Centrality-based Pathway Enrichment

Zuguang Gu

September 11, 2012

1 Introduction

Gene set enrichment analysis is broadly used in microarray data analysis [8, 5]. 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 [6] (http://http://david.abcc.ncifcrf.gov/) and `GOstats` package [3]. The second methodology to find significant pathways is to use whole expression matrix, named Gene-set Analysis (GSA). GSA methods are implemented via either a univariate or a multivariate procedure [1]. In univariate analysis, gene level statistics are initially calculated from fold changes or statistical tests (e.g., t-test). These statistics are then combined into a pathway level statistic by summation or averaging. GSEA [12] is a widely used univariate tool that utilizes a weighted Kolmogorov-Smirnov test to measure the degree of differential expression of a gene set by calculating a running sum from the top of a ranked gene list. Multivariate analysis considers the correlations between genes in the pathway and calculates the pathway level statistic directly from the expression value matrix using Hotelling's T^2 test [11] or MANOVA models [7].

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 and GSA) is extended by introducing network centralities as the weight of nodes which have been mapped from differentially expressed genes in pathways [4]. 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 non-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 Pathway Catalogue

Pathways are collected from public databases, such as PID, KEGG, BioCarta etc. In `CePa` package, four catalogues (PID, KEGG, BioCarta and Reactome) from PID database have been integrated. The pathway data are parsed from XML format file provided by the PID FTP site. The Perl code for parsing can be obtained from the author's website (http://mcube.nju.edu.cn/jwang/lab/soft/cepa/). The pathway data is stored in `PID.db`. Note only part of pathways in the XML file are listed on the PID website. Also, we have set the minimum and maximum connected nodes when extracting pathways from PID, so not all the pathways listed on the PID website are in `PID.db`.

```
> library(CePa)
> data(PID.db)
> names(PID.db)
```

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

Each pathway catalogue has been stored as a `pathway.catalogue` class object. The `print.pathway.catalogue` function simply prints the number of pathways in the catalogue. The `plot.pathway.catalogue` function visulizes general information of the catalogue (figure 1). It plot: A) Distribution of the number of member genes in each node; B) Distribution of the number of nodes in which a single gene resides; C) Relationship between node count and gene count in biological pathways.

```
> class(PID.db$NCI)
```

```
[1] "pathway.catalogue"
```

```
> PID.db$NCI
```

```
  The catalogue contains 206 pathways.
```

```
> plot(PID.db$NCI)
```

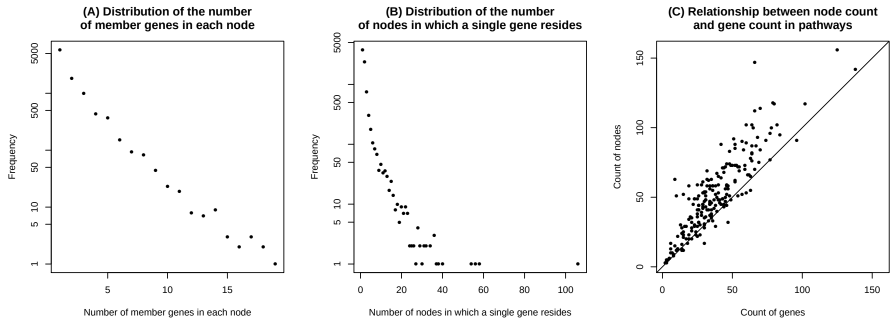


Figure 1: Meta analysis of pathway catalogue

The pathway catalogue data contains a list of pathways and each pathway contains a list of interactions. There are several parts in the pathway data where three of them is must: the pathway list, the interaction list and the mapping list. The corresponding list name are `pathList`, `interactionList` and `mapping`.

```
> names(PID.db$NCI)
```

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

You can find the version of NCI data.

```
> PID.db$NCI$version
```

```
[1] "2012_07_19 09:34::20"
```

The `pathList` is a list in which each item is a list of interaction IDs

```
> head(PID.db$NCI$pathList, n = 2)
```

```
$wnt_signaling_pathway
 [1] "203098" "203087" "203104" "203106" "203125" "203127" "203092" "203142"
 [9] "203097" "203111" "203099" "203118" "203103" "203137" "203143" "203091"
[17] "203088" "203141" "203128" "203089" "203101" "203117" "203126" "203140"
[25] "203095" "203108" "203119" "203129" "203113" "203120" "203094" "203102"
[33] "203122" "203136" "203145" "203105" "203123" "203144" "203130" "203132"
[41] "203124" "203107" "203110" "203093" "203133" "203090" "203096" "203109"
[49] "203100" "203121" "203116" "203112"

$cdc42_reg_pathway
```

```
 [1] "203416" "203396" "203420" "203393" "203405" "203418" "203392" "203415"
 [9] "203388" "203408" "203389" "203390" "203403" "203412" "203398" "203410"
[17] "203406" "203395" "203391" "203401" "203394" "203419" "203404" "203397"
[25] "203399" "203407" "203402" "203400" "203413" "203411" "203409" "203414"
```

