BANFF: An R Package for BAyesian Network Feature Finder

Zhou Lan
North Carolina State University

Yize Zhao
Weill Cornell Medical College

Jian Kang
University of Michigan

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 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 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.

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, computational 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 $\mathbf{r}$ or p-values; and the adjacency matrix $\mathbf{C}$ for network configurations are also assumed to be known. The latent feature selection indicator $\mathbf{z}$ is of our primary interest.

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

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 μ_i and variance σ_i^2, denoted $\mathrm{N}(\mu_i, \sigma_i^2)$. The distribution of parameters (μ_i, σ_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 τ_k. Also we specify $G_{0k} = \mathrm{N}(\gamma_k, \epsilon_k^2) \times \mathrm{IG}(\alpha_k, \beta_k)$, where $\mathrm{IG}(\alpha_k, \beta_k)$ denotes an inverse-Gamma distribution with shape α_k and rate β_k. The DPM model for feature selection is given by

$$\begin{aligned} [r_i|\mu_i, \sigma_i^2\ ] &\sim \mathrm{N}(\mu_i, \sigma_i^2), \\ [(\mu_i, \sigma_i^2)|z_i = k, G_k\ ] &\sim G_k, \\ G_k &\sim \mathcal{DP}(G_{0k}, \tau_k), \\ G_{0k} &= \mathrm{N}(\gamma_k, \xi_k^2) \times \mathrm{IG}(\alpha_k, \beta_k), \end{aligned} \tag{1}$$

for $i = 1, 2, ..., n$ and $k = 0, 1$. A weighted Ising prior is assigned to $\mathbf{z}$ to incorporate the

network information. The probability mass function is given by

$$\pi(\mathbf{z}|\boldsymbol{\pi},\boldsymbol{\rho},\boldsymbol{\omega},\boldsymbol{C}) \propto \exp\left[\sum_{i=1}^{n}(\widetilde{\omega}_i \log(\pi_{z_i} + \rho_{z_i}\sum_{i\neq j}\omega_j C_{ij} I[z_i = z_j])\right], \tag{2}$$

where parameter $\boldsymbol{\pi} = (\pi_0, \pi_1)$ with $0 < \pi_1 = 1 - \pi_0 < 1$ controls the sparsity of $\mathbf{z}$. Parameter $\boldsymbol{\rho} = (\rho_0, \rho_1)$ with $\rho_k > 0$ for k=0,1 characterizes the smoothness of $\mathbf{z}$ over the network. The pre-determined weights $\boldsymbol{\omega} = (\omega_1, \omega_2, \ldots, \omega_n)'$ is used to incorporate *a priori* biological knowledge to each node. And $\widetilde{\omega}_i = \sum_{j=1}^{n} c_{ij}\omega_j / \widetilde{\omega}_i c_{ij}$ is introduced to balance the contribution from $\boldsymbol{\pi}$ and $\boldsymbol{\rho}$ on the prior probability of $\mathbf{z}$. We refer to Zhao *et al.* (2014) for more details on the model constraints and model representations.

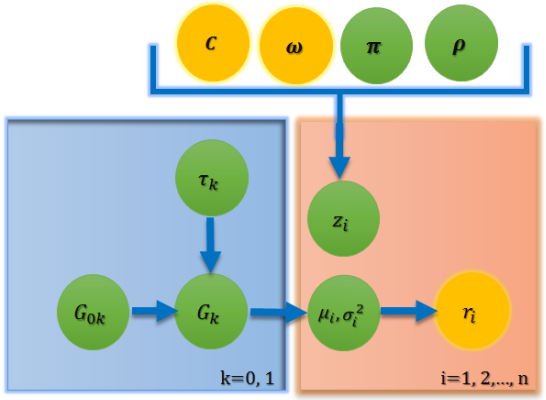

Figure 1: Graphical illustration of the hierarchical model structure. The yellow balls represent input data; the green balls represent parameters. The red box collects the feature specific variables/testing statistics that are all indexed by i, for $i = 1, \ldots, n$. The blue box collects parameters to characterize the "important" ($k = 1$) and "unimportant" ($k = 0$) features. The arrows indicate the determination relationship between these variables.

2.3. Hyper-parameter specifications

BANFF also implements specifications of hyper-parameters $\boldsymbol{\pi}$, $\boldsymbol{\rho}$ and $\boldsymbol{\omega}$ in the weighted Ising priors. Two methods have been suggested for selecting $\boldsymbol{\pi}$ and $\boldsymbol{\rho}$: (a) assigning hyper-priors on $\boldsymbol{\pi}$ and $\boldsymbol{\rho}$ to make inference; (b) employing the Bayesian model averaging for a set of possible choices of $\boldsymbol{\pi}$ and $\boldsymbol{\rho}$. In order to make this step more efficient and applicable, we modify method (b) for hyper-parameter selection in the package **BANFF**. Specifically, denote by $\{(\boldsymbol{\pi}_m, \boldsymbol{\rho}_m)\}_{m=1}^{M}$ all possible choices of hyper-parameters. Given each $(\boldsymbol{\pi}_m, \boldsymbol{\rho}_m)$, 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 $(\boldsymbol{\pi}_m, \boldsymbol{\rho}_m)$ 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 an FGM model:

$$\pi(r) = \sum_{k=0}^{1} p_k f_k(r) = \sum_{g=-L_0+1}^{L_1} \frac{\widetilde{q}_g}{\widetilde{\sigma}_g} \phi(\frac{r-\widetilde{\mu}_g}{\widetilde{\sigma}_g}),$$

