collpcm: Bayesian model selection in latent position cluster models for networks

Jason Wyse **Caitríona Ryan** **Nial Friel**

Abstract

The latent position cluster model is a popular model for the statistical analysis of network data. This model attaches a latent position to vertices in the network graph, with the motivation that the distance between two latent positions is connected to the probability of there being an edge between the corresponding vertices in the graph. Bayesian methods are used to estimate the likely configurations of the latent positions based on a prior assumption on their layout. This prior assumption can reflect a clustering or grouping in the latent positions which can be used to say something about community formation between vertices in the network graph. This package provides convenient computational facilities to fit latent position cluster models, while exploring probable community patterns in the network i.e. exploring the likely groupings or clustering present. The methods do not require specifying a number of groupings or communities in the network at the outset.

Keywords: collapsed latent position cluster model, reversible jump Markov chain Monte Carlo, Bayesian model choice, social network analysis, finite mixture model.

1. Introduction

A social network consists of nodes or actors in a graph, for example, individuals or organizations, connected by one or more specific types of interdependency, such as, friendship, business relationships or trade between countries. The analysis of network data has a rich interdisciplinary history finding application in a wide range of areas including sociology, neuroscience, protein-protien networks, physics, computer science (Wasserman and Galaskiewicz 1994; Faloutsos *et al.* 1999; Adamic *et al.* 2001; Michailidis 2012; Sizemore *et al.* 2018; Pellegrini 2019; Cheng and Park 2020), and many more. The aims of network analysis are both descriptive and inferential. For example, one might be interested in examining global structure within a network or in analysing network attributes such as the degree distribution as well as the local structure such as the identification of influential or highly connected actors in the network. Inferential goals include hypothesis testing, model comparison and making predictions, for example, how far will a virus spread through a network.

There have been many statistical models proposed for the analysis of network data; the exponential random graph model see (Wasserman and Pattison 1996) and (Robins *et al.* 2007) and its extensions to Bayesian (Caimo and Friel 2014), temporal (Lee *et al.* 2020) and multilayer settings (Caimo and Gollini 2020), and the stochastic block model (Nowicki and Snijders 2001) and its extensions (Matias *et al.* 2018; Corneli *et al.* 2019). For comprehensive perspectives and reviews on the statistical models for and analysis of network data, see Goldenberg

et al. (2010); Salter-Townshend *et al.* (2012); Fritz *et al.* (2020). An alternative and popular approach to modelling network data are latent space approaches. There has been much interest in this within the statistical network literature over recent years. For example the earlier proposal of Hoff *et al.* (2002) and more recently Orbanz and Roy (2015); Caron and Fox (2017) in a Bayesian non-parametric settings.

The model this package is concerned with is that proposed initially by Handcock *et al.* (2007). In this model, vertices in the network graph are embedded in a latent space, with proximity in latent space being a predictor of observing edges between vertices. Pairs of vertices with corresponding positions in latent space that are close together are more likely to share an edge in the network graph. (Handcock *et al.* 2007) allows for detection of communities through clustering or groupings in the latent space by assuming the a prior Gaussian finite mixture distribution on latent positions. This provides a useful interpretation of the network since the underlying latent model provides an automatic means of grouping vertices and providing uncertainty around the probability of a vertex belonging to a particular grouping. The R package `latentnet` Krivitsky and Handcock (2008, 2020) can be used to fit an LPCM.

Despite the popularity of LPCMs, a difficulty is the requirement to explicitly state the number of groupings before running an analysis. There may not be an intuition from a practioners perpective, or indeed, the number of groupings may be the objective of the investigation. This obstacle can be somewhat overcome by running separate analyses for each of a range of numbers of groups, and determining an optimal one using an information criterion. However, as argued in Ryan *et al.* (2017), this has its drawbacks; primarily though, running a single analysis can be a heavy computational task, and thus repeating this many times might be involved (especially for moderate sized networks). The motivation of the work we present in this package was to find a way to kill two birds with one stone, integrating the search over probable groupings and latent positions into a single modelling procedure. What we achieved is detailed in , and the methods in this package follow on from that paper.

We avoid going too deep into the details discussed at length in Ryan *et al.* (2017), but we give an overview of the main ideas of the modelling in Section 2. Section 3 describes the Markov chain Monte Carlo algorithm implemented in this package to fit the models. Section 4 gives an overview of key functions in the package and their usage as well as default settings and how to tweak them. Section 5 concludes by giving a walk through example demonstrating the use of the package and exploring the functionality available.

2. Bayesian Latent Position Cluster Model

2.1. Networks and network graphs

Let $\mathcal{V} = \{1, \ldots, n\}$ denote a set of n vertices. We denote all distinguishable pairs of vertices (i, j) by $\mathcal{D}$ (for dyads). If there is an edge between the vertex pair (i, j), then the pair also belongs to the edge set, $(i, j) \in \mathcal{E}$. The network consists of the vertices and the edges and is represented by the network graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. We allow the possibility for edges to be directed, in that $(i, j) \in \mathcal{E}$ is distinguished from $(j, i) \in \mathcal{E}$. An undirected network does not

distinguish these events. Self-ties are not allowed, in that $(i, i) \notin \mathcal{D}$ for any $i \in \mathcal{V}$. The dyad sets are given by

$$\mathcal{D} = \{\, (i, j) : 1 \leq i,\, j \leq n,\, i \neq j \,\},$$

while for undirected networks

$$\mathcal{D} = \{\, (i, j) : 1 \leq i \leq n, j < i \,\}.$$

In the case of a directed network we have $|\mathcal{D}| = n(n-1)$ while in the undirected case $|\mathcal{D}| = n(n-1)/2$.

An observed network and its graph are encoded through an $n \times n$ adjacency matrix, denoted by $\mathbf{Y}$. The (i, j) entry y_{ij} is equal to one if there is an edge from i to j. In the case of undirected networks this will also determine y_{ji}. The network graph $\mathcal{G}$ is completely encoded through $\mathbf{Y}$, so we will use the adjacency matrix for the remainder of this vignette.

