BANFF: An \textit{R} Package for BAyesian Network Feature Finder

Zhou Lan  
Georgia Institute of Technology  

Yize Zhao  
SAMSI  

Jian Kang  
Emory University  

Tianwei Yu  
Emory University

Abstract

Feature selection on high-dimensional networks plays an important role in understanding of biological mechanisms and disease pathologies. It has a broad range of applications. Recently, a Bayesian nonparametric mixture model (Zhao, Kang, and Yu 2014) has been successfully applied for selecting gene and gene sub-networks. We extend this method to a unified approach for feature selection on general high-dimensional networks; and we develop a powerful \textit{R} package, the Bayesian network feature finder (BANFF), providing a full package of posterior inference, model comparison, and graphical illustration of model fitting. In BANFF, we develop a parallel computing algorithm for the Markov chain Monte Carlo (MCMC) based posterior inference and an Expectation-Maximization (EM) based algorithm for posterior approximation, both of which greatly reduce the computational time for model inference. In this work, we provide detailed instruction on how to use the \textit{R} functions in BANFF along with several tutorial examples on analysis of simulated datasets and real datasets. Particularly, we demonstrate the use of BANFF on selecting features from a protein-protein interaction network and perform brain image segmentations.

\textit{Keywords}: Bayesian, network feature, Markov chain Monte Carlo.

1. Introduction

Feature selection over high-dimensional networks has become a very important research question motivated by the needs of analyzing big data in a broad range of biological and biomedical applications.

One important area is omics research, which includes genomics, transcriptomics, proteomics and metabolomics. Preexisting biological knowledge on the relationships between biological entities (genes, proteins etc.) is usually coded using the network data structure, with each
node representing a biological entity and each edge representing a relationship. Commonly used genome-scale networks include protein-protein interaction network (Rual, Venkatesan, Hao, Hirozane-Kishikawa, Dricot, Li, Berriz, Gibbons, Dreze, Ayivi-Guedehoussou et al. 2005), transcriptional regulatory network (Licatalosi and Darnell 2010), signal transduction network (Janes and Yaffe 2006), metabolomic network (Duarte, Becker, Jamshidi, Thiele, Mo, Vo, Srivas, and Palsson 2007), etc. They play important roles in the analyses and interpretation of high-throughput data. Jointly analyzing high-throughput data with the networks can incorporate existing biological knowledge to help achieve better feature selection, more robust predictive models, and more interpretable biological results (Barabási 2007; Barabási, Gulbahce, and Loscalzo 2011; Chan and Loscalzo 2012).

Another important application is neuroimaging, which includes the use of various techniques to either directly or indirectly measure the brain structure and function. The commonly used functional neuroimaging techniques, such as functional magnitude resonance imaging (fMRI) and Positron emission tomography (PET), detect neural activities at a set of locations in the brain, to which we refer as voxels/regions. The brain activities at each voxel/region are considered as biomarkers or features that are potentially associated with brain functions and neurological disorders. The spatial neighborhoods of voxels/regions constitute a spatial network. Also, the functional/structure connectivities that characterize the coherence of the brain activities can make up another level of network in the brain imaging studies. It has been shown that the brain signals over those two types of networks can be highly dependent. Thus, incorporating the networks information would be more efficient and powerful to perform selections from important imaging biomarkers that are associated with the neurological or psychiatric disorders.

Recently, many statistical methods have been proposed to perform feature selection incorporating network information (Li and Li 2008; Pan, Xie, and Shen 2010; Stingo, Chen, Tadesse, and Vannucci 2011; Ma, Shi, Li, Yi, and Shia 2010; Qu, Nettleton, and Dekkers 2012). Among those, a Bayesian nonparametric mixture model (Zhao et al. 2014) has been successfully applied to select genes and gene sub-networks under the large-scale simultaneous hypothesis testing framework (Efron 2004). This method provides a general framework and can be applied to a wide range of aforementioned biomedical applications. Under Bayesian modeling framework, this method can provide good feature selection accuracy compared to other existing methods and produce reliable uncertainty estimates. Also, taking advantages of nonparametric Bayesian modeling, this method is not sensitive to model assumptions and can make more robust posterior inference. The fast posterior approximation algorithms developed by Zhao et al. (2014) are much faster than the standard MCMC algorithm. Although such a valuable method has been built, the efficient implementation is currently not available, which becomes the main barrier to the use of this method.

To this end, we develop an R package: BAyes Network Feature Finder (BANFF) in CRAN, which is a user-friendly, computationally efficient, publicly available and well-maintained software package. The BANFF implements the standard posterior inference algorithm which combines network based Dirichlet process mixture (DPM) model fitting and the hierarchical ordered density clustering (HODC). It also provides option for the two fast computational algorithms including finite Gaussian mixture (FGM) approximation. To further speed up the MCMC posterior inference simulation, the BANFF also implements the parallel MCMC computing algorithms. Based on our experience, this method can reduce 67% computing time when using seven cores in a typical personal computer. In addition, the BANFF
provides a full package of functions which perform data preprocessing and transformation, conduct Bayesian model fitting and diagnostics, compute posterior summary statistics and generate graphical presentations of results.

In this article, we present an introduction to our R package: BANFF. In Section 2, we start with a brief review of the Bayesian nonparametric mixture model for feature selection over high-dimensional networks along with the posterior computation algorithms. In Section 3, we provide detailed usage of the package. In Section 4, we present simulation studies and real data applications. Finally, we conclude the paper in Section 5 with a brief discussion on the future work.

2. Model

2.1. Notations

We summarize notations and their definitions in the model in Table 1. For a large scale hypothesis testing problem, if only p-values are observed, they can be transformed to be testing statistics, for example, $r_i = -\phi^{-1}(p_i)$, where $\phi^{-1}(\cdot)$ is the inverse of the cumulative distribution function (CDF) of standard normal distribution. We assume that “important” features are characterized by larger testing statistics compared to “unimportant” ones. In the model, the observed data are the testing statistics $r$ or $p$-values; and the adjacency matrix $C$ for network configurations are also assumed to be known. The latent feature selection indicator $z$ is of our primary interest.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n$</td>
<td>an integer, total number of features</td>
</tr>
<tr>
<td>$p$</td>
<td>a vector of $n$ p-values, $p = (p_1, \ldots, p_n)'$</td>
</tr>
<tr>
<td>$r$</td>
<td>a vector of $n$ transforming statistics by $r_i = -\phi^{-1}(p_i)$, $r = (r_1, \ldots, r_n)'$</td>
</tr>
<tr>
<td>$z$</td>
<td>a vector of $n$ latent indicators, $z = (z_1, \ldots, z_n)'$, $z_i = 1$ indicates “important feature”, $z_i = 0$ otherwise</td>
</tr>
<tr>
<td>$C$</td>
<td>an $n \times n$ adjacency matrix, $c_{ij} = 1$ if features $i$ and $j$ are connected, $c_{ij} = 0$ otherwise</td>
</tr>
</tbody>
</table>

Table 1: A summary of notations and definitions in the model.

