Centrality-based Pathway Enrichment

Zuguang Gu

April 28, 2012

1 Introduction

Gene set enrichment analysis is broadly used in microarray data analysis [5, 3]. It aimes to find which biological functions are affected by a group of related genes behind the massive information. The most used methotology is finding these significant gene set from a 2×2 contingency table, usually by Fisher's exact test or chi-square test. This kind of analysis is known as Over-represented Analysis (ORA). It takes a list of differential expressed gene, and returns significant gene sets that the differential genes are enriched in. A lot of methods have been developed under the framework of ORA such as DAVID [4] (http://http://david.abcc.ncifcrf.gov/) and `GOstats` package [2].

For a specific form of gene sets, biological pathways are collections of correlated genes/proteins, RNAs and compounds that work together to regulate specific biological processes. Instead of just being a list of genes, a pathway contains the most important information that is how the member genes interact with each other. Thus network structure information is necessary for the intepretation of the importance of the pathways.

In this package, the original pathway enrichment method (ORA) is extended by introducing network centralities as the weight of nodes which have been mapped from differentially expressed genes in pathways. There are two advantages compared to former methods. First, for the diversity of genes' characters and the difficulties of covering the importance of genes from all aspects, we do not design a fixed measurement for each gene but set it as an optional parameter in the model. Researchers can select from candidate choices where different measurement reflects different aspect of the importance of genes. In our model, network centralities are used to measure the importance of genes in pathways. Different centrality measurements assign the importance to nodes from different aspects. For example, degree centrality measures the amount of neighbours that a node directly connects to, and betweenness centrality measures how many information streams must pass through a certain node. Generally speaking, nodes having large centrality values are central nodes in the network. It's observed that nodes represented as metabolites, proteins or genes with high centralities are essential to keep the steady state of biological networks. Moreover, different centrality measurements may relate to different biological functions. The selection of centralities for researchers depends on what kind of genes they think important. Second, we use nodes as the basic units of pathways instead of genes. We observe that nodes in the pathways include different types of molecules, such as single gene, complex and protein families. Assuming a complex or family contains ten differentially expressed member genes, in traditional ORA, these ten

genes behave as the same position as other genes represented as single nodes, and thus they have effect of ten. It is not proper because these ten genes stay in a same node in the pathway and make functions with the effect of one node. Also, a same gene may locate in different complexes in a pathway and if taking the gene with effect of one, it would greatly decrease the importance of the gene. Therefore a mapping procedure from genes to pathway nodes is applied in our model. What's more, the nodes in pathways also include none-gene nodes such as microRNAs and compounds. These nodes also contribute to the topology of the pathway. So, when analyzing pathways, all types of nodes are retained.

2 Example

In this example, we use PID pathway catalogues which have been already integrated in `CePa` package. The pathway data here contains a pathway list, a interaction list and the mapping from node ids to gene ids.

```
> library(CePa)
> set.seed(123)
> data(PID.db)
> names(PID.db)

[1] "NCI"      "BioCarta" "KEGG"     "Reactome"

> names(PID.db$NCI)

[1] "pathList"        "interactionList" "mapping"         "node.name"
[5] "node.type"       "created"
```

The `CePa` package needs a differentially expressed gene list and a background gene list. The differential gene list can be obtained through variaty of methods such as *t*-test, SAM [7] and limma [6]. The background gene list is the complete category of genes that exist on a certain microarray platform or from the whole genome. The `CePa` package contains an example gene list and a background gene list. The gene list is obtained from a microarray study by *t*-test [1].

```
> data(gene.list)
> names(gene.list)

[1] "bk"  "dif"
```

`PID.db` contains four pathway catalogues which are BioCarta, NCI, Reactome and KEGG. Among them, NCI catalogue is processed via manual curations and is recommanded by the PID database, so in the following analysis, we use the NCI pathway catalogue. In order to find significant pathways under several centrality measurements, use `cepa.all` function.

```
> res = cepa.all(gene.list$dif, gene.list$bk,
+       PID.db$NCI$pathList, PID.db$NCI$interactionList, PID.db$NCI$mapping)

  1/207, hif1_tfpathway...
  2/207, s1p_s1p5_pathway...
  3/207, wnt_signaling_pathway...
```

```
4/207, ap1_pathway...
5/207, lpa4_pathway...
...
```

The differential gene list and the background gene list should be indicated with the same identifiers (e.g. gene symbol or refseq ID). All genes in the differential gene list should exist in the background gene list. In this example, gene list must be formatted as gene symbol.

