be made report on links other than their own, which would lead to a large influx of reported data. We hope that with the information in this paper, others are encouraged to use VIMuRe and other types of Variational Inference, correctly accounting for errors, and doing the science right.

---

References

---

- 1 David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017. [arXiv:1601.00670\[cs,stat\]](https://arxiv.org/abs/1601.00670), [doi:10.1080/01621459.2017.1285773](https://doi.org/10.1080/01621459.2017.1285773).
- 2 Raina A. Brands. Cognitive social structures in social network research: A review. *Journal of Organizational Behavior*, 34:S82–S103, 2013. [doi:10.1002/job.1890](https://doi.org/10.1002/job.1890).
- 3 Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013. [doi:10.48550/arXiv.1309.0238](https://doi.org/10.48550/arXiv.1309.0238).
- 4 Aaron Clauset, Cristopher Moore, and M. E. J. Newman. Hierarchical structure and the prediction of missing links in networks. *Nature*, 453(7191):98–101, 2008. [doi:10.1038/nature06830](https://doi.org/10.1038/nature06830).
- 5 Caterina De Bacco, Martina Contisciani, Jonathan Cardoso-Silva, Hadiseh Safdari, Gabriela Lima Borges, Diego Baptista, Tracy Sweet, Jean-Gabriel Young, Jeremy Koster, Cody T Ross, Richard McElreath, Daniel Redhead, and Eleanor A Power. Latent network models to account for noisy, multiply reported social network data. 186(3):355–375, 2023. [doi:10.1093/jrsssa/qnac004](https://doi.org/10.1093/jrsssa/qnac004).
- 6 Roger Guimerà and Marta Sales-Pardo. Missing and spurious interactions and the reconstruction of complex networks. *Proceedings of the National Academy of Sciences*, 106(52):22073–22078, 2009. [doi:10.1073/pnas.0908366106](https://doi.org/10.1073/pnas.0908366106).
- 7 Clement Lee and Darren J. Wilkinson. A review of stochastic block models and extensions for graph clustering. *Applied Network Science*, 4(1):122, 2019. [doi:10.1007/s41109-019-0232-2](https://doi.org/10.1007/s41109-019-0232-2).
- 8 Peter V. Marsden. Network data and measurement. *Annual Review of Sociology*, 16(1):435–463, 1990. [doi:10.1146/annurev.so.16.080190.002251](https://doi.org/10.1146/annurev.so.16.080190.002251).
- 9 Statistics Netherlands. Income distribution (standardised income). Last Modified: 08-12-2023T12:08:42. URL: <https://www.cbs.nl/en-gb/visualisations/income-distribution>.
- 10 Duy Nguyen. An in depth introduction to variational bayes note. *SSRN Electronic Journal*, 2023. [doi:10.2139/ssrn.4541076](https://doi.org/10.2139/ssrn.4541076).
- 11 Leto Peel, Tiago P. Peixoto, and Manlio De Domenico. Statistical inference links data and theory in network science. *Nature Communications*, 13:6794, November 2022. [doi:10.1038/s41467-022-34267-9](https://doi.org/10.1038/s41467-022-34267-9).
- 12 Tiago P. Peixoto. Reconstructing networks with unknown and heterogeneous errors. *Physical Review X*, 8(4):041011, 2018. [doi:10.1103/PhysRevX.8.041011](https://doi.org/10.1103/PhysRevX.8.041011).
- 13 Daniel Redhead, Richard McElreath, and Cody T. Ross. Reliable network inference from unreliable data: A tutorial on latent network modeling using STRAND. *Psychological Methods*, 2023. [doi:10.1037/met0000519](https://doi.org/10.1037/met0000519).
- 14 Hadiseh Safdari, Martina Contisciani, and Caterina De Bacco. Generative model for reciprocity and community detection in networks. 3(2):023209. [doi:10.1103/PhysRevResearch.3.023209](https://doi.org/10.1103/PhysRevResearch.3.023209).
- 15 Brent Simpson, Barry Markovsky, and Mike Steketee. Power and the perception of social networks. *Social Networks*, 33(2):166–171, 2011. [doi:10.1016/j.socnet.2010.10.007](https://doi.org/10.1016/j.socnet.2010.10.007).
- 16 Edward Bishop Smith, Tanya Menon, and Leigh Thompson. Status differences in the cognitive activation of social networks. *Organization Science*, 23(1):67–82, 2012. [doi:10.1287/orsc.1100.0643](https://doi.org/10.1287/orsc.1100.0643).

