Bioinformatics, 35, 2019, i23–i30
doi: 10.1093/bioinformatics/btz370
ISMB/ECCB 2019

Learning a mixture of microbial networks using minorization–maximization

Sahar Tavakoli and Shibu Yooseph*

Department of Computer Science, Genomics and Bioinformatics Cluster, University of Central Florida, Orlando, FL 32816, USA

*To whom correspondence should be addressed.

Abstract

Motivation: The interactions among the constituent members of a microbial community play a major role in determining the overall behavior of the community and the abundance levels of its members. These interactions can be modeled using a network whose nodes represent microbial taxa and edges represent pairwise interactions. A microbial network is typically constructed from a sample-taxa count matrix that is obtained by sequencing multiple biological samples and identifying taxa counts. From large-scale microbiome studies, it is evident that microbial community compositions and interactions are impacted by environmental and/or host factors. Thus, it is not unreasonable to expect that a sample-taxa matrix generated as part of a large study involving multiple environmental or clinical parameters can be associated with more than one microbial network. However, to our knowledge, microbial network inference methods proposed thus far assume that the sample-taxa matrix is associated with a single network.
Results: We present a mixture model framework to address the scenario when the sample-taxa matrix is associated with K microbial networks. This count matrix is modeled using a mixture of K Multivariate Poisson Log-Normal distributions and parameters are estimated using a maximum likelihood framework. Our parameter estimation algorithm is based on the minorization–maximization principle combined with gradient ascent and block updates. Synthetic datasets were generated to assess the performance of our approach on absolute count data, compositional data and normalized data. We also addressed the recovery of sparse networks based on an l_1-penalty model.
Availability and implementation: MixMPLN is implemented in R and is freely available at https://github.com/sahatava/MixMPLN.
Contact: shibu.yooseph@ucf.edu
Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

Microbes are found almost everywhere on earth, including in environments deemed too extreme for other life forms, and they play critical roles in many biogeochemical processes (Falkowski *et al.*, 2008; Whitman *et al.*, 1998). Microbial communities are also found in association with higher life forms, including plants and animals; for instance, trillions of microbes live in or on the human body (almost as many human cells as there are in the body) (Sender et al., 2016) and ongoing research continues to reveal the important roles that many of these microbes play in human health (Huttenhower *et al.*, 2012; Qin *et al.*, 2010). Microbial communities are typically structured and composed of members of different species. The microbes in a community do not exist in isolation, but interact with each other and also compete for the available carbon and energy sources. These interactions, along with resource availability and environmental parameters (like temperature, pH and salinity) (Hibbing *et al.*, 2010; Williamson and Yooseph, 2012), determine the taxonomic composition of the microbial community and the abundance levels of its constituents. Knowledge of these interactions is crucial for understanding the overall behavior of the microbial community, and can be used to elucidate the biological mechanisms underlying microbe-associated disease progression and microbe-mediated processes (like biofilm formation).

The study of microbial communities has been greatly enabled with the advent of high-throughput next-generation DNA sequencing technologies (Bentley, 2006; Margulies *et al.*, 2005; Quail *et al.*, 2012). The taxonomic composition of a microbial community can be obtained by sequencing the DNA extracted from a biological sample that has been collected from the environment of interest. This is achieved either using a targeted approach, involving the sequencing of a taxonomic marker gene [for instance, the 16S ribosomal RNA gene, which is found in all bacteria (Woese and Fox, 1977)] or using a whole-genome shotgun sequencing approach (Venter *et al.*, 2004). Both approaches generate taxa counts that are *compositional* in nature, and that enable the estimation of the *relative* abundances of the constituent members of the community.

Microbial interactions can be modeled using a weighted graph (or network), where each node in the graph represents a taxon (or taxonomic group) and an (undirected) edge exists between two nodes if the corresponding taxa interact with, or influence, each other. The edge weight captures the strength of the interaction, with its sign reflecting whether the interaction is positive or negative. This framework can be used to model a variety of microbial interactions, including competition and co-operation. While we do not consider it here, a directed graph can also be used to represent interactions, where the edge direction indicates the direction of influence (or causality).

Microbial networks are typically constructed from sample-taxa count matrices. A sample-taxa count matrix is generated by sequencing multiple biological samples (*n* samples) collected from the environment of interest and identifying the counts of the observed taxa (*d* taxa) in each sample.

As we discuss briefly below, microbial networks can be constructed using a variety of different approaches. To our knowledge, all of these methods assume that the sample-taxa matrix is associated with a *single* underlying stochastic process (i.e. there is one underlying network topology and set of edge weights). However, this need not always be the case. In this paper, we consider an important extension to the network inference problem, whereby we develop a mixture modeling framework for inferring *K* microbial networks when the observed sample-taxa matrix is associated with *K* underlying distributions. We are motivated by large-scale human-associated and other environmental microbial community projects that are now possible due to cost-effective sequencing. For instance, human gut microbiome studies now routinely analyze large cohorts of individuals and generate microbial community data from several hundreds (to even thousands) of samples. An important research question in this area involves the definition of a 'core' microbiome associated with a particular host phenotype (Huttenhower *et al.*, 2012; Qin *et al.*, 2010). It is well understood that the gut microbiome composition is greatly influenced by many factors including diet and age, and thus it is not unreasonable to expect the associated microbial network interactions to also be different when these factors vary (e.g. the gut microbial community interaction network in vegetarian hosts can be expected to be different compared to the network in non-vegetarian hosts). A similar situation also occurs in environmental studies where the microbial interactions are influenced greatly by the physical and chemical gradients of the environment. Often the collected metadata in these studies may not be comprehensive enough to discern these interactions in a supervised manner. Our proposed mixture framework offers a principled approach to identifying these multiple microbial interaction networks from a sample-taxa matrix.

Several methods have been proposed for constructing a *single* microbial network from an input sample-taxa matrix (Layeghifard *et al.*, 2017). One approach involves using pairwise correlations (Pearson or Spearman) between taxa to define the edge weights in the graph. However, the computation of these correlation networks directly from the observed count data can be misleading because of the compositional nature of these data (Gloor *et al.*, 2017). Furthermore, for a microbial network with d nodes, while there are $d * (d-1)/2$ edge weights that need to be determined, the number of available samples n is often not large enough, with the result that the system of equations to determine all pairs of correlations is under-determined. This later issue is typically handled by assuming that the network is sparse [i.e. the number of edges is $O(d)$]. Methods based on latent variable modeling have been proposed to infer correlation networks (Fang *et al.*, 2015; Friedman and Alm, 2012). These methods use log-ratio transformations of the original count data (Aitchison, 1982) to deal with the compositional nature of these data and subsequently infer the correlation matrix (i.e. edge weights) under the assumption of sparsity. Microbial networks have also been constructed using a probabilistic graphical model framework (Jordan, 1999) that enables the modeling of conditional dependencies associated with the interactions. For instance, the assumption that the log-ratio transformed count data follow a Gaussian distribution, results in a Gaussian graphical model (GGM) framework. In this scenario, the graph structure represents the precision matrix (or inverse covariance matrix) of the underlying multivariate Gaussian distribution. This graph has the property that an edge exists between two nodes iff the corresponding entry in the precision matrix is non-zero. A zero entry in the precision matrix indicates conditional independence between the two corresponding random variables. When the graph is assumed to be sparse, the GGM inference problem can be solved using sparse precision matrix estimation algorithms (Friedman *et al.*, 2008). This approach has been used to construct microbial networks from sample-taxa matrices (Kurtz *et al.*, 2015).

