missSBM: An R Package for Handling Missing Values in the Stochastic Block Model

Pierre M Barbillon, Julien Chiquet, Timothée Tabouy

missSBM: An R Package for Handling Missing Values in the Stochastic Block Model

Pierre Barbillon
UMR MIA-Paris, Université Paris-Saclay, AgroParisTech, INRAE

Julien Chiquet
UMR MIA-Paris, Université Paris-Saclay, AgroParisTech, INRAE

Tabouy Timothée
UMR MIA-Paris, Université Paris-Saclay, AgroParisTech, INRAE

Abstract

The Stochastic Block Model (SBM) is a popular probabilistic model for random graphs. It is commonly used for clustering network data by aggregating nodes that share similar connectivity patterns into blocks. When fitting an SBM to a network which is partially observed, it is important to take into account the underlying process that generates the missing values, otherwise the inference may be biased. This paper introduces **missSBM**, an R-package fitting the SBM when the network is partially observed, i.e., the adjacency matrix contains not only 1's or 0's encoding presence or absence of edges but also `NA`'s encoding missing information between pairs of nodes. This package implements a set of algorithms for fitting the binary SBM, possibly in the presence of external covariates, by performing variational inference adpated to several observation processes. Our implementation automatically explores different block numbers to select the most relevant model according to the Integrated Classification Likelihood (ICL) criterion. The ICL criterion can also help determine which observation process better corresponds to a given dataset. Finally, **missSBM** can be used to perform imputation of missing entries in the adjacency matrix. We illustrate the package on a network data set consisting of interactions between political blogs sampled during the French presidential election in 2007.

Keywords: Network, Missing data, Stochastic Block Model.

1. Introduction

In many fields of science, networks are a natural way to represent interaction data. To cite a few examples, a network may represent social interactions such as friendship or collaboration between people in a social network, regulation between genes and their products in a gene regulatory network, or predation between animals in a food web. In this paper, we only consider networks which can be represented by graphs composed of binary edges connecting pairs of nodes (also referred to as *dyads* in the following).

To this day, there exist many pieces of software performing network-related analyses. Un-

surprisingly, the R community is extremely active in this area. Indeed, the R programming language is especially well-designed for performing data manipulation and visualization, and is thus appropriate for handling network data. Among the many available R packages related to networks, we suggest a classification into three groups[1]:

i) Packages for representation, manipulation or visualization tasks, and packages computing descriptive statistics. We mention non-exhaustively the following top representatives: **igraph** (Csardi and Nepusz 2006), **network** and **sna** (Butts 2008a,b).

ii) Packages learning the structure of a network from an external source of data, such as **huge** (Zhao, Liu, Roeder, Lafferty, and Wasserman 2012), **glasso** (Friedman, Hastie, and Tibshirani 2019), **bnlearn** (Scutari 2009) or **bnstruct** (Franzin, Sambo, and di Camillo 2017). These packages generally rely on a specific graphical modeling of the data (e.g., Gaussian graphical models (Lauritzen 1996) in **huge** and **glasso**, or Bayesian networks (Pearl 2011) in **bnlearn** and **bnstruct**).

iii) Packages fitting (probabilistic) models on network data. The **ergm** package (Hunter, Handcock, Butts, Goodreau, and Morris 2008) fits the family of exponential random graph models (ERGM) introduced in Hunter and Handcock (2006): it is part of the collection of tools around ERGM regrouped in the **statnet** metapackage (Handcock, Hunter, Butts, Goodreau, and Morris 2008); **latentnet** (Krivitsky and Handcock 2008) implements the latent space approach of Hoff, Raftery, and Handcock (2002); **mixer** (Ambroise, Grasseau, Hoebeke, Latouche, Miele, and Picard 2015) and **blockmodels** (Léger 2016) fit the Stochastic Block Model (SBM) when the distribution of the edges belongs to the exponential family (Snijders and Nowicki 1997; Nowicki and Snijders 2001). Other R packages related to the SBM and its extensions include **sbm** (Chiquet, Donnet, and Barbillon 2021), **sbmr** (Strayer 2021), **dynSBM** (Matias and Miele 2010), **blockmodeling** (Žiberna 2020), **dBlockmodeling** (Brusco 2020), **expSBM** (Rastelli and Fop 2019), **MLVSBM** (Chabert-Liddell 2021), **greed** (Étienne Côme and Jouvin 2021), **sbmSDP** (Amini 2015), **hergm** (Schweinberger and Luna 2018), **lda** (Chang 2015), **graphon** (You 2020), **GREMLINS** (Donnet and Barbillon 2021) and **noisySBM** (Rebafka and Villers 2020). Some of these packages, as well as some implementations in other programming languages, are presented in the following.

The **missSBM** package which we introduce here belongs to the third category, that is, software that fits a specific probabilistic model on network data. More specifically, **missSBM** is dedicated to the estimation of the Stochastic Block Model (SBM), a mixture of Erdős-Rényi random graphs (Erdős and Renyi 1959) offering a high degree of heterogeneity in connectivity profiles (see Abbe 2017, for a recent review). The SBM generally fits well real-world network data while keeping the advantage of being a probabilistic generative model (contrary to mechanistic approaches such as the Barabási-Albert model (Albert and Barabási 2002), defined by a preferential attachment algorithm). The main outcome of an SBM fit is a clustering of the nodes – or "blocks" – so that the nodes share the same properties within the same block. To our knowledge, the reference package for fitting the SBM with the R programming system is **blockmodels**. It includes efficient implementations of variational algorithms to fit

[1]In addition to this brief typology, the interested reader may consult the CRAN task view on the related topic of graphical modeling (Hojsgaard 2019).

different flavors of the SBM, adapted to binary network data and valued networks, with optional covariates on the edges. Two other important extensions of the SBM are available as R packages: the degree-corrected Stochastic Block Model in **randnet** (Li, Levina, and Zhu 2010) and a dynamic version of the Stochastic Block Model in **dynsbm** (Matias and Miele 2010). Beyond the R framework, there also exist Python packages and C++ libraries providing efficient codes for some particular SBM: the Python packages **CommunityDetection** (Mejean and Maison 2017) and **BipartiteSBM** (Yen and Larremore 2018) are dedicated to the estimation of special network structures using various heuristics and network models, among which the SBM. Beyond variational approaches, MCMC methods exist for inferring the SBM, solving the exact problem but being generally more computationally demanding: the Python library **graph-tool** (Peixoto 2014) includes an MCMC sampler to fit the binary SBM and its degree-corrected variant; C++ libraries **sbm_canonical_mcmc** (Young, Desrosiers, Hébert-Dufresne, Laurence, and Dubé 2017) and **bipartiteSBM-MCMC** (Yen and Larremore 2019) respectively implement a MCMC sampler for the SBM and the bipartite SBM. Finally **MODE-NET** (Decelle, Krzakala, and Zhang 2019) implements the belief propagation algorithm for inferring the degree-corrected SBM.

Despite their high quality, an important limitation of the aforementioned software is to require a network that is fully observed, that is, no missing value is supported. The main feature of **missSBM** is to deal with cases where the network data is only partially observed. More precisely, we consider situations where the adjacency matrix of the network data contains not only 1's or 0's for presence or absence of an edge, but also `NA`'s encoding missing information for some dyads. Note that this situation is different from the case considered in **noisySBM**: there, a similarity matrix is fully observed between all pairs of nodes, and the goal is to separate the 'true' interactions from noise by means of a dedicated SBM.

When inferring the SBM from network data with missing values, it is important to take into account the underlying process that generates these missing values in the estimation of the model parameters, otherwise it may be biased. More specifically, one has to identify whether the values are Missing at Random or not (MAR and MNAR, see Little and Rubin 2019). This issue has been studied in the context of network data by Handcock and Gile (2010) for the ERGM and in our methodological paper (Tabouy, Barbillon, and Chiquet 2020) for the SBM. **missSBM** is an implementation of the methodology developed therein. It also considers new sampling designs and the inclusion of covariates simultaneously in the SBM and in the observation process, which was not studied by Tabouy *et al.* (2020). Specifically, **missSBM** implements variational algorithms in the vein of Daudin, Picard, and Robin (2008) and Léger (2016) for estimating the SBM, with or without covariates, under various missing data mechanisms. This includes cases of incomplete data where the inference can be made only on the observed part of the data (MAR), or cases where it is necessary to take the sampling design into account in the inference (MNAR).

Some frameworks deal with missing data but rather from the cross-validation perspective than the sampling perspective. Cross-validation is used to perform model selection for networks such as the choice of the number of blocks or communities (Li, Levina, and Zhu 2020; Chen and Lei 2018) or the choice of the latent structure (Hoff 2007). Hence, these frameworks are quite different from ours since cross-validation is done under a MAR sampling while our main goal is to be able to infer an SBM under several MNAR sampling mechanisms.

The paper is organized as follows: Section 2 introduces the statistical framework of the binary SBM, with or without covariates, and summarizes the key points of its inference under missing

data conditions. Section 3 provides basic user guidelines for the main functions and classes of objects. We finally detail in Section 4 a case study which analyzes a network data set describing the French blogosphere during the period preceding the 2007 French presidential election, illustrating the most striking features of the package.

2. Statistical Framework

2.1. Binary Stochastic Block Model (SBM)

In an SBM, nodes from a set $\mathcal{N} \triangleq \{1, \ldots, n\}$ are distributed among a set $\mathcal{Q} \triangleq \{1, \ldots, Q\}$ of hidden blocks which model the latent structure of the graph. The group membership is described by independent categorical variables $(\mathbf{Z}_i, i \in \mathcal{N})$ with multinomial distribution $\mathcal{M}(1, \boldsymbol{\alpha} = (\alpha_1, ..., \alpha_Q))$. The probability of having an edge between any pair of nodes (or *dyad*) only depends on the blocks the two nodes belong to. Hence, the presence of an edge between i and j, indicated by the binary variable Y_{ij}, is independent of the other edges conditionally on the latent blocks:

$$Y_{ij}|\mathbf{Z}_i, \mathbf{Z}_j \sim^{ind} \mathcal{B}(\pi_{\mathbf{Z}_i\mathbf{Z}_j}), \text{ for all } (i,j) \in \mathcal{D}, \tag{1}$$

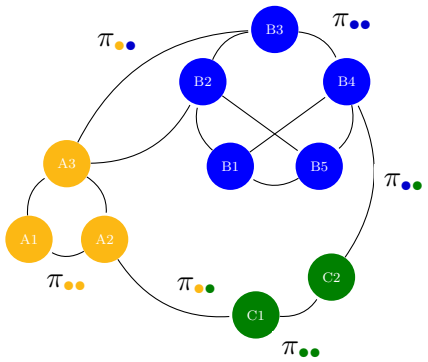
where $\mathcal{B}$ stands for the Bernoulli distribution and $\mathcal{D}$ the set of dyads. This set may be either equal to $\{(i,j) \in \mathcal{N}^2;\ i \neq j\}$ if the network is directed or to $\{(i,j) \in \mathcal{N}^2;\ i < j\}$, otherwise[2]. In the following, we denote by $\boldsymbol{\pi} = (\pi_{q\ell})_{(q,\ell)\in\mathcal{Q}^2} \in [0,1]^{Q^2}$ the connectivity matrix, $\boldsymbol{\alpha} \in \mathbb{D}^Q = \{(\alpha_1, \ldots, \alpha_Q) \in [0,1]^Q;\ \alpha_1 + \ldots \alpha_Q = 1\}$ the block proportions, $\mathbf{Z} = (\mathbf{Z}_1, ..., \mathbf{Z}_n)^T$ the $n \times Q$ membership matrix and $\mathbf{Y} = (Y_{ij})_{(i,j)\in\mathcal{D}}$ the $n \times n$ adjacency matrix. This matrix is binary, with a diagonal filled with `NA`'s and is symmetric if and only if the network is undirected. The vector encompassing all the unknown model parameters is $\boldsymbol{\theta} = (\boldsymbol{\alpha}, \boldsymbol{\pi})$. A schematic representation of the binary SBM in the undirected case is given in Figure 1, where we highlight the latent clustering.