The `interactionList` is a three-column matrix in which the first column is the interaction ID, the second column is the input node ID and the third column is the output node ID.

```
> head(PID.db$NCI$interactionList)

  interaction.id  input output
1         503376 507485 506711
2         503376 507487 507485
3         204164 202538 208490
4         204164 208487 208490
5         100688 101169 101176
6         100688 101177 101176
```

The `mapping` is the two-column matrix in which the first column is the node ID and the second column is the gene ID.

```
> head(PID.db$NCI$mapping)

  node.id  symbol
1  202230 ARHGAP6
2  201405    XIAP
3  503376  SLC7A2
4  203548   SATB1
5  201647    CRY2
6  508774     CRH
```

The pathway catalogue can also be self-defined by `set.pathway.catalogue` function. The function returns a `pathway.catalogue` class object. E.g. we only need the first ten pathways in NCI catalogue.

```
> new.catalogue = set.pathway.catalogue(pathList = PID.db$NCI$pathList[1:10],
+                 interactionList = PID.db$NCI$interactionList,
+                 mapping = PID.db$NCI$mapping)
```

In the following examples, we will use NCI catalogue as the default pathway catalogue.

3 ORA Extension

The pathway score is defined as the summation of the weights of differentially affected nodes in the pathway:

$$s = \sum_{i=1}^{n} w_i d_i \tag{1}$$

where s is the score of the pathway, w_i is the weight of the i^{th} node and reflects the importance of the node, n is the number of nodes in the pathway, and d_i identifies whether the i^{th} node is differentially affected ($= 1$) or not ($= 0$).

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 [13] and limma [10]. 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 [2].

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

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

In order to find significant pathways under several centrality measurements, we use cepa.all function.In the function, dif refers to the differential gene list, bk refers to the background gene list and the pc refers to the pathway catalogue.

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

```
  Calculate pathway scores...
    1/211, wnt_signaling_pathway...
      - equal.weight: 0.878
      - in.degree: 0.864
      - out.degree: 0.921
      - betweenness: 0.89
      - in.reach: 0.81
      - out.reach: 0.91
    ...
```

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, since PID.db is applied, gene list must be formatted as gene symbol. If background gene list is not specified, the function use whole human genome genes as default.

By default, cepa.all calls 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). The non-default centralities can be set by cen argument, and remember to set the cen.name argument to get the name of the centrality. Note you can mix the centralities in string format and function format. When you set the function object, the only parameter for the function is the network in igraph object format. The following codes are examples to set the centralities.

```
> # if you use the function, you should quote the function
> # because we need the function name as the centrality name
> cepa.all(dif = gene.list$dif, bk = gene.list$bk,
+          pc = PID.db$NCI,
+          cen = list("in.degree", quote(closeness)))
```

Moreover, if your centrality function contains more than one argument, you must wrap to a function that only have one `igraph` object argument.

```
> in.closeness = function(g) closeness(g, mode = "in")
> cepa.all(dif = gene.list$dif, bk = gene.list$bk,
+          pc = PID.db$NCI,
+          cen = list("in.degree", quote(in.closeness)))
> # If you don't like the function name to be centrality name
> # you can set by cen.name argument
> cepa.all(dif = gene.list$dif, bk = gene.list$bk,
+          pc = PID.db$NCI,
+          cen = list("in.degree", quote(in.closeness)),
+          cen.name = c("In-degree", "In-closeness"))
```

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.

The calculation would spend about 12 min. `res` is a `cepa.all` class object. To see the general information of this object:

```
> res
```

```
number of pathways: 211

Significant pathways (p.value <= 0.01):
             Number
equal.weight     20
in.degree        19
out.degree       19
betweenness      14
in.reach         19
out.reach        20
```

It will print the number of significant pathways under different centralities. For ORA extension, `cepa.all` in fact calls `cepa.ora.all` function. So the following code is same as the former code.

```
> res = cepa.ora.all(dif = gene.list$dif, bk = gene.list$bk,
+       pc = PID.db$NCI)
```

The p-values or adjusted p-values of all pathways under different centralities can be compared through the heatmap of p-values (Figure 2). Users can select methods to adjust raw p-values.

```
> plot(res)
```

By default, `CePa` use `p.adjust` to calculate adjusted p-values, so only methods valid for `p.adjust` can be applied to `CePa`. However, there is another popular method to adjust p-values: `qvalue`. `CePa` did not implement it since errors may occur when evaluating some kind of p-values. Nevertheless, users can override the default `p.adjust` to support `qvalue` by themselves, use code below:

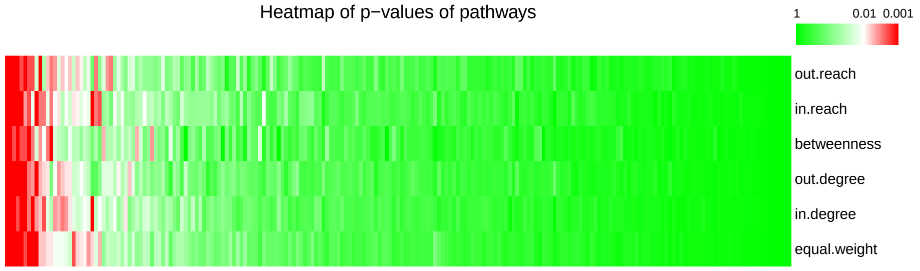


Figure 2: Heatmap of p-values of all pathways

```
> library(qvalue)
> p.adjust = function(p, method = c("holm", "hochberg", "hommel", "bonferroni",
+                    "BH", "BY", "fdr", "none", "qvalue"), ...) {
+   if(method == "qvalue") {
+       # qvalue has more arguments, pass them by ...
+       qvalue(p, ...)$qvalue
+   } else {
+       stats::p.adjust(p, method)
+   }
+ }
```

R will first look for `p.adjust` in `.GlobalEnv` environment and get your own `p.adjust`.

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 3). Here we do not set `cutoff` arguments because if adjusted method is used, the default cutoff is 0.05, while if user just wants the raw p-values, the default cutoff is 0.01.

```
> 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
wnt_signaling_pathway    0.878121878 0.86413586 0.921078921 0.890109890
cdc42_reg_pathway        0.858141858 0.84515485 0.818181818 0.852147852
mtor_4pathway            0.777222777 0.80319680 0.707292707 0.596403596
plk3_pathway             0.007992008 0.01598402 0.007992008 0.002997003
era_genomic_pathway      0.079920080 0.08891109 0.047952048 0.074925075
insulin_glucose_pathway  1.000000000 1.00000000 1.000000000 1.000000000
                          in.reach   out.reach
wnt_signaling_pathway   0.81018981 0.910089910
cdc42_reg_pathway       0.84815185 0.859140859
mtor_4pathway           0.83616384 0.422577423
```

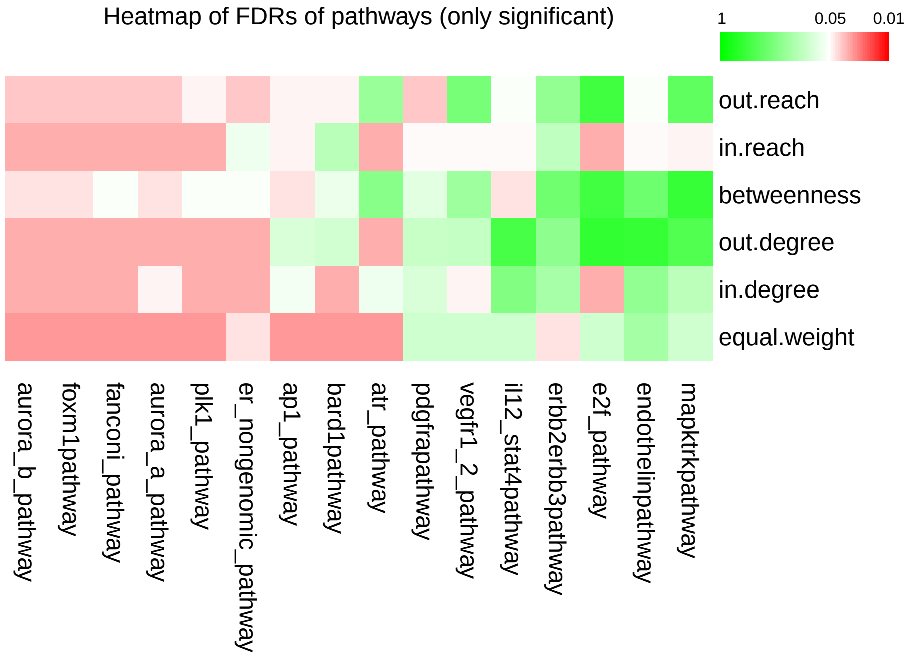


Figure 3: Heatmap of p-values of significant pathways

```
plk3_pathway            0.01398601 0.005994006
era_genomic_pathway     0.09290709 0.017982018
insulin_glucose_pathway 1.00000000 1.000000000
```

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, id = "mapktrkpathway", cen = "in.reach")
> g

  procedure: ora
  weight: in.reach
  p-value: 0.002
```

`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` function on the `cepa` object by specifying `type` argument (figure 4 and figure 5).

```
> plot(g, type = "graph")
```

```
> plot(g, type = "null")
```

By default, `type` is set to `graph`, and 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 6).

```
> 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 6).

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

If users use `plot` to draw network graphs, the function would return an `igraph` object. So if users are not satisfy with the default graph, they can visulize by their own methods. An example of custumize your network can be found at the next section.

```
> obj = plot(res, id = "mapktrkpathway", cen = "in.reach")
> class(obj)
[1] "igraph"
```

The `igraph` package provides a `write.graph` function to output graph into several formats. As I have tried, with `graphml` format, Cytoscape Web [9] (http://http://cytoscapeweb.cytoscape.org/) can make a more beautiful and interactive visualization of the network.