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 $\mathbf{r}$ by a DPM model fitting. Denote by $\wp = \{(\hat{\mu}_g, \hat{\sigma}_g^2, \hat{p}_g)\}_{g=1}^{L_0+L_1}$ parameters of the FGM models that are generated by a DPM model fitting. We rank these densities by means $\hat{\mu}_0 < \hat{\mu}_1 < \ldots < \hat{\mu}_{L_0+L_1}$. 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:

$m \leftarrow 0$
$s_l^{(0)} \leftarrow \{l\}$, for $l = 1, 2, ..., L_0 + L_1$
while $L_0 + L_1 - m >= 2$ **do**
    for $l^{(m)} \leftarrow 1$ ***to*** $L_0 + L_1 - m - 1$ **do**
        $l_i^{(m)} \leftarrow d(\widetilde{\phi}(.; s_i^{(m)}, \wp), \widetilde{\phi}(.; s_{i+1}^{(m)}, \wp))$
    end for
    $l_{min}^{(m)} \leftarrow min(l_1^{(m)}, l_2^m, ..., l_{L_0+L_1-m-1}^{(m)})$
    if there are more than one $l_{min}^{(m)}$ **then**
        $l_{min}^{(m)} \leftarrow$ sample one among all $l_{min}^{(m)}$s
    end if
    for $l^{(m)} \leftarrow 1$ ***to*** $L_0 + L_1 - m - 1$ **do**
        if $l < l^{(m)}$ **then**
            $s_l^{(m+1)} \leftarrow s_l^{(m)}$
        end if
        if $l = l^{(m)}$ **then**
            $s_l^{(m)} \cup s_{l+1}^{(m)}$
        end if
        if $l > l^{(m)}$ **then**

19: $s_l^{(m+1)} \leftarrow s_{l+1}^{(m)}$
20: **end if**
21: **end for**
22: $m \leftarrow m + 1$
23: **end while**
24: **return** $\{s_l^{(m)}\}_{l=1}^{L_0+L_1-m}$ for $m = 1, 2, ..., L_0 + L_1 - 2$

After $m = L_0 + L_1 - 2$ steps, we obtain two sets of Gaussian densities which provide an approximation to $f_k(r) = \sum_{g \in \boldsymbol{a}_k} \frac{q_g}{\widetilde{\sigma}_g} \phi(\frac{r-\widetilde{\mu}_g}{\widetilde{\sigma}_g})$.

Two Fast algorithms

DPM model fitting: By DPM model fitting, we obtain V posterior samples of the parameters of the marginal density of $\mathbf{r}$, $\wp_v = \{(\hat{\mu}_{vg}, \hat{\sigma}^2_{vg}, \hat{p}_{vg})\}_{g=1}^{L_{v0}+L_{v1}}$ for $v = 1, 2, ...V$ and for each $\wp_v$, we applied the HODC algorithm to obtain two sets of mixture Gaussian densities, denoted $\boldsymbol{a}_{v,0}$ and $\boldsymbol{a}_{v,1}$. Thus, the marginal posterior distribution of $\mathbf{z}$ is approximated by

$$\frac{1}{V}\sum_{v=1}^{V} \pi(\boldsymbol{z}|\boldsymbol{r}, \{\widetilde{\phi}(r; \boldsymbol{a}_{v,0}, \wp_v), \widetilde{\phi}(r; \boldsymbol{a}_{v,1}, \wp_v)\}),$$

where for each v, the full conditional $\pi(\boldsymbol{z}|\boldsymbol{r}, \{\widetilde{\phi}(r; \boldsymbol{a}_{v,0}, \wp_v), \widetilde{\phi}(r; \boldsymbol{a}_{v,1}, \wp_v)\})$ can be simulated by Gibbs sampling. The **BANFF** implements this procedure in `DPM.HODC`. The **BANFF** also implements parallel computing: When V posterior samples of the parameters of the marginal density of $\mathbf{r}$ 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 $\boldsymbol{z}$ for each iteration as final output.

EM algorithm: The parameter estimates $\wp = \{(\hat{\mu}_g, \hat{\sigma}^2_g, \hat{p}_g)\}_{g=1}^{L_0+L_1}$ 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