Basicly, a pathway contains a list of interactions. The `pathList` argument is a list where elements in the list is the vector of interaction IDs in the pathway. The interactions in the pathway can be got from a interaction list pool represented as `interactionList` argument. The `interactionList` argument stores the total interaction list in the pathway catalogue. It represents as a three columns data frame or matrix where the first column is the interaction id, the second column is the input node id and the third column is the output node id.

The mapping data frame provide the mapping from node id to gene id. The first column is the node id and the second column is the gene id. By default, it assumes every gene is the pathway is a single node.

By default, `cepa.all` use `equal.weight`, `in.degree`, `out.degree`, `betweenness`, `in.reach` and `out.reach` centralities as pathway nodes' weight. More centrality measurements can be used by setting it as a function (such as closeness, cluster coefficient).

In `CePa` package, the pathway data, interactions and the mappings are provided. But users must make sure the identifier of gene should be official gene symbol if they want to use the data.

In order to generate the null distribution of the pathway score, novel differential gene list is sampled from the background gene list. P-values are calculated from 1000 simulations by default.

`res` is a `cepa.all` class object. To see the general information of this object:

```
> res

number of pathways: 207

Significant pathways (p.value <= 0.01):
              Number
equal.weight      14
in.degree         22
out.degree        18
betweenness       15
in.reach          18
out.reach         20
```

The p-values or adjusted p-values of all pathways under different centralities can be compared through the heatmap of p-values (Figure 1).

```
> plot(res, adj.method = "BH")
```

By default, `plot` generates the heatmap containing all pathways. If only significant pathways are of interest, the `only.sig` argument can be set to `TRUE`. (Figure 2).

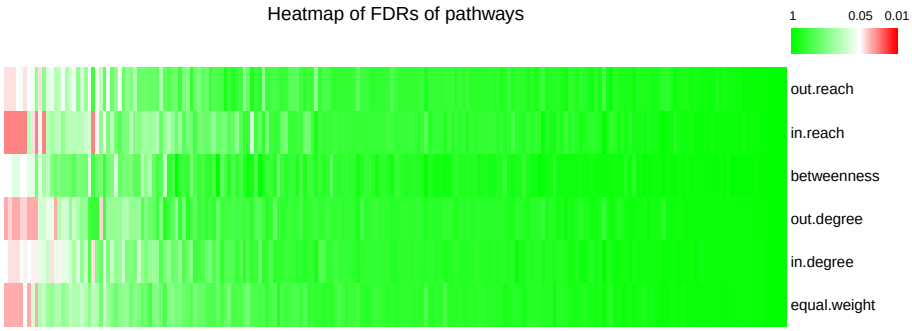

Figure 1: Heatmap of p-values of all pathways

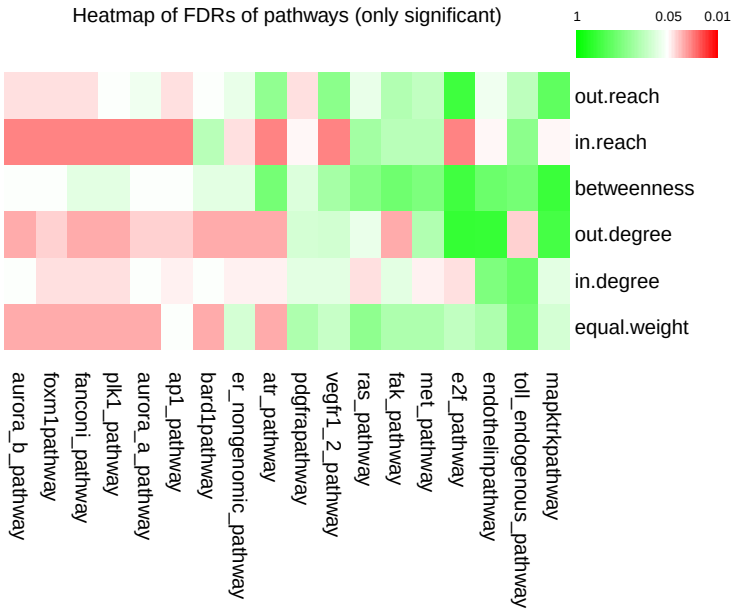

Figure 2: Heatmap of p-values of significant pathways

```
> plot(res, adj.method = "BH", only.sig = TRUE)
```