An alternate approach to constructing a microbial network, and that which we adopt in this paper, is to model the vector of observed taxa counts (in samples) using a multivariate distribution and to infer the parameters of this distribution from the observed data using a maximum likelihood framework. Any candidate multivariate distribution for this approach will have to be flexible enough to capture the underlying covariance structure to model the network interactions (i.e. allow for both positive and negative covariances); this rules out distributions like the multinomial or the Dirichlet-Multinomial, which are popular choices for modeling microbial count data in certain situations (Holmes *et al.*, 2012; La Rosa et al., 2012), but which cannot capture both types of interactions. The Multivariate Poisson Log-Normal (MPLN) distribution (Aitchison and Ho, 1989) can be used for modeling multivariate count data and its covariance structure can capture both positive and negative interactions. This distribution was used recently (Biswas et al., 2015) to model counts in a sample-taxa matrix and infer an underlying microbial network using an assumption of sparsity.

In this paper, we consider the following computational inference problem: for a sample-taxa matrix X (containing *absolute* counts of taxa) that is generated by a mixture model consisting of K MPLN component distributions, estimate the mixing coefficients and the distribution parameters of the K components. We note that the precision matrices of the K components define the K different microbial networks. We formulate this inference problem using a maximum likelihood framework, and estimate the various parameters of the constructed likelihood function directly using a numerical optimization method based on the minorization–maximization (MM) principle (Lange, 2016; Wu *et al.*, 2010; Zhou and Lange, 2010). We extend this formulation based on an l_1-penalty model and provide algorithms to infer K sparse networks. We evaluate the performance of our algorithms using both synthetic and real datasets. We also evaluate the performance of our method on compositional data obtained by subsampling from the true counts of the taxa. This evaluation models the real-world scenario, where the sample-taxa matrices that we have access to, contain only *relative* abundances of the observed taxa.

2 Materials and methods

Prior to describing our mixture modeling framework, we describe a single MPLN distribution. In our discussions, we denote matrices using upper-case letters, column vectors using bold letters (upper- or lower- case) and scalar values using normal lower-case letters.

2.1 The model

Single MPLN distribution:
Consider an MPLN distribution with parameter set $\Theta = (\boldsymbol{\mu}, \Sigma)$, where $\boldsymbol{\mu}$ represents its d-dimensional mean vector and Σ represents its $d \times d$ covariance matrix. Then, a d-dimensional count vector $\boldsymbol{X} = (x_1, ..., x_d)^T$ generated by this distribution has the following property:

$$\begin{gathered} x_j|\lambda_j \sim \mathbb{P}(\mathrm{e}^{\lambda_j}) \\ (\lambda_1, ..., \lambda_d)^T \sim \mathbb{N}_d(\boldsymbol{\mu}, \Sigma), \end{gathered} \tag{1}$$

where $\mathbb{P}(c)$ denotes a Poisson distribution with mean c and $\mathbb{N}_d(\boldsymbol{\mu}, \Sigma)$ denotes a d-dimensional multivariate normal distribution with mean $\boldsymbol{\mu}$ and covariance Σ. An MPLN distribution thus has two layers, with the observed counts being generated by a mixture of independent Poisson distributions whose (hidden) means follow a multivariate normal distribution. If we use $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, ..., \lambda_d)$ to denote the latent (or hidden) variable representing the Poisson means, then the probability density function $p(X|\Theta)$ of the MPLN distribution is given by

$$\int_{\mathbb{R}^d} \prod_{j=1}^{d} \frac{\mathrm{e}^{-\mathrm{e}^{\lambda_j}} \mathrm{e}^{\lambda_j x_j}}{x_j!} \frac{\mathrm{e}^{\left[-\frac{1}{2}(\boldsymbol{\lambda}-\boldsymbol{\mu})^T \Sigma^{-1} (\boldsymbol{\lambda}-\boldsymbol{\mu})\right]}}{\sqrt{(2\pi)^d \det(\Sigma)}} d\boldsymbol{\lambda}. \tag{2}$$

Let $\boldsymbol{X_1}, \boldsymbol{X_2}, \ldots\ldots, \boldsymbol{X_n}$ denote n independent samples drawn from an MPLN distribution, where each $\boldsymbol{X_i}$ is a d-dimensional count vector. We use $X = [\boldsymbol{X_1} \boldsymbol{X_2} \ldots\ldots \boldsymbol{X_n}]$ to denote the sample-taxa matrix generated from d taxa and n samples, and x_{ij} to denote the count of the j-th taxon in $\boldsymbol{X_i}$. We can estimate the parameter set Θ of this MPLN distribution using a likelihood framework by considering the d-dimensional latent variables $\boldsymbol{\lambda_1}, \boldsymbol{\lambda_2}, \ldots\ldots,$ and $\boldsymbol{\lambda_n}$ associated with samples $\boldsymbol{X_1}, \boldsymbol{X_2}, ..,$ and $\boldsymbol{X_n}$ respectively; let matrix $\Lambda = [\boldsymbol{\lambda_1} \boldsymbol{\lambda_2} \ldots\ldots \boldsymbol{\lambda_n}]$ and λ_{ij} denote the j-th entry in $\boldsymbol{\lambda_i}$. The log-likelihood function $L(\Theta|X, \Lambda)$ can be optimized using a simple iterative stepwise ascent (or conditional maximization) procedure to compute Θ and Λ. The estimated values of the parameters can be used to provide an approximation for $p(X|\Theta)$ as $\prod_{i=1}^{n} p(\boldsymbol{X_i}|\boldsymbol{\lambda_i}, \Theta)$, where $p(\boldsymbol{X_i}|\boldsymbol{\lambda_i}, \Theta)$ is defined as:

$$\prod_{j=1}^{d} \frac{\mathrm{e}^{-\mathrm{e}^{\lambda_{ij}}} \mathrm{e}^{\lambda_{ij} x_{ij}}}{x_{ij}!} \frac{\mathrm{e}^{\left[-\frac{1}{2}(\boldsymbol{\lambda_i}-\boldsymbol{\mu})^T \Sigma^{-1} (\boldsymbol{\lambda_i}-\boldsymbol{\mu})\right]}}{\sqrt{(2\pi)^d \det(\Sigma)}}. \tag{3}$$

An analogous approach based on optimizing the log-posterior using an iterative conditional modes algorithm has been proposed previously (Biswas *et al.*, 2015).

