Usage of AMOUNTAIN

Dong Li
dxl466@cs.bham.ac.uk
School of Computer Science, The University of Birmingham, UK

Date modified: 2016-10-17

In this example we embed parts of the examples from the `AMOUNTAIN` help page into a single document. And show how to use `AMOUNTAIN` step by step.

1 Network simulation

We follow [1] to construct gene co-expression networks for simulation study. Let n be the number of genes, and edge weights W as well as node score z follow the uniform distribution in range $[0, 1]$. A module contains k genes inside which the edge weights as well as node score follow the uniform distribution in range $[\theta, 1]$, where $\theta = \{0.5, 0.6, 0.7, 0.8, 0.9\}$.

```
> library(AMOUNTAIN)
> n=100
> k=20
> theta=0.5
> pp <- networkSimulation(n, k, theta)
> moduleid <- pp[[3]]
> netid <- 1:100
> restp<- netid[-moduleid]
> groupdesign=list(moduleid,restp)
> names(groupdesign)=c('module','background')
```

Figure 1 shows the weighted co-expression network when $n = 100, k = 20$ and red nodes indicate module members and wider edges mean larger similarities. Visualization is based on `qgraph` [2].

When simulating a two-layer network, the basic method is to connect two independent networks with a inter-layer weight matrix A, which is designed to has larger weights between two modules.

```
> n1=100
> k1=20
> theta1 = 0.5
> n2=80
> k2=10
> theta2 = 0.5
> ppresult <- twolayernetworkSimulation(n1,k1,theta1,n2,k2,theta2)
> A <- ppresult[[3]]
> pp <- ppresult[[1]]
> moduleid <- pp[[3]]
> netid <- 1:n1
> restp<- netid[-moduleid]
> pp2 <- ppresult[[2]]
> moduleid2 <- pp2[[3]]
> netid2 <- 1:n2
> restp2<- netid2[-moduleid2]
> library(qgraph)
```

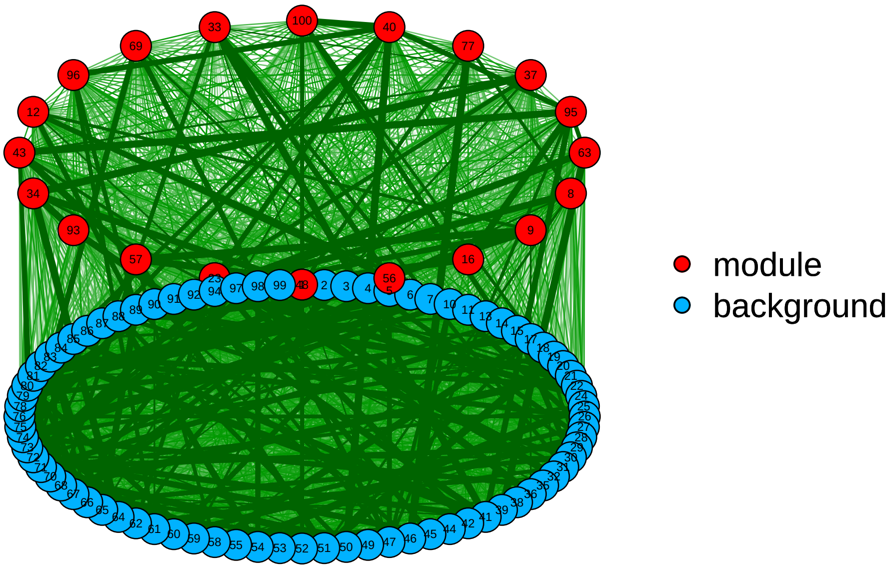

Figure 1: Simulated single layer network

```
> ## labelling the groups
> groupdesign=list(moduleid,restp,(moduleid2+n1),(restp2+n1))
> names(groupdesign)=c('module1','background1','module2',
+                      'background2')
> twolayernet<-matrix(0,nrow=(n1+n2),ncol=(n1+n2))
> twolayernet[1:n1,1:n1]<-pp[[1]]
> twolayernet[(n1+1):(n1+n2),(n1+1):(n1+n2)]<-pp2[[1]]
> twolayernet[1:n1,(n1+1):(n1+n2)] = A
> twolayernet[(n1+1):(n1+n2),1:n1] = t(A)
```

Figure 2 shows the the two-layer weighted co-expression network based on above simulation.