2.2. Network-Based DPM Model

We assume that each testing statistic $r_i$ follows a normal distribution with mean $\mu_i$ and variance $\sigma_i^2$, denoted $N(\mu_i, \sigma_i^2)$. The distribution of parameters $(\mu_i, \sigma_i^2)$ given the selection indicator $z_i = k$ is defined by random probability measure $G_k$, for $k = 0, 1$. The random measure $G_k$ follows a Dirichlet process with base measure $G_{0k}$ and scalar precision $\tau_k$. Also we specify $G_{0k} = N(\gamma_k, \epsilon_k^2) \times IG(\alpha_k, \beta_k)$, where $IG(\alpha_k, \beta_k)$ denotes an inverse-Gamma
distribution with shape $\alpha_k$ and rate $\beta_k$. The DPM model for feature selection is given by

\[
[r_i | \mu_i, \sigma_i^2] \sim N(\mu_i, \sigma_i^2), \\
[(\mu_i, \sigma_i^2) | z_i = k, G_k] \sim G_k, \\
G_k \sim \text{DP}(G_{0k}, \tau_k), \\
G_{0k} = N(\gamma_k, \xi_k^2) \times \text{IG}(\alpha_k, \beta_k),
\]

(1)

for $i = 1, 2, \ldots, n$ and $k = 0, 1$. A weighted Ising prior is assigned to $z$ to incorporate the network information. The probability mass function is given by

\[
\pi(z | \pi, \rho, \omega, C) \propto \exp \left[ \sum_{i=1}^{n} (\tilde{\omega}_i \log(\pi_{zi} + \rho_{zi} \sum_{i \neq j} \omega_j C_{ij} I[z_i = z_j])) \right],
\]

(2)

where parameter $\pi = (\pi_0, \pi_1)$ with $0 < \pi_1 = 1 - \pi_0 < 1$ controls the sparsity of $z$. Parameter $\rho = (\rho_0, \rho_1)$ with $\rho_k > 0$ for $k=0,1$ characterizes the smoothness of $z$ over the network. The pre-determined weights $\omega = (\omega_1, \omega_2, \ldots, \omega_n)'$ is used to incorporate a priori biological knowledge to each node. And $\tilde{\omega}_i = \sum_{j=1}^{n} c_{ij} \omega_j / \tilde{\omega}_i c_{ij}$ is introduced to balance the contribution from $\pi$ and $\rho$ on the prior probability of $z$. We refer to Zhao et al. (2014) for more details on the model constraints and model representations.
we modify method (b) for hyper-parameter selection in the package BANFF. Specifically, denote by \( \{(\pi_m, \rho_m)\}_{m=1}^{M} \) all possible choices of hyper-parameters. Given each \((\pi_m, \rho_m)\), we integrate out all the parameters in the model and compute the marginal density of testing statistics using the Monte Carlo simulation. The optimal choice of \((\pi_m, \rho_m)\) is then selected by maximizing the marginal density. Hyper-parameter specification is the very first step before any posterior computational algorithms. BANFF has already embed this procedure in the main posterior inference functions `Network.STD()` and `Network.Fast()`. Also, it includes a separate function `HyperPara.Select()` for implementing the hyper-parameter specification. We provide more details in Section 3.

2.4. Fast computation algorithms

Since the standard posterior computation algorithm (Zhao et al. 2014) (NET-DPM-1) is very computationally expensive when the dimension of feature space is very large, Zhao et al. (2014) proposed two fast algorithms to fit finite Gaussian mixture models (FGM), referred as NET-DPM-2 and NET-DPM-3. BANFF implements NET-DPM-3 in function `Networks.Fast()` for fast computation. This algorithm consists of two parts: (1) the FGM approximation and (2) applying the HODC algorithm for fitting the density function of “important” and “unimportant” features and further perform feature selection. We will present this algorithm by introducing the HODC algorithm first; and then discuss the details of FGM approximation that is guided by a DPM model fitting or model-based clustering fitting in the following section.

Hierarchical Ordered Density Clustering (HODC) Algorithm

As discussed by (Zhao et al. 2014), the marginal distribution of \( r_i \) could be also approximated by a DPM model:

\[
\pi(r) = \sum_{k=0}^{1} p_k f_k(r) = \sum_{g=-L_0+1}^{L_1} \frac{q_g}{\sigma_g} \phi\left( \frac{r - \tilde{\mu}_g}{\sigma_g} \right),
\]

where \( L_0 \) and \( L_1 \) represent the number of mixture components that contribute to the densities of testing statistics of “unimportant” feature and “important” feature, respectively. Without incorporating network information, we can have an approximation to the marginal density of \( r \) by a DPM model fitting. Denote by \( \varphi = \{(\tilde{\mu}_g, \tilde{\sigma}_g, \tilde{\rho}_g)\}_{g=1}^{L_0+L_1} \) parameters of the FGM models that are generated by a DPM model fitting. We rank these densities by means \( \tilde{\mu}_0 < \tilde{\mu}_1 < \ldots < \tilde{\mu}_{L_0+L_1} \). A distance metric of density functions are defined by

\[
d(f, f') = \int_{-\infty}^{\infty} [f(x) - f'(x)]^2 dx.
\]

The pseudo code is provided as follows:

1: \( m \leftarrow 0 \)
2: \( s_l^{(0)} \leftarrow \{l\}, \) for \( l = 1, 2, \ldots, L_0 + L_1 \)
3: \( \text{while } L_0 + L_1 - m >= 2 \) do
4: \( \text{for } l^{(m)} \leftarrow 1 \) to \( L_0 + L_1 - m - 1 \) do
5: \( l^{(m)} \leftarrow d(\tilde{\phi}(; s_l^{(m)}, \varphi), \tilde{\phi}(; s_{l+1}^{(m)}, \varphi)) \)
6: \( \text{end for} \)
7: \( l_{min} \leftarrow \min(l^{(m)}, l_2^{(m)}, \ldots, l_{L_0+L_1-m-1}^{(m)}) \)
8: \( \text{if there are more than one } l_{min} \) then
After \( m = L_0 + L_1 - 2 \) steps, we obtain two sets of Gaussian densities which provide an approximation to
\[
 f_k(r) = \sum_{g \in a_k} \frac{q_g}{\sigma_g} \phi\left(\frac{r - \hat{\mu}_g}{\sigma_g}\right).
\]