Mixture of *K* MPLN distributions (MixMPLN):
In our framework, we consider a mixture model involving K MPLN distributions. Let $\pi_1, ..., \pi_K$ represent the mixing coefficients of the K components (where $\sum_{l=1}^{K} \pi_l = 1$), and let p_l and Θ_l denote the l-th component distribution and its parameter set. A d-dimensional sample vector $\boldsymbol{X}$ generated from this mixture has the following distribution:

$$p(\boldsymbol{X}|\pi_1, ..., \pi_K, \Theta_1, ..., \Theta_K) = \sum_{l=1}^{K} \pi_l p_l(\boldsymbol{X}|\Theta_l). \tag{4}$$

For n independent samples $X = [\boldsymbol{X_1} \boldsymbol{X_2} \ldots\ldots \boldsymbol{X_n}]$, the probability distribution is given by

$$p(X|\pi_1, ..., \pi_K, \Theta_1, ..., \Theta_K) = \prod_{i=1}^{n} \sum_{l=1}^{K} \pi_l p_l(\boldsymbol{X_i}|\Theta_l). \tag{5}$$

The general log-likelihood function is thus

$$\begin{gathered} L(\pi_1, ..., \pi_K, \Lambda_1, ..., \Lambda_K, \Theta_1, ..., \Theta_K|X) = \\ \log\left(\prod_{i=1}^{n} \sum_{l=1}^{K} \pi_l p_l(\boldsymbol{X_i}|\boldsymbol{\lambda_{il}}, \Theta_l)\right), \end{gathered} \tag{6}$$

where $\boldsymbol{\lambda_{il}}$ is the d-dimensional latent variable associated with $\boldsymbol{X_i}$ in component l. We use a maximum log-likelihood framework to estimate the parameters of the MixMPLN model from the observed data X by optimizing the function $L(\pi_1, ..., \pi_K, \Lambda_1, ..., \Lambda_K, \Theta_1, ..., \Theta_K|X)$.

2.2 Optimizing the log-likelihood function using the MM principle

The MM principle is a general technique (Hunter and Lange, 2004; Lange, 2016) that has proven to be useful for tackling function optimization problems (MM stands for minorization–maximization in maximization problems and for majorization–minimization in minimization problems). For our scenario, let $h(\theta)$ denote a function to be maximized. The MM principle proposes to maximize the minorizer function $g(\theta|\theta^t)$ instead of maximizing $h(\theta)$ directly; here, θ^t denotes a fixed value of the parameter θ. The function $g(\theta|\theta^t)$ is said to be a minorizer of $h(\theta)$ if:

$$\begin{gathered} h(\theta^t) = g(\theta^t|\theta^t) \\ h(\theta) \geq g(\theta|\theta^t), \theta \neq \theta^t. \end{gathered} \tag{7}$$

Therefore, the first step in our MM approach is to find a minorizer function which has the required property. For this, we use the following observation that follows from the concavity property of the log function (Lange, 2016; Zhou and Lange, 2010) for m non-negative values $\beta_1, \beta_2, \ldots\ldots, \beta_m$:

$$\log\left(\sum_{i=1}^{m} \beta_i\right) \geq \sum_{i=1}^{m} \frac{\beta_i^{\,t}}{\sum_{j=1}^{m} \beta_j^{\,t}} \log\left(\frac{\sum_{j=1}^{m} \beta_j^{\,t}}{\beta_i^{\,t}} \beta_i\right). \tag{8}$$

Equation (6) can thus be lower-bounded based on this observation:

$$\log\left(\prod_{i=1}^{n} \sum_{l=1}^{K} \pi_l p_l(\boldsymbol{X_i}|\boldsymbol{\lambda_{il}}, \Theta_l)\right) \geq \sum_{i=1}^{n} \sum_{l=1}^{K} w_{il}{}^{t} \log\left(\frac{\pi_l}{w_{il}{}^{t}} p_l(\boldsymbol{X_i}|\boldsymbol{\lambda_{il}}, \Theta_l)\right), \tag{9}$$

where, weight w_{il}^t is defined as follows:

$$w_{il}{}^{t} = \frac{\pi_l{}^{t} p_l(\boldsymbol{X_i}|\boldsymbol{\lambda_{il}}{}^{t}, \Theta_l^t)}{\sum_{l=1}^{K} \pi_l{}^{t} p_l(\boldsymbol{X_i}|\boldsymbol{\lambda_{il}}{}^{t}, \Theta_l{}^{t})}. \tag{10}$$

We use the function on the right-hand side of Equation (9) as the minorizer function for our problem. Thus, we will define the new objective function (L) to be maximized as follows:

$$\begin{aligned} & L(\pi_1, \pi_2, ..., \pi_K, \Lambda_1, ..., \Lambda_K, \Theta_1, \Theta_2, ..., \Theta_K) \\ & = \sum_{i=1}^{n} \sum_{l=1}^{K} w_{il}{}^{t} \log\left(\frac{\pi_l}{w_{il}{}^{t}} p_l(\boldsymbol{X_i}|\boldsymbol{\lambda_{il}}, \Theta_l)\right) = \sum_{i=1}^{n} \sum_{l=1}^{K} w_{il}{}^{t} \left(\log\left(\frac{1}{w_{il}{}^{t}}\right)\right) \\ & + \sum_{i=1}^{n} \sum_{l=1}^{K} w_{il}{}^{t} (\log(\pi_l)) + \sum_{i=1}^{n} \sum_{l=1}^{K} w_{il}{}^{t} (\log(p_l(\boldsymbol{X_i}|\boldsymbol{\lambda_{il}}, \Theta_l))). \end{aligned} \tag{11}$$

2.3 Steps of the MixMPLN optimization algorithm

We used a coordinate ascent approach in conjunction with a block update strategy to optimize L. We present the details of parameter initialization and subsequent iterative updates below.

2.3.1 Parameter initialization

The n samples $\boldsymbol{X}_1\boldsymbol{X}_2\ldots\ldots\boldsymbol{X}_n$ are partitioned into K clusters (components) using the K-means algorithm. Then, the samples assigned to a component are used to estimate the parameters of that component using the moments of an MPLN distribution (Aitchison and Ho, 1989), given by the equations below; here, σ_{ij} denotes the ij-th entry in Σ.

$$\begin{gathered} E(x_i) = \exp\left(\mu_i + \frac{1}{2}\sigma_{ii}\right) = \alpha_i \\ var(x_i) = \alpha_i + {\alpha_i}^2\{\exp(\sigma_{ii}) - 1\} \\ cov(x_i, x_j) = \alpha_i\alpha_j\{\exp(\sigma_{ij}) - 1\} \end{gathered}. \tag{12}$$

2.3.2 Parameter estimation in iteration $(t+1)$

The portion of function L in Equation (11) that is dependent on the π_i's can be optimized.

$$L_1(\pi_1, \pi_2, \ldots, \pi_l) = \sum_{i=1}^{n}\sum_{l=1}^{K} {w_{il}}^t(\log(\pi_l))). \tag{13}$$

Since $\sum_l \pi_l = 1$, we can optimize L_1 by introducing a Lagrange multiplier and identifying a stationary point of the subsequent Lagrangian (Bilmes *et al.*, 1998). This yields

$${\pi_l}^{t+1} = \frac{1}{n}\sum_{i=1}^{n} w_{il}^t, \tag{14}$$

where w_{il}^t is calculated using Equation (10).