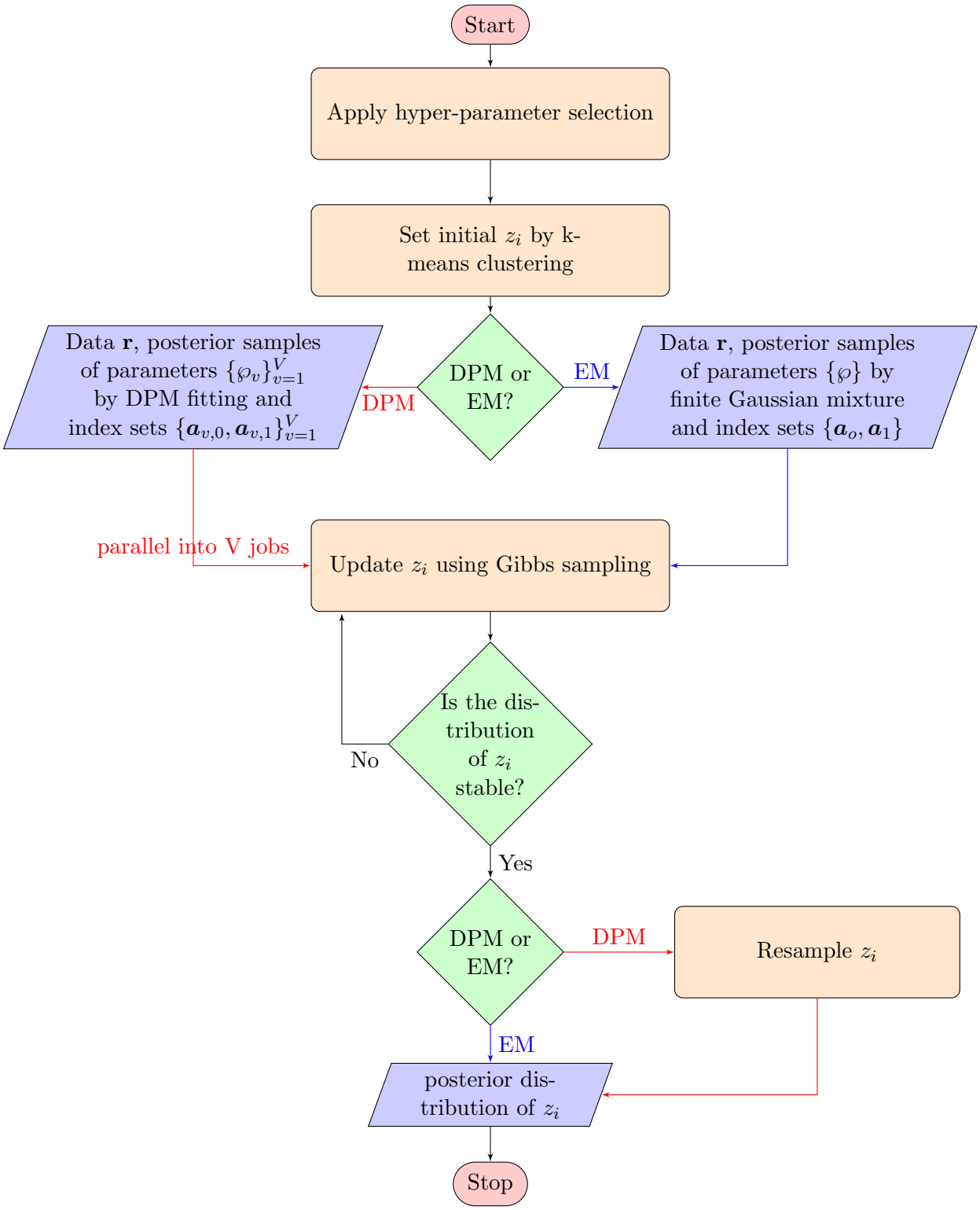

Figure 2: Fast posterior approximation algorithms

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, DPparallel=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)

DPM.HODC(v, pvalue, 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))
```

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 $\mathbf{C}$ indicating the networking configuration, where $c_{ij} = 1$ if features i

and j are connected over the network, $c_{ij} = 0$, otherwise. Of note, $\mathbf{C}$ 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 γ_k, ξ_k, β_k, τ_k and α_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 $\boldsymbol{\pi} = (\pi_0, \pi_1)$ with $0 < \pi_1 = 1 - \pi_0 < 1$ which controls the sparsity of $\mathbf{z}$, the parameter $\boldsymbol{\rho} = (\rho_0, \rho_1)$ with $\rho_k > 0$ for $k = 0, 1$ characterizes the smoothness of $\mathbf{z}$ over the network. The functions take arguments `piall` and `rhoall`: collections of possible choices on π_0 and ρ_k for $k = 0, 1$ where we assume that ρ_0 and ρ_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 $\{\wp_v\}_{v=1}^V$. 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 $\{\boldsymbol{a}_{v,0}, \boldsymbol{a}_{v,1}\}_{v=1}^V$ 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 $\{\boldsymbol{a}_0, \boldsymbol{a}_1\}$ 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 ($\mathbf{a}_0$) and important clusters ($\mathbf{a}_1$). The inputs of HODC algorithm (the parameter estimates $\wp$ 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 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 Mean Halfwidth
test
var1 passed 91.1 2.86

ii.  Hyper-Parameter Selection:
pi0= 0.8 rho0= 0.5 rho1= 1

iii.  Classification Table:
classification
0 1
74 26
```

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. π_0 and $\boldsymbol{\rho}$. 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.