Two Fast algorithms

**DPM model fitting:** By DPM model fitting, we obtain \( V \) posterior samples of the parameters of the marginal density of \( r \), \( \varphi_v = \{(\hat{\mu}_g, \hat{\sigma}_g^2, \hat{p}_g)\}_{g=1}^{L_0+L_1} \) for \( v = 1, 2, \ldots, V \) and for each \( \varphi_v \), we applied the HODC algorithm to obtain two sets of mixture Gaussian densities, denoted \( a_{v,0} \) and \( a_{v,1} \). Thus, the marginal posterior distribution of \( z \) is approximated by
\[
\frac{1}{V} \sum_{v=1}^{V} \pi(z|r, \{\tilde{\phi}(r; a_{v,0}, \varphi_v), \tilde{\phi}(r; a_{v,1}, \varphi_v)\}),
\]
where for each \( v \), the full conditional \( \pi(z|r, \{\tilde{\phi}(r; a_{v,0}, \varphi_v), \tilde{\phi}(r; a_{v,1}, \varphi_v)\}) \) can be simulated by Gibbs sampling. The BANFF implements this procedure in function `DPM.HODC()`. The BANFF also implements parallel computing: When \( V \) posterior samples of the parameters of the marginal density of \( r \) is obtained, we subsequently update the \( z_i \) by each parameters as the input of Ising prior. For the \( V \) outputs, we then average the \( z \) for each iteration as final output.

**EM algorithm:** The parameter estimates \( \varphi = \{(\hat{\mu}_g, \hat{\sigma}_g^2, \hat{p}_g)\}_{g=1}^{L_0+L_1} \) can also be obtained by an EM algorithm which has developed for model-based clustering (Fraley and Raftery 2002; Reynolds 2009). We maximize BIC to determine \( L_0 + L_1 \). The BANFF implements this algorithm in function `EM.HODC()`. Other steps such as HODC algorithm and Ising prior is same as DPM model fitting. But the process of averaging is avoided.

**Combination:** The above two algorithms are combined and implemented in function `Networks.Fast()` in BANFF. Figure 2 presents the flowchart of this algorithm. The red
arrows represent the route of generating inputs by DPM model fitting and the blue ones represent the route of generating inputs by an EM algorithm. The common routes are using black arrows. For the details of the implementation, please refer to Section 3. Specifically, function `Networks.Fast()` will call `DPM.HODC()` or `EM.HODC()` when the corresponding option `algorithms="DPM"` or `algorithms="EM"` is chosen first, and subsequently update the $z_i(v)$ by parameter $\wp(v)$ as the input of Ising prior.

2.5. Standard posterior computation algorithm

The standard posterior computation algorithms can be developed based on an equivalent model representation. This has been discussed in Zhao et al. (2014) (NET-DPM-1). Please refer Section 2.2 and Appendix B1 of Zhao et al. (2014) for details. BANFF has implemented this algorithm in function `Networks.STD()`. Please see Section 3 for more details.

3. Implementation

The developed package BANFF imports or depends on the packages pscl (Jackman 2014), tmvtnorm (Wilhelm and G 2014), DPpackage (Jara, Hanson, Quintana, Müller, and Rosner 2011; Jara 2007), coda (Plummer, Best, Cowles, and Vines 2006), mclust (Fraley, Raftery, Murphy, and Scrucca 2012), igraph (Csardi and Nepusz 2006), network (Butts, Handcock, and Hunter 2014), foreach (Analytics and Weston 2014b) and doParallel (Analytics and Weston 2014a). It contains four major functions:

- `Networks.STD()`, `Networks.Fast()`, `EM.HODC()`, and `DPM.HODC()`

along with a couple of additional functions for summarizing results and graphical presentations. `Networks.STD()` implements the standard algorithm to perform feature selection and sub-network selection; `Networks.Fast()` implements a hybrid fast algorithm to perform feature selection and sub-network selection described; `EM.HODC()` implements the HODC algorithm guided by the EM algorithm for the model based clustering; `DPM.HODC()` implements the HODC algorithm guided by a DPM model fitting. The major function usage are summarized as follows:

`Networks.STD(pvalue, net, iter = 5000, nburns = 2000, piall = c(0.75, 0.8, 0.85, 0.9), rhoall = c(0.5, 1, 5, 10, 15), status=FALSE, fit, show.steps=1, showlikelihood=FALSE, likelihood.frequency=100)`

`Networks.Fast(pvalue, net, iter = 5000, nburns = 2000, algorithms=c("EM","DPM"), v = 20, DParallel=FALSE, n.nodes=1, DPM.mcmc=list(nburn=2000, nsave=1, nskip=0, ndisplay=10), DPM.prior=list(a0=2, b0=1, m2=rep(0,1), s2=diag(100000,1), psiinv2=solve(diag(0.5,1)), nu1=4, nu2=4, tau1=1, tau2=100), piall = c(0.8, 0.85, 0.9, 0.95), rhoall = c(1, 2, 5, 10, 15), show.steps=10, showlikelihood=FALSE, likelihood.frequency=100)`

`EM.HODC(pvalue)`

Start

Apply hyper-parameter selection

Set initial $z_i$ by k-means clustering

Data $r$, posterior samples of parameters $\{\varphi_v\}_{v=1}^V$ by DPM fitting and index sets $\{a_{v,0}, a_{v,1}\}_{v=1}^V$

DPM or EM?

Update $z_i$ using Gibbs sampling

Is the distribution of $z_i$ stable?

DPM or EM?

posterior distribution of $z_i$

Resample $z_i$

Stop

parallel into $V$ jobs

Figure 2: Fast posterior approximation algorithms
3.1. Data input

There are two common arguments: `pvalue` and `net` in both functions `Networks.STD()` and `Networks.Fast()` as the input data. The argument `pvalue` specifies a vector of p-values obtained from large scale statistical hypothesis testing. The argument `net` specifies the adjacency matrix $C$ indicating the networking configuration, where $c_{ij} = 1$ if features $i$ and $j$ are connected over the network, $c_{ij} = 0$, otherwise. Of note, $C$ is symmetric and its diagonal elements are all zeros.