2.2. Accounting for External Covariates

On top of information about connections between nodes, it is common for network data to be accompanied with additional information on nodes or dyads, that we call *covariates*: for instance in social networks, nodes may belong to different categories (gender, occupation, nationality). Covariates on dyads usually represent similarity or dissimilarity between nodes: for example in a context of spatial data where nodes correspond to entities with explicit geographic locations, dyad covariates may be the distances between the nodes.

Depending on the analysis, we may want to detect a connectivity pattern beyond the covariate effect. To do so, we implemented in **missSBM** a variant of Model (1) for including covariates. Let $\mathbf{X}_{ij} \in \mathbb{R}^m$ denote the vector of m covariates for dyad (i,j). If the covariates correspond to the nodes, i.e., $\mathbf{X}_i \in \mathbb{R}^N$ is associated with node i for all $i \in \mathcal{N}$, they are transferred onto the dyad level through a symmetric "similarity" function $\phi(\cdot,\cdot) : \mathbb{R}^N \times \mathbb{R}^N \to \mathbb{R}^m$: $\mathbf{X}_{ij} \triangleq \phi(\mathbf{X}_i, \mathbf{X}_j)$. In the following, $\mathbf{X} \triangleq [\mathbf{X}_{ij}]_{i,j\in\mathcal{N}} \in (\mathbb{R}^m)^{n\times n}$ denotes the array of covariates.

[2] Although self-edges (Y_{ii}) could be defined in the SBM, they are not considered in **missSBM** since they are scarce in real data.



- $\mathcal{Q} = \{\bullet, \bullet, \bullet\}$ blocks
- $\alpha_\bullet = \mathbb{P}(i \in \bullet), \bullet \in \mathcal{Q}, i = 1, \ldots, n$
- $\pi_{\bullet\bullet} = \mathbb{P}(Y_{ij} = 1 | i \in \bullet, j \in \bullet)$

Figure 1: Schematic representation of an undirected network following the stochastic block model with 3 blocks. Colors are blocks in which nodes are dispatched with probabilities $\boldsymbol{\alpha}$ and the distribution of the dyads depends on colors of nodes with probabilities $\boldsymbol{\pi}$.

An SBM including the effect of these covariates is

$$\begin{array}{rcll} \mathbf{Z}_i & \sim^{\text{iid}} & \mathcal{M}(1, \boldsymbol{\alpha}), & \text{for all } i \in \mathcal{N}, \\ Y_{ij} \mid \mathbf{Z}_i, \mathbf{Z}_j, \mathbf{X}_{ij} & \sim^{\text{ind}} & \mathcal{B}(g(\gamma_{z_i z_j} + \boldsymbol{\beta}^\top \mathbf{X}_{ij}), & \text{for all } (i,j) \in \mathcal{D}, \end{array} \tag{2}$$

where $\gamma_{q\ell} \in \mathbb{R}$, $\boldsymbol{\beta} \in \mathbb{R}^m$, $\boldsymbol{\alpha} = (\alpha_1, ..., \alpha_Q)$, $g(x) = (1 + e^{-x})^{-1}$. The vector of unknown parameters is now defined by $\boldsymbol{\theta} = (\boldsymbol{\gamma}, \boldsymbol{\beta}, \boldsymbol{\alpha})$. Note the connection with logistic regression: Model (2) assumes a logistic link between the presence of an edge Y_{ij} and the corresponding covariates. The intercept term $(\gamma_{q\ell})_{q\ell}$ depends on the blocks of the nodes and describes the heterogeneity of the connections that is not explained by the regression term $\beta^T \mathbf{X}_{ij}$.

Connections with Similar Models. The SBM was originally introduced by Nowicki and Snijders (2001). Many extensions have been proposed since, including other distributions on dyads on top of the incorporation of covariates (Mariadassou, Robin, and Vacher 2010). Contrary to the latent space model of Hoff *et al.* (2002) where the latent space is continuous, the latent variables in the SBM lie in a discrete space. When covariates are included as in Equation (2), their effect is removed and the blocks shall then explain the structure in the network *beyond* the covariates. This approach is similar to Vu, Hunter, and Schweinberger (2013) and opposite to the one used by Tallberg (2004) or Binkiewicz, Vogelstein, and Rohe (2017) where the covariates help learn the underlying clustering of the nodes since their distribution is assumed to depend on the same latent variables as the SBM.

2.3. Missing Data and SBM

The main feature of **missSBM** is to deal with some processes that generate missing values in order to provide more accurate estimates of the parameters underlying an incompletely observed network. The number of nodes n is assumed to be known and the missing information only concerns the dyads. Hence the sampled data can be encoded in an adjacency matrix $\mathbf{Y}$ where missing information – dyads whose value is unobserved – is encoded by `NA`'s. We also define the $n \times n$ observation matrix $\mathbf{R}$ such as $R_{ij} = 1$ if the dyad Y_{ij} is observed and $R_{ij} = 0$

otherwise. For convenience we define $\mathbf{Y}^o = \{Y_{ij} : R_{ij} = 1\}$ and $\mathbf{Y}^m = \{Y_{ij} : R_{ij} = 0\}$ the respective sets of observed and unobserved dyads.

In our framework, an observation process – or sampling design – is a stochastic process that generates $\mathbf{R}$. We then rely on the standard missing data theory of Little and Rubin (2019) to classify those designs either into Missing Completely At Random (MCAR), Missing At Random (MAR) or Missing Not At Random (MNAR) cases. This framework has to be extended to handle the latent variables $\mathbf{Z}$ in the SBM as we did in Tabouy *et al.* (2020):

$$\text{Sampling design for SBM is } \begin{cases} \text{MCAR} & \text{if } \mathbf{R} \perp\!\!\!\perp (\mathbf{Y}, \mathbf{Z}) \mid \mathbf{X}, \\ \text{MAR} & \text{if } \mathbf{R} \perp\!\!\!\perp (\mathbf{Y}^{\text{m}}, \mathbf{Z}) \mid (\mathbf{Y}^{\text{o}}, \mathbf{X}), \\ \text{MNAR} & \text{otherwise.} \end{cases} \tag{3}$$

The notation $\perp\!\!\!\perp$ stands for independence between random variables. Note that MCAR missingness is a particular case of MAR missingness. This definition provides the general case when covariates $\mathbf{X}$ are available. Otherwise, the definition remains valid just by removing $\mathbf{X}$. Denoting by $\boldsymbol{\psi}$ the set of parameters associated with the distribution that generates the sampling matrix $\mathbf{R}$, we assume that $\boldsymbol{\psi}$ and $\boldsymbol{\theta}$ live in a product space, so that we can derive the following proposition:

Proposition 1. *From* (3)*, if the sampling is MAR or MCAR then maximizing* $p_{\boldsymbol{\theta},\boldsymbol{\psi}}(\mathbf{Y}^{\text{o}}, \mathbf{R}, \mathbf{X})$ *or* $p_{\boldsymbol{\theta}}(\mathbf{Y}^{\text{o}}, \mathbf{X})$ *in* $\boldsymbol{\theta}$ *is equivalent for* $\boldsymbol{\theta}$*.*

This proposition was proven in Tabouy *et al.* (2020) in the absence of covariate. The generalization to handle covariates is straightforward. In words, the inference can be conducted on the observed part of the network data when the sampling is M(C)AR without incurring any bias. In these cases, adaptating the existing algorithms for SBM inference is simple. MNAR sampling designs require, however, more refined inference strategies since the observation process has to be included in the inference.

2.4. Examples of Sampling Designs for Networks

This section reviews a set of stochastic processes defining sampling designs available in **missSBM**. These sampling designs may depend either on *i*) the values of the dyads in the network; *ii*) the latent clustering of the nodes; or *iii*) some covariates, via the vector of parameters $\boldsymbol{\psi}$. All examples detailed in the following assume that the observations are independent conditionally on $\mathbf{Y}$, $\mathbf{Z}$ and $\mathbf{X}$ (either observations of the dyads for dyad-centered sampling designs, or observations of the nodes for node-centered sampling designs). From a practical viewpoint, the sampling designs implemented in **missSBM** allow the user to either *i*) generate new data by partially observing an existing network according to a predefined sampling design, with user-defined parameters $\boldsymbol{\psi}$, or *ii*) to fit an SBM model under missing data condition, assuming that the missing entries arise from a given type of sampling design for which the unknown parameters $\boldsymbol{\psi}$ are estimated jointly with the SBM parameters $\boldsymbol{\theta}$.

Dyad-Centered Sampling Designs

- **Dyad Sampling** (MCAR): each dyad $(i, j) \in \mathcal{D}$ has the same probability $\mathbb{P}(R_{ij} = 1) \triangleq \psi \in [0, 1]$ to be observed.

- **Double Standard Sampling** (MNAR): let $\boldsymbol{\psi} \triangleq (\rho_1, \rho_0) \in [0,1]^2$. Double standard sampling consists in observing dyads with probabilities

$$\mathbb{P}(R_{ij} = 1 | Y_{ij} = 1) = \rho_1, \qquad \mathbb{P}(R_{ij} = 1 | Y_{ij} = 0) = \rho_0.$$

  The probability for sampling a dyad thus intrinsically depends on the presence/absence of the corresponding edge. This double standard sampling is especially likely in real world applications, if it is easier to observe an existing connection than the absence of connection. For instance, in protein-protein networks, the sampling effort is more important to determine the absence of a link than its existence.

- **Block-Dyad Sampling** (MNAR): this sampling consists in observing all dyads with probabilities $\boldsymbol{\psi} \triangleq (\psi_{q\ell})_{(q,\ell)\in\mathcal{Q}^2} \in [0,1]^{Q^2}$ depending on the underlying clustering of the network:

$$\psi_{q\ell} = \mathbb{P}(R_{ij} = 1 \mid Z_{iq} = 1, Z_{j\ell} = 1).$$

- **Covar-Dyad Sampling** (MAR): let us define $\boldsymbol{\psi} \triangleq (\alpha, \kappa) \in \mathbb{R} \times \mathbb{R}^m$. Here the probability for observing a dyad is driven by the effect of a given covariate:

$$\mathbb{P}(R_{ij} = 1 | \mathbf{X}_{ij}) = g(\alpha + \kappa^t \mathbf{X}_{ij}),$$

  where we recall that $g(x) = (1 + e^{-x})^{-1}$. Under this sampling, the external covariates may impact both a connection and the ability to observe it. In this case, the sampling remains MAR provided that the covariates are available.