Considering the part of the L function that is dependent on Λ and Θ, we have the following term to maximize:

$$L_2(\Lambda_1, \ldots, \Lambda_K, \Theta_1, \ldots, \Theta_K) = \sum_{i=1}^{n}\sum_{l=1}^{K} {w_{il}}^t\left(\log(p_l(\boldsymbol{X}_i|\boldsymbol{\lambda}_{il}, \Theta_l))\right). \tag{15}$$

Expanding p_l using Equation (3), we have:

$$\begin{aligned} &L_2(\Lambda_1, \ldots, \Lambda_K, \Theta_1, \ldots, \Theta_K) \\ &= \sum_{i=1}^{n}\sum_{j=1}^{d}\sum_{l=1}^{K} -{w_{il}}^t e^{\lambda_{ijl}} + \sum_{i=1}^{n}\sum_{j=1}^{d}\sum_{l=1}^{K} {w_{il}}^t \lambda_{ijl}x_{ij} \\ &- \sum_{i=1}^{n}\sum_{j=1}^{d}\sum_{l=1}^{K} {w_{il}}^t \log(x_{ij}!) - \frac{d}{2}\sum_{i=1}^{n}\sum_{l=1}^{K} {w_{il}}^t \log(2\pi) \\ &+ \frac{1}{2}\sum_{i=1}^{n}\sum_{l=1}^{K} {w_{il}}^t \log\left(det\left(\Sigma_l^{-1}\right)\right) \\ &- \frac{1}{2}\sum_{i=1}^{n}\sum_{l=1}^{K} {w_{il}}^t (\boldsymbol{\lambda}_{il} - \boldsymbol{\mu}_l)^T \Sigma_l^{-1}(\boldsymbol{\lambda}_{il} - \boldsymbol{\mu}_l), \end{aligned} \tag{16}$$

where λ_{ijl} is the j-th entry in $\boldsymbol{\lambda}_{il}$. We solve for the parameters separately using the partial derivation method. Calculation of the partial derivative of L_2 with respect to λ_{ijl} gives us:

$$\begin{gathered} \frac{\partial}{\partial \lambda_{ijl}} L_2 = 0 \Rightarrow \\ -e^{\lambda_{ijl}} + x_{ij} - \langle(\boldsymbol{\lambda}_{il} - \boldsymbol{\mu}_l), \Sigma_{.jl}^{-1}\rangle = 0, \end{gathered} \tag{17}$$

where $\langle \boldsymbol{a}, \boldsymbol{b}\rangle$ denotes the inner product of vectors $\boldsymbol{a}$ and $\boldsymbol{b}$, and $\Sigma_{.jl}^{-1}$ denotes the j-th column of Σ_l^{-1}. We use the Newton–Raphson method to estimate λ_{ijl}^{t+1} from this equation, using values from iteration t for $\boldsymbol{\Sigma}^{-1}$, $\boldsymbol{\mu}$, and $\lambda_{ij'l}$ (where $j \neq j'$). Partial derivation with respect to $\boldsymbol{\mu}_l$ gives us:

$$\begin{gathered} \frac{\partial}{\partial \boldsymbol{\mu}_l} L_2 = 0 \Rightarrow \\ \sum_{i=1}^{n} {w_{il}}^t(\boldsymbol{\lambda}_{il} - \boldsymbol{\mu}_l) = 0 \Rightarrow \\ \boldsymbol{\mu}_l = \frac{\sum_{i=1}^{n} {w_{il}}^t \boldsymbol{\lambda}_{il}}{\sum_{i=1}^{n} {w_{il}}^t}. \end{gathered} \tag{18}$$

Thus, ${\boldsymbol{\mu}_l}^{t+1}$ can be estimated from ${w_{il}}^t$ and ${\boldsymbol{\lambda}_{il}}^{t+1}$.

And finally, partial derivation with respect to Σ_l^{-1} results in:

$$\begin{aligned} &\frac{\partial}{\partial \Sigma_l^{-1}} L_2 = 0 \Rightarrow \\ &\frac{\partial}{\partial \Sigma_l^{-1}} \left\{ \frac{1}{2}\sum_{i=1}^{N} {w_{il}}^t \log(det(\Sigma_l^{-1})) \right. \\ &\left. -\frac{1}{2} trace\left(\Sigma_l^{-1}\sum_{i=1}^{N} {w_{il}}^t (\boldsymbol{\lambda}_{il} - \boldsymbol{\mu}_l)(\boldsymbol{\lambda}_{il} - \boldsymbol{\mu}_l)^T\right)\right\} = 0. \end{aligned} \tag{19}$$

We can solve this equation using matrix derivative rules to compute an estimate for the covariance matrix Σ_l in iteration $(t+1)$ as:

$$\Sigma_l^{t+1} = \frac{\sum_{i=1}^{n} {w_{il}}^t({\boldsymbol{\lambda}_{il}}^{t+1} - {\boldsymbol{\mu}_l}^{t+1})({\boldsymbol{\lambda}_{il}}^{t+1} - {\boldsymbol{\mu}_l}^{t+1})^T}{\sum_{i=1}^{n} {w_{il}}^t}. \tag{20}$$

2.4 Inferring sparse networks using an l_1-penalty model

We extended the MixMPLN framework by incorporating an l_1-norm penalty as follows:

$$p(\boldsymbol{X}|\pi_1, \ldots, \pi_K, \Theta_1, \ldots, \Theta_K) = \sum_{l=1}^{K}\left[\pi_l p_l(\boldsymbol{X}|\Theta_l) + \rho_l\|\Sigma_l^{-1}\|_1\right], \tag{21}$$

where $\|\Sigma_l^{-1}\|_1$ is the l_1-norm of the precision matrix of the l-th component, and $\rho_1, \ldots, \rho_K$ are tuning parameters that can be selected independently. This framework allows for the inference of sparse networks associated with the K components. For a fixed tuning parameter of ρ and a multivariate Gaussian distribution, the problem of selecting a precision matrix with the l_1-norm penalty can be stated as (Friedman *et al.*, 2008):

$$\underset{\Sigma^{-1}}{\operatorname{argmax}}\{\log(det(\Sigma^{-1})) - trace(S\Sigma^{-1}) - \rho\|\Sigma^{-1}\|_1\}, \tag{22}$$

where S denotes the empirical covariance matrix.