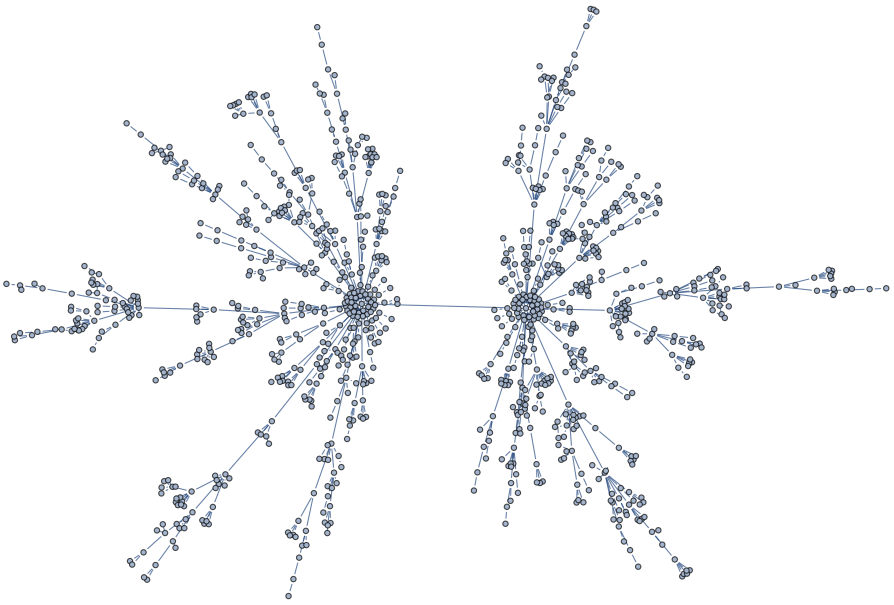

Figure 3: Simulated scale-free network of 1,000 Genes

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 π_0 are (0.8, 0.85, 0.9, 0.95) and these for ρ_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.

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

	NET-EM-Fast	STD-EM
TPR	0.925	0.573
FPR	0.016	0.014
FDR	0.035	0.016

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

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.

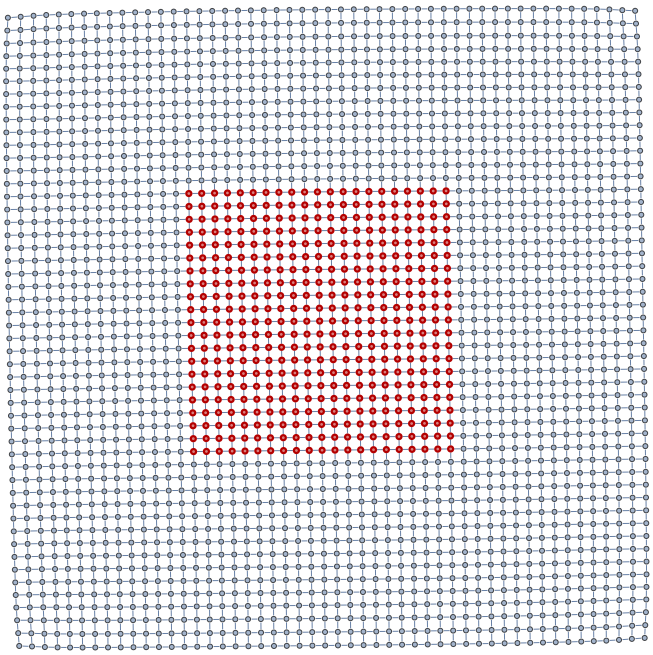

Figure 4: Network of a 50×50 image, where the red nodes represent "important" ones

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 adja-

cency 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" voxels from $\mathrm{N}(0.5, 0.2)$ and $\mathrm{N}(0, 0.2)$, 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.

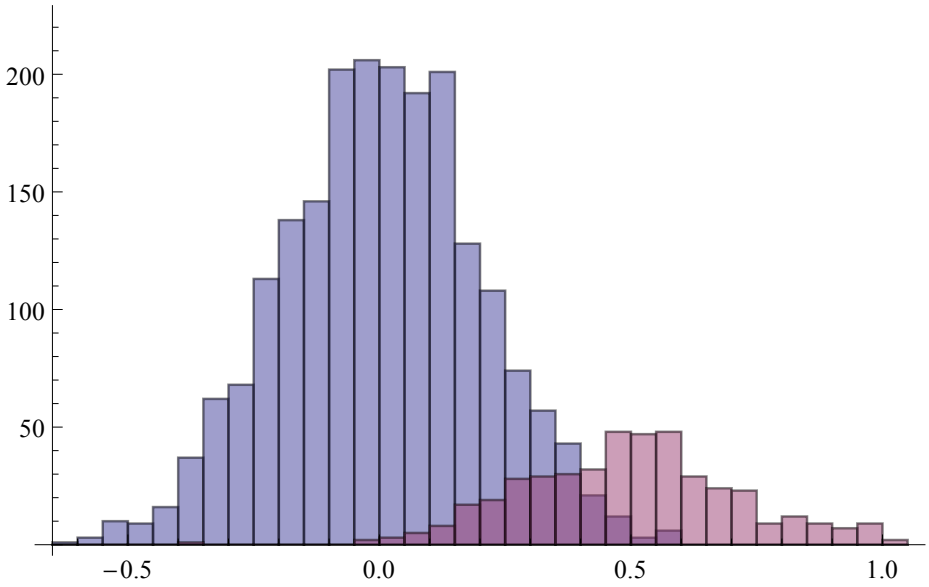

Figure 5: The histogram of input test statistics of typical 50×50 image data in image segmentation 1. The pinks bars represent "important" and blue ones represents "unimportant"

The possible choices of π_0 are (0.8, 0.85, 0.9, 0.95) and these for ρ_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

```
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))
```

where `total0`, `total1` and `total2` return the object obtained by NET-STD, NET-DPM-Fast and NET-EM-Fast, respectively.