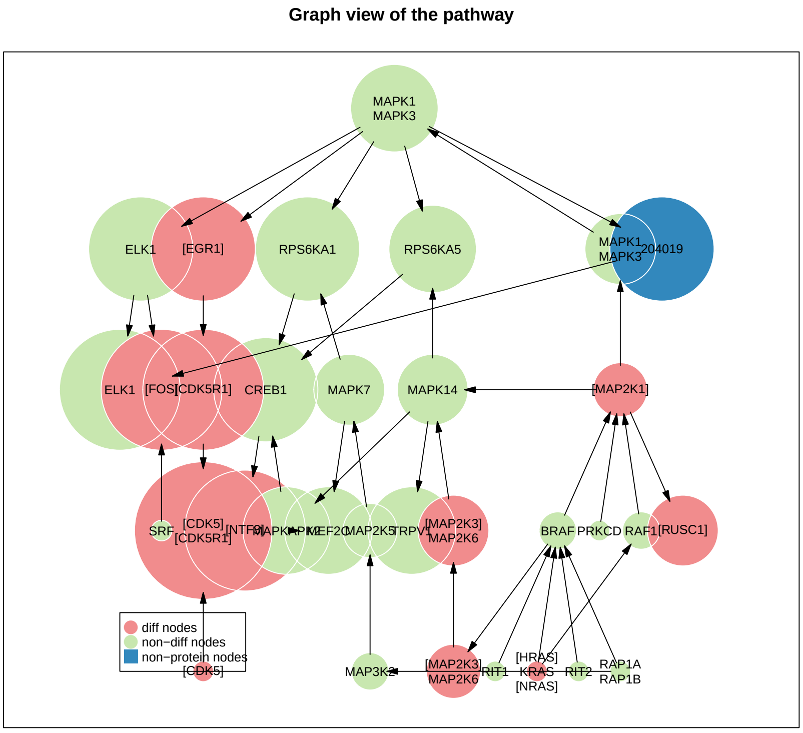


Figure 4: Network visualization of a pathway

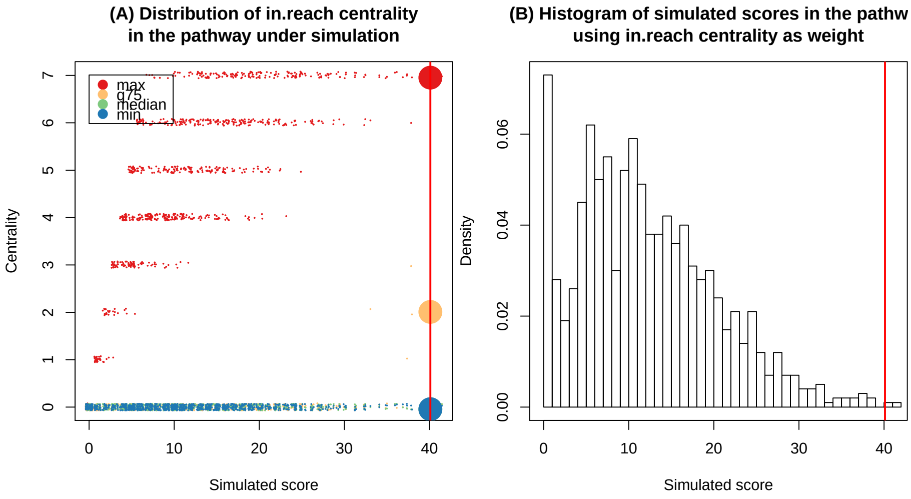


Figure 5: Null distribution of pathway score

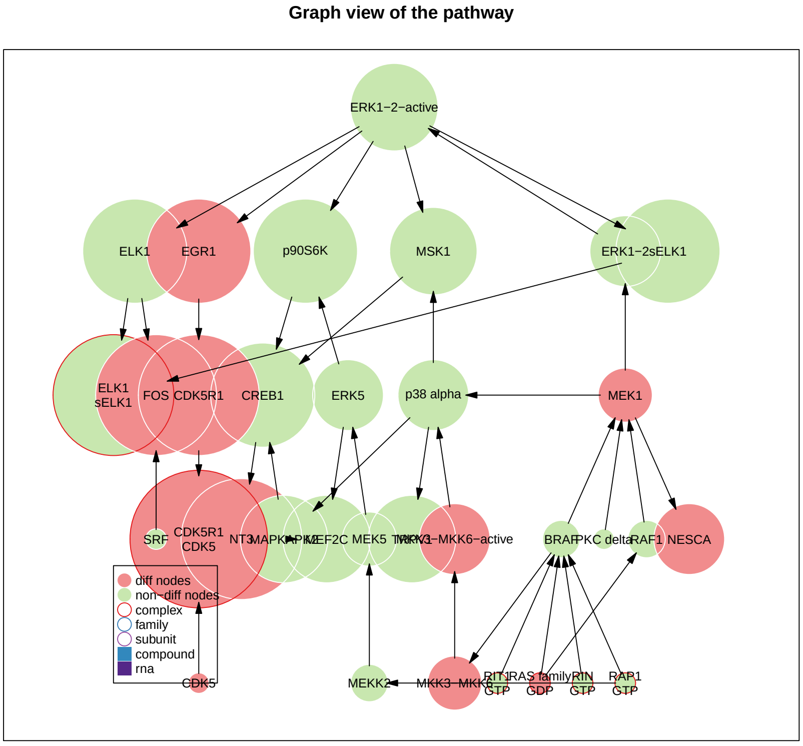


Figure 6: Network visualization of a pathway, with node name and node type specified

```
> write.graph(obj, file = "example-network.xml", format = "graphml")
> write.graph(obj, file = "example-network.gml", format = "gml")
```