In our implementation of the extended MixMPLN framework, we calculated the empirical covariance matrix for each component using Equation (20) in each iteration. We used the 'glasso' and 'huge' packages in R to solve the sparse precision matrix selection problem (Friedman *et al.*, 2008; Zhao *et al.*, 2012). We applied three different strategies using each of the two packages. In MixMPLN + glasso(cross validation), we used cross validation to determine the value of ρ (i.e. we picked that ρ value which gave the best log-likelihood value among the subsamples). In MixMPLN + glasso (fixed tuning parameter), we used $\rho = 2d^2/(N\|\Sigma_{\text{init}}^{-1}\|_1)$, as proposed in Biswas *et al.* (2015). In this formulation, $\Sigma_{\text{init}}^{-1}$ is the estimated precision matrix after the initialization step of MixMPLN. In MixMPLN + glasso(iterative tuning parameter), we used the same formulation to initialize the ρ value, but then updated it in each iteration based on the new estimated precision matrix in that iteration.

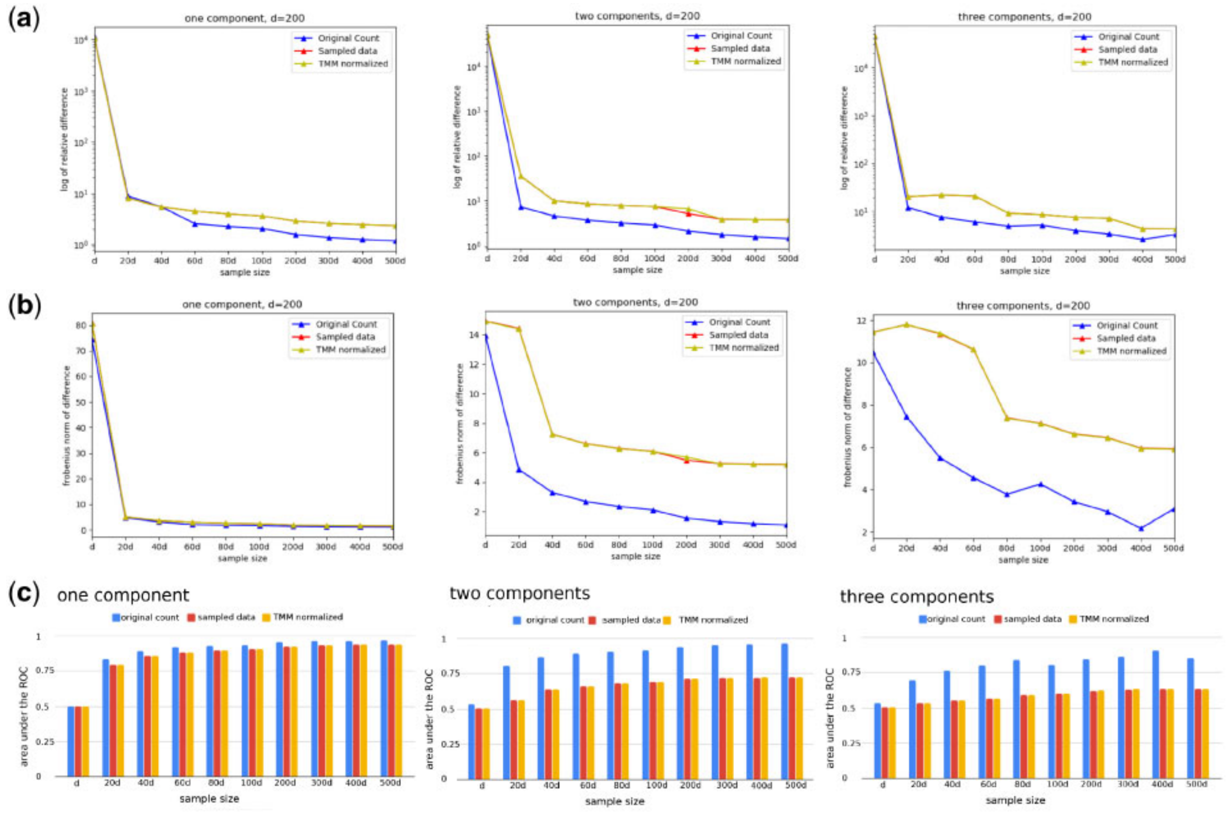

Fig. 1. Performance on synthetic data with $d = 200, sp = 0.7, K = 1, \; 2, \; 3$. (**a**) Relative difference between the predicted and true precision matrices. (**b**) Frobenius norm of the difference. (**c**) Area under the ROC

In MixMPLN + huge(StARS), we used the stability approach to regularization selection (StARS) method (Liu *et al.*, 2010). This method selects the coefficient which results in the most edge stability in the final graph) (Kurtz *et al.*, 2015). The tuning parameter selection method in MixMPLN + huge(fixed tuning parameter) and MixMPLN + huge(fixed tuning parameter) was the same as the corresponding implementations using glasso.

2.5 Performance evaluation and datasets

Synthetic sample-taxa count matrices were generated in order to assess the performance of the various MixMPLN algorithms. We evaluated the convergence properties of the algorithms as a function of increasing sample sizes. Since, in practice, sample-taxa count matrices generated from biological samples are compositional in nature, we also evaluated the effect of sampling from the true (or absolute) counts, and the subsequent application of data normalization, on network recovery and convergence. In addition, we also evaluated the accuracy in recovering sparse networks. Finally, we applied our mixture model framework to analyze a real dataset.

2.5.1 Datasets

Synthetic data generation: Each sample-taxa count matrix X was produced by combining samples generated from K component MPLN distributions, where component l generated n_l samples (d-length count vectors), and such that $\pi_l = \frac{n_l}{n}$ and $n = \Sigma_{l=1}^{K} n_l$. For each component l, the $d \times d$ covariance matrix of its MPLN distribution was derived from a randomly generated $d \times d$ positive definite precision matrix containing a fixed number of zero entries (as given by the sparsity level sp which denotes the fraction matrix entries that are zero). The mean vector for each MPLN component was a random d-length vector. For a sample-taxa matrix X generated this way, it is assumed that each entry in the matrix is the true (or absolute) abundance of taxon j in sample i. We refer to X as the *original count* matrix. To simulate compositional count data and sequencing depth differences between the biological samples, we generated a *sampled data* matrix XS from X by first normalizing each entry in a sample (by dividing by sample size) and subsequently scaling that value by a sample-specific random number $Y \in [5000, \; 10000]$ (to model sequencing depth for the biological sample). Finally, to study the effect of data normalization, we applied the trimmed mean of M-values (TMM) normalization procedure [from the ‘edgeR’ package (Robinson and Oshlack, 2010)] to XS to generate the *TMM normalized data* matrix TS.

Real data: We also applied our mixture modeling framework to a sample-taxa count matrix produced by a recent microbiome study (Yooseph *et al.*, 2015) that explored connections between gut microbiome composition and the risk of *Plasmodium falciparum* infection. In this study, stool samples from a cohort composed of 195 Malian adults and children were collected and analyzed. The samples were assayed by sequencing the 16S ribosomal RNA gene to determine the bacterial communities they contained. This generated a sample-taxa count matrix with 195 samples and 221 bacterial genera which we analyzed in this paper.