2 Module identification for single layer network

The algorithm for module identification requires the input With edge weights W and vertex weight $\mathbf{z}$ for weighted network G. Here We show how to use the following function in the package to find active module for above simulated single layer network. With groundtruth in hand we can evaluate the quality of identified modules by F-score [4]. In order to get higher quality, we need to tune parameter α in the elastic net palnety and $\lambda = 1$ in the objective function. The common way to select two optimal parameters is grid search.

```
> n = 100
> k = 20
> theta = 0.5
> pp <- networkSimulation(n,k,theta)
> moduleid <- pp[[3]]
> alphaset = seq(0.1,0.9,by=0.1)
> lambdaset <-  2^seq(-5,5)
> ## using a grid search to select lambda and alpha
> Fscores <- matrix(0,nrow = length(alphaset),
+                       ncol = length(lambdaset))
> for (j in 1:length(alphaset)) {
+         for (k in 1:length(lambdaset)) {
```

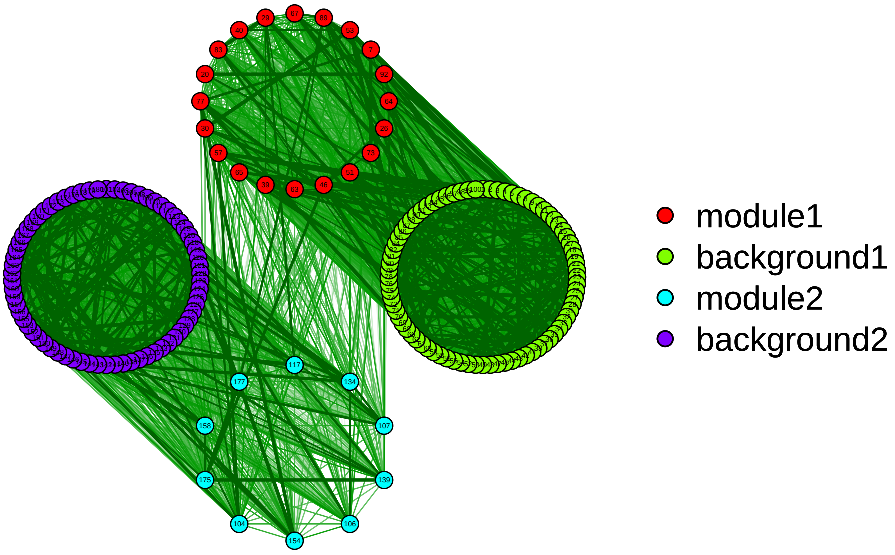

Figure 2: Simulated two-layer network

```
+                 x <- moduleIdentificationGPFixSS(pp[[1]],pp[[2]],
+                     rep(1/n,n),maxiter = 500,
+                     a=alphaset[j],lambda = lambdaset[k])
+                 predictedid<-which(x[[2]]!=0)
+             recall <- length(intersect(predictedid,moduleid))/
+                   length(moduleid)
+                 precise <- length(intersect(predictedid,moduleid))/
+                   length(predictedid)
+                 Fscores[j,k] <- 2*precise*recall/(precise+recall)
+         }
+ }
```

We can show $gridFscore$ by 3-D plot Figure 3 to see how these parameters affect the performance. By certain combination of these two parameters, we can almost exactly find the target model nodes with $F - score = 1$.

3 Module identification for two-layer network

The basic idea to identification modules on two-layer network is to find two active modules on each layer, at the same time with maximal inter-later links. we have function $moduleIdentificationGPFixSSTwolayer$ in the package. Following the two-layer network simulation in section 1, we call the method.

```
> ## network simulation is the same as before
> modres=moduleIdentificationGPFixSSTwolayer(pp[[1]],pp[[2]],
+              rep(1/n1,n1),pp2[[1]],pp2[[2]],rep(1/n2,n2),A)
> predictedid<-which(modres[[1]]!=0)
> recall = length(intersect(predictedid,moduleid))/
+     length(moduleid)
> precise = length(intersect(predictedid,moduleid))/
+     length(predictedid)
> F1 = 2*precise*recall/
+     (precise+recall)
> predictedid2<-which(modres[[2]]!=0)
```

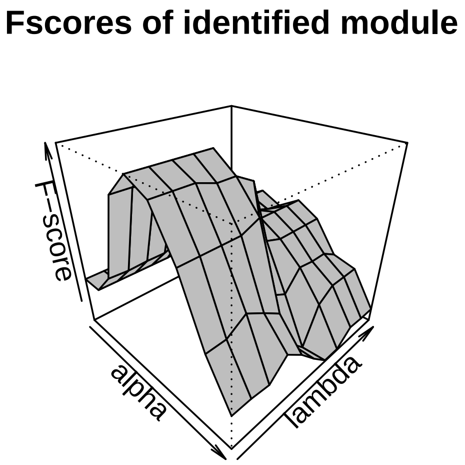

Figure 3: Simulated two-layer network

```
> recall2 = length(intersect(predictedid2,moduleid2))/
+     length(moduleid2)
> precise2 = length(intersect(predictedid2,moduleid2))/
+     length(predictedid2)
> F2 = 2*precise2*recall2/(precise2+recall2)
```

And we can also select optimal paramters combination in a more sophscated way based on the example in section 2.