2.2. Augmentation with latent positions

A latent position cluster model (LPCM) attaches a latent (unobserved) position $\mathbf{x}$ in a latent space which is $\mathbb{R}^d$ in this package. The latent positions are used to state the probability of an edge between vertices i and j: the linear predictor

$$\eta_{ij} = \beta - ||\mathbf{x}_i - \mathbf{x}_j|| \tag{1}$$

gives this probability through a logistic link function

$$\Pr\left(Y_{ij} = 1 | \mathbf{x}_i, \mathbf{x}_j, \beta\right) = \frac{1}{1 + \mathrm{e}^{-\eta_{ij}}}.$$

The appearance of the euclidean norm $||\cdot||$ in (1) measuring the latent distance between actors i and j has an appealing intuitive interpretation; vertices farther apart in latent space are less likely to have an edge. The parameter β is often referred to as the abundance; high values of β imply a high probability of forming ties (hence abundant). As one of the main motivations for LPCMs is visualisation and interpretation, the most popular choice is $d = 2$, similar to projection of multivariate datasets onto lower dimensional subspaces like principal components analysis with a two-dimensional subspace. This package was designed with this and practitioners in mind. We thus don't tackle the question of model selection for d here (see Durante and Dunson (2014) for an approach to this problem using a shrinkage prior in a dynamic network setting). The package provides most functionality for $d = 2, 1$ (in that order) with much less functionality for $d \geq 3$.A *local independence* assumption is made which assumes edges arise independently over pairs of actors in the network. The likelihood of observing the adjacency $\mathbf{Y}$ then factors as a product over outcomes for dyads $\mathcal{D}$.

$$p(\mathbf{Y} \,|\, \mathbf{X}, \beta) = \prod_{(i,j) \in \mathcal{D}} \Pr\left(Y_{ij} = y_{ij} | \mathbf{x}_i, \mathbf{x}_j, \beta\right) \tag{2}$$

where $\mathbf{X}$ is used to collectively denote the joint positions $\mathbf{x}_1, \ldots, \mathbf{x}_n$.

To provide a facility for community representation, a mixture prior is assumed for the latent positions $\mathbf{X}$: a finite mixture of G d-dimensional Gaussians with spherical precision in place of the is used, giving joint prior on the latent positions of

$$\pi(\mathbf{X}|\boldsymbol{\theta}, G) = \prod_{i=1}^{n} \left(\sum_{g=1}^{G} \omega_g \, \mathcal{N}(\, \mathbf{x}_i \,;\, \boldsymbol{\mu}_g, 1/\tau_g \mathbf{I}) \right). \tag{3}$$

The parameter $\boldsymbol{\theta}$ will be taken to denote the mixture weights ω_g (which sum to one: $\sum_{g=1}^{G} \omega_g = 1$), the component centres and precisions $\boldsymbol{\mu}_g, \tau_g$, $g = 1, \ldots, G$. Clustering in the network can be captured by clustering in the latent positions; different clusters are represented by the components of the finite mixture. One introduces labels $\mathbf{c} = (c_1, \ldots, c_n)$, denoting the component to which each actor belongs. Using the labels, the joint prior density of the latent positions and labels is

$$\pi(\mathbf{X}, \mathbf{c}|\boldsymbol{\theta}, G) = \prod_{g=1}^{G} \prod_{i\,:\,c_i=g} \omega_g \, \mathcal{N}(\mathbf{x}_i; \boldsymbol{\mu}_g, 1/\tau_g \mathbf{I}) \tag{4}$$

2.3. Prior assumptions

Independent priors are assumed for the component weights (Dirichlet), centres (Gaussian) and precisions (gamma) over the G groups:

$$(\omega_1, \ldots, \omega_g) \sim \mathcal{D}(\alpha, \ldots, \alpha)$$

$$\boldsymbol{\mu}_g|\tau_g \sim \mathcal{N}(\mathbf{0}, 1/(\kappa\tau_g)\mathbf{I}) \qquad g = 1, \ldots, G$$

$$\tau_g \sim \mathcal{G}(\delta/2, \gamma/2) \qquad g = 1, \ldots, G$$

where α, κ, δ and γ are parameters to be chosen. Denote these collectively through $\boldsymbol{\nu}$. Choosing $\alpha = 3$, $\delta = 2$ and $\gamma = 0.103$ corresponds to the prior choices made in (Handcock *et al.* 2007) (their parameters are denoted $\nu = 3$, $\alpha = 2$ and $\sigma_0^2 = 0.103$, respectively). Our specification of the prior precision on the component means $\boldsymbol{\mu}_g$ is different. We scale the within cluster precision τ_g by a factor κ. We note that values of κ less than 1 imply that the cluster means are more dispersed than the cluster members. The prior assumed for the intercept parameter β in the linear predictor (1) is $\mathcal{N}(0, \psi^2)$ as in (Handcock *et al.* 2007). All priors above are conditional on G, the number of components in the finite mixture for the latent positions. We also assume prior probability mass function $\pi(G|\boldsymbol{\lambda})$ parameterised by hyperparameter $\boldsymbol{\lambda}$ which could inlcude a maximum number of groups $G_{\max}$ as well as a parameter for the (truncated) probability mass function.

2.4. Hierarchical model

The Bayesian model can be written in hierarchical form

$$\pi(\mathbf{X}, \mathbf{c}, \beta, \boldsymbol{\theta}, G|\mathbf{Y}, \boldsymbol{\nu}, \boldsymbol{\lambda}) \propto p(\mathbf{Y}|\mathbf{X}, \beta)\, \pi(\beta|\psi^2)\, \pi(\mathbf{X}, \mathbf{c}|\boldsymbol{\theta}, G)\, \pi(\boldsymbol{\theta}|\boldsymbol{\nu}, G)\, \pi(G|\boldsymbol{\lambda}). \tag{5}$$

The form of the priors assumed in Section 2.3 imply that the entire $\boldsymbol{\theta}$ vector can be marginalized out of the right hand side of the relation in closed form, meaning that

$$\pi(\mathbf{X}, \mathbf{c}, G|\mathbf{Y}, \boldsymbol{\nu}) = \int \pi(\mathbf{X}, \mathbf{c}, \boldsymbol{\theta}, G|\mathbf{Y}, \boldsymbol{\nu}) \, \mathrm{d}\boldsymbol{\theta}$$

is available to us up to a normalizing constant. The resulting form is

$$\pi(\mathbf{X}, \mathbf{c}, \boldsymbol{\theta}, G|\mathbf{Y}, \boldsymbol{\nu}) \propto p(\mathbf{Y}|\mathbf{X}, \beta) \, \pi(G) \, \frac{\Gamma(G\alpha)}{\Gamma(n + G\alpha)} \prod_{g=1}^{G} \frac{\Gamma(n_g + \alpha)}{\Gamma(\alpha)} \lambda_g(\mathbf{X}, \mathbf{c}|\boldsymbol{\nu}) \tag{6}$$

$$\lambda_g(\mathbf{X}, \mathbf{c}|\boldsymbol{\nu}) = \pi^{-n_g d/2} \frac{\gamma^{\delta/2}}{(n_g/\kappa + 1)^{d/2}} \frac{\Gamma\left((n_g d + \delta)/2\right)}{\Gamma\left(\delta/2\right)} \left[\sum_{i:c_i=g} ||\mathbf{x}_i||^2 - \frac{||\sum_{i:c_i=g} \mathbf{x}_i||^2}{n_g + \kappa} + \gamma \right]^{-(n_g d + \delta)/2}$$

where $n_g = |\{i : c_i = g\}|$. This is a form of product partition model (Quintana and Iglesias 2003). Ryan *et al.* (2017) describes sampling from (2.4) using different MCMC strateges. We summarize these in Section 3.