Instead of analysis a list of pathways, users can also be focused on a single pathway under a single centrality by identifying the id of the pathway in the catalogue.

```
> res.pathway = cepa(dif = gene.list$dif, bk = gene.list$bk,
+                pc = PID.db$NCI, "mapktrkpathway",
+                cen = "in.reach")
```

Similarly, cepa function here directly calls cepa.ora.

4 GSA extension

In the traditional univariate GSA procedure, the score s of the pathway is defined as:

$$s = f(\mathbf{g}) \tag{2}$$

where f transforms the gene-level statistic to a pathway-level statistic (e.g. by summation, averaging) and $\mathbf{g}$ is the gene-level statistic vector which typically comprises t-values. In ORA, $\mathbf{g}$ is a binary variant and $f(\mathbf{g})$ is summation. In our model to extend GSA, gene-level statistic is first transformed to node-level statistic. We define the vector of the node-level statistics as $\mathbf{d}$. When nodes in pathways comprise multiple genes, the node-level statistic can be considered as the largest principle component of the corresponding member genes. Using centrality as the weight, the score is defined as

$$s = f(\mathbf{wd}) \tag{3}$$

where $\mathbf{w}$ is the weight vector and the transformation function f acts upon the product of $\mathbf{w}$ and $\mathbf{d}$. Equation 3 incorporates centrality weight into the original node-level statistic. The null distribution of the pathway score could then be generated by permuting the gene expression matrix.

Since GSA procedure need a complete expression matrix, we first read the P53 microarray data set. The P53_symbol.gct and P53.cls can be downloaded from http://mcube.nju.edu.cn/jwang/lab/soft/cepa/. read.gct and read.cls are simple functions to read expression data and phenotype data.

```
> eset = read.gct("P53_symbol.gct")
> # some process of the names of genes
> rownames(eset) = gsub("\\s+.*$", "", rownames(eset))
> label = read.cls("P53.cls", treatment="MUT", control="WT")
```

Here, we also use cepa.all to do batch pathway analysis. The following code spent about 38 min with 1000 sample permutations.

```
> res = cepa.all(mat = eset, label = label, pc = PID.db$NCI,
+                   nlevel = "tvalue_sq", plevel = "mean")

  Calculate gene level values.
  Calculate pathway score...
    1/211, wnt_signaling_pathway...
    Calculate node level value and permutate sample labels...
    17 genes measured in the pathway...
    - equal.weight: 0.587
    - in.degree: 0.652
    - out.degree: 0.777
    - betweenness: 0.466
    - in.reach: 0.56
    - out.reach: 0.696
  ...
```

Here, we use `mat` and `label` arguments instead of `dif` and `bk` arguments. In fact, when specifying `mat` and `label` arguments, `cepa.all` calls `cepa.univaraite.all`.

In GSA procedure, first a node level statistic should be calculated. In `CePa` package, there are three methods to calculate node level statistics. User can choose from `tvalue`, `tvalue_abs` and `tvalue_sq`. `tvalue_abs` is choosen as the default node level method because it can capture two directional regulations. After we get the node level statistics in the pathway, a pathway level transformation should be applied. User can choose from `max`, `min`, `median`, `sum`, `mean` and `rank`. `mean` is taken as default.

The node level statistic can be self-defined. The self-defined function should only contain two argumetns, one for vector of expression value in treatment class and one for that in control class. E.g. we set the node level statistic as kind of robust t-value:

```
> robust_tvalue = function(x, y) {
+     qx = quantile(x, c(0.1, 0.9))
+     qy = quantile(y, c(0.1, 0.9))
+
+     x = x[(x <= qx[2]) & (x >= qx[1])]
+     y = y[(y <= qy[2]) & (y >= qy[1])]
+
+     n1 = length(x)
+     n2 = length(y)
+     v1 = var(x)
+     v2 = var(y)
+     ifelse(v1 + v2 == 0, 0, (mean(x) - mean(y)) / sqrt(v1/n1 + v2/n2))
+ }
> res = cepa.all(mat = eset, label = label, pc = PID.db$NCI,
+                nlevel = robust_tvalue, plevel = "mean")
```

Similarly, the pathway level transformation can also be self-defined:

```
> trim_mean = function(x) mean(x, trim = 0.2)
> res = cepa.all(mat = eset, label = label, pc = PID.db$NCI,
+                nlevel = "tvalue_abs", plevel = trim_mean)
```