Node-Centered Sampling Designs

A node-centered sampling consists in observing some nodes sampled with probabilities given by the sampling design. Observing a node means observing all the dyads involving that node. For all $i \in \mathcal{N}$, we denote by V_i the indicator variable for observing node i. Hence if $V_i = 1$ we have $R_{ij} = R_{ji} = 1$ for all $j \in \mathcal{N}$. Node-centered sampling designs are likely in social sciences since a network is sampled through direct interviews. During an interview, individuals (nodes) indicate to whom they are connected. Some individuals may then indicate a connection with an individual not available for an interview. The resulting missing dyads concern dyads between individuals who were not interviewed. Even if the connection is oriented (directed network), we assume that an individual, when interviewed, provides its ingoing and outgoing connections.

- **Node Sampling** (MCAR): the probabilities for observing nodes are uniform: $\mathbb{P}(V_i = 1) = \psi \in [0,1]$ for all $i \in \mathcal{N}$.

- **Snowball sampling** (MAR): a first batch of nodes is sampled as in node sampling. Then, a second batch is composed of the neighbors of the first batch (the set of nodes linked to at least a node of the first batch). Other batches can then be obtained through several sampling steps which are called waves. These successive waves are then MAR and not MCAR since they are built on the basis of the previously observed part of Y.

- **Degree Sampling** (MNAR): for all node $i \in \mathcal{N}$, $\mathbb{P}(V_i = 1) = \rho_i$ where $(\rho_1, \ldots, \rho_n) \in [0,1]^n$ are such that $\rho_i = g(a + bD_i)$ for all $i \in \mathcal{N}$ where $\psi \triangleq (a, b) \in \mathbb{R}^2$ and $D_i = \sum_j Y_{ij}$. This sampling may be the consequence of a situation where popular individuals are more likely to be interviewed.

- **Block-Node Sampling** (MNAR): this sampling consists in observing all dyads corresponding to nodes selected with probabilities $\psi \triangleq (\psi_1, \ldots, \psi_Q) \in [0,1]^Q$ such that $\psi_q = \mathbb{P}(V_i = 1 \mid Z_{iq} = 1)$ for all $(i, q) \in \mathcal{N} \times \mathcal{Q}$. This sampling may happen if some communities that shape the connections in the network are not equally reachable.

- **Covar-Node Sampling** (MAR): let $\psi \triangleq (\nu, \eta) \in \mathbb{R} \times \mathbb{R}^N$. The probability to observe a node is
$$\mathbb{P}(V_i = 1 | \mathbf{X}_i) = g(\nu + \eta^t \mathbf{X}_i).$$
In this sampling, some external information shapes the sampling process of the nodes. Even if the covariates also impact the probabilities of connection, as in the Covar-dyad sampling, the sampling design is MAR provided that the covariates are available.

2.5. Estimation Procedure: a Variational EM

The SBM is a latent state space model which can be seen as a mixture model for random graphs. Therefore, the EM algorithm (Dempster, Laird, and Rubin 1977) is the natural choice for the inference since it generally proves very useful for inferring various types of mixture models. It is based on the evaluation of the expectation of the complete log-likelihood of the model, with respect to the conditional distribution of the latent variables given the data. However, this expectation is intractable in the SBM due to the structure of dependency between the latent variables $\mathbf{Z}$ and the network $\mathbf{Y}$. In fact, it would require to sum over all possible clusterings for all pairs of nodes, which is out of reach even for a moderate number of nodes or blocks. To address this shortcoming when the network $\mathbf{Y}$ is fully observed, Daudin *et al.* (2008) introduced a *variational* EM, based on the variational principles of Jordan, Ghahramani, Jaakkola, and Saul (1998). The idea is to maximize a lower bound of the log-likelihood based on an approximation of the true conditional distribution of the latent variable $\mathbf{Z}$. In the case of an SBM with missing data, the level of difficulty is higher since the set of latent variables encompasses both $\mathbf{Z}$ (the latent blocks) and $\mathbf{Y}^{\mathrm{m}}$ (the missing dyads). We propose here a variational distribution of the conditional distribution $p_{\boldsymbol{\theta},\boldsymbol{\psi}}(\mathbf{Z}, \mathbf{Y}^{\mathrm{m}} | \mathbf{Y}^{\mathrm{o}})$ where complete independence is forced on $\mathbf{Z}$ and $\mathbf{Y}^{\mathrm{m}}$, using a multinomial, respectively a Bernoulli distribution for $\mathbf{Z}$ and $\mathbf{Y}^{\mathrm{m}}$. We denote by $m(\cdot)$ and $b(\cdot)$ the probability density functions of, respectively, the multinomial and the Bernoulli distributions which gives the following expression of the variational distribution:

$$\tilde{p}_{\boldsymbol{\tau},\boldsymbol{\nu}}(\mathbf{Z}, \mathbf{Y}^{\mathrm{m}}) = \tilde{p}_{\boldsymbol{\tau}}(\mathbf{Z})\ \tilde{p}_{\boldsymbol{\nu}}(\mathbf{Y}^{\mathrm{m}}) = \prod_{i \in \mathcal{N}} m(\mathbf{Z}_i; \tau_i) \prod_{(i,j) \in \mathcal{D}^{\mathrm{m}}} b(Y_{ij}; \nu_{ij}),$$

where $\boldsymbol{\tau} = \{\tau_i = (\tau_{i1}, \ldots, \tau_{i\mathcal{Q}}) \in [0,1]^{\mathcal{Q}} : \sum_{q=1}^{\mathcal{Q}} \tau_{iq} = 1, i \in \mathcal{N}\}$ and $\boldsymbol{\nu} = \{\nu_{ij} \in [0,1], (i,j) \in \mathcal{D}^{\mathrm{m}}\}$ are the two sets of variational parameters respectively associated with $\mathbf{Z}$ and $\mathbf{Y}^{\mathrm{m}}$. Interestingly, $\boldsymbol{\tau}$'s are proxies for the posterior probabilities of the group memberships for all nodes, and $\boldsymbol{\nu}$'s correspond to the imputed values of the missing dyads in the network data. This variational distribution was chosen in order to replace the intractable E-step of the EM

algorithm with a tractable variational E-step. This approximation leads to the following lower bound J of the log-likelihood, where KL is the Kullback-Leibler divergence between the true conditional distribution and its variational approximation:

$$\begin{aligned}\log p_{\boldsymbol{\theta},\boldsymbol{\psi}}(\mathbf{Y}^{\mathrm{o}}, \mathbf{R}) \geq \\ J_{\boldsymbol{\tau},\boldsymbol{\nu},\boldsymbol{\theta},\boldsymbol{\psi}}(\mathbf{Y}^{\mathrm{o}}, \mathbf{R}) &\triangleq \log p_{\boldsymbol{\theta},\boldsymbol{\psi}}(\mathbf{Y}^{\mathrm{o}}, \mathbf{R}) - \mathrm{KL}(\tilde{p}_{\boldsymbol{\tau},\boldsymbol{\nu}}(\mathbf{Z}, \mathbf{Y}^{\mathrm{m}}) \parallel p_{\boldsymbol{\theta}}(\mathbf{Z}, \mathbf{Y}^{\mathrm{m}} \| \mathbf{Y}^{\mathrm{o}})), \\ &= \mathbb{E}_{\tilde{p}_{\boldsymbol{\tau},\boldsymbol{\nu}}}\left[\log p_{\boldsymbol{\theta},\boldsymbol{\psi}}(\mathbf{Y}^{\mathrm{o}}, \mathbf{R}, \mathbf{Y}^{\mathrm{m}}, \mathbf{Z})\right] - \mathbb{E}_{\tilde{p}_{\boldsymbol{\tau},\boldsymbol{\nu}}}\left[\log \tilde{p}_{\boldsymbol{\tau},\boldsymbol{\nu}}(\mathbf{R}, \mathbf{Y}^{\mathrm{m}})\right].\end{aligned}$$

If we choose $\tilde{p} = p_{\boldsymbol{\theta},\boldsymbol{\psi}}(\mathbf{Z}, \mathbf{Y}^{\mathrm{m}} | \mathbf{Y}^{\mathrm{o}})$, the true conditional distribution of the latent variables $\mathbf{Z}, \mathbf{Y}^{\mathrm{m}}$, we retrieve the standard EM algorithm, requiring the evaluation of the intractable quantity $\mathbb{E}_{p_{\boldsymbol{\theta},\boldsymbol{\psi}}(\mathbf{Z},\mathbf{Y}^{\mathrm{m}}|\mathbf{Y}^{\mathrm{o}})}\left[\log p_{\boldsymbol{\theta},\boldsymbol{\psi}}(\mathbf{Y}^{\mathrm{o}}, \mathbf{R}, \mathbf{Y}^{\mathrm{m}}, \mathbf{Z})\right]$. Note that we alleviated the notations above by not explicitly writing the possible conditioning on covariates in the log-likelihoods.

Based on this approximation, the variational EM algorithm consists in alternating updates of the variational parameters $\{\boldsymbol{\tau}, \boldsymbol{\nu}\}$ (the VE-step) with updates of the model parameters $\boldsymbol{\theta}, \boldsymbol{\psi}$ (the M-step) maximizing J. Steps VE and M are iterated until convergence like in a standard EM. The algorithm converges to a local maximum of the lower bound of the log-likelihood. This variational is translated into a collection of algorithms for handling missing data with all sampling designs introduced in Section 2.4. When an algorithm reaches convergence, we obtain estimates of the parameters involved in the SBM ($\boldsymbol{\theta}$), in the sampling process ($\boldsymbol{\psi}$), and also estimates of the variational parameters which bring information on the clustering ($\boldsymbol{\tau}$) and on the missing dyads ($\boldsymbol{\nu}$). The variational estimator in the SBM is proven to be asymptotically normal in Bickel, Choi, Chang, Zhang *et al.* (2013) when the network is fully observed. The extension to the MCAR case is proven in Mariadassou, Tabouy *et al.* (2020).

Initialization. It is well known that EM-like algorithms have a great sensitivity to the initialization step, which therefore requires a special attention. In **missSBM**, the initial clustering is obtained by applying the popular Absolute Eigenvalues Spectral Clustering (detailed in Rohe, Chatterjee, and Yu 2011) to the adjacency matrix where the `NA`'s are replaced by zero. The initial clustering can also be provided by the user. As specified in the next paragraph, the exploration of the number of blocks also provides several other relevant initializations.