2.5. Hyperpriors for hyperparameter uncertainty

Exploration of the range of possible values of the hyperparameters $\boldsymbol{\nu}, \boldsymbol{\lambda}$ can be important for some applications. Of the hyperparameters in the model, in our experience, γ appears to be the one whose prior specification has the strongest influence on the posterior. This mirrors closely the findings of (Richardson and Green 1997) (Section 5.1), although their prior specification is slightly different to the one adopted here. The posterior of the number of groups and the prior choice of γ are closely connected, since γ effectively controls the volume in latent space that clusters can occupy. Small values place higher prior mass on clusters occupying a smaller volume of latent space (hence a higher number of groups), while large values favour a smaller number of groups. However, universal calibration of γ is not possible for all problems *a priori*. Incorporating a hyperprior on γ can mitigate this calibration issue. In the package a Gamma(s, r) hyperprior is assumed, with default values of $s = 16, r = 16/0.103$.

3. Estimation using MCMC

Approximate sampling from the posterior (6) can be carried out using MCMC methods. There are four types of updates in our collapsed sampler

(i) updating the abundance parameter β from the observed data likelihood

(ii) updating the latent positions of actors $\mathbf{x}_1, \ldots, \mathbf{x}_n$

(iii) updating actor labels $c_1, \ldots, c_n$ in the finite mixture prior

(iv) updating the number of components G in the mixture, by absorbing components or ejecting new ones.

Update for abundance

The intercept parameter is updated using a random walk Metropolis-Hastings step. A proposal value β^* is drawn from a $\mathcal{N}(\beta, \sigma^2_\beta)$ distribution, where β is the current value of the intercept in the chain. The proposed value is accepted with probability

$$\min\left[1, \frac{p(\mathbf{Y}|\mathbf{X}, \beta^*)\,\pi(\beta^*)}{p(\mathbf{Y}|\mathbf{X}, \beta\,)\,\pi(\beta\,)}\right].$$

Note that the calculation of $p(\mathbf{Y}|\mathbf{X}, \beta)$ is an $O(n^2)$ computation. This is a major drawback when considering the potential applicability of the LPCM in larger networks. Some approaches have been proposed in the literature to circumvent this bottleneck, most notably, the case-control approximation of (Raftery *et al.* 2012) (include Riccardo paper here). We do not consider this problem explicitly in this paper, however, we do note that the log of the likelihood (2) is

$$\log p(\mathbf{Y}|\mathbf{X}, \beta) = \sum_{(i,j)\in\mathcal{D}} \log \Pr\left(Y_{ij} = y_{ij}|\mathbf{x}_i, \mathbf{x}_j, \beta\right). \tag{7}$$

The calculation of this sum (7) is *embarrassingly parallelizable* i.e. the sum over pairs $(i, j) \in \mathcal{D}$ may be split over P available processors at the time of compute giving in good cases a factor P reduction in compute times for the β update. This could be a suggested approach to assuage the quadratic order calculation. Of course, the practicalities of parallelization mean that a favourable increase in efficiency will be implementation and example dependent.

Update for latent positions

The latent positions are updated once each per sweep of the MCMC algorithm using a random walk Metropolis-Hastings update. For actor $i, i = 1, \ldots, n$, a new $\mathbf{x}_i^*$ is proposed from a $\mathcal{N}(\mathbf{x}_i, \sigma^2_{\mathbf{x}}\mathbf{I})$ distribution, where $\mathbf{x}_i$ is the current position of actor i in the latent space. The updated value is accepted with probability

$$\min\left[1, \frac{p_i(\mathbf{Y}|\mathbf{X}^*, \beta)\,\lambda_{c_i}(\mathbf{X}^*, \mathbf{c})}{p_i(\mathbf{Y}|\mathbf{X}\,, \beta)\,\lambda_{c_i}(\mathbf{X}\,, \mathbf{c})}\right]$$

where

$$p_i(\mathbf{Y}|\mathbf{X}, \beta) = \prod_{j\neq i} \Pr(Y_{ij} = y_{ij}|\mathbf{x}_i, \mathbf{x}_j, \beta)$$

if the network is undirected and

$$p_i(\mathbf{Y}|\mathbf{X}, \beta) = \prod_{j\neq i} \Pr(Y_{ij} = y_{ij}|\mathbf{x}_i, \mathbf{x}_j, \beta)\ \Pr(Y_{ji} = y_{ji}|\mathbf{x}_i, \mathbf{x}_j, \beta)$$

if directed.

Updates for actor labels

The label of each actor is sampled from its full conditional $\pi(c_i|\mathbf{c}_{-i}, \mathbf{X}, G)$ in a Gibbs step in each sweep of the algorithm. There is the possibility of label switching due to the non-identifiability of the mixture prior. This will be discussed further in Section 3.3. These Gibbs moves may only move one actor at a time between components. Moves which can move

many actors at a time between clusters are also used. These follow the general prescriptions of (Nobile and Fearnside 2007) moves M1, M2 and M3. As demonstrated by (Nobile and Fearnside 2007) (Section 3.4), such moves can improve the mixing of the chain.

Updating the number of components in the mixture prior

The moves to update the number of components in the mixture comprises two reversible *eject* and *absorb* moves. If the current number of clusters is G, then it is proposed to eject a component from one of the existing components with probability η_G^{ej}; the probability of proposing an absorb move is $1 - \eta_G^{\text{ej}}$. For all G except 1 and some maximum realistic number $G_{\max}$ components, we use $\eta_G^{\text{ej}} = 0.5$.