2.5.2 Benchmarking criteria

Let $A_{d \times d}$ denote the true precision matrix and $B_{d \times d}$ an inferred (or predicted) precision matrix. For evaluating the performance of our algorithms on synthetic data, we used three different criteria to compare the inferred precision matrices with the original precision matrices that were used to generate the sample-taxa matrices.

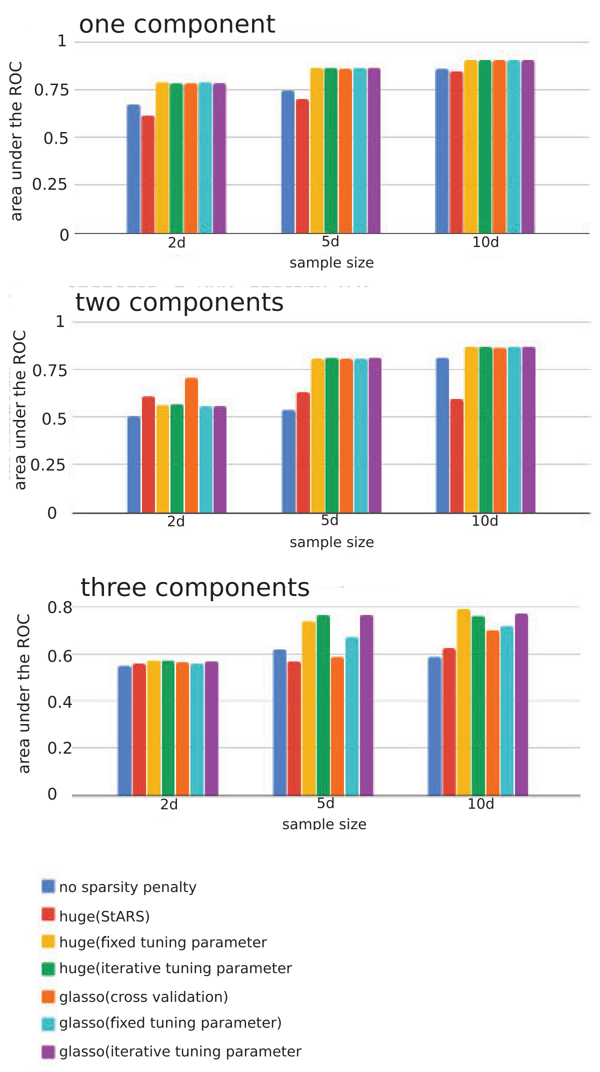

Fig. 2. AUC values for the synthetic datasets generated using $d = 200, sp = 0.9, K = 1,\ 2,\ 3$

- *Relative difference* between two matrices A and B defined as $\frac{1}{d^2}\sum_{i=1}^{d}\sum_{j=1}^{d}\frac{|a_{ij}-b_{ij}|}{|a_{ij}|\cdot|b_{ij}|}$.
- *Frobenius norm of the difference* between the partial correlations of matrices A and B. Frobenius norm of a matrix M is defined as $\|M\|_F = \sqrt{\sum_i\sum_j M_{ij}{}^2}$. For a precision matrix C, its partial correlation matrix P is calculated as $P[i,j] = -C[i,j]/\sqrt{C[i,i]*C[j,j]}$.
- *Area under the ROC (AUC)*: this measure was used to assess the recovery of the edges of the microbial network (i.e. identification of the zero entries in the precision matrix). The AUC was calculated by applying varying thresholds to the original and estimated precision matrices to define zero and non-zero entries. As any non-zero entry in the estimated precision matrix represents an edge in the graph, the specificity and sensitivity of detecting an edge at different thresholds can be computed and used to plot the receiver-operating characteristic curve (ROC).

For the above measures, smaller values for relative difference and Frobenius norm indicate increased proximity to the ground truth. Values closer to 1 for the AUC plots indicate increased accuracy in reconstructing the network topology. When $K > 1$, we first matched the predicted precision matrix (of a component) to the nearest true precision matrix from the set of K true precision matrices. We used the Frobenius norm measure for this. After pairing the true and predicted matrices, we report their mean value statistics.

Table 1. Performance on synthetic data with $d = 200$, $sp = 0.9, K = 1,\ 2,\ 3$

	$n = 2d$	$n = 5d$	$n = 10d$
One component Frobenius norm			
MixMPLN	14.16	13.99	7.34
MixMPLN + huge(StARS)	5.62	16.02	7.15
MixMPLN + huge(fixed ρ)	4.86	2.87	2.08
MixMPLN + huge(iterative ρ)	5.62	3.01	2.15
MixMPLN + glasso(cross validation)	4.17	2.96	2.09
MixMPLN + glasso(fixed ρ)	4.86	2.87	2.08
MixMPLN + glasso(iterative ρ)	5.62	3.01	2.15
One component relative distance			
MixMPLN	27.36	30.23	11.75
MixMPLN + huge(StARS)	1.96	30.31	9.62
MixMPLN + huge(fixed ρ)	3.10	1.04	0.72
MixMPLN + huge(iterative ρ)	4.80	1.49	1.07
MixMPLN + glasso(cross validation)	1.52	1.23	0.77
MixMPLN + glasso(fixed ρ)	3.10	1.04	0.72
MixMPLN + glasso(iterative ρ)	4.80	1.49	1.07
Two components Frobenius norm			
MixMPLN	73.67	19.76	8.79
MixMPLN + huge(StARS)	5.60	5.42	15.29
MixMPLN + huge(fixed ρ)	27.24	5.28	3.48
MixMPLN + huge(iterative ρ)	25.94	4.33	2.88
MixMPLN + glasso(cross validation)	5.04	3.73	2.84
MixMPLN + glasso(fixed ρ)	28.83	5.28	3.48
MixMPLN + glasso(iterative ρ)	27.63	4.33	2.88
Two components relative distance			
MixMPLN	14443.03	56.52	14.42
MixMPLN + huge(StARS)	1.98	1.83	25.50
MixMPLN + huge(fixed ρ)	957.78	4.68	2.52
MixMPLN + huge(iterative ρ)	772.93	2.73	1.29
MixMPLN + glasso(cross validation)	1.82	1.30	1.04
MixMPLN + glasso(fixed ρ)	920.39	4.68	2.52
MixMPLN + glasso(iterative ρ)	842.48	2.73	1.29
Three components Frobenius norm			
MixMPLN	20.07	14.66	13.61
MixMPLN + huge(StARS)	6.93	5.90	5.46
MixMPLN + huge(fixed ρ)	19.15	8.58	6.04
MixMPLN + huge(iterative ρ)	18.56	6.11	4.97
MixMPLN + glasso(cross validation)	6.48	6.25	5.88
MixMPLN + glasso(fixed ρ)	19.67	11.02	8.03
MixMPLN + glasso(iterative ρ)	19.44	6.11	4.98
Three components relative distance			
MixMPLN	48131.48	4571.65	24.22
MixMPLN + huge(StARS)	8959.39	2.18	1.87
MixMPLN + huge(fixed ρ)	16096.08	12.68	6.70
MixMPLN + huge(iterative ρ)	5891.79	5.63	3.04
MixMPLN + glasso(cross validation)	2.60	2.46	3.55
MixMPLN + glasso(fixed ρ)	11828.42	2029.51	10.91
MixMPLN + glasso(iterative ρ)	5836.67	5.63	3.32

