`sirDAG`: Surrogate Intervention Recovery of a Directed Acyclic Graph (DAG)

Min Jin Ha

October 28, 2018

1 Overview

This vignette describes how to use `R/sirDAG` package to estimate high-dimensional directed acyclic graphs (DAGs) for gene expression with surrogate intervention data such as cis-eQTL SNP genotype data. We first estimate a skeleton from gene expression data using PenPC algorithm and then orient the edges in the skeleton using `sirDAG` algorithm. For the whole procedure we need to download three packages.

```
> library(PEN) # Installed from PEN_1.01.tar.gz
> library(PenPC) # Installed from PenPC_1.0.tar.gz
> library(sirDAG)
```

2 Example

We illustrate the usage of `sirDAG` package using simulated data under the scenario of $p = 100$, $n = 30$, ER model for the true DAG with $pE = 2/(p-1)$ and multiple eQTLs per gene (total no. of eQTLs across all genes are 200, $m = 200$).

```
> load("simul_p100n30r27pE2_ER_M.Rdata")
> names(simul)
```

```
 [1] "A"
 [2] "B"
 [3] "Y"
 [4] "X"
 [5] "G"
 [6] "Gy"
 [7] "cross"
 [8] "markers.nms"
 [9] "allqtls"
[10] "pheno.names"
```

The list `simul` includes objects:

- $p \times p$ matrix `A` for the gene-gene coefficents
- $p \times m$ matrix `B` for the SNP-gene coefficients
- $n \times p$ matrix `Y` for gene expression data
- $n \times m$ matrix `X` for SNP genotype data

- graphNEL object `G` that includes the augmented DAG for gene expression and eQTL
- graphNEL object `Gy` that includes the DAG for gene expression only
- `cross`, `markers.nms`, and `allqtls` are used for simulation studies for QDG algorithm (Neto et. al., 2008)

Before fitting DAG, we scale the gene expression and genotype data.

```
> n=30
> p=100
> datX = simul$X
> datY = simul$Y
> meandat = apply(datX,2,mean)
> normdat = sqrt(rowSums((t(datX)-meandat)^2)/n)
> datX= scale(datX,meandat,normdat)
> meandat = apply(datY,2,mean)
> normdat = sqrt(rowSums((t(datY)-meandat)^2)/n)
> datY= scale(datY,meandat,normdat)
```

2.1 Skeleton Estimation

We estimate the skeleton of gene expression using PenPC algorithm. In the first step of PenPC, neighborhood selections for all nodes are performed to uncover Markov blanket.

```
> coef = ne.PEN(dat=datY,nlambda=100,ntau=10,V=1:p,order=TRUE,verbose=FALSE)
> # Set edge weight
> edgeWeights   = matrix(apply(abs(cbind(c(coef),c(t(coef)))),1,max),ncol=p)
```

Then we perform partial correlation tests for the p-value cutoff $\alpha = 0.01$ to eliminate the false positive edges.

```
> alpha = 0.01
> indepTest = gaussCItest
> suffStat  = list(C = cor(datY), n = n)
> sFitNI  = skeletonPENstable(suffStat, indepTest, as.integer(p)
+                               , alpha,edgeWeights=edgeWeights, verbose=FALSE)

> sFitNI

Object of class 'pcAlgo', from Call:
skeletonPENstable(suffStat = suffStat, indepTest = indepTest,
    p = as.integer(p), alpha = alpha, verbose = FALSE, edgeWeights = edgeWeights)
Number of undirected edges:  36
Number of directed edges:    0
Total number of edges:       36
```

2.2 Edge orientation

We orient the undirected edges in the skeleton using `sirDAG-MAP` or `sirDAG-NG`.

```
> gSkel = sFitNI@graph
```

First, we try `sirDAG-MAP`.

```
> fit.sirDAG_MAP = directionPosterior(gInput=gSkel,B=simul$B,datX=datX,datY=datY
+                                   ,exhaustive.cut=10,no.T=2^10,verbose=F,acyclic=T
+                                   ,bf.cut1=1,bf.cut2=1,Cluster=T)
```

By default, we perform `sirDAG-MAP`. If we set `NormalGamma=T`, `sirDAG-NG` is performed with predetermined set of hypoerparameters of the prior, `nu`, `delta`, and `gg`, where `gg` is set to be the sample size n for the unit g-prior.

```
>    fit.sirDAG_NG = directionPosterior(gInput=gSkel,B=simul$B,datX=datX,datY=datY
+                                ,exhaustive.cut=10,no.T=2^10,verbose=F,acyclic=T
+                                ,bf.cut1=1,bf.cut2=1,Cluster=T
+                                ,NormalGamma=T,nu=1,delta=1,gg=n)
```

The output values includes `posterior` for posterior probabilities for each edge directions, `g` for the reuslting DAG (or PDAG), `OK` is to check whether the edge orientation is performed.

```
> names(fit.sirDAG_MAP)

[1] "posterior" "g"
[3] "OK"

> names(fit.sirDAG_NG)

[1] "posterior" "g"
[3] "OK"
```

We can disply the networks by the plot function using `igraph` package.

```
> gMAP = graph_from_graphnel(fit.sirDAG_MAP$g)
> gNG = graph_from_graphnel(fit.sirDAG_NG$g)
> par(mfrow=c(1,2),mar=c(0,0,0,0))
> l = layout_on_grid(gMAP)
> plot(gMAP,layout=l, vertex.color="green",edge.color=1,vertex.size=10
+      ,vertex.label.cex=0.7,margin=0,vertex.frame.color="green")
> plot(gNG,layout=l, vertex.color="green",edge.color=1,vertex.size=10
+      ,vertex.label.cex=0.7,margin=0,vertex.frame.color="green")
```