Selection of the Number of Blocks. A main difficulty met when conducting SBM inference lies in the estimation of the number of blocks, generally unknown to the user. To remedy this problem, we use the Integrated Classification Likelihood (ICL) criterion defined in Biernacki, Celeux, and Govaert (2000) and routinely used in the framework of mixture models. Note that Saldana, Yu, and Feng (2017), Wang, Bickel *et al.* (2017), Hu, Zhang, Qin, Yan, and Zhu (2020) and Côme, Jouvin, Latouche, and Bouveyron (2021) propose alternative methods for selecting the number of blocks. The ICL criterion was adapted to the SBM under missing data condition in Tabouy *et al.* (2020), and is recalled here: for a model with Q blocks, a sampling design parametrized by K parameters (size of $\boldsymbol{\psi}$) and $(\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\psi}}) = \arg\max_{(\boldsymbol{\theta},\boldsymbol{\psi})} \log p_{\boldsymbol{\theta},\psi}(\mathbf{Y}^{\mathrm{o}}, \mathbf{Y}^{\mathrm{m}}, \mathbf{R}, \mathbf{Z})$, then

$$\mathrm{ICL}(Q) = -2\mathbb{E}_{\tilde{p}_{\boldsymbol{\tau},\boldsymbol{\nu}}}\left[\log p_{\hat{\boldsymbol{\theta}},\hat{\boldsymbol{\psi}}}(\mathbf{Y}^{\mathrm{o}}, \mathbf{Y}^{\mathrm{m}}, \mathbf{R}, \mathbf{Z} | Q, K)\right] + \mathrm{pen}_{\mathrm{ICL}}(Q, K),$$

$$\mathrm{pen}_{\mathrm{ICL}} = \begin{cases} \left(K + \frac{Q(Q+1)}{2}\right) \log\left(\frac{n(n-1)}{2}\right) + (Q-1)\log(n) & \text{for dyad-centered sampling,} \\ \frac{Q(Q+1)}{2} \log\left(\frac{n(n-1)}{2}\right) + (K+Q-1)\log(n) & \text{for node-centered sampling.} \end{cases}$$

We also implemented in **missSBM** an exploration procedure designed to avoid getting stuck in local minima by producing a convex and robust ICL curve. This exploration procedure is divided into two steps, forward and backward: the forward step creates new initializations for each number Q of block considered, by splitting blocks obtained from estimations with $Q-1$ blocks. On the other hand, the backward step tries new initializations for each Q by merging groups of the model with $Q+1$ groups. The best model in terms of ICL is always retained. The procedure can be iterated until a satisfying shape of the ICL curve is met.

Implementation Details. **missSBM** adopts an oriented-object programming spirit for representing most models by means of R6-classes and the **R6** package of Chang (2017), which paves the way for future extensions. This approach is non-visible to the user, who essentially only has to deal with classical R functions. The time consuming pieces of code are written in C++ using the **armadillo** library for linear algebra (Sanderson and Curtin 2016), in conjunction with the **Rcpp** and **RcppArmadillo** packages (Eddelbuettel and François 2011; Eddelbuettel and Sanderson 2014) to interface C++ with R. The numerical performance of our implementation is of the same order as existing variational implementations of the binary SBM (like **blockmodels**). It allows to deal with networks with up to a couple of thousands of nodes. To give an idea to the reader, Figure 2 reports the timings for adjusting an SBM to the network data considered in Section 4 (the 2007 French political blogosphere, 194 nodes), for a varying number of blocks with a single Intel Core i9-9900 CPU at 3.10GHz, replicated 50 times per block-size. We report in the discussion some interesting tracks for improving the speed in the future and eventually tackle larger networks.

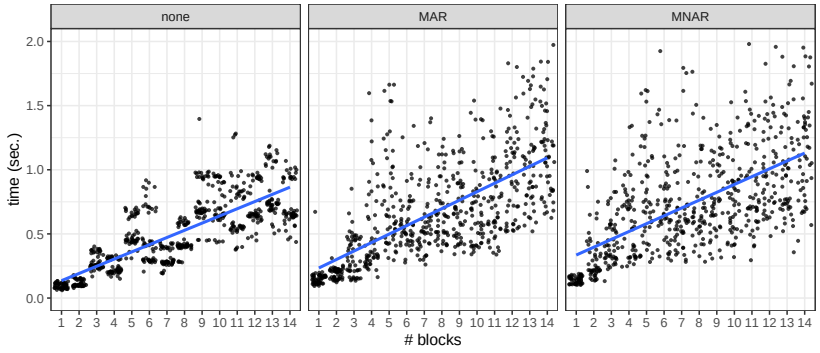


Figure 2: Timings for adjusting binary SBM with **missSBM** on the French political blogosphere with a single core for a varying number of blocks (50 replicated runs per # blocks).

3. Guidelines for Users

This section gives an overview of the basic usage of the package. In particular, it describes inputs and outputs of the main functions at play in the procedure that fits an SBM from partially observed network data. Along the section, we also describe the classes of objects included in the package to ease the manipulation of the resulting fitted models. As a complement to this section and to the usual R package documentation, a full documentation including

a vignette is available as a **pkgdown** website at URL http://grosssbm.github.io/missSBM.

3.1. Parameter Estimation, Prediction and Clustering

Estimation of an SBM from a partially observed network is done by means of the function `estimateMissSBM`, with the following usage:

```
estimateMissSBM(
  adjacencyMatrix    , # observed network data: matrix with NA entries
  vBlocks            , # vector of block numbers for model exploration
  sampling   = "dyad", # string describing the sampling design
  covariates = list(), # optional list of covariates (dyad or nodal)
  control    = list()) # optional list for tuning the algorithm
```

Standard Usage. This function takes as a first argument a square `base::matrix` or a sparsely encoded `Matrix::dgCMatrix`, possibly with `NA` entries corresponding to missing (unobserved) values of the network. The second argument `vBlocks` contains the successive explored values for the number of blocks, this number being generally unknown. The third argument – `sampling` – specifies how `NA` entries are taken into account in the estimation. It must be one of the following character strings, corresponding to one of the sampling designs (or observation processes) depicted in Section 2.4:

```
missSBM:::available_samplings

## [1] "dyad"            "covar-dyad"      "node"            "covar-node"
## [5] "block-node"      "block-dyad"      "double-standard" "degree"
## [9] "snowball"
```

Argument `covariates` is an optional list with as many entries as covariates. If the covariates are nodal, each entry must be a size-n vector; if the covariates are defined between pairs of nodes, each entry must be an $n \times n$ matrix.

Advanced Tuning. The argument `control` is a `list` to control the estimation and finely tune the variational EM algorithm, with the following entries:

i) `useCov`: Logical indicating whether the covariates should be incorporated within the SBM (or just in the sampling);

ii) `clusterInit`: Initial method for clustering. Either a character ("spectral") or a list with `length(vBlocks)` vectors, each with size `ncol(adjacencyMatrix)`, providing a user-defined clustering. Default is "spectral", for absolute spectral clustering;

iii) `similarity`: An $\mathbb{R} \times \mathbb{R} \mapsto \mathbb{R}$ function to compute similarities between nodal covariates (default is `missSBM:::l1_similarity`, that is, $(x, y) \mapsto -|x - y|$);

iv) `threshold`: Optimization stops when a V-EM step changes the objective function or the connection parameters by less than `threshold` (default is 1.10^{-2});

v) `maxIter`: Optimization stops when the number of iteration exceeds `maxIter` (default is 50);

vi) `fixPointIter`: Number of iterations in the fixed-point algorithm used to solve the variational E step (default is 3);

vii) `exploration`: Character indicating what kind of exploration should be used among `"forward"`, `"backward"`, `"both"` or `"none"` (default is `"both"`);

viii) `iterates`: Integer for the number of iterations during the exploration process. Only relevant when `exploration` is different from `"none"` (default is 1);

ix) `trace`: Logical for verbosity (default is `TRUE`).

Return Value: Classes `missSBM_collection` and `missSBM_fit`. An output of the function `estimateMissSBM` is an instance of an **R6** object with class `missSBM_collection`, which can be handled as a standard `list`. Its fields are described in Table 1.

Field	**Description**
`models`	a `list` of models with class `missSBM_fit`
`ICL`	the `vector` of ICL associated to the models in the collection
`bestModel`	best model according to the ICL (a `missSBM_fit` object)
`optimizationStatus`	a `data.frame` summarizing the optimization process for all models
Method	**Description**
`plot(object, type)`	plot either `"icl"`, `"elbo"` or `"monitoring"`

Table 1: Structure of `missSBM_collection` (output of `estimateMissSBM`).

Among the fields of `missSBM_collection`, `models` is a list with as many elements as in `vBlocks`. These elements are **R6** objects with class `missSBM_fit`, the fields of which are detailed in Table 2. It gives access to the results of the inference for a fixed number of blocks.

We finally give additional details on fields `fittedSBM` and `fittedSampling` in Table 3. Note that `fittedSBM` enjoys some standard `plot`, `print`, `coef`, `predict` and `fitted` methods, most of them inherited from the class `simpleSBM_fit` of the package **sbm**.

Models Exploration. At the end of the estimation process, it is common that the algorithm gets stuck in some local minima for some values of Q, the current number of blocks. The consequence is "non-smooth" ICL curve which is theoretically convex and rather smooth. This may lead the user to choose a sub-optimal number of blocks. Thus, the ICL criterion is automatically "smoothed" after a first pass across all the models. The idea is to apply a split and merge strategy to the path of models stored in `missSBM_collection` in order to find a better initialization for each value of Q in the V-EM algorithm, so that it converges to the global minimum. This model exploration can be tuned with the `control` argument, see paragraph *Advanced tuning*. The default goes back and forth a single time.

Field	**Description**	**R6 object / class**
`fittedSBM`	the adjusted Stochastic Block Model	`simpleSBM_fit_missSBM`
`fittedSampling`	the estimated sampling process	`networkSampling`
`imputednetwork`	the adjacency matrix with imputed values	`dgCMatrix`
`monitoring`	status of the optimization process	`data.frame`
`vICL`	ICL criterion associated to Q	`double`
`vBound`	value of the variational bound $\mathcal{J}_{\boldsymbol{\tau},\boldsymbol{\theta}}$ at each step	`double`
`vExpec`	value of $\mathbb{E}_{\tilde{p}_{\boldsymbol{\tau}}}[\log(p_{\boldsymbol{\theta}}(\mathbf{Y},\mathbf{Z}))]$ at each step	`double`
`penalty`	penalty of the model with Q blocks	`double`
Method	**Description**	
`plot(object, type)`	plot either `"imputed"`, `"expected"`, `"meso"` or `"monitoring"`	
`print(object)`	a print method recalling important fields and methods available	
`coef(object, type)`	model parameters (`type="mixture"`, `"connectivity"`, `"covariates"` or `"sampling"`)	
`predict(object)`	estimated adjacency matrix (with imputation)	
`fitted(object)`	expected value of the SBM	

Table 2: Selection of fields in object `missSBM_fit` with descriptions and types.

R6 object	**Field**	**Description**	**Correspondence**
`fittedSBM`	`probMemberships`	estimated probability of block belonging	$\{\boldsymbol{\tau}_i\}_{i\in\mathcal{N}}$
	`connectParam`	connectivity matrix	$\hat{\boldsymbol{\pi}}$
	`expectation`	imputed adjacency matrix	$\mathbf{Y}^o \cup \{\nu_{ij}\}_{(i,j)\in\mathcal{D}^m}$
	`covarParam`	regression parameter	$\hat{\boldsymbol{\beta}}$
	`memberships`	vector of blocks memberships	$(\texttt{which.max}(\boldsymbol{\tau}_i))_{i\in\mathcal{N}}$
	`blockProp`	block proportions	$\hat{\boldsymbol{\alpha}}$
`fittedSampling`	`parameters`	sampling parameter	$\hat{\boldsymbol{\psi}}$

Table 3: Selection of fields in `fittedSBM`, `fittedSampling` and mathematical counterparts.