	STD-DPM	STD-EM	NET-DPM-Fast	NET-EM-Fast	NET-STD
TPR	0.64	0.73	0.94	0.99	0.98
FPR	0.24	0.03	0.02	0.01	0.01
FDR	0.27	0.09	0.03	0.01	<0.01
Time (hrs)	0.20	0.10	1.00	0.40	6

Table 3: 50×50 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 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".

Figure 6: Output of `Plot.Subnetwork()` for a 50×50 image network. The left panel is the true "important" sub-network (region) and the right panel is a selected sub-network (region). The red area represents "important" nodes (voxels)

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 $\mathrm{N}(0,1)$. We consider two scenarios of "important" voxels:

$$\begin{aligned} \text{Mixture of Gaussian Distributions:} \quad & 0.4\mathrm{N}(3,1) + 0.6\mathrm{N}(2,0.5), \\ \text{Mixture of Non-Gaussian Distributions:} \quad & 0.4\mathrm{G}(5,2) + 0.6\mathrm{G}(6,3) \end{aligned}$$

where $\mathrm{G}(\alpha, \beta)$ denotes a gamma distribution with shape parameter α and rate parameter β. 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.

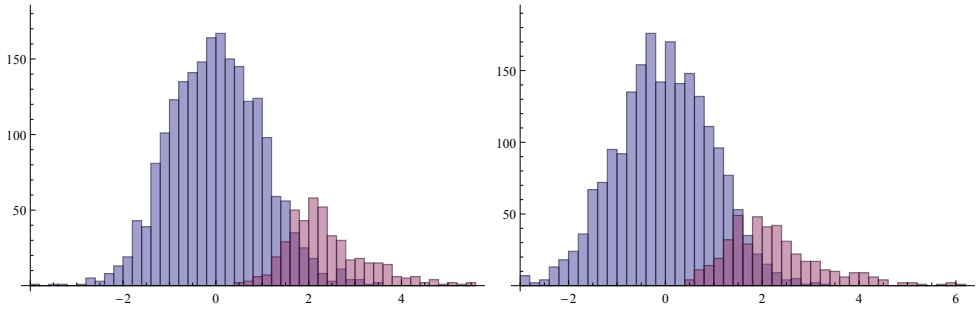

Figure 7: Histogram of simulated test statistics of a 50×50 image from different mixture distributions. The left is the Gaussian data, and the right is the non-Gaussian data. The pink bars represent "important" voxels and blue ones represent "unimportant" voxels

	Gaussian data				Non-Gaussian data			
	STD-EM	STD-DPM	NET-EM-Fast	NET-STD	STD-EM	STD-DPM	NET-EM-Fast	NET-STD
TPR	0.96	0.75	0.99	0.98	0.62	0.72	0.94	0.94
FPR	0.24	0.22	0.01	0.03	0.08	0.20	<0.01	0.04
FDR	0.19	0.25	0.01	0.01	0.17	0.23	0.02	0.03

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

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 0.70 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 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.

```
SummaryAccuracy(Trace,No.Sets,Type.Accuracy=c("Node","Sub-network"),True.Node,
TruePositive.Net,FalsePositive.Net,
Type.Net.Accuracy=c("Marginal","Sample"),Tolerance)
```

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> 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