The eject move chooses one of the G existing clusters g at random. It will be attempted to potentially reallocate members of g to a new component $G+1$. A probability p is sampled from a beta $\mathcal{B}(a, a)$ distribution. The elements of component g are each put into component $G+1$ with probability p. The value of a is chosen from a precomputed lookup table, so that "empty components are proposed relatively often" (see Nobile and Fearnside (2007), Wyse and Friel (2012)). The proposal mechanism creates a new label vector $\mathbf{c}^* \in \{1, \ldots, G+1\}^n$ resulting in the acceptance probability $\min[1, \rho]$, where

$$\rho = \frac{\lambda_g(\mathbf{X}, \mathbf{c}^*)\lambda_{G+1}(\mathbf{X}, \mathbf{c}^*)\pi(G+1)}{\lambda_g(\mathbf{X}, \mathbf{c})\pi(G)} \frac{1 - \eta_G^{\text{ej}}}{\eta_G^{\text{ej}}} \frac{\Gamma(a)^2}{\Gamma(2a)} \frac{\Gamma(2a + n_g)}{\Gamma(a + n_g^*)\Gamma(a + n_{G+1}^*)}.$$

If the move is accepted a random label swap is made between component $G+1$ and one of the other components.

In proposing an absorb move, two components g and k are selected at random from the $G+1$ available. Suppose that the current label vector is $\mathbf{c}$. It is proposed to combine these into one component, in other words, g absorbs k if $g < k$ and vice-versa. Actors which are labelled k are relabelled g, giving the proposed label vector $\mathbf{c}^*$. Then the move is accepted with probability $\min[1, \upsilon]$ where

$$\upsilon = \frac{\lambda_g(\mathbf{X}, \mathbf{c}^*)\pi(G)}{\lambda_g(\mathbf{X}, \mathbf{c})\lambda_k(\mathbf{X}, \mathbf{c})\pi(G+1)} \frac{\eta_{G+1}^{\text{ej}}}{1 - \eta_{G+1}^{\text{ej}}} \frac{\Gamma(2a)}{\Gamma(a)^2} \frac{\Gamma(a + n_g)\Gamma(a + n_k)}{\Gamma(2a + n_g^*)},$$

and $n_g^* = n_g + n_k$. If the move is accepted, all elements of the label vector with a value of k upwards are decremented by 1.

3.1. Updates of hyperparameters

In an extra sampling step, the component marginal precisions τ_g can be "uncollapsed" and sampled at each iteration (still leaving the $\boldsymbol{\mu}_g$ collapsed). Assuming a Gamma$(s/2, r/2)$ hyperprior for γ, first sample τ_g from the conditional

$$\tau_g \,|\, G, \mathbf{c}, \mathbf{X}, \gamma \sim \mathcal{G}\left(\frac{n_g d + \delta}{2}, \frac{1}{2}\left[\sum_{i:c_i=g} ||\mathbf{x}_i||^2 - \frac{||\sum_{i:c_i=g} \mathbf{x}_i||^2}{n_g + \kappa} + \gamma\right]\right)$$

and then sample

$$\gamma \,|\, G, \tau_{1:G} \sim \mathcal{G}\left(\frac{G\,\delta + s}{2}\,,\, \frac{1}{2}\left[\sum_{g=1}^{G} \tau_g + r\right]\right)$$

in each sweep of the MCMC algorithm described below. Uncertainty in κ could also potentially be incorporated using this type of approach, whereby one would additionally sample the $\boldsymbol{\mu}_g$ from their full conditionals in order to sample κ (having assumed a hyperprior for it).

3.2. Adaptive MCMC phase for proposal tuning

We use an adapting phase within an adaptive MCMC (Andrieu and Thoms 2008) at the beginning of the run in order to tune the proposal standard deviations for the abundance parameter update (σ^2_β) and the latent position updates ($\sigma^2_{\mathbf{x}}$). This leads to much improved performance. The adaptation is terminated after the burn-in period so as to satisfy the requirements of adaptive chains to preserve invariance.

The proposal standard deviations are adapted every few hundred iterations. At one of these adaptation times, at iteration t of the burn-in, let $\Delta = \min\{t^{-1/2}, 0.01\}$. If the acceptance rate of the move in question so far is greater than a target rate of 0.234 (Roberts *et al.* 1997), let $f = +1$ and otherwise let $f = -1$. Then multiply the proopsal standard deviation by $\exp(f\Delta)$. This has the effect of increasing the standard deviation if the acceptance rate is higher than the target rate, and decreasing it otherwise.

3.3. Model invariance and post-processing

The likelihood given by (2) is invariant to rotations, reflections or translations of the latent positions $\mathbf{X}$. This is because the linear predictor (1) depends only on the distance between the latent positions. When computing estimates of posterior quantities involving the latent positions via ergodic averages it is thus necessary to post-process the samples generated by the MCMC algorithm. A Procrustes transformation Sibson (1979) is used to match each sample to a reference set of positions $\mathbf{X}_{\text{ref}}$. The MCMC sample iterate giving the highest likelihood (2) is used as a reference configuration.

A different form of invariance is present in the mixture prior (4). Any permutation σ of $\{1, \ldots, G\}$ applied to the labels $\mathbf{c}$ will produce the same value of the prior i.e. $\pi(\mathbf{X}, \mathbf{c}|\boldsymbol{\theta}) = \pi(\mathbf{X}, \mathbf{c}_\sigma|\boldsymbol{\theta})$ where $\mathbf{c}_\sigma = (\sigma(c_1), \sigma(c_2), \ldots, \sigma(c_n))$. Again, to estimate posterior auntities related to the labels, the samples of labels must be post-processed. To do this we use the label switching algorithm advocated by Nobile and Fearnside (2007) and detailed in the Appendix of Wyse and Friel (2012). This algorithm finds the best permutation for each sample by minimizing an evolving cost function based on component assignment agreement. The MCMC sampler produces models with different values of G. In order to post-process labels, we condition on a value of G and correct label switching for all samples having G components. We employ our own implementation of this algorithm, however there is a dedicated **R** package **label.switching** Papastamoulis (2016) which has a number of easily accessible algorithms to carry out post processing of label samples and which we have found very useful in other work.

4. Structure of the package

The key function in the package is the `collpcm.fit` function. The output of this function is an object of class `collpcm`. The `collpcm` class has associated `plot` function. In the remainder of this section, we give an overview of the essential parts of these functions and more important arguments. Many of the arguments relating to model settings and MCMC are set through a call to `collpcm.control`. Prior to writing this vignette, the only information available to users was that contained in the help files. We now link the function syntax with the notation of Section 2.