Parallel Computing. Some internal components of `estimateMissSBM` (initialization, estimation, exploration) use the **future** framework (Bengtsson 2020) to speed-up the whole process with parallel computing. The `future::plan` must be set by the user (default is sequential with no parallelism). For instance, in order to run **missSBM** on 10 cores with forking on Unix systems, the following command should be used before calling `estimateMissSBM()`.

```
future::plan("multicore", workers = 10)
```

3.2. Partial Observation ("sampling") of Some Network Data

The function `observeNetwork` generates missing entries in a fully observed adjacency matrix according to a given network sampling design. The usage is the following:

```
observeNetwork(
  # the original adjacency matrix of the network to be partially observed
  adjacencyMatrix,
  # a character for the sampling design which defines the observation process
  sampling,
  # a set of parameters associated to the sampling design
  parameters,
  # an optional clustering membership vector required for block samplings
  clusters = NULL,
  # an optional list of covariates that may influence the observation process
  covariates = list(),
  # an optional function to compute similarities between node covariates
  similarity = l1_similarity,
  # an optional intercept term added in case of the presence of covariates
  intercept = 0)
```

Note that the dimension (or `length`) of `parameters` depends on the sampling design selected, as described in Section 2.4. Argument `clusters` only needs to be specified for `"block-dyad"` and `"block-node"` sampling designs. Arguments `covariates` is by-default `list()`, `covarSimilarity` is set to the `l1_similarity` function defined by $(x, y) \in \mathbb{R}^d \times \mathbb{R}^d \mapsto -|x - y|_{\ell^1} \in \mathbb{R}^d$, and finally `intercept` is set to 0. These last three arguments only need to be specified in the case of an SBM with covariate(s). Note that the intercept is not included in `parameters` and must be specified independently. The output of `observeNetwork` is a `matrix` encoding the adjacency-matrix, with `NA` values for dyads not observed by the observation process, which can be provided as input to `estimateMissSBM`.

4. Illustration: the 2007 French political blogosphere

This section illustrates the features of **missSBM** by conducting the analysis of a real-world network data. It is a sub-network of the French political blogosphere, extracted from a snapshot of over 1100 blogs collected during a period preceding the 2007 French presidential election, and manually classified by the "Observatoire Présidentielle project" (see Zanghi,

Ambroise, and Miele 2008). The network is composed of 194 blogs representing nodes in the network and 1432 edges indicating that at least one of the two blogs references the other. On top of **missSBM**, our analysis relies on **aricode** (Chiquet, Dervieux, Rigaill, and Sundqvist 2020) for computing clustering comparison measures and **tidyverse** for performing data manipulations and producing graphical outputs. The **igraph** package, imported by **missSBM**, is needed for basic graph manipulations. The **future** package is used for parallel computing. We also fix the seed for reproducibility:

```
library(missSBM)
library(aricode)
library(tidyverse); theme_set(theme_bw())
library(igraph)
library(future)
set.seed(03052008)
```

We set our `future::plan` to `multicore` with 10 workers [3].

```
future::plan("multicore", workers = 10)
```

The `frenchblog2007` data set is provided with **missSBM**[4] as an igraph object. We extract the adjacency matrix corresponding to the blog network after removing the isolated nodes. We also extract the political party of each blog from the vertex attribute `party`, which gives us a natural classification of the nodes that could be used as a reference in our analyses.

```
data("frenchblog2007", package = "missSBM")
frenchblog2007 <-
  delete_vertices(frenchblog2007, which(degree(frenchblog2007) == 0))
blog  <- as_adj(frenchblog2007)
party <- vertex.attributes(frenchblog2007)$party
```

For this network, we explore models with a number of blocks varying from 1 to 18:

```
blocks <- 1:18
```

Standard SBM Estimation. At this stage, the data set has no missing entry: every dyad and every node is observed. The adjacency matrix $\mathbf{Y}$ of the fully-observed network is stored in the variable `blog`. We first perform a standard SBM estimation on the fully observed network, including smoothing of the ICL.

```
sbm_full <- estimateMissSBM(blog, blocks, "node")
```

We inspect the result of the optimization process by plotting the ELBO (variational lower bound of the log-likelihood) against the number of iterations in the V-EM algorithm, accumulated along all the numbers of blocks considered (Figure 3). The ELBO is typically expected

[3] Use `multisession` if you work on Windows or from Rstudio

[4] Earlier versions of this data set were available in packages **mixer** and **sand**.

to increase with the number of blocks at the end of the successive optimizations, which is indeed the case here:

```
plot(sbm_full, type = "monitoring")
```

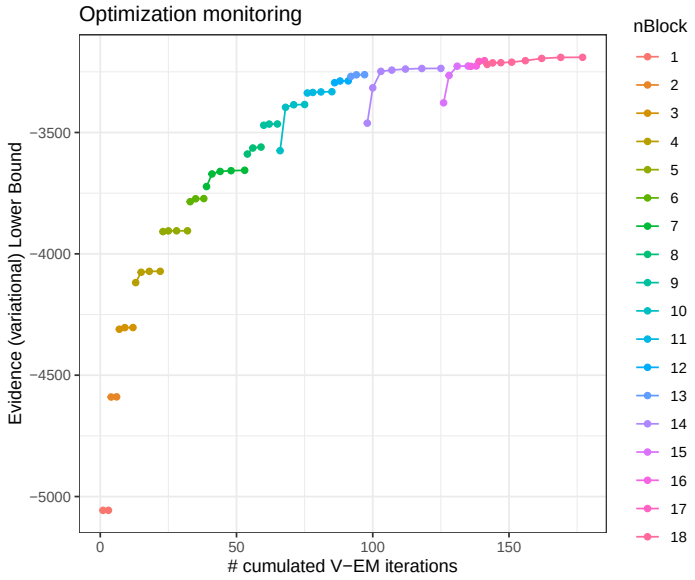


Figure 3: Evolution of the ELBO during the optimization for the successive numbers of blocks

The ICL criterion is minimal for an SBM with 10 blocks:

```
which.min(sbm_full$ICL)

## [1] 10
```

The corresponding model is stored in the field `$bestModel` of `sbm_full` as an object with class `missSBM_fit`. Printing this object results in a summary of the most important accessible fields and methods:

```
sbm_full$bestModel

## missSBM-fit
## ========================================================================
## Structure for storing an SBM fitted under missing data condition
## ========================================================================
## * Useful fields (first 2 special objects themselves with methods)
##   $fittedSBM (the adjusted stochastic block model)
##   $fittedSampling (the estimated sampling process)
##   $imputedNetwork (the adjacency matrix with imputed values)
##   $monitoring, $ICL, $loglik, $vExpec, $penalty
## * S3 methods
##   plot, coef, fitted, predict, print
```

In particular, one can access various parameters (e.g. block proportion/mixture parameters),

```
coef(sbm_full$bestModel, type = "mixture")

##  [1] 0.02741279 0.03589227 0.08965712 0.06583498 0.14612002 0.12372170
##  [7] 0.05670103 0.12990416 0.13922946 0.18552647
```

or plot diverse outputs, for instance a matrix view of the original network with columns and rows reordered according to the block memberships of the nodes, see Figure 4).

```
plot(sbm_full$bestModel, dimLabels = list(row = "blogs", col = "blogs"))
```

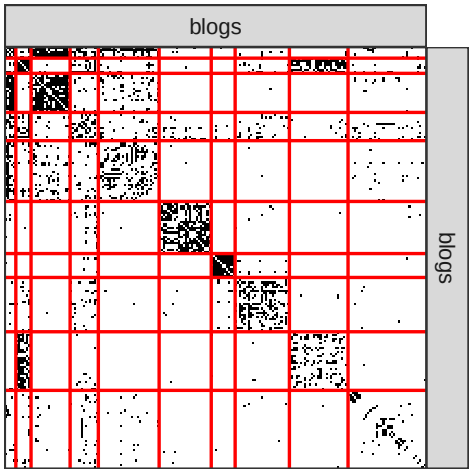


Figure 4: Network data reorganized by the estimated block memberships

Partial Observation of an Existing Network. For illustrative purposes, we sample a sub-graph from the blog network to mimic missing data and create a new adjacency matrix with missing entries. Since data consists in interactions between blogs (who references who), it is natural to sample the network with a node-centered sampling design: the following code generates a realization of a 194×194 sampling matrix according to the *block-node sampling*, where the blocks correspond to the clustering estimated by the SBM fitted on the full data set. The sampling rate is either low (0.2) in small blocks or high (0.8) in large blocks.

```
samplingParameters <-
  ifelse(sbm_full$bestModel$fittedSBM$blockProp < .1, 0.2, .8)
blog_obs <-
  observeNetwork(
    adjacencyMatrix = blog,
    sampling        = "block-node",
```

```
    parameters      = samplingParameters,
    clusters        = sbm_full$bestModel$fittedSBM$memberships
  )
```

Estimation of a Partially Observed Network. We now perform SBM inference under missing data conditions by fitting two types of model: first, the SBM under the MNAR *block-node* sampling design, i.e., under the design that truly generated the missing entries; second, the SBM under the MCAR *node* sampling design which basically performs inference only on the observed part of the network, neglecting the process that generates the missing values. The estimation is run on both models with the same settings as for the fully observed data. The ICL curve is smoothed with five iterations of forward-backward exploration since the presence of missing values typically increases the chances for falling into local minima.

```
sbm_block <-
  estimateMissSBM(blog_obs, blocks, "block-node", control = list(iterates=5))
sbm_node <-
  estimateMissSBM(blog_obs, blocks, "node", control = list(iterates=5))
```

Sampling Design Comparison. We plot on Figure 5 the ICL of the three models (`"fully observed"`, `"block-node"` sampling and `"node"` sampling) to show how it can be compared to select which sampling design fits at best the data:

```
rbind(
  tibble(Q = blocks, ICL = sbm_node$ICL , sampling ="node"),
  tibble(Q = blocks, ICL = sbm_block$ICL, sampling = "block-node"),
  tibble(Q = blocks, ICL = sbm_full$ICL , sampling = "fully observed")
) %>% ggplot(aes(x = Q, y = ICL, color = sampling)) +
  geom_line() + geom_point() + ggtitle("Model Selection") +
  labs(x = "# blocks", y = "Integrated Classification Likelihood")
```

Note that the curves associated with the *block-node sampling* and the *node sampling* are quite close for small numbers of blocks (less than 10) and then depart from each other: the choice of the sampling design is a tough question for the network data at play. We also represent the ICL curve for the collection of SBM estimated on the fully observed network: although the values of ICL cannot be compared between this model and the two obtained on the partially observed network (data sets are not the same), we underline that the numbers of blocks selected in the different cases remain comparable. A model with 10 blocks is selected with the fully observed network, while a model with only 9 blocks is chosen for the block-node sampling SBM. Indeed, due to the partial sampling, some blocks are less well represented than others, and it seems more likely to gather some blocks together considering the information available. Regarding the clustering obtained by the three variants, we compare them with the Adjusted Rand Index (ARI, Rand 1971) computed with the **aricode** package (Chiquet *et al.* 2020). We use the classification of the fully-observed SBM as a reference, since its clustering was used to sample the network with the block-node sampling design. We typically expect