Alzhelmer'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> 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))
```

where `net` returns the adjacency matrix of the network of Brain Image and total returns the Fast algorithm inference result.

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.

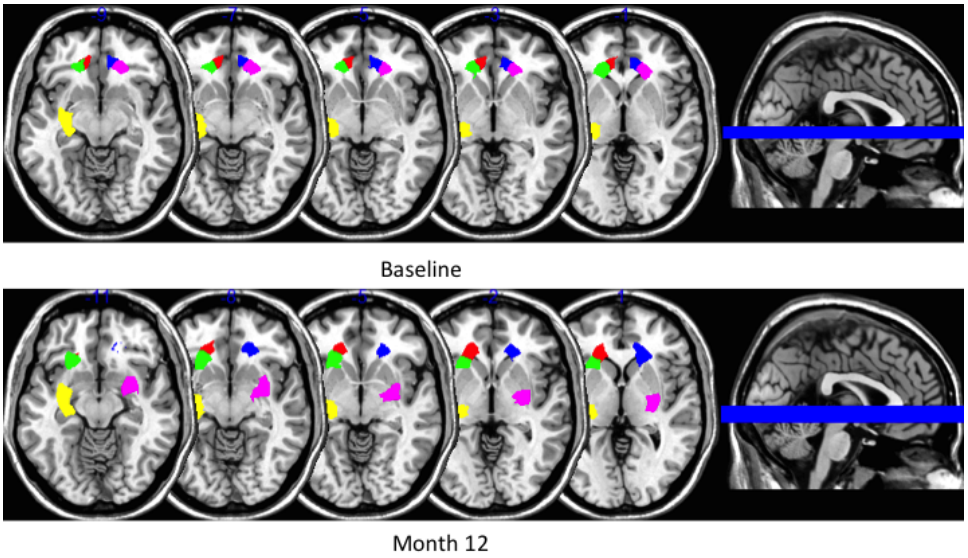

Figure 8: Top five important regions identified by BANFF at different time points. This indicates that there is an increased functional co-activation between the regions in frontal lobe and the middle temporal gyrus; and the co-activation pattern has changed over time

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> 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 `results` returns the inference result by Fast Algorithm ,`mynet` returns the adjacency matrix of selected the sub-network. And the sub-network is plotted implemented by R package **igraph**.

Here, we provide function `Subnetwork.Select()` for selecting sub-networks centering on one node within certain steps, where `center` is an argument for providing the centered node and `steps` is for the given steps. We can also set `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.

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 nonparametric 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.

Gene ID	Symbol	Degree	Gene ID	Symbol	Degree	Gene ID	Symbol	Degree	Gene ID	Symbol	Degree
2	A2M	16	1912	PHC2	8	5813	PURA	2	10528	NOP56	2
25	ABL1	38	1915	EEF1A1	16	5814	PURB	2	10529	NEBL	2
27	ABL2	14	1951	CELSR3	6	5981	RFC1	6	10576	CCT2	2
58	ACTA1	8	1956	EGFR	30	5982	RFC2	2	10891	PPARGC1A	6
60	ACTB	44	2017	CTTN	6	5984	RFC4	2	10957	PNRC1	6
71	ACTG1	14	2033	EP300	14	5992	RFX4	2	11034	DSTN	4
81	ACTN4	2	2048	EPHB2	4	6129	RPL7	4	11051	NUDT21	4
87	ACTN1	8	2064	ERBB2	20	6130	RPL7A	2	11064	CNTRL	2
88	ACTN2	8	2065	ERBB3	16	6139	RPL17	2	11137	PWP1	2
90	ACVR1	4	2079	ERH	2	6146	RPL22	6	11177	BAZ1A	2
163	AP2B1	10	2099	ESR1	86	6159	RPL29	4	11187	PKP3	2
164	AP1G1	2	2100	ESR2	276	6166	RPL36AL	2	11331	PHB2	2
196	AHR	4	2101	ESRRA	20	6175	RPLP0	2	23054	NCOA6	10
207	AKT1	18	2104	ESRRG	16	6188	RPS3	2	23076	RRP1B	2
310	ANXA7	14	2130	EWSR1	8	6191	RPS4X	2	23082	PPRC1	2
324	APC	14	2137	EXTL3	8	6195	RPS6KA1	6	23223	RRP12	4
326	AIRE	12	2151	F2RL2	8	6197	RPS6KA3	2	23524	SRRM2	2
334	APLP2	2	2167	FABP4	4	6203	RPS9	2	25930	PTPN23	2
356	FASLG	6	2175	FANCA	14	6205	RPS11	2	25959	KANK2	2
357	SHROOM2	4	2185	PTK2B	4	6207	RPS13	2	26168	SENP3	2
367	AR	40	2203	FBP1	6	6217	RPS16	2	26354	GNL3	2
373	TRIM23	6	2213	FCGR2B	6	6222	RPS18	2	27043	PELP1	2
468	ATF4	12	2268	FGR	8	6230	RPS25	2	27072	VPS41	2
493	ATP2B4	2	2308	FOXO1	6	6231	RPS26	2	27332	ZNF638	2
546	ATRX	2	2316	FLNA	16	6233	RPS27A	6	29079	MED4	4
547	KIF1A	6	2332	FMR1	4	6282	S100A11	2	29855	UBN1	2
563	AZGP1	2	2335	FN1	4	6407	SEMG2	2	51065	RPS27L	2
573	BAG1	6	2353	FOS	12	6597	SMARCA4	6	51474	LIMA1	2
578	BAK1	4	2494	NR5A2	16	6667	SP1	12	51490	C9orf114	4
595	CCND1	8	2495	FTH1	8	6712	SPTBN2	6	51503	CWC15	2
598	BCL2L1	6	2516	NR5A1	6	6714	SRC	76	51663	ZFR	2
610	HCN2	6	2534	FYN	48	6780	STAU1	6	54512	EXOSC4	4
658	BMPR1B	2	2547	XRCC6	6	6829	SUPT5H	4	55037	PTCD3	4
672	BRCA1	24	2597	GAPDH	12	6838	SURF6	2	55131	RBM28	2
705	BYSL	4	2617	GARS	4	7083	TK1	10	55226	NAT10	2
708	C1QBP	2	2670	GFAP	2	7178	TPT1	2	55379	LRRC59	2
773	CACNA1A	10	2690	GHR	6	7276	TTR	10	55629	PNRC2	10
831	CAST	10	2697	GJA1	4	7323	UBE2D3	4	55729	ATF7IP	6
836	CASP3	12	2934	GSN	12	7965	AIMP2	4	55922	NKRF	2
857	CAV1	6	2959	GTF2B	2	8161	COIL	14	55971	BAIAP2L1	2
867	CBL	14	2976	GTF3C2	2	8202	NCOA3	16	56164	STK31	2
976	CD97	2	3014	H2AFX	6	8204	NRIP1	18	56829	ZC3HAV1	2
998	CDC42	8	3065	HDAC1	4	8241	RBM10	4	56945	MRPS22	2