3 Results

We implemented the MixMPLN algorithms in R, and assessed their performance using the synthetic datasets. For our assessments, we

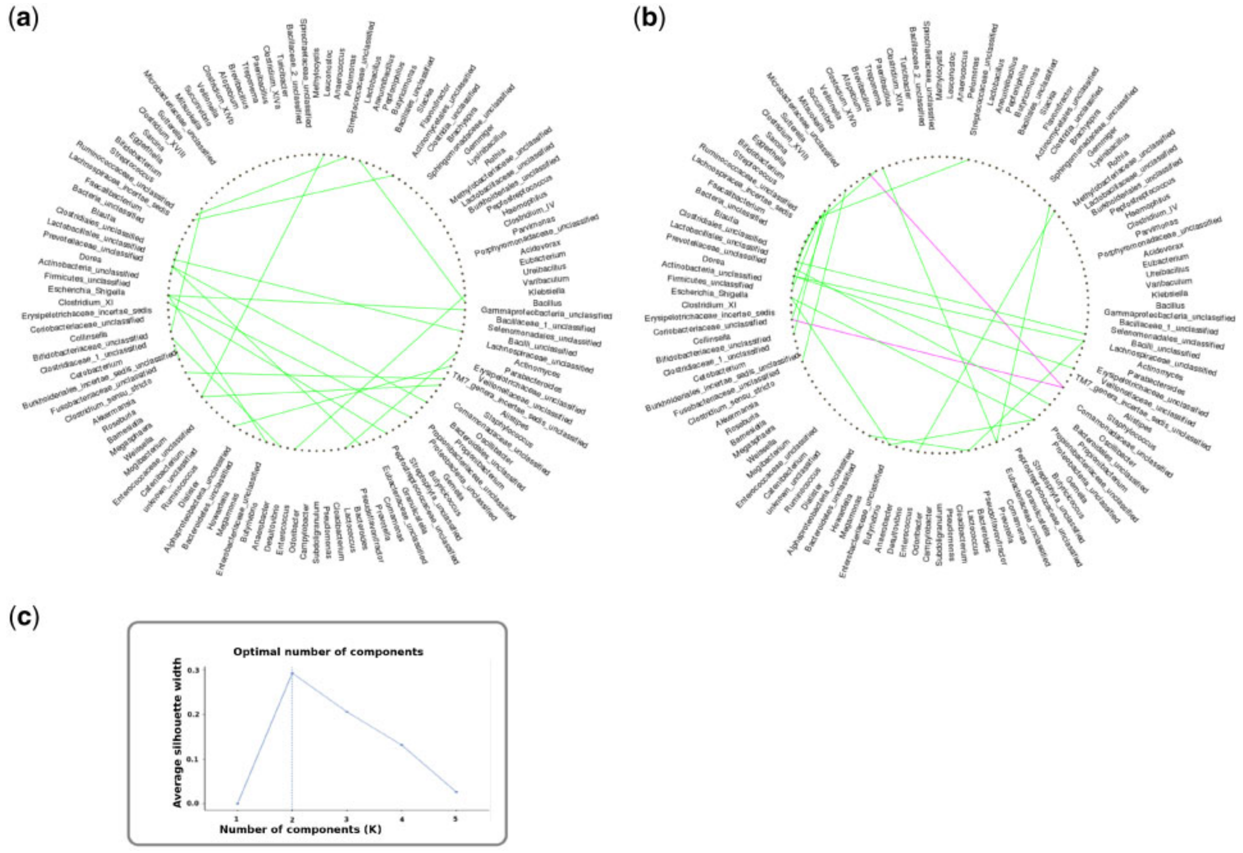

Fig. 3. Application of MixMPLN + glasso with cross validation to a real dataset. Green and red edges represent positive and negative entries respectively in the estimated partial correlation matrices. (**a**) Graph of Component 1 which contains 158 samples; (**b**) graph of Component 2 which contains 37 samples. The threshold to select the edges is 0.3; (**c**) Selection of the optimal number of the components

generated sample-taxa count matrices X (and their corresponding XS and TS matrices) for the following four sets of parameters:

- $(d = 30, sp = 0.5, n = [d, 100000d], K = 1,\ 2,\ 3)$,
- $(d = 100, sp = 0.7, n = [d, 500d], K = 1,\ 2,\ 3)$,
- $(d = 200, sp = 0.7, n = [d, 500d], K = 1,\ 2,\ 3)$,
- $(d = 200, sp = 0.9, d = 2d, 5d, 10d, K = 1,\ 2,\ 3)$.

For each dataset, each component mixing coefficient was $1/K$. In addition, five replicates were generated for each parameter combination. In total, 465 datasets were generated and analyzed. The datasets with $sp = 0.9$ were used to assess the performance of the MixMPLN algorithms in recovering sparse networks.

First we evaluated the ability of the MixMPLN algorithm to recover the true precision matrices with increasing sample size n. For this, we used the original count data, the sampled data and the TMM normalized data. Figure 1 shows the benchmark results for the parameter combination of $d = 200, sp = 0.7$ and $K = 1,\ 2,\ 3$; Supplementary Figures S1 and S2 show the corresponding results for $d = 30, sp = 0.5, K = 1,\ 2,\ 3$ and for $d = 100, sp = 0.7, K = 1,\ 2,\ 3$. The three benchmark criteria (relative difference, Frobenius norm and AUC) show that the entries in predicted precision matrices approach their true values as the sample size n increases. A strong convergence trend is seen using the original count data (blue curves/bar chart), with the AUC values approaching 0.97, 0.96 and 0.9 for $K = 1,\ 2,\ 3$ respectively. The drop in performance with increasing K is not unexpected given the smaller fraction of data available per component to infer the component parameters. The figure also shows that the accuracy of recovery of the true precision matrices is not as high when using the sampled data (red curves/bar chart) and the TMM normalized data (orange curves/bar chart). In addition, from our analysis, it is not immediately evident that applying a TMM type data normalization is advantageous for the purpose of inferring the underlying covariance structure of the network.

We also evaluated the performance of MixMPLN and its l_1-norm penalty variants in recovering sparse networks. Table 1 and Figure 2 show the performance of these methods for $d = 200$, $K = 1,\ 2,\ 3$ and sparsity level $sp = 0.9$. Sample sizes of $n = 2d, 5d, 10d$ were used in these evaluations. As can be seen, the performance for the methods generally improve with increasing n, and the l_1-norm penalty variants perform better than the unpenalized MixMPLN model on these data. The performances of the l_1-norm penalty variants are generally quite comparable to each other [with MixMPLN + huge(StARS) having a slightly lower performance compared to the others].