4.1. Main fitting function

The main fitting function is `collpcm.fit`. The argument `Y` takes an object of class `network`, while `G` gives the number of components (which can be left blank if wished and initialised by default). Argument `Gmax` can be used to give an upper bound on the number of components that can be explored by the MCMC search. The dimension of the latent space is given by argument `d`. The package can work well when $d = 1$ or $d = 2$. For any higher values of d, visualisation of the latent positions is not possible. Arguments to control MCMC settings can be passed through `control`. One can also pass a reference latent position configuration to Procrustes match to using `Xref`. This is included as it could be useful for comparisons with other latent space approaches, or for orienting plots (with an auxiliary run) in a specific way. The `collpcm` object returned from `collpcm.fit` is a list with a number of named slots. These are detailed in Table 1

4.2. Adjusting the default settings

The default settings can be changed through the `collpcm.control` function. A call to this function can include a list with some tailored settings. Alternatively, a blank call will return a list of default settings. This function can be used to do any or all of:

- specify priors and hyperpriors
- specify initial values if desired
- give the MCMC specifics such as run lengths, burn-in and thinning rate
- set proposal distributions for the MCMC run
- give further MCMC settings such as details for adaptive MCMC chains
- determine what summaries should be computed and reported.

Table 2 gives an outline of all settings available through `collpcm.control` and their default settings for reference.

4.3. Plotting the output

There is an S3 plotting method for the `collpcm` class. In the case where models with different G are explored in the MCMC chain, this function makes a plot of the latent positions of

Slot	**What it contains**
`call`	Arguments passed to the function and initialisations (potentially default)
`sample`	Samples and products of the MCMC run (see below)
`Gpost`	The approximate posterior of G from the MCMC samples
`Xpostmean`	The posterior mean latent positons
`XpostMKL`	The maximumd Kullback-Liebler positions
`acceptance.rates`	Acceptance rates for the MCMC sampler
`adapted.sd.prop`	Final proposal standard deviations after (potential) adaptation
`timings`	Timings in seconds for each part of the processing
	Entries of the `sample` *slot*
`sample$G`	Samples of the number of components in the mixture for latent positions
`sample$beta`	Samples of the parameter β
`sample$llike`	Value of the log-likelihood at each stored sample
`sample$labels`	Post processed matrix of component labels
`sample$Gslot`	Indexing vector giving the value of G referencing entries in lists
`sample$label.probs`	Posterior mass functions for vertex membership for each visited G
`sample$Xref`	The reference latent positions used for Procrustes matching
`sample$X`	Samples of the latent positions of vertices
`sample$gamma`	If `gamma.update` is `TRUE`, samples of γ

Table 1: Description of what is in the list returned from a call to `collpcm.fit`.

Setting	**Description**	**Default**
	Specifying priors and hyperparameters	
`G`	Initial number of components	$\texttt{G} \sim \mathcal{U}\{2,3,4,5\}$
`Gmax`	Maximum number of components	$\lfloor n/2 \rfloor$
`Gprior`	Prior probability mass function	$\mathcal{P}(\text{rate} = 1.0)$
`xi`	Prior mean of β	0
`psi`	Prior standard deviation of β	$\sqrt{2}$
`gamma`	Twice the prior rate of the component precisions	0.103
`delta`	Twice the prior shape of the component precisions	2
`alpha`	Symmetric Dirichlet prior on component weights	3
`kappa`	Scaling of τ_g to give prior mean on $\boldsymbol{\mu}_g$	0.1
	Initialisation	
`betainit`	Initial value of β for MCMC	$\mathcal{N}(0, 0.01^2)$
`Xinit`	Initial value of latent positions	independent $\mathcal{N}(\mathbf{0}, \mathbf{I})$
	MCMC settings	
`sample`	Final number of samples requested	5000
`burn`	Number of burn-in iterations	5000
`interval`	Interval between storing each sample (thinning)	10
`model.search`	Do a search over the number of components G	`TRUE`
`pilot`	Number of pilot iterations for only adaptation	0
	Proposal densities for random walk Metropolis updating	
`sd.beta.prop`	Proposal standard deviation for β update	$\sqrt{0.5}$
`sd.X.prop`	Proposal standard deviation (spherical) for $\mathbf{x}_i$ updates	1
	Updates and storage	
`gamma.update`	Take a hyperprior on γ and update	`TRUE`
`adapt`	Use adaptive phase to tune proposal standard deviations	`TRUE`
`adapt.interval`	The iteration interval for adaptive tuning	200
`store.sparse`	Do a sparse run and only store summaries	`FALSE`
`MKL`	Use maxiumum Kullback-Liebler positions for plotting	`TRUE`
	Progress reports	
`verbose`	Print progress report to screen during run	`FALSE`

Table 2: All settings that can be modified through the `collpcm.control` function.

network vertices for the most frequently observed value of G in the samples from the MCMC chain. The latent positons plotted are either the maximum Kullback-Liebler positions or the posterior mean positions, depending on the settings given to `collpcm.control` (see Table 2). By default, a pie chart is shown for each vertex, representing a degree of membership in each component by coloured slices. This can be disabled (and a modal colour will be displayed for each vertex) by passing `pie = FALSE`.

4.4. A summary plot

The function `collpcm.summaryplot` takes an object of class `collpcm` and produces a two-by-two panel summary plot showing MCMC traceplots for visual inspection. This can be used to diagnose obvious poor mixing and cases where the chain should be run for longer. Most of the data shown in this plot is in the `sample` slot of the return from a `collpcm.fit` call.

5. Example application

In this section we walk through an example application using one of the datasets from the package. For this we use the well known monks network of (Sampson 1968) which contains 18 vertices. This is available as the `Monks` dataset in the package. Two other datasets are available, the well known (Zachary 1977) `Karate` club network and the (Lusseau *et al.* 2003) `Dolphins` networks. We begin by loading the data.

```
> # load the Sampson network
> data(Monks)
```

The `Monks` data is loaded as a `network` object, which is the format accepted by `collpcm.fit`. We can now pass this to the main fitting function. As always, it is important to run the MCMC for long enough, and take a good size thinning interval. The MCMC part of the package is implemented in C, and so should be able to handle networks of a couple of hundred vertexes/actors with a good processor and some patience. We'll run this example for 10,000 initial burn-in iterations, and then 500,000 iterations storing every 100th. We use `verbose` to print a progress report to screen. We take the dimension of the latent space as $d = 2$.

```
> # set seed
> set.seed(1984)
>
> # run the model printing run updates to screen
> fit <- collpcm.fit( Monks, d=2,
+       control=list( verbose=TRUE, sample=5000, interval=10^2, burn=10^4 )
```

A summary plot of the MCMC run can be obtained using the `collpcm.summaryplot` function. This can be useful to diagnose obvious convergence/mixing issues.

```
> collpcm.summaryplot(fit)
```

This produces the plot shown in Figure 1. This plot shows traceplots for G, β and the log-likelihood over the MCMC run, as well as the latent positions for the most visited model. Pie charts showing degree of component membership are *not* shown in this network plot.