1022	CDK7	6	3188	HNRNPH2	2	8295	TRRAP	2	57418	WDR18	6
1026	CDKN1A	10	3191	HNRNPL	2	8431	NR0B2	8	57473	ZNF512B	18
1051	CEBPB	8	3312	HSPA8	10	8467	SMARCA5	4	57530	CGN	2
1072	CFL1	14	3320	HSP90AA1	10	8568	RRP1	2	57619	SHROOM3	2
1073	CFL2	6	3480	IGF1R	2	8648	NCOA1	16	57644	MYH7B	2
1107	CHD3	12	3912	LAMB1	4	8668	EIF3I	6	58513	EPS15L1	2
1147	CHUK	10	4089	SMAD4	14	8735	MYH13	2	64425	POLR1E	2
1173	AP2M1	12	4193	MDM2	8	8805	TRIM24	6	64960	MRPS15	2
1374	CPT1A	2	4288	MKI67	6	8945	BTRC	4	65266	WNK4	2
1386	ATF2	8	4298	MLLT1	2	9221	NOLC1	4	79009	DDX50	2
1387	CREBBP	22	4331	MNAT1	6	9328	GTF3C5	2	79784	MYH14	2
1400	CRMP1	12	4622	MYH4	2	9329	GTF3C4	2	79809	TTC21B	2
1432	MAPK14	16	4625	MYH7	2	9330	GTF3C3	2	80829	ZFP91	2
1434	CSE1L	8	4637	MYL6	4	9343	EFTUD2	2	81631	MAP1LC3B	4
1445	CSK	8	4641	MYO1C	2	9349	RPL23	4	81887	LAS1L	2
1460	CSNK2B	12	4642	MYO1D	2	9440	MED17	4	83939	EIF2A	2
1487	CTBP1	6	4736	RPL10A	2	9513	FXR2	14	85366	MYLK2	2
1488	CTBP2	4	4744	NEFH	2	9611	NCOR1	8	90850	ZNF598	2
1600	DAB1	6	4878	NPPA	4	9612	NCOR2	8	91748	ELMSAN1	4
1605	DAG1	6	5216	PFN1	4	9648	GCC2	2	114294	LACTB	2
1627	DBN1	24	5295	PIK3R1	42	9699	RIMS2	4	117246	FTSJ3	2
1639	DCTN1	4	5304	PIP	4	9774	BCLAF1	2	124245	ZC3H18	2
1660	DHX9	4	5335	PLCG1	32	9883	POM121	10	130507	UBR3	2
1748	DLX4	10	5394	EXOSC10	2	10073	SNUPN	2	135112	NCOA7	2
1759	DNM1	4	5433	POLR2D	4	10095	ARPC1B	2	220164	DOK6	6
1789	DNMT3B	6	5469	MED1	6	10270	AKAP8	2			
1795	DOCK3	8	5594	MAPK1	16	10432	RBM14	2			
1822	ATN1	6	5597	MAPK6	10	10454	TAB1	8			
1854	DUT	6	5705	PSMC5	8	10499	NCOA2	8			
1857	DVL3	2	5718	PSMD12	2	10514	MYBBP1A	4			

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

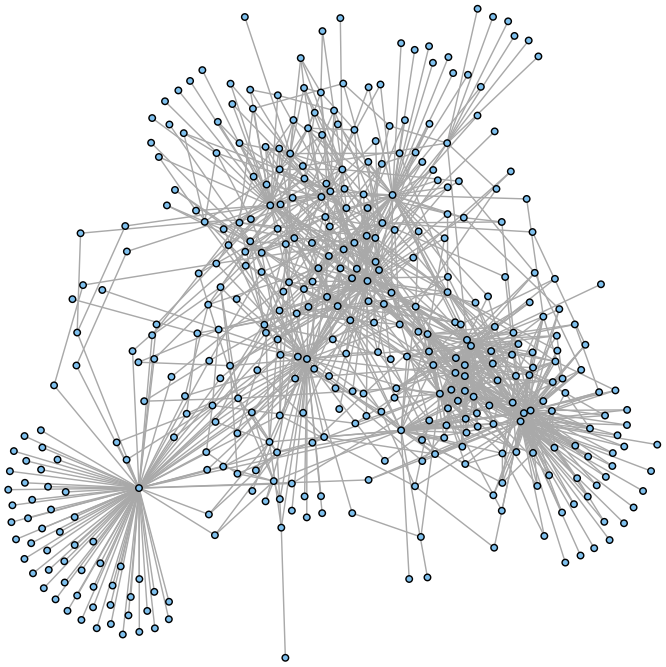

Figure 9: Gene sub-network centered on estrogen receptor (ESR1 and ESR2) including neighborhood genes within two-steps

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

Analytics R, Weston S (2014a). *doParallel: Foreach parallel adaptor for the parallel package*. R package version 1.0.8, URL http://CRAN.R-project.org/package=doParallel.

Analytics R, Weston S (2014b). *foreach: Foreach looping construct for R*. R package version 1.4.2, URL http://CRAN.R-project.org/package=foreach.

Barabási AL (2007). "Network medicine—from obesity to the "diseasome"." *New England Journal of Medicine*, **357**(4), 404–407.

Barabási AL, Gulbahce N, Loscalzo J (2011). "Network medicine: a network-based approach to human disease." *Nature Reviews Genetics*, **12**(1), 56–68.

Butts CT, Handcock MS, Hunter DR (2014). *network: Classes for Relational Data*. Irvine, CA. R package version 1.10.2, URL http://statnet.org/.

Chan SY, Loscalzo J (2012). "The emerging paradigm of network medicine in the study of human disease." *Circulation research*, **111**(3), 359–374.

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

Duarte NC, Becker SA, Jamshidi N, Thiele I, Mo ML, Vo TD, Srivas R, Palsson BØ (2007). "Global reconstruction of the human metabolic network based on genomic and bibliomic data." *Proceedings of the National Academy of Sciences*, **104**(6), 1777–1782.

Efron B (2004). "Large-scale simultaneous hypothesis testing." *Journal of the American Statistical Association*, **99**(465).

Fraley C, Raftery AE (2002). "Model-based clustering, discriminant analysis, and density estimation." *Journal of the American Statistical Association*, **97**(458), 611–631.

Fraley C, Raftery AE, Murphy TB, Scrucca L (2012). *mclust Version 4 for R: Normal Mixture Modeling for Model-Based Clustering, Classification, and Density Estimation*.

Gluz O, Liedtke C, Gottschalk N, Pusztai L, Nitz U, Harbeck N (2009). "Triple-negative breast cancer—current status and future directions." *Annals of Oncology*, p. mdp492.