The numeric values of p-values can be obtained via `p.table`. The function just returns the raw p-values.

```
> pt = p.table(res)
> head(pt)
```

```
                       equal.weight   in.degree  out.degree betweenness
hif1_tfpathway          0.744255744 0.817182817 0.798201798 0.800199800
s1p_s1p5_pathway        0.136863137 0.198801199 0.138861139 0.062937063
wnt_signaling_pathway   0.886113886 0.869130869 0.898101898 0.896103896
ap1_pathway             0.001998002 0.001998002 0.001998002 0.000999001
lpa4_pathway            0.824175824 0.820179820 0.795204795 0.722277722
avb3_opn_pathway        0.752247752 0.577422577 0.644355644 0.351648352
                          in.reach   out.reach
hif1_tfpathway        0.650349650 0.807192807
s1p_s1p5_pathway      0.070929071 0.136863137
wnt_signaling_pathway 0.818181818 0.920079920
ap1_pathway           0.000999001 0.000999001
lpa4_pathway          0.776223776 0.746253746
avb3_opn_pathway      0.679320679 0.602397602
```

We can get the result for single pathway under specific centrality from the `cepa.all` object by identifying the index for the pathway and the index for the centrality.

```
> g = get.cepa(res, "mapktrkpathway", "in.degree")
> g
```

```
  diff nodes in pathway: 11
  all nodes in pathway: 33
  diff genes in pathway: 10
  all genes in pathway: 33


  weight: in.degree
  p-value: 5.0e-03
```

`g` is a `cepa` class object. It stores information of the evaluation of a single pathway under a single centrality. The distribution of the pathway score and the network graph can be generated by `plot` on the `cepa` object (Figure 3).

```
> plot(g)
```

By default, the node labels is combined from member genes. The exact name for each node can be set by `node.name` argument. Also, more detailed categories of the nodes can be set by `node.type` argument (Figure 4).

```
> plot(g, node.name = PID.db$NCI$node.name, node.type = PID.db$NCI$node.type)
```

For simplicity, the plotting for the `cepa` object can be directly applied on the `cepa.all` object by specifying the index of the pathway and the index of the centrality (Figure 5).

```
> plot(res, id = "mapktrkpathway", cen = "in.degree",
+      node.name = PID.db$NCI$node.name, node.type = PID.db$NCI$node.type)
```