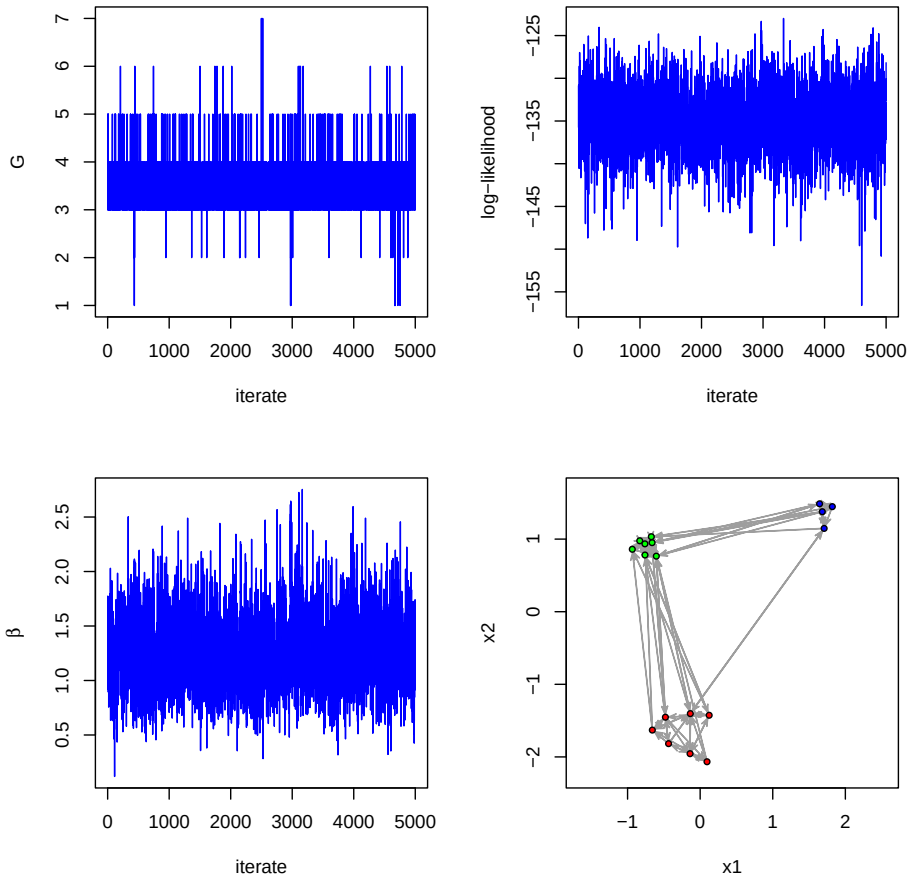

Figure 1: Output of `collpcm.summaryplot` after run on the `Monks` data.

The S3 `plot` function for the `collpcm` class, plots the latent positions for the most visited model. In this case, it defaults to maximum Kullback-Liebler post processed positions (as it was in `collpcm.summaryplot`).

```
> plot(fit)
```

The output is shown in Figure 2. In this case pie charts are overlain on each vertex, but the uncertainty is small for the `Monks` network in the modal model with $G = 3$ groups.

The plot function can be used to explore values of G other than the modal one. For example

```
> plot(fit, G=2)
Warning message:
In plot.collpcm(fit, G = 2) :
  The posterior probability for 2 groups is small: this plot is based on less
  than 100 visits to this model
```

where if the number of visits to the model is small, a warning message will be printed. The plot produced by this code is shown in Figure 3. The orientation of the latent space is similar

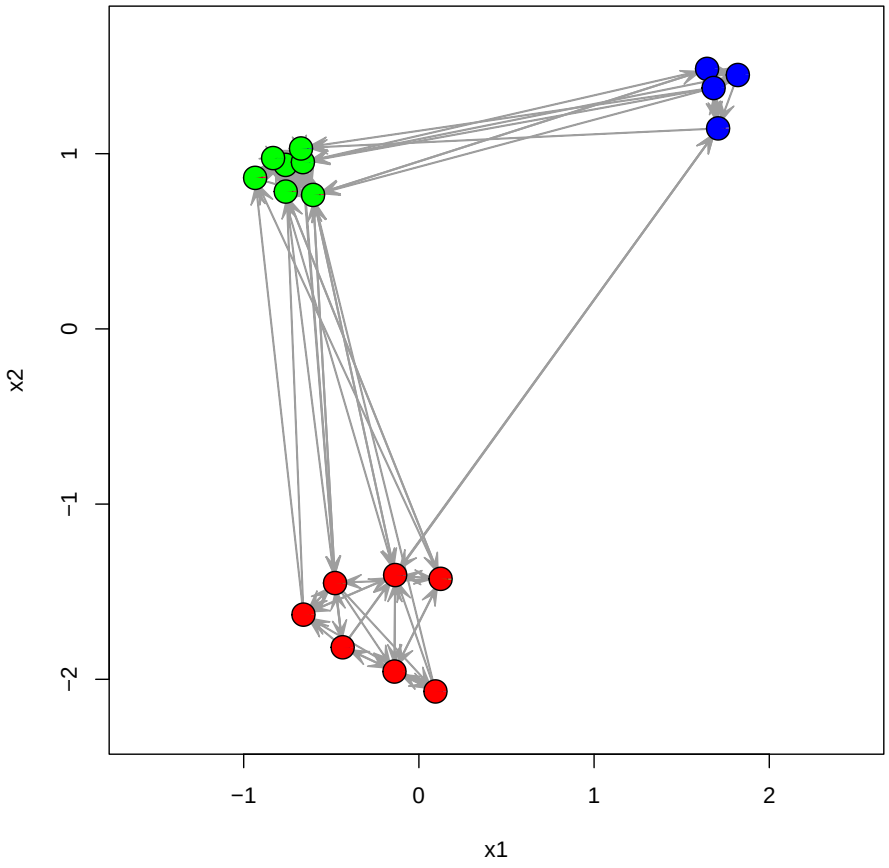

Figure 2: Output of the S3 `plot` function.