3.2. Parameter specifications

Appropriate parameter specifications are vitally important for obtaining accurate feature selection results. Model (1) involves a set of hyper-parameters including $\gamma_k$, $\xi_k$, $\beta_k$, $\tau_k$ and $\alpha_k$ for $k = 0, 1$. In `Networks.STD()` and `Networks.Fast()`, we set $\tau_0 = 10$, $\tau_1 = 2$ and $\beta_0 = \beta_1 = 10$. The parameters $\{\gamma_0, \xi_0, \gamma_1, \xi_1\}$ are specified by clustering the sample into two clusters using k-means and computing the sample mean and variance accordingly. Also, we set $\alpha_k = \beta_k/\xi_k^2 + 1$, for $k = 0, 1$.

As discussed in Section 2.3, the parameter $\pi = (\pi_0, \pi_1)$ with $0 < \pi_1 = 1 - \pi_0 < 1$ which controls the sparsity of $z$, the parameter $\rho = (\rho_0, \rho_1)$ with $\rho_k > 0$ for $k = 0, 1$ characterizes the smoothness of $z$ over the network. The functions take arguments `piall` and `rhoall`: collections of possible choices on $\pi_0$ and $\rho_k$ for $k = 0, 1$ where we assume that $\rho_0$ and $\rho_1$ take the same set of possible values. The optimal combinations of $\{\pi_0, \rho_0, \rho_1\}$ are then taken by maximizing the marginal likelihood which are computed by Monte Carlo integration. Although we provide default specifications on `piall` and `rhoall` in function `Networks.STD()` and `Networks.Fast()`, a good choice might still depend on the real data applications and specific data sets. We provide a few examples in Section 4 where we conduct simulation studies for different scenarios and detailed analyses of real data sets.

3.3. DPM model fitting or EM?

In Section 2.5, two fast computational algorithms are developed based on the DPM model fitting and the EM algorithm respectively. In `Networks.Fast()`, an argument `algorithms` is introduced to specify which algorithm will be called in the programming: "DPM" or "EM".

If we set `algorithm="DPM"`, `Networks.Fast()` will perform the DPM model fitting by calling function `DPdensity()` in the package `DPpackage` (Jara et al. 2011; Jara 2007) which generates $V$ posterior samples of parameters $\{\psi_v\}_{v=1}^V$. Arguments `DPM.mcmc` and `DPM.prior` are provided to control the iteration information and prior information for the MCMC chain of DPM fitting, respectively. Please refer the setting of arguments `mcmc` and `prior` of function `DPdensity()` in the manual of `DPpackage` (Jara et al. 2011; Jara 2007) for more details. By applying the HODC algorithm, index sets $\{a_v,0,a_v,1\}_{v=1}^V$ which specify the clustering results are generated for each set subsequently. To reduce the computational time, we offer the parallel computing option in `Network.Fast()` which is controlled by
DPparallel; and another option n.cores is introduced to determine the number of CPU cores to be used. The parallel computing implementation depends on packages doParallel (Analytics and Weston 2014a) and foreach (Analytics and Weston 2014b).

If we set algorithms="EM", Network.fast() will implement the EM algorithm by calling function Mclust() of the package mclust (Fraley et al. 2012) which produces a set parameter estimations \( \{\wp\} \). Index sets \( \{a_0, a_1\} \) are generated by the HODC algorithm subsequently.

The above two approaches have their own advantages and limitations. The implementation of the EM algorithm is straightforward and fast. However, it is less accurate to estimate densities either when the network dimension is small or the signal of “important” features is relatively weak. In contrast, the DPM model fitting is more accurate to estimate densities, however, it requires more computation and takes a longer time to run. With the parallel computing option, the entire computation can speed it up very well.

3.4. Options for monitoring the MCMC convergence

We provide arguments showlikelihood and likelihood.frequency to compute the log-likelihood periodically and produce a trace plot of the log-likelihood to help monitor the convergence of the MCMC chain. BANFF also has a function Likelihood.History() that computes the likelihood values using the output of Networks.Fast() or Networks.STD() to generate the trace plot. In case that the simulated Markov chain is not converged, we can continue the posterior simulation instead of starting over by setting status=TRUE and fit=total$model, where total is the output of Networks.STD().

3.5. Other input

To control the MCMC posterior simulations, we have the arguments iter and nburns to specify the total number of iterations and burn-in iterations, respectively; and argument v is introduced to set the number of posterior samples produced by a DPM model fitting. To monitor the whole computational process, we provide argument show.steps that customize the iteration information being printed out on the console during the process. These two settings only control the number of iterations being printed, has nothing to do with saving the results. The default settings for these arguments are iter=5000, nburns=2000, v=20, show.steps = 10.

3.6. Outputs

- **Networks.Fast()**: a list of four elements: trace, statistics, where trace includes posterior samples of \( z_i \) and statistics contains several associated summary statistics including mean, median, variance and quantiles, convergence which contains convergence diagnostic results and graph which is the igraph object of full network.

- **Networks.STD()**: a list of two elements: trace and model, where trace provides posterior samples of cluster label \( g_i \), model provides a list of parameters at current state in the Markov chain, convergence which contains convergence diagnostic results and graph which is the igraph object of full network.
• **DPM.HODC()**: a list of four elements indicating density clustering results by the HODC algorithm: `mean`, `variance`, `pro` and `classification` providing mean estimates, variance estimates, probability estimates and classifications configurations of unimportant ($a_0$) and important clusters ($a_1$). The inputs of HODC algorithm (the parameter estimates $\varphi$ and classification information) are obtained by DPM model fitting.

• **EM.HODC()**: Similar to **DPM.HODC()** but the inputs of HODC algorithm are obtained by EM algorithm.

3.7. Outputs Processing

The objects of class "Network.Fast" and "Network.STD" outputted by **Network.Fast()** and **Network.STD()**, we provide S3 methods including `summary()` and `plot()` for processing the objects class "Network.Fast" and "Network.STD". A summary of convergence test conducted by the method of Heidelberger and Welch diagnostic (Heidelberger and Welch 1983), Hyper-Parameter Selection and a classification table would be shown on the Console, when applying `summary()` and a selected sub-network would be plotted when applying `plot()`. Here is a typical example when applying `summary()`, showing the MCMC chain is converged and the results of hyper-parameter selection and node-classification:

i. Convergence results:

Stationarity start p-value
test iteration
var1 passed 1 0.277

Halfwidth test
Mean Halfwidth test
var1 passed 91.1 2.86

ii. Hyper-Parameter Selection:

$pi0 = 0.8$ $rho0 = 0.5$ $rho1 = 1$

iii. Classification Table:

<table>
<thead>
<tr>
<th>classification</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>74</td>
<td>26</td>
</tr>
</tbody>
</table>

We will talk about the plot outputted by processing method `plot()` in later sections.

