BANS: Bayesian Node-wise Selection for Multi-layered Gaussian Graphicl Models

Min Jin Ha

June 15, 2019

1 Overview

```
> library(BANS)
```

This vignette describes how to use R/BANS to estimate multi-layered Gaussian Graphical Models using Bayesian node-wise selection.

2 BANS with an example

We illustrate the usage of R/BANS pakcage using The Cancer Genome Atlas (TCGA) mRNA expression and protein expression data for 428 clear cell renel cell carcinoma (ccRCC) patients. The multi-omic data (matched sample in row) corresponding to PI3K pathway can be loaded.

```
> data("TCGA_KIRC_CNA_mRNA_RPPA")
> dim(mRNA);dim(RPPA)
```

```
[1] 428  10

[1] 428  14
```

We make an input data by combining mRNA and protein expression datasets.

```
> Y = cbind(mRNA,RPPA)
> colnames(Y) = c(paste("mRNA_",colnames(mRNA),sep="")
+ ,paste("RPPA_",colnames(RPPA),sep=""))
> dim(Y)
```

```
[1] 428  24
```

In this way, we set the nodes from 1-10 are for genes corresponding to mRNA expression and 11-24 are for proteins corresponding to protein expression. Using this multi-platform data, we aim to construct two-layered Gaussian graphical models using the `ch.chaingraph` function. For the input parameters, we fix λ (`lambda`) and δ (`delta`) as the shape and scale of the prior on the precision matrix. We also specify the priors on γ and η in the options `eta.prob` and `gamma.prob`. By setting `burnin.S` and `inf.S`, we can choose the number of iterations of the MCMC samples for burn-in and inference. Note that in this function, we implemented MCMC sampling to estimate the posterior distributions with the symmetric constraint in the structure of zeros of the precision matrix. Therefore the function should run jointly for a layer. For the first layer that has no upstream layer (e.g. mRNA expression), the Bayesian neighborhood selection for UG is performed by setting `v.pa=NULL`. For the first layer (mRNA expression), we fit UG.

```
> fit1 = ch.chaingraph(v.ch=1:10,v.pa=NULL,Y=Y,eta.prob=0.1
+                        ,gamma.prob=0.1,lambda=5,delta=2,burnin.S=10,inf.S=20)
```

For the sencond layer (protein expression), we fit directed edges from the mRNA expression to protein expression and the residual undirected protein network structure.

```
> fit2 = ch.chaingraph(v.ch=11:24,v.pa=1:10,Y=Y,eta.prob=0.1
+                        ,gamma.prob=0.1,lambda=5,delta=2,burnin.S=10,inf.S=20)
```

The output is as follows:

```
> names(fit1)

[1] "Gamma" "eta"   "A"     "B"     "kappa"

> names(fit2)

[1] "Gamma" "eta"   "A"     "B"     "kappa"
```

`Gamma` and `B` are `|v.ch|`$\times$`|v.pa|`$\times$`inf.S` arrays for the binary indicators γ and regression coefficients b for directed edges between mRNA and protein. If it is for the first layer, those arrays are NULL. `eta` and A are `|v.ch|`$\times$`|v.ch|`$\times$`inf.S` arrays for the binary indicators η and regression coefficients α for undirected edges within layer, and `kappa` is a `inf.S`$\times$`|v.ch|` matrix for the precision parameters.

3 BANS-parallel

To make BANS node-wise parallelizable, we also implemented node-wise regression learning without the symmetric constraint of the precision matrix. The additional input parameters are lists for node numbers for layers and parent layers. `chlist` is the length q (number of layers) list with node number information for each layer. `palist` is also the length q list with node number information for each corresponding parent layers. For the first layer, `palist` is set to NULL. And `v` is the node that we want to regress on all the other variables using our BANS model. From BANS-parallel, we estimate undirected edges for the node `v` within the layer (v$\in \tau$), and directed edges between all nodes in τ and its parent layers (see the BANS model in proposition 1).

```
> v=12
> chlist = list(1:10,11:24)
> palist = list(NULL,1:10)
> vfit = v.chaingraph(v=v,chlist=chlist,palist=palist,Y=Y
+                     ,lambda=5,delta=2,burnin.S=10,inf.S=20)
> names(vfit)

[1] "Gamma" "eta"   "A"     "B"     "kappa"
```

`Gamma` and `B` objects are $|\tau| \times |\texttt{pa}_\tau| \times$ `|inf.S|` arrays for the directed edges between τ that includes the node `v` and its parent layers. `eta` and `A` are `inf.S`$\times|\tau|$ matrices for undirected edges connected to `v` within the same layer τ.

4 Posterior Inference on the Signs of Edges

To make further inference on the signs/weights of edges, we implemented structured MCMC, conditional on the predetermined multi-layered graphical structure. We first compute posterior marginal probability of edge inclusion for each edge using the results from the structural estimations for all layers.

```
> p = ncol(Y)
> B = matrix(0,p,p)
> B[chlist[[1]],chlist[[1]]] = apply(fit1$eta,c(1,2),mean)
> B[chlist[[2]],chlist[[2]]] = apply(fit2$eta,c(1,2),mean)
> B[as.matrix(expand.grid(chlist[[1]],palist[[1]]))] = apply(fit2$Gamma,c(1,2),mean)
```

Then, with cutoff of 0.5, we construct the adjacency matrix for the estimated graphical structure and perform structured MCMC as follows.

```
> G = 1*(B>0.5)
> t = 1 # first layer
> sfit1 = ch.chaingraph.str(v.ch=chlist[[t]],v.pa=palist[[t]]
+                           ,Y=Y,G=G,lambda=5,delta=2,burnin.S=10,inf.S=20)
> t = 2 # second layer
> sfit2 = ch.chaingraph.str(v.ch=chlist[[t]],v.pa=palist[[t]]
+                           ,Y=Y,G=G,lambda=5,delta=2,burnin.S=10,inf.S=20)
```

Using the structured estimation, we can build weighted/signed multi-layered networks.