## A Adapted Equations

### A.1 Updating Rho with $\tau$

$$P(Y_{ij}|Y_{\setminus Y_{ij}}, X, Z, \lambda, \theta, \underbrace{\tau}_{\text{new}}, \eta)$$

$$\propto P(Y_{ij}) \prod_k \left[ \prod_m \left( \text{Pois}(z_{mk}^1; \theta_m \lambda_k) \text{Pois}(z_{ijm}^2; \eta X_{jim}) \underbrace{\text{Pois}(z_{ijm}^3; \tau_m \chi_{ij})}_{\text{new}} \right)^{Y_{ij,k}} \delta(z_{mk}^1 + z_{ijm}^2 + \underbrace{z_{ijm}^3}_{\text{new}} - X_{ijm})^{Y_{ij,k}} \right] \quad (70)$$

Similar to what is done in Subsection 3.2, this ends up as:

$$\rho_{ij,k} \propto \exp \left\{ \begin{aligned} & \log p_{ij,k} + \sum_m \left( X_{ijm} \hat{z}_{mk}^1 (\mathbb{E}_{q(\theta_m)}[\log \theta_m] + \mathbb{E}_{q(\lambda_k)}[\log \lambda_k]) \right) \\ & + \sum_m X_{ijm} \hat{z}_{ijm}^2 \mathbb{E}_{q(\eta)}[\log \eta X_{jim}] + \underbrace{\sum_m X_{ijm} z_{ijm}^3 \mathbb{E}_{q(\tau_m)}[\log \tau_m \chi_{ij}]}_{\text{new}} \\ & - \frac{\phi_k^{\text{shape}}}{\phi_k^{\text{rate}}} \sum_m \frac{\gamma_m^{\text{shape}}}{\gamma_m^{\text{rate}}} - \frac{\nu^{\text{shape}}}{\nu^{\text{rate}}} \sum_m X_{jim} - \underbrace{\chi_{ij} \sum_m \frac{\xi^{\text{shape}}}{\xi^{\text{rate}}}}_{\text{new}} \\ & - \sum_m \mathbb{E}_{q(z_{mk}^1)}[\log z_{mk}^1!] - \sum_m \mathbb{E}_{q(z_{ijm}^2)}[\log z_{ijm}^2!] - \underbrace{\sum_m \mathbb{E}_{q(z_{ijm}^3)}[\log z_{ijm}^3!]}_{\text{new}} \end{aligned} \right\} \quad (71)$$

As described in the main text, none of these additions are dependent on  $k$ , and we normalise around  $k$ , so we do not need to change the  $\rho$  updating formula.

### A.2 Deriving the expected value of the Marginal Distribution with $\tau$

Taking  $\Theta$  to represent latent variables for the the link going in the other direction;

$$\begin{aligned} \mathbb{E}[X_{ijm}|Y_{ij=k}] &= \mu_{ijm} = \sum_{X_{ijm}, X_{jim}} X_{ijm} P(X_{jim}|\Theta) \\ &= \sum_{X_{jim}} P(X_{jim}|\Theta) \cdot \sum_{X_{ijm}} X_{ijm} P(X_{ijm}|X_{jim}, \Theta) \\ &= \sum_{X_{jim}} P(X_{jim}|\Theta) \cdot [\theta_m \lambda_k |_{Y_{ij=k}} + \underbrace{\tau_m \chi_{ij}}_{\text{new}} + \eta X_{jim}] \\ &= \theta_m \lambda_k |_{Y_{ij=k}} + \underbrace{\tau_m \chi_{ij}}_{\text{new}} + \eta \sum_{X_{jim}} X_{jim} P(X_{jim}|\Theta) \\ &= \theta_m \lambda_k |_{Y_{ij=k}} + \underbrace{\tau_m \chi_{ij}}_{\text{new}} + \eta \mu_{jim} \\ &= \theta_m \lambda_k |_{Y_{ij=k}} + \underbrace{\tau_m \chi_{ij}}_{\text{new}} + \eta (\theta_m \lambda_k |_{Y_{ji=k}} + \underbrace{\tau_m \chi_{ij}}_{\text{new}} + \eta \mu_{jim}) \end{aligned} \quad (72)$$