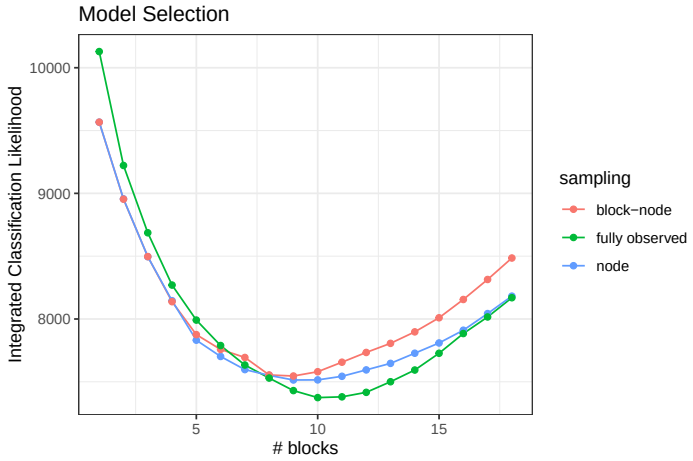


Figure 5: ICL for models with fully observed data, block-node sampling and node sampling

that a model relying on a better modeling of the missing values shall lead to a clustering closer to the reference. This is indeed the case when looking at the next piece of code, where it is shown that the ARI with the reference clustering is higher for the block-sampling SBM than for the MCAR node sampling SBM:

```
ARI(sbm_block$bestModel$fittedSBM$memberships,
    sbm_full$bestModel$fittedSBM$memberships)

[1] 0.6570555

ARI(sbm_node$bestModel$fittedSBM$memberships ,
    sbm_full$bestModel$fittedSBM$memberships)

[1] 0.5436008
```

Extraction of the SBM with Block-Sampling Design. The model that we finally retain is thus a block-sampling with 9 blocks.

```
myModel <- sbm_block$bestModel
```

`myModel` is an object with class `missSBM_fit` with two special elements used for storing the results of the estimation of both the SBM (field `fittedSBM`) and the sampling design (`fittedSampling`). These two elements are special objects themselves with dedicated fields and methods which are recalled to the user thanks to the `print`/`show` methods:

```
myModel$fittedSBM

Simple Stochastic Block Model -- bernoulli variant
```

```
=============================================================================
Dimension = ( 194 ) - ( 9 ) blocks and no covariate(s).
=============================================================================
* Useful fields
  $nbNodes, $modelName, $dimLabels, $nbBlocks, $nbCovariates, $nbDyads
  $blockProp, $connectParam, $covarParam, $covarList, $covarEffect
  $expectation, $indMemberships, $memberships
* R6 and S3 methods
  $rNetwork, $rMemberships, $rEdges, plot, print, coef
```

```
myModel$fittedSampling

block-node-model for network sampling
=============================================================================
Structure for handling network sampling in missSBM.
=============================================================================
* Useful fields
  $type, $parameters, $df
  $penalty, $vExpec
```

Representation and Validation. With `myModel`, we now have at hand a tool for analyzing the clustering of the French political blogosphere. The first output is the connectivity matrix of the network, which puts into light the community structure of the blogosphere. Indeed, it is revealed by a diagonal filled with high probabilities and off-diagonal with low probabilities. Thus, nodes (blogs) in blocks connect with a high probability with other nodes in the same block and with a low probability with nodes in other blocks. Such a network concentrates most of its edges between nodes of the same blocks. This can be seen by displaying the probability of connection predicted by the SBM at the whole network scale (see Figure 6):

```
plot(myModel, type = "expected", dimLabels = list(row="blogs", col="blogs"))
```

For validation, we suggest comparing the clustering of the model with the node attribute corresponding to the political parties to which the blogs belong. First, we remark that the SBM fitted on missing entries carries a little bit less information regarding the political party than the SBM adjusted on the fully observed network:

```
ARI(party, myModel$fittedSBM$memberships)

[1] 0.4126328

ARI(party, sbm_full$bestModel$fittedSBM$memberships)

[1] 0.463709
```

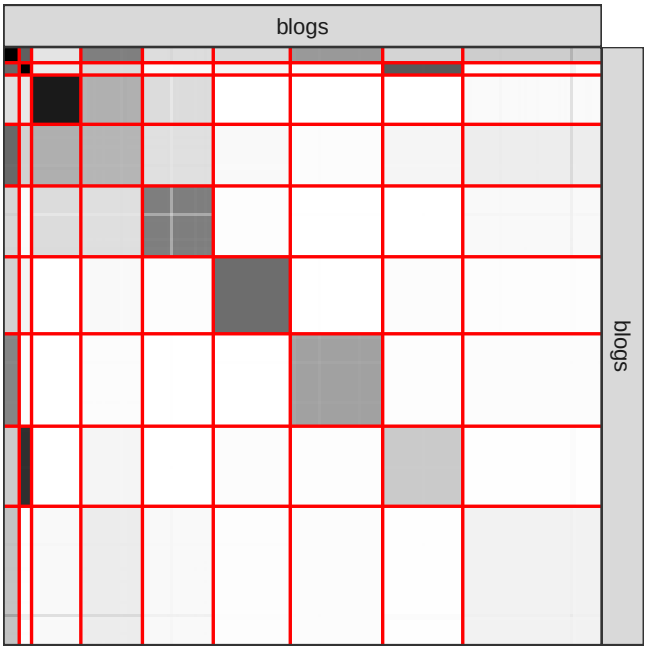


Figure 6: Probabilities of connection predicted by the SBM with block-node sampling

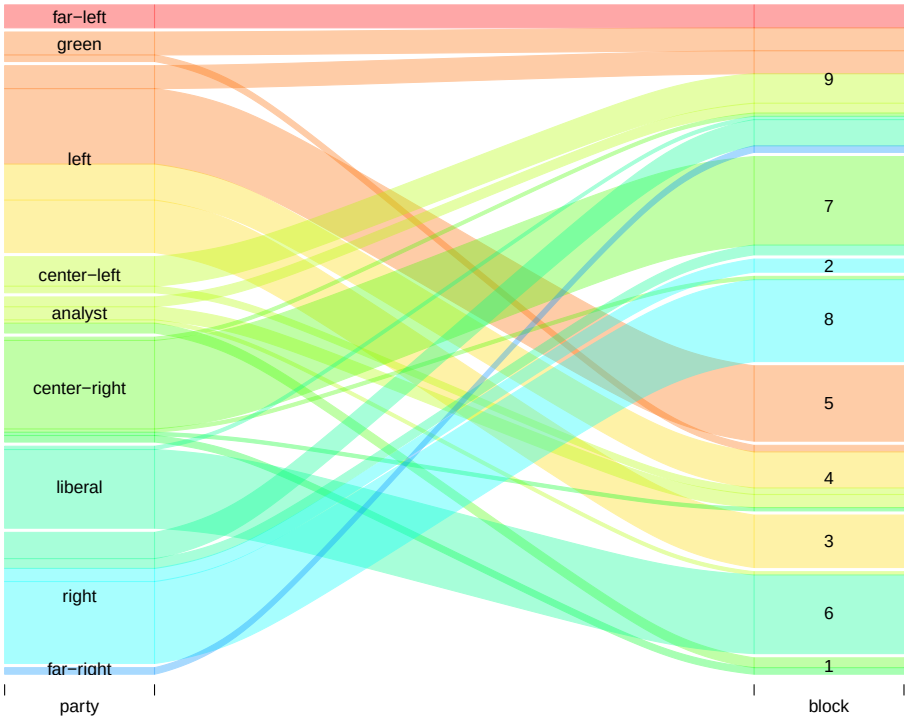


Figure 7: Alluvial plot between block-node sampling clustering and political parties.

A more detailed comparison between blocks inferred by the SBM and political parties is reported in Figure 7 with an alluvial diagram.

Also remember that **missSBM** performs imputation of the missing dyads in the adjacency matrix. Thus, we can compare the imputed values with the values of the dyad in the fully observed network to validate the performance of our approach. Using the R package **pROC** (Robin, Turck, Hainard, Tiberti, Lisacek, Sanchez, and Muller 2011), we check the quality of the imputation. We replicate this experiment 500 times with a sampling rate varying between ≈ 0.4 and ≈ 0.9 (always with block-sampling design), fixing the number of blocks to the best one found on the fully observed network. The Area Under the Curve (AUC) is plotted in Figure 8 against the sampling rate, showing the robustness and the good performance of the imputation method.

```
library(pROC)
library(parallel)
cl0     <- sbm_full$bestModel$fittedSBM$memberships
nBlocks <- sbm_full$bestModel$fittedSBM$nbBlocks
future::plan("sequential")
res_auc <- mclapply(1:500, function(i) {
  subGraph   <- observeNetwork(blog, "block-node", runif(nBlocks), cl0)
  missing    <- which(as.matrix(is.na(subGraph)))
  true_dyads <- blog[missing]
  sbm_block  <- estimateMissSBM(subGraph, nBlocks, "block-node",
                    control = list(cores = 1, trace = 0))
  imputed_dyads <- sbm_block$bestModel$imputedNetwork[missing]
  c(rate = 1 - length(missing)/length(blog),
    auc  = auc(true_dyads, imputed_dyads, quiet = TRUE))
}, mc.cores = 10)

purrr::reduce(res_auc, rbind) %>% as.data.frame() %>%
  ggplot() + aes(x = rate, y = auc) + geom_point(size = 0.25) +
  geom_smooth(method = "lm", formula = y ~ x, se = FALSE) +
  labs(x = "sampling rate", y = "Area under the ROC curve")
```

Dealing with Covariates. In order to illustrate how we can deal with covariates in the observation process, we now consider a sampling which depends on the political parties of the blog. For illustrative purposes, we extract a sub-graph that only contains the nodes whose political party is either `left` or `right`:

```
blog_subgraph <-
  frenchblog2007 %>%
  igraph::induced_subgraph(V(frenchblog2007)$party %in% c( "right", "left"))
blog_subgraph <-
  delete_vertices(blog_subgraph, which(degree(blog_subgraph) == 0))
```

The sub-graph is given in Figure 9, generated with the following piece of code:

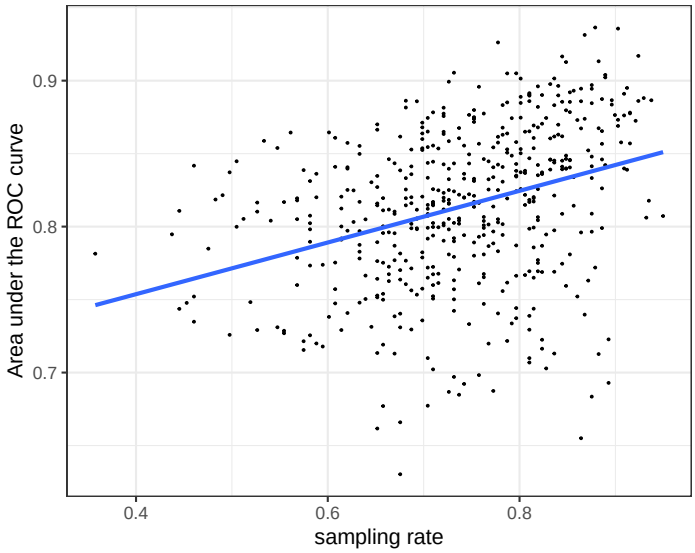


Figure 8: Area Under the Curve (AUC) of the imputation as a function of the sampling rate.

```
plot(blog_subgraph, vertex.shape="none", vertex.label=V(blog_subgraph)$party,
                    vertex.label.color = "steel blue", vertex.label.font=1.5,
                    vertex.label.cex=.6, edge.color="gray70", edge.width = 1)
```