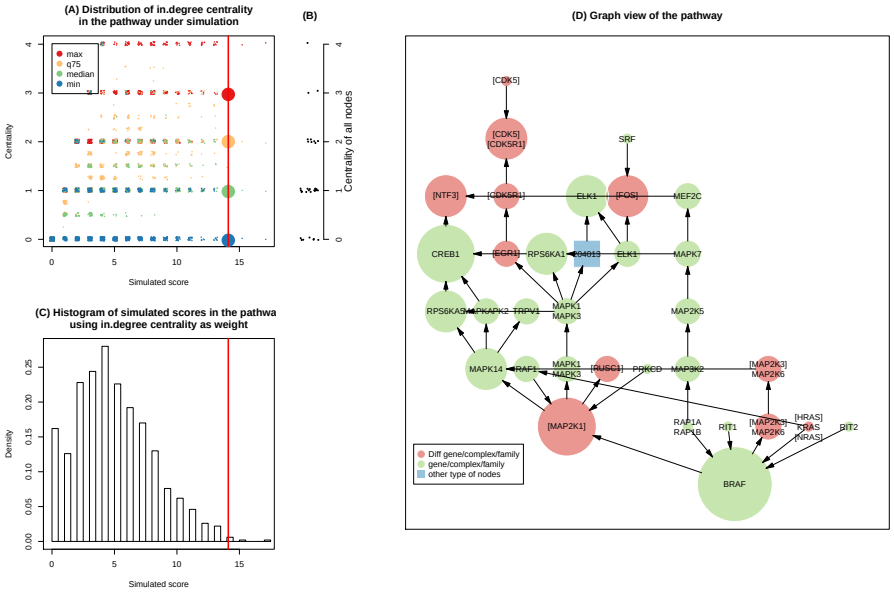

Figure 3: Summary of a single pathway under one centrality

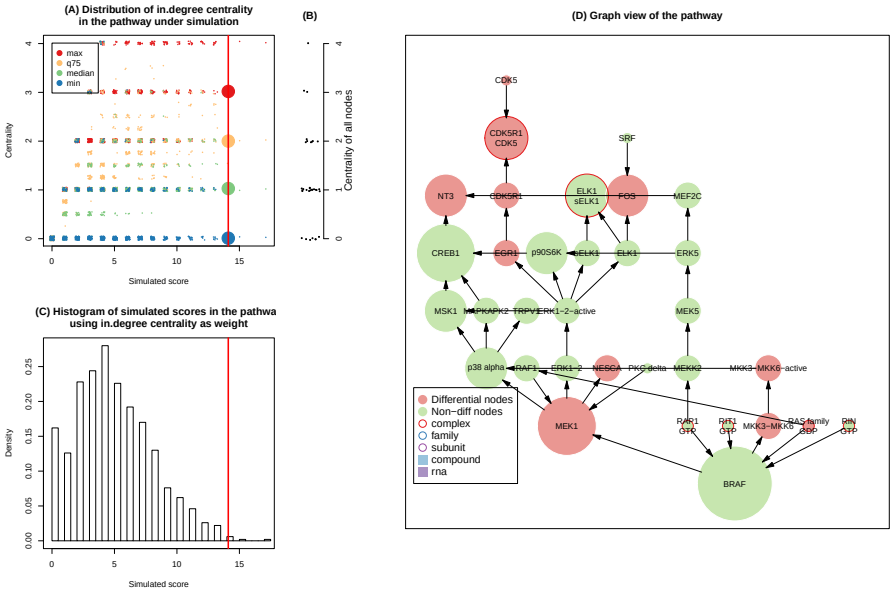

Figure 4: Summary of a single pathway under one centrality, with node name and node type specified

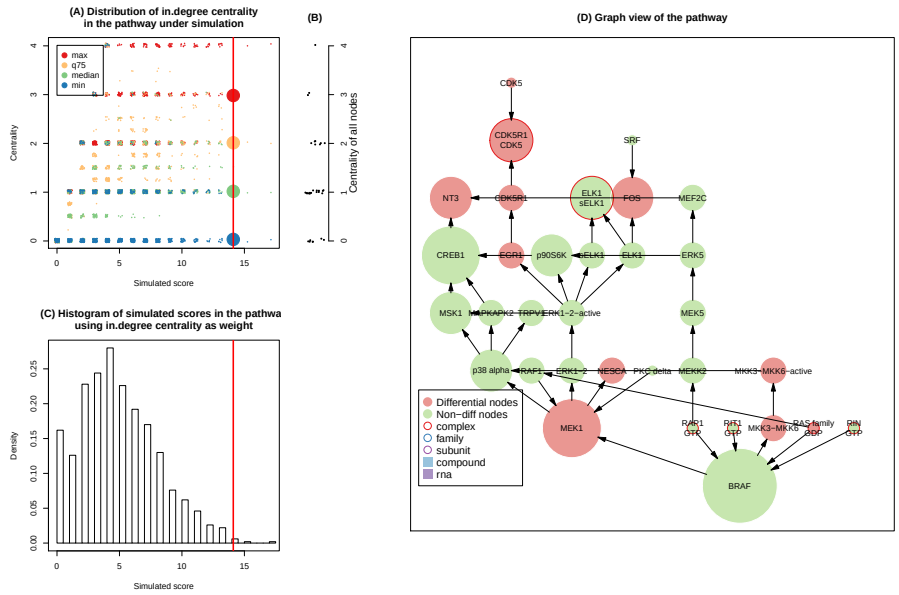

Figure 5: Summary of a single pathway under one centrality, directly from `cepa.all` object

3 The `report` function

One of the advantages of `CePa` package is that it can generate a detailed report in HTML format. The function `report` is used to generate report. The report will locate in the current working directory. By default it only generate figures of the significant pathways, but this can be changed by setting `sig.only` argument to `FALSE`.

```
> report(res)

  generate images for ap1_pathway ...
  generate images for epopathway ...
  generate images for il12_stat4pathway ...
  generate images for foxm1pathway ...
  generate images for mapktrkpathway ...
  generate images for aurora_a_pathway ...
  ...


> report(res, sig.only = FALSE)
```

4 Parallel computing

Since `CePa` evaluates pathways independently, the process can be realized through parallel computing. In R statistical environment, there are many packages focusing on parallel computing such as `snow`, `multicore`, etc. Here we demonstrate how to apply the parallel version of `CePa`, taking `multicore` for example.

```
> library(multicore)
> # identify how many cores you want to use in your computer
> ncores = 4
```