4 Module identification for real-world data

The usage of the package functions is the same for real-world data, but we need to be aware about two aspects. First of all the way to calculate edges score and nodes score in weighted network can make an impact on the performance. Different input W and $\mathbf{z}$ in the objective function may lead to different modules. For instance, we may need to try several different forms of W other than using just the correlation matrix in practice. We tried the same differential analysis method of gene pairs as [3]. See [4] for more details.

Secondly we do not have groundtruth about module membership for real world data. In this case we may need to select the proper parameter so that the desired module size can be archived. When fixing $\lambda = 0.01$, we use a binary search method to select α for elastic net penalty which control the sparsity of the module.

```
> ## binary search parameter to fix module size to 100~200
> abegin = 0.01
> aend = 0.9
> maxsize = 200
> minsize = 100
> for (i in 1:100) {
+         x <- moduleIdentificationGPFixSS(W,z,rep(1/n,n),
+                 a=(abegin+aend)/2,lambda = 0.001,maxiter = 500)
+         predictedid<-which(x[[2]]!=0)
+         if(length(predictedid) > maxsize){
+                 abegin = (abegin+aend)/2
+         }else if (length(predictedid) < minsize){
+                 aend = (abegin+aend)/2
+         }else
+                   break
+ }
```

5 Biological explanation

Finally we can do gene annotation enrichment analysis with intergative tools like DAVID[1] or Enrichr[2], to see whether a module gene list can be explained by existing biological process, pathways or even diseases.

6 Session info

- R version 3.3.1 (2016-06-21), `x86_64-w64-mingw32`
- Locale: `LC_COLLATE=C`, `LC_CTYPE=English_United States.1252`, `LC_MONETARY=English_United States.1252`, `LC_NUMERIC=C`, `LC_TIME=English_United States.1252`
- Base packages: base, datasets, grDevices, graphics, methods, stats, utils
- Other packages: AMOUNTAIN 1.0.0, qgraph 1.3.5
- Loaded via a namespace (and not attached): BiocStyle 2.2.0, Formula 1.2-1, Hmisc 3.17-4, MASS 7.3-45, Matrix 1.2-7.1, RColorBrewer 1.1-2, Rcpp 0.12.7, abind 1.4-5, acepack 1.3-3.3, arm 1.9-1, boot 1.3-18, chron 2.3-47, cluster 2.0.5, coda 0.18-1, colorspace 1.2-7, corpcor 1.6.8, d3Network 0.5.2.1, data.table 1.9.6, ellipse 0.3-8, fdrtool 1.2.15, foreign 0.8-67, ggm 2.3, ggplot2 2.1.0, glasso 1.8, grid 3.3.1, gridExtra 2.2.1, gtable 0.2.0, gtools 3.5.0, huge 1.2.7, igraph 1.0.1, jpeg 0.1-8, lattice 0.20-34, latticeExtra 0.6-28, lavaan 0.5-22, lme4 1.1-12, magrittr 1.5, matrixcalc 1.0-3, mi 1.0, minqa 1.2.4, mnormt 1.5-5, munsell 0.4.3, network 1.13.0, nlme 3.1-128, nloptr 1.0.4, nnet 7.3-12, parallel 3.3.1, pbivnorm 0.6.0, plyr 1.8.4, png 0.1-7, psych 1.6.9, quadprog 1.5-5, reshape2 1.4.1, rjson 0.2.15, rpart 4.1-10, scales 0.4.0, sem 3.1-8, sna 2.4, splines 3.3.1, statnet.common 3.3.0, stats4 3.3.1, stringi 1.1.2, stringr 1.1.0, survival 2.39-5, tools 3.3.1, whisker 0.3-2

References

[1] Li W, Liu C C, Zhang T, et al. Integrative analysis of many weighted co-expression networks using tensor computation. PLoS Comput Biol, 2011, 7(6): e1001106.

[2] Epskamp S, Cramer A O J, Waldorp L J, et al. qgraph: Network visualizations of relationships in psychometric data. Journal of Statistical Software, 2012, 48(4): 1-18.

[3] Hsu, Chia-Lang, Hsueh-Fen Juan, and Hsuan-Cheng Huang. Functional Analysis and Characterization of Differential Coexpression Networks. Scientific reports 5 (2015).

[4] Dong Li et al. Active modules for multilayer weighted gene co-expression networks: a continuous optimization approach. 2016.

---

[1] https://david.ncifcrf.gov

[2] http://amp.pharm.mssm.edu/Enrichr