Solving for  $\mu_{ijm}$  yields:

$$\mu_{ijm} (1 - \eta^2) = (\theta_m \lambda_k |_{Y_{ij=k}} + \underbrace{\tau_m \chi_{ij}}_{\text{new}} + \eta \theta_m \lambda_k |_{Y_{ji=k}} + \underbrace{\eta \tau_m \chi_{ij}}_{\text{new}}) \quad (73)$$

Which gives:

$$\mu_{ijm} = \frac{(\theta_m \lambda_k |_{Y_{ij}=k} + \underbrace{\tau_m \chi_{ij}}_{\text{new}} + \eta \theta_m \lambda_k |_{Y_{ji}=k} + \underbrace{\eta \tau_m \chi_{ji}}_{\text{new}})}{(1 - \eta^2)} \quad (74)$$

### A.3 Updating ELBO with $\tau$

The ELBO formula is adapted to include  $\tau$  as:

$$\text{ELBO}(q) = \mathbb{E}_q[\log \mathcal{L}(\lambda, \theta, \underbrace{\tau}_{\text{new}}, \eta, Y)] - \mathbb{E}_q[\log q(\lambda, \theta, \underbrace{\tau}_{\text{new}}, \eta, Y)] \quad (75)$$

The Full Posterior Distribution based on our prior information changes to:

$$\begin{aligned} \mathcal{L}(\lambda, \theta, \eta, Y) &= \prod_{i,j} P(\{X_{ijm}\}_m | \{X_{jim}\}_m, Y_{ij}, \lambda, \{\theta_m\}_m, \underbrace{\{\tau_m\}_m}_{\text{new}}, \eta) \cdot P(Y_{ij}; p_{ij}) \\ &\cdot \prod_k P(\lambda_k; a_k, b_k) \prod_m P(\theta_m; \alpha_m, \beta_m) \underbrace{\prod_m P(\tau_m; f_m, g_m)}_{\text{new}} P(\eta; c, d) \end{aligned} \quad (76)$$

The variational parameters which we adjust throughout are:

$$q(\lambda, \theta, \eta, Y) = q(\eta; \nu^{\text{shape}}, \nu^{\text{rate}}) \prod_k q(\lambda_k; \phi_k^{\text{shape}}, \phi_k^{\text{rate}}) \prod_m q(\theta_m; \gamma^{\text{shape}}, \gamma^{\text{rate}}) \underbrace{\prod_m q(\tau_m; \xi^{\text{shape}}, \xi^{\text{rate}})}_{\text{new}} \prod_{i,j} q(Y_{ij}; \rho_{ij}) \quad (77)$$

In the same manner as described in Subsection 3.3, Equation 75 becomes:

$$\begin{aligned} \text{ELBO}(q) &\propto \\ &\mathbb{E}_q \left[ \sum_{i,j,m,k} Y_{ij,k} \left( X_{ijm} \log(\theta_m \lambda_k + \underbrace{\tau_m \chi_{ij}}_{\text{new}} + \eta X_{jim}) - (\theta_m \lambda_k + \underbrace{\tau_m \chi_{ij}}_{\text{new}} + \eta X_{jim}) \right) \right] \\ &+ \mathbb{E}_q \left[ \sum_{i,j,k} Y_{ij,k} \log(p_{ij,k}) \right] \\ &+ \mathbb{E}_q \left[ c \log(\eta) - d\eta + \sum_k (a_k \log(\lambda_k) - b_k \lambda_k) + \sum_m (\alpha_m \log(\theta_m) - \beta_m \theta_m) + \underbrace{\sum_m (f_m \log(\tau_m) - g_m \tau_m)}_{\text{new}} \right] \\ &- \mathbb{E}_q \left[ \sum_{i,j,k} (Y_{ij,k} \log(\rho_{ij,k})) + \nu^{\text{shape}} \log(\nu^{\text{rate}}) - \log(\Gamma(\nu^{\text{shape}})) + \nu^{\text{shape}} \log \eta - \nu^{\text{rate}} \eta \right] \\ &- \mathbb{E}_q \left[ \sum_k (\phi_k^{\text{shape}} \log(\phi_k^{\text{rate}})) - \log(\Gamma(\phi_k^{\text{shape}})) + \phi_k^{\text{shape}} \log \lambda_k - \phi_k^{\text{rate}} \lambda_k \right] \\ &- \mathbb{E}_q \left[ \sum_m (\gamma_m^{\text{shape}} \log(\gamma_m^{\text{rate}})) - \log(\Gamma(\gamma_m^{\text{shape}})) + \gamma_m^{\text{shape}} \log \theta_m - \gamma_m^{\text{rate}} \theta_m \right] \\ &- \underbrace{\mathbb{E}_q \left[ \sum_m (\xi_m^{\text{shape}} \log(\xi_m^{\text{rate}})) - \log(\Gamma(\xi_m^{\text{shape}})) + \xi_m^{\text{shape}} \log \tau_m - \xi_m^{\text{rate}} \tau_m \right]}_{\text{new}} \end{aligned} \quad (78)$$

Finally, this converts to what we see in Equation 67:

$$\begin{aligned}
\text{ELBO}(q) \propto & \\
& \sum_{i,j,m,k} \left[ \rho_{ij,k} \left( X_{ijm} (\mathbb{E}_q[\log(\theta_m \lambda_k)] + \underbrace{\mathbb{E}_q[\log(\tau_m \chi_{ij}) + \mathbb{E}_q[\log(\eta X_{jim})]}_{\text{new}}}) \right. \right. \\
& \quad \left. \left. - \left( \frac{\gamma_m^{shape}}{\gamma_m^{rate}} \frac{\phi_k^{shape}}{\phi_k^{rate}} + \frac{\nu^{shape}}{\nu^{rate}} X_{jim} + \underbrace{\frac{\xi^{shape}}{\xi^{rate}} \chi_{ij}}_{\text{new}} \right) \right) \right] \\
& + \Psi(\nu^{shape})(c - \nu^{shape}) + \log(\Gamma(\nu^{shape})) - c \log(\nu^{rate}) + \nu^{shape} \left( 1 - \frac{d}{\nu^{rate}} \right) \\
& + \sum_k \left[ \Psi(\phi_k^{shape})(a_{kl} - \phi_k^{shape}) + \log(\Gamma(\phi_k^{shape})) - a_{kl} \log(\phi_k^{rate}) + \phi_k^{rate} \left( 1 - \frac{b_{kl}}{\phi_k^{rate}} \right) \right] \\
& + \sum_m \left[ \Psi(\gamma_m^{shape})(\alpha_{ml} - \gamma_m^{shape}) + \log(\Gamma(\gamma_m^{shape})) - \alpha_{ml} \log(\gamma_m^{rate}) + \gamma_m^{rate} \left( 1 - \frac{\beta_{ml}}{\gamma_m^{rate}} \right) \right] \\
& + \underbrace{\sum_m \left[ \Psi(\xi_m^{shape})(f_{ml} - \xi_m^{shape}) + \log(\Gamma(\xi_m^{shape})) - f_{ml} \log(\xi_m^{rate}) + \xi_m^{rate} \left( 1 - \frac{g_{ml}}{\xi_m^{rate}} \right) \right]}_{\text{new}} \\
& + \sum_{i,j,k} [\rho_{ij,k} (\log(p_{ij,k}) - \log(\rho_{ij,k}))]
\end{aligned} \tag{79}$$

## B Code

The code used for this project is based on the work by De Bacco et al. [5]. It has been adapted and can be found on the gitlab of Utrecht University at: <https://git.science.uu.nl/c.j.bouma/masters-thesis-vimure>