Since there are a list of pathways, we would link to divide them into several approximately equal groups, so we have a `divide` function.

```
> divide = function(x, k) {
+         if(length(x) ==1 && is.numeric(x)) {
+                 x = 1:x
+         }
+         if(length(x) < k) {
+                 stop("o")
+         }
+         w = floor(length(x)/k)
+         q = length(x) - k*w
+         d = matrix(0, nrow=k, ncol=2)
+         n = 1
+         for(i in 1:k) {
+                 d[i, 1] = n
+                 d[i, 2] = n+w-1+ifelse(q>0, 1, 0)
+                 n = d[i,2]+1
+                 q = ifelse(q > 0, q-1, 0)
+         }
+         d[k,2] = length(x)
+         return(d)
+ }
```

In the `divide` function, the first argument is a vector, usually a index vector, and the second argument identify how many part you want to divide into. ALso, the first argument can be a positive integer. For example, we want to divide `1:10` into two groups.

```
> divide(1:10, 2)
```

```
     [,1] [,2]
[1,]    1    5
[2,]    6   10
```

The function returns a matrix. Rows correspond to groups and columns correspond to the start index and the end index. If the vector can not be divided equally, the function would return an approximately division.

```
> divide(1:10, 3)
```

```
     [,1] [,2]
[1,]    1    4
[2,]    5    7
[3,]    8   10
```

Now we can divide the complete NCI pathway catalogue into several groups.

```
> NCI = PID.db$NCI
> d = divide(1:length(NCI$pathList), ncores)
```

Then we use `mclapply` which is something like a parallel version of `lapply` to do parallele computing.

```
> res = mclapply(1:ncores, function(i) {
+                     cepa.all(dif, bk, NCI$pathList[d[i,1]:d[i,2]],
+                              NCI$interactionList,
+                              NCI$mapping)},
+                mc.cores=ncores)
```

In the `mclapply`, calculation in each core would returns a `cepa.all` object. Thus, `res` is a list of `cepa.all` objects. We need some code to transform it into a single `cepa.all` object containing all pathways.

```
> pathway.name = character(0)
> centrality = names(res[[1]]$pathway.result)
> pathway.result = list()
> for(i in 1:length(res)) {
+     pathway.name = c(pathway.name, res[[i]]$pathway.name)
+     for(cen in centrality) {
+         pathway.result[[cen]] = c(pathway.result[[cen]],
+                                   res[[i]]$pathway.result[[cen]])
+     }
+ }
> obj = list(pathway.name = pathway.name,
+            pathway.result = pathway.result)
> class(obj) = "cepa.all"
```

OK, now the `obj` is a `cepa.all` object just like the one generated from non-parallel `CePa`.

References

[1] J. Burchard, C. Zhang, A. M. Liu, R. T. P. Poon, N. P. Y. Lee, K.-F. Wong, P. C. Sham, B. Y. Lam, M. D. Ferguson, G. Tokiwa, R. Smith, B. Leeson, R. Beard, J. R. Lamb, L. Lim, M. Mao, H. Dai, and J. M. Luk. microRNA-122 as a regulator of mitochondrial metabolic gene network in hepatocellular carcinoma. *Molecular systems biology*, 6:402, Aug. 2010.

[2] S. Falcon and R. Gentleman. Using GOstats to test gene lists for GO term association. *Bioinformatics*, 23(2):257–8, 2007.

[3] D. W. Huang, B. T. Sherman, and R. A. Lempicki. Bioinformatics enrichment tools: paths toward the comprehensive functional analysis of large gene lists. *Nucleic acids research*, 37(1):1–13, Jan. 2009.

[4] D. W. Huang, B. T. Sherman, and R. A. Lempicki. Systematic and integrative analysis of large gene lists using DAVID bioinformatics resources. *Nature protocols*, 4(1):44–57, Jan. 2009.

[5] P. Khatri and S. DrÄČghici. Ontological analysis of gene expression data: current tools, limitations, and open problems. *Bioinformatics*, 21(18):3587–95, Sept. 2005.

[6] G. K. Smyth. Limma: linear models for microarray data. pages 397–420, 2005.

[7] V. G. Tusher, R. Tibshirani, and G. Chu. Significance analysis of microarrays applied to the ionizing radiation response. *Proceedings of the National Academy of Sciences of the United States of America*, 98(9):5116–21, Apr. 2001.