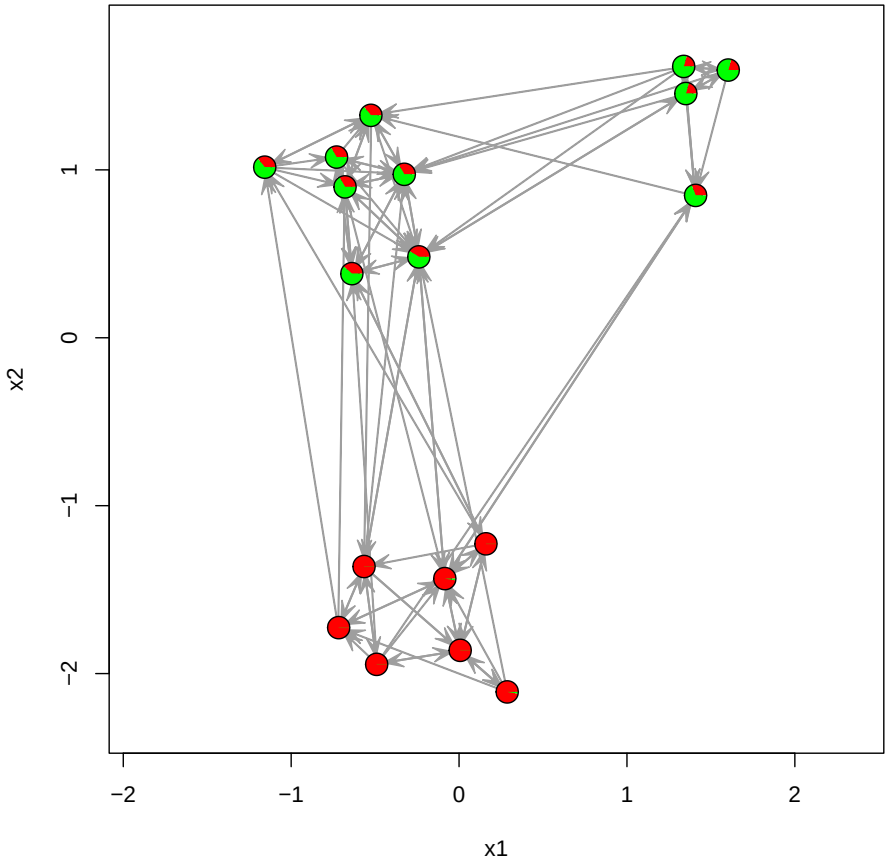

Figure 3: Result of plotting for the $G = 2$ samples.

over sampels corresponding to different values of G since the same reference configuration has been used for Procrustes matching of *all* latent positions. Additionally, the colouring of the pie charts by groups has been made to be as comparable as possible over plots for different values of G for exploration of the groupings.

The S3 `summary` function for the `collpcm` class gives an overview of the MCMC run including acceptance rates and timings:

```
> summary(fit)

Summary of the collapsed LPCM run:

Posterior for the number of groups:
  Number groups Posterior probability
              1                0.0036
              2                0.0078
              3                0.7712
              4                0.1830
```

```
              5                0.0308
              6                0.0032
              7                0.0004

MCMC inferences based on :
         Samples:             5000
         Burn-in:             10000
         Interval:            100
                             -------
         Iterations:          510000

Acceptance rates for the various moves (% accepted):
         Latent postions:                21.32
         Beta (intercept):               23.82
         Move 1:                         1.91
         Move 2:                         15.74
         Move 3:                         86.58
         Eject move:                     4.05
         Absorb move:                    4.08

Run times for various parts of analysis in seconds
        MCMC for samples :                        20.72
        Post-processing procedures
        Label switching :                         0.15
        Procrustes matching :                     0.92
```

There is also a `print` function for the `collpcm` class which prints the estimated posterior of G:

```
> print(fit)

Summary of the collapsed LPCM MCMC run:

Posterior for the number of groups:
  Number groups Posterior probability
              1                0.0036
              2                0.0078
              3                0.7712
              4                0.1830
              5                0.0308
              6                0.0032
              7                0.0004
```

References

Adamic LA, Lukose RM, Puniyani AR, Huberman BA (2001). "Search in power-law networks." *Physical Review E*, **64**, 046135. doi:10.1103/PhysRevE.64.046135.

Andrieu C, Thoms JA (2008). "A tutorial on adaptive MCMC." *Statistics and Computing*, **18**, 343–373.

Caimo A, Friel N (2014). "Bergm: Bayesian Exponential Random Graphs in R." *Journal of Statistical Software*, **61**(2), 1–25. URL https://www.jstatsoft.org/v61/i02/.

Caimo A, Gollini I (2020). "A multilayer exponential random graph modelling approach for weighted networks." *Computational Statistics and Data Analysis*, **142**, 106825. ISSN 0167-9473. doi:https://doi.org/10.1016/j.csda.2019.106825. URL https://www.sciencedirect.com/science/article/pii/S0167947319301720.

Caron F, Fox EB (2017). "Sparse graphs using exchangeable random measures." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **79**(5), 1295–1366. doi:https://doi.org/10.1111/rssb.12233. URL https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/rssb.12233.

Cheng S, Park B (2020). "Flows and Boundaries: A Network Approach to Studying Occupational Mobility in the Labor Market." *American Journal of Sociology*, **126**(3), 577–631. doi:10.1086/712406. https://doi.org/10.1086/712406, URL https://doi.org/10.1086/712406.

Corneli M, Bouveyron C, Latouche P, Rossi F (2019). "The dynamic stochastic topic block model for dynamic networks with textual edges." *Statistics and Computing*, **29**, 677–695. doi:https://doi.org/10.1007/s11222-018-9832-4.

Durante D, Dunson D (2014). "Bayesian Logistic Gaussian Process Models for Dynamic Networks." In S Kaski, J Corander (eds.), *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*, volume 33 of *Proceedings of Machine*

Learning Research, pp. 194–201. PMLR, Reykjavik, Iceland. URL http://proceedings.mlr.press/v33/durante14.html.

Faloutsos M, Faloutsos P, Faloutsos C (1999). "On Power-law Relationships of the Internet Topology." *SIGCOMM Computer Communication Review*, **29**(4), 251–262. ISSN 0146-4833. doi:10.1145/316194.316229.

Fritz C, Lebacher M, Kauermann G (2020). "Tempus volat, hora fugit: A survey of tie-oriented dynamic network models in discrete and continuous time." *Statistica Neerlandica*, **74**(3), 275–299. doi:https://doi.org/10.1111/stan.12198. https://onlinelibrary.wiley.com/doi/pdf/10.1111/stan.12198, URL https://onlinelibrary.wiley.com/doi/abs/10.1111/stan.12198.

Goldenberg A, Zheng AX, Fienberg SE, Airoldi EM (2010). "A Survey of Statistical Network Models." *Foundations and Trends in Machine Learning*, **2**(2), 129–233. ISSN 1935-8237. doi:10.1561/2200000005. URL http://dx.doi.org/10.1561/2200000005.