4. Examples

In this section, we illustrate **BANFF** via several examples on analysis of simulated data and real data. Specifically, using **BANFF**, we repeat one simulation study in Zhao et al. (2014) for gene selection and perform additional simulation studies for biomedical image segmentation.
We provide detailed instructions on parameter specifications in the model, e.g. $\pi_0$ and $\rho$. We provide R code for all examples.

In this section, we refer to the standard posterior computation algorithm for network feature selection as NET-STD; and refer to the two fast computation algorithms based on the EM algorithm and the DPM model fitting as NET-EM-Fast and NET-DPM-Fast, respectively. In contrast, without incorporating the network information in the model we also implement the algorithm that directly perform the FGM approximations using the DPM model fitting or the EM algorithm combined with the HODC algorithm. We refer to this algorithm as DPM-STD or EM-STD accordingly.

4.1. Simulation study 1: gene selection

We use BANFF to repeat the simulation study 1 conducted by Zhao et al. (2014). The goal is to evaluate the feature selection accuracy on a simulated scale-free network of 1,000 genes. The network is generated based on the rich-get-rich algorithm, implementing the function `barabasi.game()` in R package igraph. The “important” genes are generated from the Ising model\(^2\) with the sparsity parameter $\pi_0 = 0.8$, smoothness parameters $\rho = (\rho_0, \rho_1) = (5, 10)$. The simulated network is shown in Figure 3.

![Figure 3: Simulated scale-free network of 1,000 Genes](image)

We generate test statistics for “important” genes and “unimportant” genes from $N(0.5, 0.2)$ and $N(0, 0.2)$, respectively. The possible choices for $\pi_0$ are $(0.8, 0.85, 0.9, 0.95)$ and these for $\rho_k$ are $(1, 2, 5, 10, 15)$ for both $k = 0$ and $k = 1$. We run 5,000 iterations with 2,000 burn-in. For this simulation study, we compare NET-EM-Fast and STD-EM to show the power of incorporating the network information in the model. The arguments in function `Networks.Fast()` are specified as follows:
R> library("BANFF")
R> total=Networks.Fast(pvalue,net,iter=5000,
+ nburns=2000, algorithms="EM","piall=c(0.8, 0.85, 0.9, 0.95),
+ rhoall=c(1, 2, 5, 10, 15))

We summarize the feature selection accuracy over the simulated gene network in Table 2, where the true positive rate (TPR) is defined as the proportion of selected features among all “important” features. The false positive rate (FPR) refers to the proportion of selected features among all “unimportant” features. The false discovery rate (FDR) is the proportion of “unimportant” features among all selected features.

<table>
<thead>
<tr>
<th></th>
<th>NET-EM-Fast</th>
<th>STD-EM</th>
</tr>
</thead>
<tbody>
<tr>
<td>TPR</td>
<td>0.925</td>
<td>0.573</td>
</tr>
<tr>
<td>FPR</td>
<td>0.016</td>
<td>0.014</td>
</tr>
<tr>
<td>FDR</td>
<td>0.035</td>
<td>0.016</td>
</tr>
</tbody>
</table>

Table 2: Summary of the gene network feature selection accuracy.

This simulation study reproduces the results that are reported in Zhao et al. (2014), indicating that with incorporating the network information, the NET-EM-Fast can produce much higher TPR compared to STD-EM. Also, the FPR and the FDR are comparable between the two methods.

4.2. Simulation study 2: image segmentation

In this section, we apply BANFF to the biomedical statistic image segmentation problem. Here, the statistic image is defined as a 2D/3D array of test statistics or p-values that can either characterize the brain activities or show the contrasts between different brain or body tissues. Our goal is to identify a collection of spatially contiguous “important” voxels, e.g. regions, that show significance for the corresponding hypothesis testing problem.

To define the network on an image, the voxels are considered as nodes and each voxel only connects with the voxels in its neighborhood. Take a 50 × 50 image for example, the number of voxels is 2,500; and the voxel $(i,j)$ for $1 \leq i,j \leq 50$ is connected with voxels $(i-1,j),(i+1,j),(i,j-1)$ and $(i,j+1)$. The network of a 50 × 50 image is shown in Figure 4.

With the assistance of the function graph.lattice() in the R package igraph, we can quickly obtain the 2500 × 2500 adjacency matrix using the codes below (net is the adjacency matrix):

R> library("igraph")
R> g <- graph.lattice(length=50,dim=2)
R> net <- as(get.adjacency(g,attr=NULL),"matrix")

50 × 50 image segmentation 1

We generate test statistics of “important” voxels and the test statistics of “unimportant”
Figure 4: Network of a 50 × 50 image, where the red nodes represent “important” ones

voxels from N(0.5, 0.2) and N(0, 0.2), respectively. We simulate 50 data-sets accordingly for 50 × 50 image. The histogram of input test statistics of typical 50 × 50 image with such setting is in Figure 5.

The possible choices of $\pi_0$ are (0.8, 0.85, 0.9, 0.95) and these for $\rho_k$ for both $k = 0$ and $k = 1$ are (0.25, 0.5, 1, 2, 5). For each of the fast computing algorithms: NET-DPM-Fast and NET-EM-Fast, we run 5,000 iterations with 2,000 burn-in. By tracking the likelihood value by the function `Likelihood.History()` in BANFF, the Markov chain simulated by the NET-STD gets stable and shows mixing well within 200 iterations. Thus, we only run STD-NET 500 iterations with 200 burn-in. The arguments setting of the functions of `Networks.STD()` and `Networks.Fast()` can be set accordingly as

\begin{verbatim}
R> library("BANFF")
R> total0=Networks.STD(pvalue,net,iter=500,
+ nburns=200, piall=c(0.8, 0.85, 0.9, 0.95)
+ ,rhoall=c(0.25, 0.5, 1, 2, 5))
R> total1=Networks.Fast(pvalue,net,iter=5000,
+ nburns=2000, algorithms="DPM",DPparallel=TRUE,n.cores=8,
+ piall=c(0.8, 0.85, 0.9, 0.95), rhoall=c(0.25, 0.5, 1, 2, 5))
R> total2=Networks.Fast(pvalue,net,iter=5000,
+ nburns=2000, algorithms="EM",
+ piall=c(0.8, 0.85, 0.9, 0.95), rhoall=c(0.25, 0.5, 1, 2, 5))
\end{verbatim}

where \texttt{total0, total1} and \texttt{total2} return the object obtained by NET-STD, NET-DPM-Fast and NET-EM-Fast, respectively.
Figure 5: The histogram of input test statistics of typical $50 \times 50$ image data in image segmentation 1. The pink bars represent “important” and blue ones represent “unimportant”

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>TPR</td>
<td>0.64</td>
<td>0.73</td>
<td>0.94</td>
<td>0.99</td>
<td>0.98</td>
</tr>
<tr>
<td>FPR</td>
<td>0.24</td>
<td>0.03</td>
<td>0.02</td>
<td>0.01</td>
<td>0.01</td>
</tr>
<tr>
<td>FDR</td>
<td>0.27</td>
<td>0.09</td>
<td>0.03</td>
<td>0.01</td>
<td>&lt;0.01</td>
</tr>
<tr>
<td>Time (hrs)</td>
<td>0.20</td>
<td>0.10</td>
<td>1.00</td>
<td>0.40</td>
<td>6</td>
</tr>
</tbody>
</table>

Table 3: $50 \times 50$ image segmentation accuracy of node (voxel) selection 1.