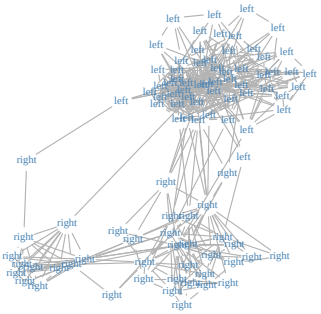


Figure 9: Subnetwork extracted from the French political blogosphere (only blogs with attribute party in $\{\text{left}, \text{right}\}$ were kept)

We then build a simple binary covariate on nodes indicating its party (1/0 = left/right).

```
dummy_party <- (V(blog_subgraph)$party == "left") * 1
```

Now, the observation process of the network is assumed to depend on this covariate so that a node belonging to the `left` party is more likely to be observed than a node belonging to the `right` party:

```
blog_subgraph_obs <-
  blog_subgraph %>% as_adj() %>%
  observeNetwork(sampling="covar-node", parameters = 10,
                 covariates = list(dummy_party))
blocks <- 2:8
future::plan("multicore", workers = 10)
```

Since the covariate is nodal, a nodal observation process is considered. When it comes to take into account the effect of this covariate on the probability of connection between two nodes in the SBM as in Equation (2), the covariate has to be transferred at the dyad level. This consists in computing an $n \times n$ matrix whose elements are equal to one if the corresponding two nodes belong to the same political party, zero otherwise. This matrix computation is directly handled in **missSBM**, when the option `useCov` is `TRUE`.

From the observed network `blog_subgraph_obs`, four modeling choices (labeled i to iv) are possible whether the covariate is taken into account in the SBM or not, and whether the sampling is set as depending on the covariate or not. For the latter choice, we know that the sampling which depends on the covariate is more appropriate but this information is generally not available *a priori*. We then run the estimation in these four scenarios below and print the estimated parameters related to the SBM and the sampling.

i) Take the covariate into account in both the sampling and the SBM:

```
sbm_covar1 <- estimateMissSBM(blog_subgraph_obs, blocks,
  "covar-node", covariates = list(dummy_party),
  control = list(useCov = TRUE, iterates = 2))
sbm_covar1$bestModel$fittedSampling$parameters # sampling parameters
sbm_covar1$bestModel$fittedSBM$covarParam      # regression parameter
```

```
## [1] -0.3629055  2.9469030
## [1] 4.945801
```

ii) Take the covariate into account in the sampling only:

```
sbm_covar2 <- estimateMissSBM(blog_subgraph_obs, blocks,
  "covar-node", covariates = list(dummy_party),
  control = list(useCov = FALSE, iterates = 2))
sbm_covar2$bestModel$fittedSampling$parameters # sampling parameters
```

```
## [1] -0.3629055  2.9469030
```

iii) Take the covariate into account in the SBM only:

```
sbm_covar3 <- estimateMissSBM(blog_subgraph_obs, blocks,
  "node", covariates = list(dummy_party),
  control = list(useCov = TRUE, iterates = 2))
sbm_covar3$bestModel$fittedSampling$parameters # sampling parameter
sbm_covar3$bestModel$fittedSBM$covarParam      # regression parameter
```

```
## [1] 0.71875
## [1] 4.945801
```

iv) Ignore the covariate in both the sampling and the SBM:

```
sbm_covar4 <- estimateMissSBM(blog_subgraph_obs, blocks,
  "node", control = list(useCov = FALSE, iterates = 2))
sbm_covar4$bestModel$fittedSampling$parameters # sampling parameter
```

```
## [1] 0.71875
```

For models *i*) and *ii*), we notice that the estimates of the sampling parameters are quite accurate. For models *iii*) and *iv*), the estimation of the unique sampling parameter boils down to the proportion of observed nodes. For models *i*) and *iii*), the estimate of the parameter β in Equation (2) is positive and quite high. Thus, we conclude that the probability of connection between two nodes is strengthened by the fact that the nodes belong to the same party.

Figure 10 shows clearly that the `"covar-node"` sampling is uncovered from the data since the ICL criterion favors this sampling no matter if the covariate is taken into account in the SBM or not. The ICL criterion also points out that the models including the covariate effect in the SBM are better.

```
rbind(
 tibble(Q=blocks, ICL=sbm_covar1$ICL, sampling="covar-node", useCov='true' ),
 tibble(Q=blocks, ICL=sbm_covar2$ICL, sampling="covar-node", useCov='false'),
 tibble(Q=blocks, ICL=sbm_covar3$ICL, sampling="node"      , useCov='true' ),
 tibble(Q=blocks, ICL=sbm_covar4$ICL, sampling="node"      , useCov='false')
) %>% ggplot(aes(x = Q, y = ICL, color = sampling, shape = useCov))  +
  geom_line() + geom_point() + labs(x = "#blocks", y = "ICLs")
```

Finally, we compare the clustering obtained with the four models together by computing pairwise ARIs. We also compare them with the clustering obtained by the SBM fitted on the fully observed network, adjusted as follows:

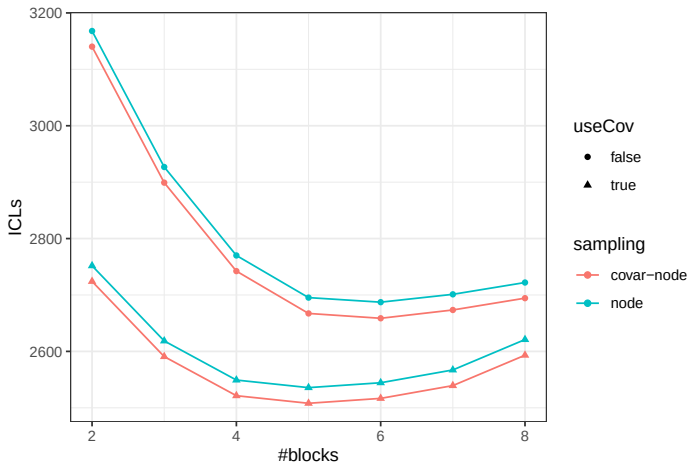


Figure 10: ICL criterion for different numbers of blocks under the four models which make different use of the covariate political party.

```
sbm_covar_full <- as_adj(blog_subgraph) %>%
  estimateMissSBM(blocks, "node", covariates =  list(dummy_party))
```

For the sake of clarity, the ARIs are displayed in Table 4. We obtain the same clustering with model *i*) and *iii*), which is the closest to the one obtained with the fully observed network. This is expected since these models also take into account the effect of the covariate.

Table 4: Clustering comparison with adjusted Rand indices (ARIs) between models taking the covariate into account (models *i*), *ii*), *iii*), *iv*) and model with fully observed network).

	i	ii	iii	iv
Full	0.64	0.48	0.64	0.44
i	NA	0.42	1.00	0.39
ii	NA	NA	0.42	0.95
iii	NA	NA	NA	0.39

5. Discussion

The R package **missSBM** enables the estimation of an SBM from partially observed binary networks even for some observation processes which generate MNAR data.

Although version `1.0.0` of **missSBM** only deals with binary networks, we deploy a structure that lets the possibility to easily include other variants in the future. In particular, extending **missSBM** to weighted SBM where the distribution on the edges belongs to the exponential family (e.g. Poisson distribution or Gaussian distribution, see Mariadassou *et al.* 2010) should be straightforward in the MAR case. To pave the way of such a generalization, **missSBM** relies on the package **sbm** which is designed to offer a collection of methods and algorithms to

manipulate more general SBM, yet *in the absence of missing data, or at least, with imputed data*. Hence, **missSBM** could take advantage in the future of improvements and new advances in **sbm**, while focusing on the handling of missing data specifically, by dealing only with the modeling of the observation process and the imputation of the missing entries. The modular object-oriented coding of **missSBM** also allows the developer to make it easily evolving in the way it can take into account the missing data: new observation processes could be incorporated by providing new sampling design which should contain the functions corresponding to the sampling specific steps in the VEM algorithm. Other dependency structures between the SBM and the covariates could also be modeled and integrated. For example, the latent variables $\mathbf{Z}$ could directly depend on the covariates $\mathbf{X}$ instead of the adjacency matrix $\mathbf{Y}$.

Some recent work is concerned with boosting the speed of the variational inference (see e.g. Blei, Kucukelbir, and McAuliffe 2017). This work could be adapted in the SBM context with or without missing data in order to enable our package to deal with larger networks. Resorting to minorization-maximization algorithms within the Variational E-step as in Vu *et al.* (2013) could also be interesting for speeding up the algorithm. Finally, a common drawback of the variational inference is to provide too narrow standard errors for the estimated parameters (Westling and McCormick 2019). Moreover, there is no uncertainty measure on the stability of the node clustering. We consider it as future work to provide the user with confidence intervals and stability measures of the clustering based on resampling techniques such as the bootstrap.

Acknowledgments

The authors thank all members of MIRES group for fruitful discussions on network sampling designs. This work is supported by public grants overseen by the French National research Agency (ANR) as part of the "Investissement d'Avenir" program, through the "IDI 2017" project funded by the IDEX Paris-Saclay, ANR-11-IDEX-0003-02, and second by the "EcoNet" project, ANR-18-CE02-0010.

References

Abbe E (2017). "Community detection and stochastic block models: recent developments." *The Journal of Machine Learning Research*, **18**(1), 6446–6531.

Albert R, Barabási AL (2002). "Statistical mechanics of complex networks." *Reviews of modern physics*, **74**(1), 47.

Ambroise C, Grasseau G, Hoebeke M, Latouche P, Miele V, Picard Fea (2015). *MixeR: Random Graph Clustering*. R package version 1.8.

Amini AA (2015). *sbmSDP: Semidefinite Programming for Fitting Block Models of Equal Block Sizes*. R package version 0.2, URL https://CRAN.R-project.org/package=sbmSDP.

Bengtsson H (2020). "A Unifying Framework for Parallel and Distributed Processing in R using Futures." 2008.00553, URL https://arxiv.org/abs/2008.00553.

Bickel P, Choi D, Chang X, Zhang H, *et al.* (2013). "Asymptotic normality of maximum likelihood and its variational approximation for stochastic blockmodels." *The Annals of Statistics*, **41**(4), 1922–1943.

Biernacki C, Celeux G, Govaert G (2000). "Assessing a mixture model for clustering with the integrated completed likelihood." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **22**(7), 719–725.

Binkiewicz N, Vogelstein JT, Rohe K (2017). "Covariate-assisted spectral clustering." *Biometrika*, **104**(2), 361–377.

Blei DM, Kucukelbir A, McAuliffe JD (2017). "Variational inference: A review for statisticians." *Journal of the American statistical Association*, **112**(518), 859–877.

Brusco M (2020). *dBlockmodeling: Deterministic Blockmodeling of Signed, One-Mode and Two-Mode Networks*. R package version 0.2.0, URL https://CRAN.R-project.org/package=dBlockmodeling.