Handcock MS, Raftery AE, Tantrum JM (2007). "Model-based clustering for social networks." *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, **170**(2), 301–354.

Hoff PD, Raftery AE, Handcock MS (2002). "Latent space approaches to social network analysis." *Journal of the American Statistical Association*, **97**(460), 1090–1098.

Krivitsky PN, Handcock MS (2008). "Fitting latent cluster models for networks with latentnet." *Journal of Statistical Software*, **24**(5), 1–23.

Krivitsky PN, Handcock MS (2020). *latentnet: Latent Position and Cluster Models for Statistical Networks*. The Statnet Project (https://CRAN.R-project.org/package=latentnet). R package version 2.10.5.

Lee KH, Xue L, Hunter DR (2020). "Model-based clustering of time-evolving networks through temporal exponential-family random graph models." *Journal of Multivariate Analysis*, **175**, 104540. ISSN 0047-259X. doi:https://doi.org/10.1016/j.jmva.2019.104540. URL https://www.sciencedirect.com/science/article/pii/S0047259X18306717.

Lusseau D, Schneider K, Boisseau O, Haase P, Slooten E, Dawson S (2003). "The bottlenose dolphin community of Doubtful Sound features a large proportion of long-lasting associations." *Behavioral Ecology and Sociobiology*, **54**(4), 396–405.

Matias C, Rebafka T, Villers F (2018). "A semiparametric extension of the stochastic block model for longitudinal networks." *Biometrika*, **105**(3), 665–680. ISSN 0006-3444. doi:10.1093/biomet/asy016. https://academic.oup.com/biomet/article-pdf/105/3/665/25470042/asy016_supp.pdf, URL https://doi.org/10.1093/biomet/asy016.

Michailidis G (2012). "Statistical Challenges in Biological Networks." *Journal of Computational and Graphical Statistics*, **21**(4), 840–855. doi:10.1080/10618600.2012.738614. http://www.tandfonline.com/doi/pdf/10.1080/10618600.2012.738614.

Nobile A, Fearnside A (2007). "Bayesian finite mixtures with an unknown number of components: the allocation sampler." *Statistics and Computing*, **17**(2), 147–162.

Nowicki K, Snijders T (2001). "Estimation and prediction for stochastic blockstructures." *Journal of the American Statistical Association*, **96**(455), 1077–1087.

Orbanz P, Roy DM (2015). "Bayesian Models of Graphs, Arrays and Other Exchangeable Random Structures." *IEEE transactions on pattern analysis and machine intelligence*, **37**(2), 437—461. ISSN 0162-8828. doi:10.1109/tpami.2014.2334607. URL https://doi.org/10.1109/TPAMI.2014.2334607.

Papastamoulis P (2016). "label.switching: An R Package for Dealing with the Label Switching Problem in MCMC Outputs." *Journal of Statistical Software, Code Snippets*, **69**(1), 1–24. ISSN 1548-7660. doi:10.18637/jss.v069.c01. URL https://www.jstatsoft.org/v069/c01.

Pellegrini M (2019). "Community Detection in Biological Networks." In S Ranganathan, M Gribskov, K Nakai, C Schönbach (eds.), *Encyclopedia of Bioinformatics and Computational Biology*, pp. 978–987. Academic Press, Oxford. ISBN 978-0-12-811432-2. doi:https://doi.org/10.1016/B978-0-12-809633-8.20428-7. URL https://www.sciencedirect.com/science/article/pii/B9780128096338204287.

Quintana FA, Iglesias PL (2003). "Bayesian clustering and product partition models." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **65**(2), 557–574. doi:https://doi.org/10.1111/1467-9868.00402.

Raftery AE, Niu X, Hoff PD, Yeung K (2012). "Fast inference for the latent space network model using a case-control approximate likelihood." *Journal of Computational and Graphical Statistics*, **21**(4), 901–919.

Richardson S, Green P (1997). "On Bayesian analysis of mixtures with an unknown number of components (with discussion)." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **59**(4), 731–792.

Roberts GO, Gelman A, Gilks WR (1997). "Weak Convergence And Optimal Scaling Of Random Walk Metropolis Algorithms."

Robins G, Snijders T, Wang P, Handcock MS, Pattison P (2007). "Recent developments in exponential random graph (p*) models for social networks." *Social Networks*, **29**(2), 192–215.

Ryan C, Wyse J, Friel N (2017). "Bayesian model selection for the latent position cluster model for social networks." *Network Science*, **5**(1), 70–91.

Salter-Townshend M, White A, Gollini I, Murphy TB (2012). "Review of Statistical Network Analysis: Models, Algorithms, and Software." *Stat. Anal. Data Min.*, **5**(4), 243–264. ISSN 1932-1864. doi:10.1002/sam.11146. URL https://doi.org/10.1002/sam.11146.

Sampson S (1968). *A novitiate in a period of change: An experimental and case study of social relationships*. Ph.D. thesis, Cornell University, September.

Sibson R (1979). "Studies in the robustness of multidimensional scaling: Perturbational analysis of classical scaling." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **41**(2), 217–229.

Sizemore AE, Giusti C, Kahn A, Vettel JM, Betzel RF, Bassett DS (2018). "Cliques and Cavities in the Human Connectome." *J. Comput. Neurosci.*, **44**(1), 115–145. ISSN 0929-5313. doi:10.1007/s10827-017-0672-6. URL https://doi.org/10.1007/s10827-017-0672-6.

Wasserman S, Galaskiewicz J (1994). *Advances in social network analysis: Research in the social and behavioral sciences*. Sage Publications, Thousand Oaks, California.

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

Wyse J, Friel N (2012). "Block clustering with collapsed latent block models." *Statistics and Computing*, **22**(2), 415–428. ISSN 0960-3174. doi:10.1007/s11222-011-9233-4.

Zachary WW (1977). "An information flow model for conflict and fission in small groups." *Journal of Anthropological Research*, **33**(4), 452–473.

Affiliation:

Jason Wyse
Discipline of Statistics and Information Systems and ADAPT centre
School of Computer Science and Statistics
Trinity College Dublin
College Green
Dublin 2, Ireland
E-mail: wyseja@tcd.ie

Caitríona Ryan
Hamilton Institute
Maynooth University
Maynooth
Ireland E-mail: caitriona.ryan@i-form.ie

Nial Friel
School of Mathematics and Statistics and Insight centre for data analytics
University College Dublin
Belfield
Dublin 4
Ireland
E-mail: nial.friel@ucd.ie