The TPR, FPR and FDR of selecting each node (voxel) and typical computation time are summarized in Table 3 to demonstrate the high selection accuracy via our method. Since we are also interested in discovering “important” sub-network (region), it is not sufficient to show the accuracy of “important” nodes selection. Therefore, we also present the results of “important” sub-network (region) selection. In “important” sub-network (region) selection, we define a correct sub-network selection if more than $\tau\%$ voxels in the “important” sub-network are selected and less than $1 - \tau\%$ voxels which are connected to the “important” sub-network are selected. All the other selections are considered as incorrect selection. The $\tau\%$ here is called “Tolerance”.

Under the Tolerance $\tau\%=90\%$, the correct selection rates of NET-DPM-Fast, NET-EM-Fast and NET-STD are 0.66, 0.75 and 0.85, respectively. And those of STD-DPM and STD-EM are 0.10 and 0.00, respectively.

As shown in Figure 6, we compare the true “important” sub-network and a typical selected sub-network that include over 90% voxels in the true sub-network. This figure can be generated by using function `Plot.Subnetwork()` in BANFF. Such figure could also be obtained when applying S3 method `plot()` when processing the object classed by "Networks.STD" or "Networks.Fast".
This simulation study indicates a substantial improvement in accuracy when applying the Ising prior in segmentation, compared to the image segmentation without using the network information. Our method produces much higher TPR and lower FPR and FDR both in “important” nodes and sub-network selection. Also, NET-STD provides a much better selection accuracy for the “important” sub-network, compared with other methods.

50 × 50 image segmentation 2

We modify the distribution of test statistics in this simulation study to demonstrate the performance. Specifically, we generate testing statistics of “unimportant” voxels from the standard Gaussian distribution $N(0, 1)$. We consider two scenarios of “important” voxels:

- Mixture of Gaussian Distributions: $0.4N(3, 1) + 0.6N(2, 0.5)$,
- Mixture of Non-Gaussian Distributions: $0.4G(5, 2) + 0.6G(6, 3)$

where $G(\alpha, \beta)$ denotes a gamma distribution with shape parameter $\alpha$ and rate parameter $\beta$. The histogram of input testing statistics in a typical 50 × 50 image is shown in Figure 7, from which we can see that the distribution of “important” testing statistics is skewed and non-Gaussian data has a longer tail than Gaussian data.

The simulation method and argument specifications are set the same as those in 50 × 50 image segmentation 1. We summarize nodes selection results of image segmentation 2 in Table 4. For Gaussian mixture, under the Tolerance $\tau\% = 90\%$, the correct selection rates of NET-DPM-Fast, NET-EM-Fast and NET-STD are 0.84 and 0.86, respectively and those of STD-DPM and STD-EM are 0.58 and 0.00, respectively. For Non-Gaussian mixture, under the Tolerance $\tau\% = 90\%$, the correct selection rates of NET-DPM-Fast, NET-EM-Fast and NET-STD are 0.84 and 0.86, respectively and those of STD-DPM and STD-EM are 0.58 and 0.00, respectively.
Figure 7: Histogram of simulated test statistics of a 50 × 50 image from different mixture distributions. The left is the Gaussian data, and the right is the non-Gaussian data. The pink bars represent “important” voxels and blue ones represent “unimportant” voxels.

<table>
<thead>
<tr>
<th></th>
<th>Gaussian data</th>
<th>Non-Gaussian data</th>
</tr>
</thead>
<tbody>
<tr>
<td>TPR</td>
<td>0.96</td>
<td>0.75</td>
</tr>
<tr>
<td>FPR</td>
<td>0.24</td>
<td>0.22</td>
</tr>
<tr>
<td>FDR</td>
<td>0.19</td>
<td>0.25</td>
</tr>
</tbody>
</table>

Table 4: 50 × 50 image segmentation accuracy of node (voxel) selection 2.

NET-STD are 0.70 and 0.72, respectively and those of STD-DPM and STD-EM are 0.51 and 0.07, respectively. The results indicate that both node selection and sub-network discovery using the Ising prior are more accurate than those without using the Ising prior. The performance of the NET-STD is slightly improved, however, a longer computational time is required for this improvement. Therefore, if the NET-Fast achieves a satisfactory result under certain requirements, the NET-Fast is recommended compared to the NET-STD.

4.3. Summary accuracy

We implement function `SummaryAccuracy` in BANFF to compute the accuracy of node or sub-network selection.


In summary, this function provides three major options for computing the accuracy:

- **Type.Accuracy**: type of selection accuracy:
  - When `Type.Accuracy="Node"`, node selection accuracy is computed.
  - When `Type.Accuracy="Sub-network"`, sub-network selection accuracy is computed.

- **Type.Net.Accuracy**: method to compute selection accuracy:
  - When `Type.Net.Accuracy="Marginal"`, the selection accuracy is computed using the marginal distribution of selection indicators.
When `Type.Net.Accuracy="Sample"`, the sample-specific selection accuracy is first computed for each posterior sample of selection indicators. The selection accuracy is then computed by taking average over all sample specific selection accuracies.

- **Tolerance**: percentage of voxels that are correctly selected in a true positive sub-network:
  Tolerance ranges from 0.0 to 1.0.