Butts C (2008a). "network: A Package for Managing Relational Data in R." *Journal of Statistical Software*, **24**(2), 1–36. ISSN 1548-7660. doi:10.18637/jss.v024.i02.

Butts C (2008b). "Social Network Analysis with sna." *Journal of Statistical Software*, **24**(6), 1–51. ISSN 1548-7660. doi:10.18637/jss.v024.i06.

Chabert-Liddell SC (2021). *MLVSBM: A Stochastic Block Model for Multilevel Networks*. R package version 0.2.1, URL https://CRAN.R-project.org/package=MLVSBM.

Chang J (2015). *lda: Collapsed Gibbs Sampling Methods for Topic Models*. R package version 1.4.2, URL https://CRAN.R-project.org/package=lda.

Chang W (2017). *R6: Classes with Reference Semantics*. R package version 2.2.2.

Chen K, Lei J (2018). "Network cross-validation for determining the number of communities in network data." *Journal of the American Statistical Association*, **113**(521), 241–251.

Chiquet J, Dervieux V, Rigaill G, Sundqvist M (2020). *aricode: Efficient Computations of Standard Clustering Comparison Measures*. R package version 1.0.0.

Chiquet J, Donnet S, Barbillon P (2021). *sbm: Stochastic Blockmodels*. R package version 0.4.0-9100, URL https://grosssbm.github.io/sbm/.

Côme E, Jouvin N, Latouche P, Bouveyron C (2021). "Hierarchical clustering with discrete latent variable models and the integrated classification likelihood." *Advances in Data Analysis and Classification*, pp. 1–30.

Csardi G, Nepusz T (2006). "The igraph software package for complex network research." *InterJournal*, **Complex Systems**, 1695. URL http://igraph.org.

Daudin JJ, Picard F, Robin S (2008). "A mixture model for random graphs." *Statistics and Computing*, **18**(2), 173–183.

Decelle A, Krzakala F, Zhang P (2019). "MODE-NET: MOdules DEtection in NETworks." URL https://cran.r-project.org/web/views/gR.html.

Dempster AP, Laird NM, Rubin DB (1977). "Maximum likelihood from incomplete data via the EM algorithm." *Journal of the Royal Statistical Society. Series B.*, **39**(1), 1–38.

Donnet S, Barbillon P (2021). *GREMLINS: Generalized Multipartite Networks*. R package version 0.2.1, URL https://demiperimetre.github.io/GREMLINS/.

Eddelbuettel D, François R (2011). "Rcpp: Seamless R and C++ integration." *Journal of Statistical Software*, **40**(8), 1–18. doi:10.18637/jss.v040.i08.

Eddelbuettel D, Sanderson C (2014). "RcppArmadillo: Accelerating R with high-performance C++ linear algebra." *Computational Statistics & Data Analysis*, **71**, 1054–1063. URL http://dx.doi.org/10.1016/j.csda.2013.02.005.

Erdős P, Renyi A (1959). "On random graphs." *Publicationes Mathematicae*, **6**, 290–297.

Franzin A, Sambo F, di Camillo B (2017). "bnstruct: an R package for Bayesian Network Structure Learning in the Presence of Missing Data." *Bioinformatics*, **33**(8), 1250–1252. doi:10.1093/bioinformatics/btw807.

Friedman J, Hastie T, Tibshirani R (2019). *glasso: Graphical Lasso: Estimation of Gaussian Graphical Models*. R package version 1.11, URL https://CRAN.R-project.org/package=glasso.

Handcock M, Hunter D, Butts C, Goodreau S, Morris M (2008). "statnet: Software tools for the representation, visualization, analysis and simulation of network data." *Journal of Statistical Software*, **24**(1), 1–11. ISSN 1548-7660. doi:10.18637/jss.v024.i01.

Handcock MS, Gile KJ (2010). "Modeling social networks from sampled data." *The Annals of Applied Statistics*, **4**(1), 5–25.

Hoff P (2007). "Modeling homophily and stochastic equivalence in symmetric relational data." In *Advances in neural information processing systems*, pp. 657–664.

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

Hojsgaard S (2019). "CRAN Task View: gRaphical Models in R." URL https://cran.r-project.org/web/views/gR.html.

Hu J, Zhang J, Qin H, Yan T, Zhu J (2020). "Using maximum entry-wise deviation to test the goodness of fit for stochastic block models." *Journal of the American Statistical Association*, pp. 1–10.

Hunter D, Handcock M, Butts C, Goodreau S, Morris M (2008). "ergm: A package to fit, simulate and diagnose exponential-family models for networks." *Journal of Statistical Software*, **24**(3), 1–29. ISSN 1548-7660. doi:10.18637/jss.v024.i03.

Hunter DR, Handcock MS (2006). "Inference in curved exponential family models for networks." *Journal of Computational and Graphical Statistics*, **15**(3), 565–583.

Jordan MI, Ghahramani Z, Jaakkola TS, Saul LK (1998). "An introduction to variational methods for graphical models." In *Learning in graphical models*, pp. 105–161. Springer-Verlag.

Krivitsky P, Handcock M (2008). "Fitting latent cluster models for networks with latentnet." *Journal of Statistical Software*, **24**(5), 1–23. ISSN 1548-7660. doi:10.18637/jss.v024.i05.

Lauritzen LS (1996). *Graphical models*. Clarendon Press.

Léger JB (2016). "Blockmodels: A R-package for estimating in latent block model and stochastic block model, with various probability functions, with or without covariates." *Technical report*. URL https://arxiv.org/abs/1602.07587.

Li T, Levina E, Zhu J (2010). *randnet: Random Network Model Selection and Parameter Tuning*. R package version 0.2.

Li T, Levina E, Zhu J (2020). "Network cross-validation by edge sampling." *Biometrika*, **107**(2), 257–276.

Little RJ, Rubin DB (2019). *Statistical analysis with missing data*, volume 793. John Wiley & Sons.

Mariadassou M, Robin S, Vacher C (2010). "Uncovering latent structure in valued graphs: A variational approach." *Ann. Appl. Stat.*, **4**(2), 715–742.

Mariadassou M, Tabouy T, *et al.* (2020). "Consistency and asymptotic normality of stochastic block models estimators from sampled data." *Electronic Journal of Statistics*, **14**(2), 3672–3704.

Matias C, Miele V (2010). *dynsbm: Dynamic Stochastic Block Models*. R package version 0.2.

Mejean A, Maison J (2017). "GitHub: CommunityDectection." URL https://github.com/Jonas1312/CommunityDetection.

Nowicki K, Snijders TAB (2001). "Estimation and Prediction for Stochastic Blockstructures." *Journal of the American Statistical Association*, **96**(455), 1077–1087.

Pearl J (2011). "Bayesian networks."

Peixoto TP (2014). "The graph-tool python library." *figshare*. doi:10.6084/m9.figshare.1164194. URL http://figshare.com/articles/graph_tool/1164194.

Rand WM (1971). "Objective criteria for the evaluation of clustering methods." *Journal of the American Statistical Association*, **66**(336), 846–850.

Rastelli R, Fop M (2019). *expSBM: An Exponential Stochastic Block Model for Interaction Lengths*. R package version 1.3.5, URL https://CRAN.R-project.org/package=expSBM.

Rebafka T, Villers F (2020). *noisySBM: Noisy Stochastic Block Mode: Graph Inference by Multiple Testing*. R package version 0.1.4, URL https://CRAN.R-project.org/package=noisySBM.

Robin X, Turck N, Hainard A, Tiberti N, Lisacek F, Sanchez JC, Muller M (2011). "pROC: an Open-Source Package for R and S+ to Analyze and Compare ROC Curves." *BMC Bioinformatics*, **12**, 77.

Rohe K, Chatterjee S, Yu B (2011). "Spectral clustering and the high-dimensional stochastic block model." *The Annals of Statistics*.

Saldana DF, Yu Y, Feng Y (2017). "How many communities are there?" *Journal of Computational and Graphical Statistics*, **26**(1), 171–181.

Sanderson C, Curtin R (2016). "Armadillo: a template-based C++ library for linear algebra." *Journal of Open Source Software*, **1**(2), 26.

Schweinberger M, Luna P (2018). "HERGM: Hierarchical exponential-family random graph models." *Journal of Statistical Software*, **85**(1).

Scutari M (2009). "Learning Bayesian networks with the bnlearn R package." *arXiv preprint arXiv:0908.3817*.

Snijders TA, Nowicki K (1997). "Estimation and prediction for stochastic blockmodels for graphs with latent block structure." *J. class.*, **14**(1), 75–100.

Strayer N (2021). *sbmr: Fit and investigate Stochastic Block Models in R*. R package version 0.0.2.0, URL https://tbilab.github.io/sbmr.

Tabouy T, Barbillon P, Chiquet J (2020). "Variational inference for stochastic block models from sampled data." *Journal of the American Statistical Association*, **115**(529), 455–466.

Tallberg C (2004). "A Bayesian approach to modeling stochastic blockstructures with covariates." *Journal of Mathematical Sociology*, **29**(1), 1–23.

Vu DQ, Hunter DR, Schweinberger M (2013). "Model-based clustering of large networks." *The Annals of Applied Statistics*, **7**(2), 1010.

Wang YR, Bickel PJ, *et al.* (2017). "Likelihood-based model selection for stochastic block models." *The Annals of Statistics*, **45**(2), 500–528.

Westling T, McCormick TH (2019). "Beyond prediction: A framework for inference with variational approximations in mixture models." *Journal of Computational and Graphical Statistics*, **28**(4), 778–789. doi:10.1080/10618600.2019.1609977.

Yen TC, Larremore D (2019). "GitHub : bipartiteSBM-MCMC." URL https://github.com/junipertcy/bipartiteSBM-MCMC.

Yen TC, Larremore DB (2018). "Blockmodeling on a Bipartite Network with Bipartite Prior." URL https://docs.netscied.tw/det_k_bisbm/index.html.

You K (2020). *graphon: A Collection of Graphon Estimation Methods*. R package version 0.3.4, URL https://CRAN.R-project.org/package=graphon.

Young JG, Desrosiers P, Hébert-Dufresne L, Laurence E, Dubé LJ (2017). "Finite-size analysis of the detectability limit of the stochastic block model." *Physical Review E*, **95**(6), 062304.

Zanghi H, Ambroise C, Miele V (2008). "Fast online graph clustering via Erdős–Rényi mixture." *Pattern Recognition*, **41**(12), 3592 – 3599. ISSN 0031-3203. doi:https://doi.org/10.1016/j.patcog.2008.06.019.

Zhao T, Liu H, Roeder K, Lafferty J, Wasserman L (2012). "The huge package for high-dimensional undirected graph estimation in R." *The Journal of Machine Learning Research*, **13**(1), 1059–1062.

Étienne Côme, Jouvin N (2021). *greed: Clustering and Model Selection with the Integrated Classification Likelihood*. R package version 0.5.1, URL https://CRAN.R-project.org/package=greed.

Žiberna A (2020). *Generalized and Classical Blockmodeling of Valued Networks*. R package version 1.0.0.

Affiliation:

Pierre Barbillon, Julien Chiquet & Timothée Tabouy
MIA Paris, Université Paris-Saclay, AgroParisTech, INRAE
E-mails: pierre.barbillon@agroparistech.fr, julien.chiquet@inrae.fr, timothee.tabouy@gmail.com