Since MixMPLN + glass(cross validation) performs better than the other approaches for $n = 2d$, we decided to apply this method to analyze the real dataset. We used the silhouette method from the 'factoextra' R package (Kassambara, 2017) to compute the optimal number of components from this sample-taxa matrix. Figure 3 shows the results of our analysis. We find that there is strong evidence for two underlying (and different) microbial networks ($K = 2$ components) associated with this sample-taxa data. We assigned component membership to the samples based on their final weights w_{il} [Equation (10)]. This resulted in Component 1 containing 158 samples and Component 2 containing 37 samples. The average age of the individuals in Component 1 was 9 years while that of individuals in Component 2 was 0.7 years. Our reconstructed networks are consistent with the observation that infants have a different gut microbiome composition compared to older children and adults (Yooseph *et al.*, 2015). The constructed networks include edges involving bacterial groups like *Bifidobacterium, Staphylococcus, Streptococcus* and *Escherichia–Shigella* that are known to be key players in early gut microbiome development. Our method identifies both positive and negative interactions between pairs of taxa (red and green edges) for the chosen threshold of 0.3; Supplementary Figure S3 shows the graph structures for other selected threshold values. The biological significance of these interactions needs to be investigated further.

4 Conclusion

We presented a mixture model framework and network inference algorithms to analyze sample-taxa matrices that are associated with K microbial networks. Future work will include the development of Bayesian approaches for model selection for this problem.

Conflict of Interest: none declared.

References

Aitchison,J. (1982) The statistical analysis of compositional data. *J. R. Stat. Soc. Series B (Methodol.)*, **44**, 139–177.

Aitchison,J. and Ho,C.H. (1989) The multivariate Poisson-log normal distribution. *Biometrika*, **76**, 643–653.

Bentley,D.R. (2006) Whole-genome re-sequencing. *Curr. Opin. Genet. Dev.*, **16**, 545–552.

Bilmes,J.A. *et al.* (1998) A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models. *Int. Comput. Sci. Inst.*, **4**, 126.

Biswas,S. *et al.* (2015) Learning microbial interaction networks from metagenomic count data. In: *International Conference on Research in Computational Molecular Biology, Warsaw, Poland*. pp. 32–43. Springer.

Falkowski,P.G. *et al.* (2008) The microbial engines that drive earth's biogeochemical cycles. *Science*, **320**, 1034–1039.

Fang,H. *et al.* (2015) CCLasso: correlation inference for compositional data through Lasso. *Bioinformatics*, **31**, 3172–3180.

Friedman,J. *et al.* (2008) Sparse inverse covariance estimation with the graphical Lasso. *Biostatistics*, **9**, 432–441.

Friedman,J. and Alm,E.J. (2012) Inferring correlation networks from genomic survey data. *PLoS Comput. Biol.*, **8**, e1002687.

Gloor,G.B. *et al.* (2017) Microbiome datasets are compositional: and this is not optional. *Front. Microbiol.*, **8**, 2224.

Hibbing,M.E. *et al.* (2010) Bacterial competition: surviving and thriving in the microbial jungle. *Nat. Rev. Microbiol.*, **8**, 15–25.

Holmes,I. *et al.* (2012) Dirichlet multinomial mixtures: generative models for microbial metagenomics. *PLoS One*, **7**, e30126.

Hunter,D.R. and Lange,K. (2004) A tutorial on MM algorithms. *Am. Stat.*, **58**, 30–37.

Huttenhower,C. *et al.* (2012) Structure, function and diversity of the healthy human microbiome. *Nature*, **486**, 207–214.

Jordan,M.I. (1999) *Learning in Graphical Models*. MIT Press, Cambridge, Massachusetts. ISBN 0-262-60032-3.

Kassambara,A. (2017) *Practical Guide to Cluster Analysis in R: Unsupervised Machine Learning*. Vol. **1**. STHDA. CreateSpace Independent Publishing Platform.

Kurtz,Z.D. *et al.* (2015) Sparse and compositionally robust inference of microbial ecological networks. *PLoS Comput. Biol.*, **11**, e1004226.

La Rosa,P.S. *et al.* (2012) Hypothesis testing and power calculations for taxonomic-based human microbiome data. *PLoS One*, **7**, e52078.

Lange,K. (2016) *MM Optimization Algorithms*. Vol. **147**. SIAM, Philadelphia.

Layeghifard,M. *et al.* (2017) Disentangling interactions in the microbiome: a network perspective. *Trends Microbiol.*, **25**, 217–228.

Liu,H. *et al.* (2010) Stability approach to regularization selection (StARS) for high dimensional graphical models. In: *NIPS'10 Proceedings of the 23rd International Conference on Neural Information Processing Systems*, Vol. 2. Curran Associates Inc., Vancouver, British Columbia, Canada, pp. 1432–1440.

Margulies,M. *et al.* (2005) Genome sequencing in microfabricated high-density picolitre reactors. *Nature*, **437**, 376–380.

Qin,J. *et al.* (2010) A human gut microbial gene catalogue established by metagenomic sequencing. *Nature*, **464**, 59–65.

Quail,M.A. *et al.* (2012) A tale of three next generation sequencing platforms: comparison of Ion Torrent, Pacific Biosciences and Illumina MiSeq sequencers. *BMC Genomics*, **13**, 341.

Robinson,M.D. and Oshlack,A. (2010) A scaling normalization method for differential expression analysis of RNA-seq data. *Genome Biol.*, **11**, R25.

Sender,R. *et al.* (2016) Revised estimates for the number of human and bacteria cells in the body. *PLoS Biol.*, **14**, e1002533.

Venter,J.C. *et al.* (2004) Environmental genome shotgun sequencing of the Sargasso sea. *Science*, **304**, 66–74.

Whitman,W.B. *et al.* (1998) Prokaryotes: the unseen majority. *Proc. Natl. Acad. Sci. USA*, **95**, 6578–6583.

Williamson,S.J., and Yooseph, S. (2012) From bacterial to microbial ecosystems (metagenomics). In: van Helden,J., Toussaint,A. and Thieffy,D. (eds.), *Bacterial Molecular Networks*. Springer Nature, Switzerland, pp. 35–55.

Woese,C.R. and Fox,G.E. (1977) Phylogenetic structure of the prokaryotic domain: the primary kingdoms. *Proc. Natl. Acad. Sci. USA*, **74**, 5088–5090.

Wu,T.T. *et al.* (2010) The MM alternative to EM. *Stat. Sci.*, **25**, 492–505.

Yooseph,S. *et al.* (2015) Stool microbiota composition is associated with the prospective risk of plasmodium falciparum infection. *BMC Genomics*, **16**, 631.

Zhao,T. *et al.* (2012) The huge package for high-dimensional undirected graph estimation in R. *J. Mach. Learn. Res.*, **13**, 1059–1062.

Zhou,H. and Lange,K. (2010) MM algorithms for some discrete multivariate distributions. *J. Comput. Graph. Stat.*, **19**, 645–665.