In simulation studies, we use the following R code to obtain the sub-network selection accuracy.

```r
R> SummaryAccuracy(Trace,No.Sets,Type.Accuracy=c("Sub-network"),
+ TruePositive.Net,FalsePositive.Net,
+ Type.Net.Accuracy=c("Iteration"),Tolerance=0.90)
```

### 4.4. Application 1: Brain Image Segmentation

Alzheimer’s disease (AD) is a neurodegenerative disorder characterized by progressive impairment of memory and other cognitive functions. An intermediate stage between the expected cognitive decline of normal aging and the AD is referred as mild cognitive impairment (MCI). One important question of interests to select the neuroimaging biomarkers in order to measure the progression from MCI to AD. To address this question, we applied the BANFF to analyze a longitudinal Positron emission tomography (PET) image data set in the Alzheimer’s disease neuroimaging initiative (ADNI) study. The data set we analyzed consists of 51 AD and 121 MCI patients at baseline and 12 months. The data has been pre-processed and registered into a standard template consisting of 185,405 voxels in 90 automated anatomical labeling (AAL) regions (Tzourio-Mazoyer, Landeau, Papathanassiou, Crivello, Etard, Delcroix, Mazoyer, and Joliot 2002). We obtained the large scale testing statistical image by fitting logistic regression on the disease status (AD versus MCI) using the PET image intensity in each voxel at each time points. We used BANFF to identify co-activation regions that are strongly associated with the risk of progression from MCI and AD.

The code for implementing BANFF for Brain Image Segmentation is summarized below:

```r
R> library("BANFF")
R> net<- Grid.Adjmatrix.Transfer(grid, euclidean.dist=1)
R> total=Networks.Fast(pvalue,net,iter=5000,algorithms="EM"
+ piall=c(0.8, 0.85, 0.9, 0.95),rhoall=c(0.5,1,5,10,15))
```

Here, we also provide a function `Grid.Adjmatrix.Transfer()` which can fast transfer information from coordinate system into adjacency matrix in BANFF, where `grid` is the argument for information from coordinate system and `euclidean.dist` is the maximum node distance to be considered as connected.

Figure 8 shows the top five important regions identified by BANFF.
4.5. Application 2: Gene Network Feature Selection

To demonstrate the utility of BANFF on high-throughput biological data, we applied the method on a human breast cancer dataset. The network we used was a protein-protein interaction (PPI) network obtained from the HINT database. The HINT database is manually curated from several databases and the connections are filtered both systematically and manually to remove low-quality/erroneous interactions. The network contained 8,292 proteins and 27,493 high-quality binary interactions.

The gene expression data we used was the GSE18864 dataset obtained from the Gene Expression Omnibus (GEO) database. The dataset contains the expression profiles of two cohorts of breast cancer patients previously reported by (Li, Zou, Li, Haibe-Kains, Tian, Li, Desmedt, Sotiriou, Szallasi, Iglehart et al. 2010). It contains the gene expression profiles of 24 sporadic triple negative breast cancer (TNBC) samples and 51 primary breast tumor samples. TNBC is characterized by the lack of expression of estrogen receptor (ESR1 and ESR2) and the human epidermal growth factor receptor 2 (ERBB2, or HER2) (Gluz, Liedtke, Gottschalk, Pusztai, Nitz, and Harbeck 2009). Therefore, we selected sub-networks centering on each of those genes within 2 steps. The purpose of the analysis is to compare TNBC with primary breast tumors.

The code for implementing BANFF for Gene Network Feature Selection is summarized below:

```r
R> library("BANFF")
R> results=Networks.Fast(pvalue,net,iter=500,nburns=0,algorithms="EM", + piall=c(0.8, 0.85, 0.9, 0.95),rhoall=c(1, 2, 5, 10, 15))
R> mynet=Subnetwork.Select(net,trace,center,infinite=FALSE,steps=2)
R> plot(g=graph.adjacency(mynet$adj,mode="undirected"),vertex.size=1)
```
where \texttt{results} returns the inference result by Fast Algorithm, \texttt{mynet} returns the adjacency matrix of selected the sub-network. And the sub-network is plotted implemented by R package \texttt{igraph}.

Here, we provide function \texttt{Subnetwork.Select()} for selecting sub-networks centering on one node within certain steps, where \texttt{center} is an argument for providing the centered node and \texttt{steps} is for the given steps. We can also set \texttt{infinite=TRUE} if we expand the centered node by an infinite pattern.

We also summarized the genes involved in the sub-network centered on estrogen receptor (ESR1 and ESR2) within 2 steps, including their Gene IDs, Symbols and degree scores in Table 5. And the plot illustrating the selected sub-network is in Figure 9.

![Gene sub-network centered on estrogen receptor (ESR1 and ESR2) including neighborhood genes within two-steps](image-url)