Print the general result of the analysis and plot figures (figure 7).

```
> res

number of pathways: 211
```

```
Significant pathways (p.value <= 0.01):
             Number
equal.weight      6
in.degree         6
out.degree        7
betweenness       5
in.reach          7
out.reach         7
```

```
> plot(res, only.sig = TRUE, adj.method = "BH", cutoff = 0.1)
```

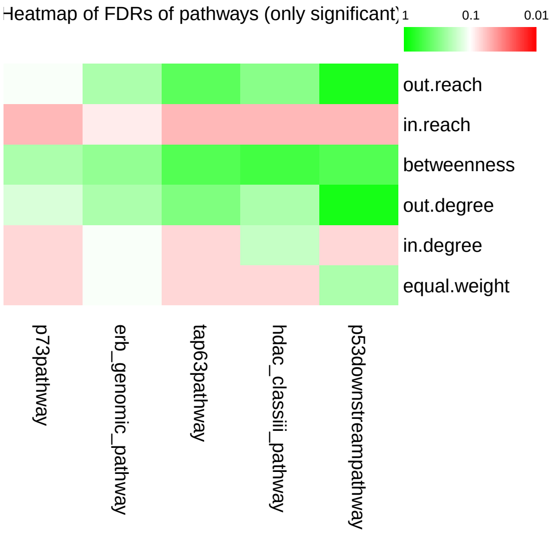


Figure 7: Heatmap of FDRs of significant pathways

If we are instread in p53 downstream pathway. First we extract this pathway under "in.degree" centrality from `res`.

```
> g = get.cepa(res, id = "p53downstreampathway", cen="in.degree")
> g

  procedure: gsa.univariate
  weight: in.degree
  p-value: 9.990e-04

> plot(g)
```

Figure 8 illustrates the graph of p53 downstream pathway. Since the pathway is evaluated under GSA procedure, the color of each node is continues in which red refers to up-regulated, green refers to down-regulated and white refers to no-change.

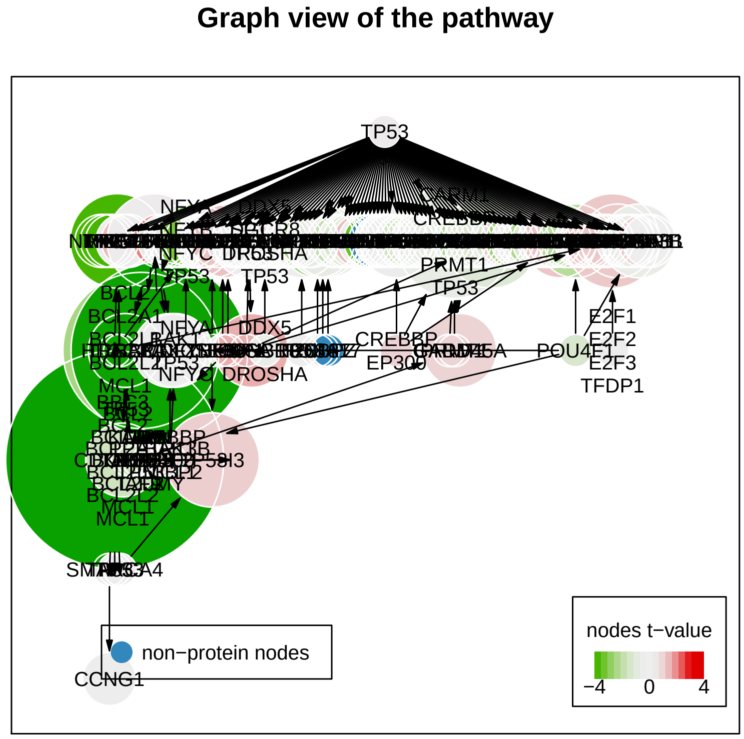


Figure 8: Network visulization of a pathway

Maybe due to to many nodes in the graph, Figure 8 is really hard to read. However, since the plotting function returns an `igraph` object. We can customize it handly (Figure 9).

```
> g2 = plot(g)
> # only label those nodes with high centralities
> V(g2)$label[V(g2)$size < quantile(V(g2)$size, 0.95)] = ""
> # we do not need margins
> par(mar = c(0,0,0,0))
> plot(g2)
```