Heidelberger P, Welch PD (1983). "Simulation run length control in the presence of an initial transient." *Operations Research*, **31**(6), 1109–1144.

Jackman S (2014). *pscl: Classes and Methods for R Developed in the Political Science Computational Laboratory, Stanford University*. Department of Political Science, Stanford University, Stanford, California. R package version 1.4.6, URL http://pscl.stanford.edu/.

Janes KA, Yaffe MB (2006). "Data-driven modelling of signal-transduction networks." *Nature Reviews Molecular Cell Biology*, **7**(11), 820–828.

Jara A (2007). "Applied Bayesian Non- and Semi-parametric Inference Using DPpackage." *R News*, **7**(3), 17–26. URL http://CRAN.R-project.org/doc/Rnews/.

Jara A, Hanson T, Quintana F, Müller P, Rosner G (2011). "DPpackage: Bayesian Semi- and Nonparametric Modeling in R." *Journal of Statistical Software*, **40**(5), 1–30. URL http://www.jstatsoft.org/v40/i05/.

Li C, Li H (2008). "Network-constrained regularization and variable selection for analysis of genomic data." *Bioinformatics*, **24**(9), 1175–1182.

Li Y, Zou L, Li Q, Haibe-Kains B, Tian R, Li Y, Desmedt C, Sotiriou C, Szallasi Z, Iglehart JD, *et al.* (2010). "Amplification of LAPTM4B and YWHAZ contributes to chemotherapy resistance and recurrence of breast cancer." *Nature medicine*, **16**(2), 214–218.

Licatalosi DD, Darnell RB (2010). "RNA processing and its regulation: global insights into biological networks." *Nature Reviews Genetics*, **11**(1), 75–87.

Ma S, Shi M, Li Y, Yi D, Shia BC (2010). "Incorporating gene co-expression network in identification of cancer prognosis markers." *BMC bioinformatics*, **11**(1), 271.

Pan W, Xie B, Shen X (2010). "Incorporating predictor network in penalized regression with application to microarray data." *Biometrics*, **66**(2), 474–484.

Plummer M, Best N, Cowles K, Vines K (2006). "CODA: Convergence Diagnosis and Output Analysis for MCMC." *R News*, **6**(1), 7–11. URL http://CRAN.R-project.org/doc/Rnews/.

Qu L, Nettleton D, Dekkers J (2012). "A Hierarchical Semiparametric Model for Incorporating Intergene Information for Analysis of Genomic Data." *Biometrics*, **68**(4), 1168–1177.

Reynolds D (2009). "Gaussian mixture models." *Encyclopedia of Biometrics*, pp. 659–663.

Rual JF, Venkatesan K, Hao T, Hirozane-Kishikawa T, Dricot A, Li N, Berriz GF, Gibbons FD, Dreze M, Ayivi-Guedehoussou N, *et al.* (2005). "Towards a proteome-scale map of the human protein–protein interaction network." *Nature*, **437**(7062), 1173–1178.

Stingo FC, Chen YA, Tadesse MG, Vannucci M (2011). "Incorporating biological information into linear models: A Bayesian approach to the selection of pathways and genes." *The annals of applied statistics*, **5**(3).

Tzourio-Mazoyer N, Landeau B, Papathanassiou D, Crivello F, Etard O, Delcroix N, Mazoyer B, Joliot M (2002). "Automated anatomical labeling of activations in SPM using a macroscopic anatomical parcellation of the MNI MRI single-subject brain." *Neuroimage*, **15**(1), 273–289.

Wilhelm S, G MB (2014). *tmvtnorm: Truncated Multivariate Normal and Student t Distribution*. R package version 1.4-9, URL http://CRAN.R-project.org/package=tmvtnorm.

Zhao Y, Kang J, Yu T (2014). "A Bayesian nonparametric mixture model for selecting genes and gene subnetworks." *The Annals of Applied Statistics*, **8**(2), 999–1021.

Affiliation:

Jian Kang
Department of Biostatistics
Kidney Epidemiology and Cost Center
School of Public Health
University of Michigan
1415 Washington Heights
Ann Arbor, Michigan 48109-2029
E-mail: jiankang@umich.edu
URL: http://www-personal.umich.edu/~jiankang/

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