**Figure 9:** Gene sub-network centered on estrogen receptor (ESR1 and ESR2) including neighborhood genes within two-steps
<table>
<thead>
<tr>
<th>Gene ID</th>
<th>Symbol</th>
<th>Degree</th>
</tr>
</thead>
<tbody>
<tr>
<td>4637</td>
<td>MYL6</td>
<td>4</td>
</tr>
<tr>
<td>10576</td>
<td>CCT2</td>
<td>2</td>
</tr>
<tr>
<td>57418</td>
<td>WDR18</td>
<td>6</td>
</tr>
<tr>
<td>9612</td>
<td>NCOR2</td>
<td>8</td>
</tr>
<tr>
<td>6197</td>
<td>RPS6KA3</td>
<td>2</td>
</tr>
<tr>
<td>11331</td>
<td>PHB2</td>
<td>2</td>
</tr>
<tr>
<td>7323</td>
<td>UBE2D3</td>
<td>4</td>
</tr>
<tr>
<td>3014</td>
<td>H2AFX</td>
<td>6</td>
</tr>
<tr>
<td>6188</td>
<td>RPS3</td>
<td>2</td>
</tr>
<tr>
<td>9343</td>
<td>EFTUD2</td>
<td>2</td>
</tr>
<tr>
<td>27072</td>
<td>VPS41</td>
<td>2</td>
</tr>
<tr>
<td>8161</td>
<td>COIL</td>
<td>14</td>
</tr>
<tr>
<td>55379</td>
<td>LRRC59</td>
<td>2</td>
</tr>
<tr>
<td>5814</td>
<td>PURB</td>
<td>2</td>
</tr>
<tr>
<td>51474</td>
<td>LIMA1</td>
<td>2</td>
</tr>
<tr>
<td>55729</td>
<td>ATF7IP</td>
<td>6</td>
</tr>
<tr>
<td>54512</td>
<td>EXOSC4</td>
<td>4</td>
</tr>
<tr>
<td>91748</td>
<td>ELMSAN1</td>
<td>4</td>
</tr>
<tr>
<td>4625</td>
<td>MYH7</td>
<td>2</td>
</tr>
<tr>
<td>57473</td>
<td>ZNF512B</td>
<td>18</td>
</tr>
<tr>
<td>2959</td>
<td>GTF2B</td>
<td>2</td>
</tr>
<tr>
<td>80829</td>
<td>ZFP91</td>
<td>2</td>
</tr>
<tr>
<td>9883</td>
<td>POM121</td>
<td>10</td>
</tr>
<tr>
<td>1915</td>
<td>EEF1A1</td>
<td>16</td>
</tr>
<tr>
<td>6139</td>
<td>RPS3</td>
<td>2</td>
</tr>
<tr>
<td>6217</td>
<td>RPS16</td>
<td>2</td>
</tr>
<tr>
<td>2151</td>
<td>F2RL2</td>
<td>2</td>
</tr>
<tr>
<td>2368</td>
<td>PDCD2</td>
<td>2</td>
</tr>
<tr>
<td>2595</td>
<td>PTEN2C2</td>
<td>2</td>
</tr>
<tr>
<td>4193</td>
<td>MDM2</td>
<td>4</td>
</tr>
<tr>
<td>5594</td>
<td>MAPK1</td>
<td>16</td>
</tr>
<tr>
<td>6097</td>
<td>SMARCA4</td>
<td>6</td>
</tr>
<tr>
<td>2100</td>
<td>ERBB2</td>
<td>20</td>
</tr>
<tr>
<td>4622</td>
<td>PDCD2</td>
<td>2</td>
</tr>
<tr>
<td>85686</td>
<td>MYLK2</td>
<td>2</td>
</tr>
<tr>
<td>65266</td>
<td>WNK4</td>
<td>2</td>
</tr>
<tr>
<td>7083</td>
<td>TK1</td>
<td>10</td>
</tr>
<tr>
<td>7178</td>
<td>TGFAP</td>
<td>2</td>
</tr>
<tr>
<td>7276</td>
<td>TTR</td>
<td>10</td>
</tr>
<tr>
<td>7332</td>
<td>UBED03</td>
<td>4</td>
</tr>
<tr>
<td>7965</td>
<td>AMPK</td>
<td>4</td>
</tr>
<tr>
<td>8161</td>
<td>COIL</td>
<td>14</td>
</tr>
<tr>
<td>8204</td>
<td>NRP1</td>
<td>8</td>
</tr>
<tr>
<td>8241</td>
<td>RBM10</td>
<td>4</td>
</tr>
<tr>
<td>8792</td>
<td>TRAF3P</td>
<td>2</td>
</tr>
<tr>
<td>8431</td>
<td>NR0B2</td>
<td>8</td>
</tr>
<tr>
<td>8647</td>
<td>SMURF2</td>
<td>2</td>
</tr>
<tr>
<td>8568</td>
<td>RR1P</td>
<td>4</td>
</tr>
<tr>
<td>8648</td>
<td>NCOA2</td>
<td>14</td>
</tr>
<tr>
<td>8849</td>
<td>NCOA7</td>
<td>2</td>
</tr>
<tr>
<td>9774</td>
<td>BCLAF1</td>
<td>2</td>
</tr>
<tr>
<td>9817</td>
<td>RBM13L2</td>
<td>2</td>
</tr>
<tr>
<td>9787</td>
<td>ZNF512A</td>
<td>2</td>
</tr>
<tr>
<td>51490</td>
<td>Cbfb</td>
<td>4</td>
</tr>
<tr>
<td>56829</td>
<td>ZSBHAV1</td>
<td>2</td>
</tr>
<tr>
<td>51503</td>
<td>CWC15</td>
<td>2</td>
</tr>
<tr>
<td>7144</td>
<td>SRC</td>
<td>4</td>
</tr>
<tr>
<td>8241</td>
<td>RBM10</td>
<td>4</td>
</tr>
<tr>
<td>8792</td>
<td>TRAF3P</td>
<td>2</td>
</tr>
<tr>
<td>8431</td>
<td>NR0B2</td>
<td>8</td>
</tr>
<tr>
<td>8647</td>
<td>SMURF2</td>
<td>2</td>
</tr>
<tr>
<td>8568</td>
<td>RR1P</td>
<td>4</td>
</tr>
<tr>
<td>8648</td>
<td>NCOA2</td>
<td>14</td>
</tr>
<tr>
<td>8849</td>
<td>NCOA7</td>
<td>2</td>
</tr>
<tr>
<td>9787</td>
<td>BCLAF1</td>
<td>2</td>
</tr>
<tr>
<td>9817</td>
<td>RBM13L2</td>
<td>2</td>
</tr>
<tr>
<td>9787</td>
<td>ZNF512A</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 5: Summary of the sub-network centered on estrogen receptor (ESR1 and ESR2) within 2 steps.
5. Conclusion and Extension

In this article, we introduced an R package BANFF for BAyes Network Feature Finder. We provided detailed discussions on models, algorithms, implementations and applications in gene network and brain image segmentation. Specifically, we presented a Bayesian non-parametric mixture model (Zhao et al. 2014) and discussed its generalization for the network feature selection for other applications. In addition to the standard algorithm implementing the fully Bayesian inference (NET-STD), we also provided the implementations of two fast algorithms based on FGM approximation through DPM model fitting (NET-DPM) or EM algorithm (NET-EM) respectively, leading to four main functions in BANFF. We illustrated how to use this package via simulation studies, which also demonstrate the network feature selection accuracy is substantially improved by using BANFF compared with other existing methods. In the future, other supportive functions are being created for multiple demands such as simulation studies and network visualization.

This package can be of interest to other applications with data including testing statistics or p-values along with their network information. For example, BANFF can identify “important” social sub-network when proving testing statistics of each person and their known social network information.

In this current version, package BANFF includes BAyesian Network Feature Finder mainly based on Bayesian nonparametric mixture model (Zhao et al. 2014). More Bayesian inference methods for network feature finding could be considered in order to extend the package.

6. Acknowledgement

Jian Kang’s research was partially supported by the National Center for Advancing Translational Sciences of the National Institutes of Health under Award Number UL1TR000454 and NIH grant 1R01MH105561. Tianwei Yu’s research was partially supported by NIH grant P20HL113451 and U19AI090023.

References


Affiliation:
Jian Kang
Department of Biostatistics and Bioinformatics
Rollins School of Public Health Emory University
1518 Clifton Road. Atlanta, Georgia 30322. USA
E-mail: jian.kang@emory.edu
URL: http://www.sph.emory.edu/faculty/JKANG30

Tianwei Yu
Department of Biostatistics and Bioinformatics
Rollins School of Public Health Emory University
1518 Clifton Road. Atlanta, Georgia 30322. USA
E-mail: tianwei.yu@emory.edu
URL: http://www.sph.emory.edu/faculty/TYU8