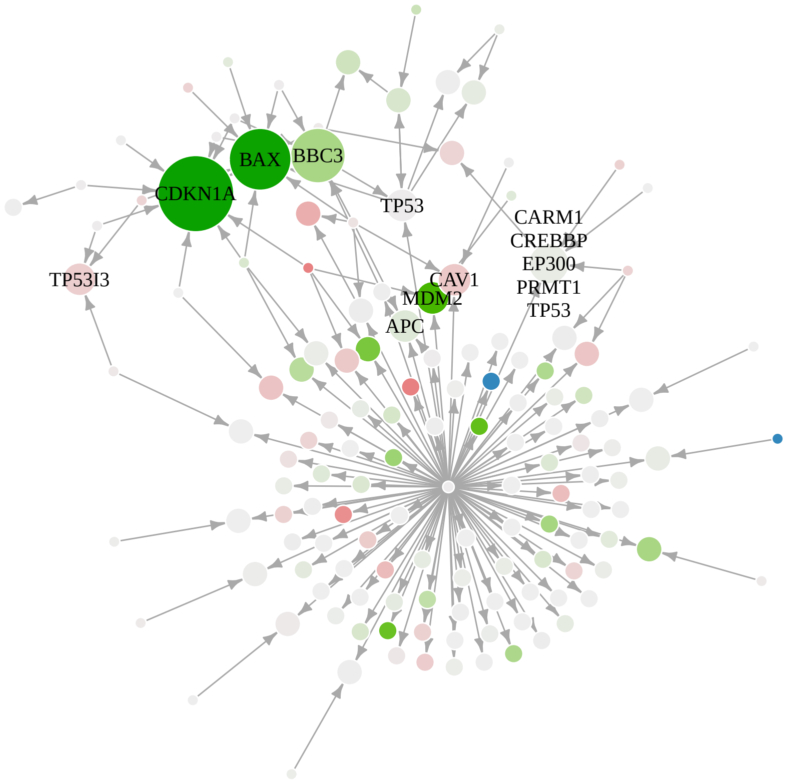


Figure 9: customize the network graph

5 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 `only.sig` 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, adj.method = "BH", cutoff = 0.2)
> report(res, only.sig = FALSE)
```

An example of the report can be found in figure 10. After `CePa` version 0.4, the network for pathways can be viewed interactively by Cytoscape Web [9] (figure 11).

6 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. After version 4.0, the package implemented a `cepa.all.parallel` function to do parallel computing.

`cepa.all.parallel` use `snow` package.

All the arguments for `cepa.all.parallel` are same as the arguments for `cepa.all` except the `ncores` arguments. The `ncores` specifies the number of cores for parallel computing.

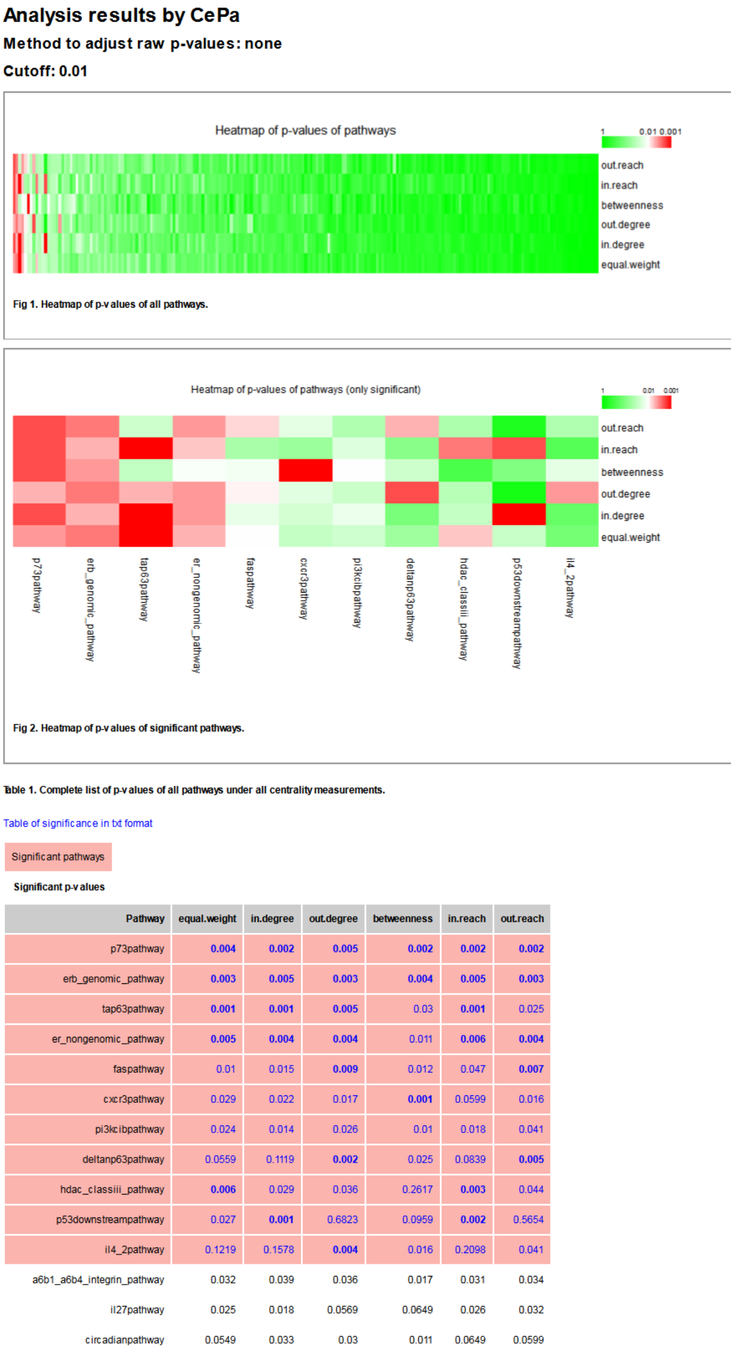
```
> res = cepa.all.parallel(dif = gene.list$dif, bk = gene.list$bk,
+       pc = PID.db$NCI, ncores = 4)
> res = cepa.all.parallel(mat = eset, label = label, pc = PID.db$NCI,
        nlevel = "tvalue_sq", plevel = "mean", ncores = 4)
```

The returned value `res` is a `cepa.all` class object.

References

[1] M. Ackermann and K. Strimmer. A general modular framework for gene set enrichment analysis. *BMC bioinformatics*, 10:47, Jan. 2009.

[2] 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.



Analysis results by CePa

Method to adjust raw p-values: none

Cutoff: 0.01

Fig 1. Heatmap of p-values of all pathways.

Fig 2. Heatmap of p-values of significant pathways.

Table 1. Complete list of p-values of all pathways under all centrality measurements.

Table of significance in txt format

Significant pathways

Significant p-values

Pathway	equal.weight	in.degree	out.degree	betweenness	in.reach	out.reach
p73pathway	**0.004**	**0.002**	**0.005**	**0.002**	**0.002**	**0.002**
erb_genomic_pathway	**0.003**	**0.005**	**0.003**	**0.004**	**0.005**	**0.003**
tap63pathway	**0.001**	**0.001**	**0.005**	0.03	**0.001**	0.025
er_nongenomic_pathway	**0.005**	**0.004**	**0.004**	0.011	**0.006**	**0.004**
faspathway	0.01	0.015	**0.009**	0.012	0.047	**0.007**
cxcr3pathway	0.029	0.022	0.017	**0.001**	0.0599	0.016
pi3kcibpathway	0.024	0.014	0.026	0.01	0.018	0.041
deltanp63pathway	0.0559	0.1119	**0.002**	0.025	0.0839	**0.005**
hdac_classiii_pathway	**0.006**	0.029	0.036	0.2617	**0.003**	0.044
p53downstreampathway	0.027	**0.001**	0.6823	0.0959	**0.002**	0.5654
il4_2pathway	0.1219	0.1578	**0.004**	0.016	0.2098	0.041
a6b1_a6b4_integrin_pathway	0.032	0.039	0.036	0.017	0.031	0.034
il27pathway	0.025	0.018	0.0569	0.0649	0.026	0.032
circadianpathway	0.0549	0.033	0.03	0.011	0.0649	0.0599

Figure 10: An report of the CePa analysis

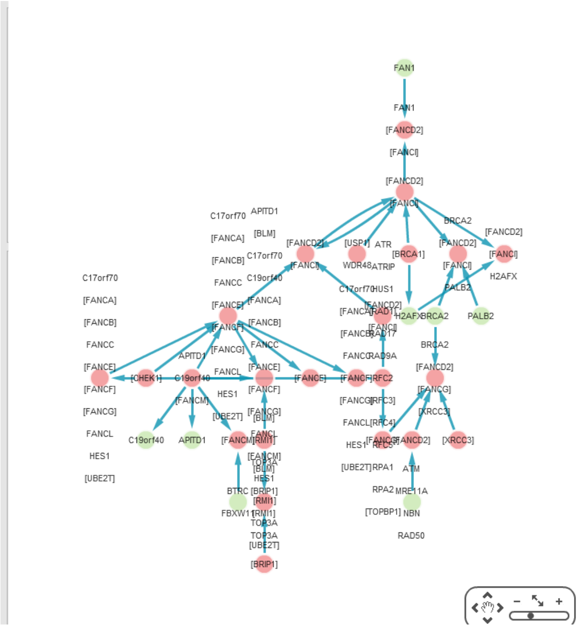


Figure 11: Network is visualized by Cytoscape Web

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

[4] Z. Gu, J. Liu, K. Cao, J. Zhang, and J. Wang. Centrality-based pathway enrichment: a systematic approach for finding significant pathways dominated by key genes. *BMC Systems Biology*, 6(1):56, 2012.

[5] 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.

[6] 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.

[7] M. Hummel, R. Meister, and U. Mansmann. GlobalANCOVA: exploration and assessment of gene group effects. *Bioinformatics*, 24(1):78–85, Jan. 2008.

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

[9] C. T. Lopes, M. Franz, F. Kazi, S. L. Donaldson, Q. Morris, and G. D. Bader. Cytoscape web: an interactive web-based network browser. *Bioinformatics*, 26(18):2347–2348, 2010.

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

[11] S. Song and M. Black. *pcot2: Principal Coordinates and Hotelling's T-Square method*. R package version 1.24.0.

[12] A. Subramanian, P. Tamayo, V. K. Mootha, S. Mukherjee, B. L. Ebert, M. A. Gillette, A. Paulovich, S. L. Pomeroy, T. R. Golub, E. S. Lander, and J. P. Mesirov. Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. *Proceedings of the National Academy of Sciences of the United States of America*, 102(43):15545–50, Oct. 2005.

[13] 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.