Package 'CePa'

October 8, 2024

Type Package

Title Centrality-Based Pathway Enrichment

Version 0.8.1

Date 2024-10-07

Depends R (>= 3.6.0)

Imports igraph (>= 0.6), stats, graphics, methods, grDevices, parallel, Rgraphviz, graph

Description It aims to find significant pathways through network topology information. It has several advantages compared with current pathway enrichment tools. First, pathway node instead of single gene is taken as the basic unit when analysing networks to meet the fact that genes must be constructed into complexes to hold normal functions. Second, multiple network centrality measures are applied simultaneously to measure importance of nodes from different aspects to make a full view on the biological system. CePa extends standard pathway enrichment methods, which include both over-representation analysis procedure and gene-set analysis procedure. <doi:10.1093/bioinformatics/btt008>.

LazyLoad yes

URL https://github.com/jokergoo/CePa

License GPL (>= 2)

NeedsCompilation no

Author Zuguang Gu [aut, cre] (<https://orcid.org/0000-0002-7395-8709>)

Maintainer Zuguang Gu <z.gu@dkfz.de>

Repository CRAN

Date/Publication 2024-10-08 10:10:27 UTC

Contents

`CePa-package` *Centrality-based pathway enrichment*

Description

Centrality-based pathway enrichment

Details

Gene set enrichment analysis is broadly used in microarray data analysis aimed to find which biological functions are affected by a group of related genes behind the massive information. A lot of methods have been developed under the framework of over-represented analysis (ORA) such as `GOstats` and `GSEABase`. 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 work. 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.

The core function of the package is `cepa.all`. There is also a parallel version `cepa.all.parallel`. User can refer to the vignette to find how to use it (`vignette("CePa")`).

Author(s)

Zuguang Gu <z.gu@dkfz.de>

References

Gu Z, Liu J, Cao K, Zhang J, Wang J. Centrality-based pathway enrichment: a systematic approach for finding significant pathways dominated by key genes. BMC Syst Biol. 2012 Jun 6;6(1):56.

Examples

```
## Not run:

# load the pathway database
data(PID.db)

# if you only have a differential gene list or other genes of interest
# in form of a list, you can apply the centrality-extended ORA method
res = cepa.all(dif = dif, bk = bk, pc = PID.db$NCI)
# in the above code, dif is your differential gene list, bk is your background
# gene list which always be whole set of genes on a certain microarray. If you
# do not have a background gene list, do not set it and the function would use
```

```
# the whole human genome genes as default. pc is the pathway catalogue which in
# this example is the NCI catalogue gathered from PID database.

# after about 20 min, you can obtain a detailed report of the analysis
report(res)

# if you have a complete gene expression data, you can apply the centrality-extended
# GSA methods
res = cepa.all(mat = mat, label = label, pc = PID.db$NCI)
# mat is your expression value matrix, label is the design of the microarray experiment.
# By default, we use absolute value of t-value as the statistic for each gene and
# use the mean value of genes' statistics as the pathway statistic.

# after about 50 min, you can obtain a detailed report of the analysis
report(res)


## End(Not run)
```

---

cepa *Apply CePa algorithm on a single pathway*

Description

Apply CePa algorithm on a single pathway

Usage

```
cepa(dif = NULL, bk = NULL, mat = NULL, label = NULL, pc, pathway = NULL,
    id = NULL, cen = "equal.weight",
    cen.name = if(is.function(cen)) deparse(substitute(cen))
    else if(mode(cen) == "name") deparse(cen) else cen,
    nlevel = "tvalue_abs", plevel = "mean", iter = 1000)
```

Arguments

dif	differential gene list
bk	background gene list. If background gene list are not specified, use whole human genes
mat	expression matrix in which rows are genes and columns are samples
label	a sampleLabel object identify the design of the microarray experiment
pc	a pathway.catalogue object storing information of pathways
pathway	an igraph object or edge list
id	identify which pathway should be analysis in the pathway catalogue
cen	centrality measuments, it can ce a string, or function has been quote
cen.name	centrality measurement names. This argument should be set if the cen is a function.

nlevel	node level transformation, should be one of "tvalue", "tvalue_sq", "tvalue_abs". Also self-defined functions are allowed, see cepa.univariate for detail.
plevel	pathway level transformation, should be one of "max", "min", "median", "sum", "mean", "rank". Also, self-defined functions are allowed, see cepa.univariate for detail.
iter	number of simulations

Details

The function is a wrapper of cepa.ora and cepa.univariate. Selection of which function depends on the arguments specified.

If dif, bk, pc, pathway, id, cen, cen.name and iter are specified, the arguments are passed to cepa.ora. The centrality-extension of over-representation analysis (ORA) will be applied on the list of differential genes.

If mat, label, pc, pathway, id, cen, cen.name, nlevel, plevel and iter are specified, the arguments are passed to cepa.univariate. The centrality-extension of gene-set analysis (GSA) will be applied on the whole gene expressions.

This function is always called by cepa.all. But you can still use it if you want to analysis a single pathway under a specific centrality.

Value

A cepa class object

Author(s)

Zuguang Gu <z.gu@dkfz.de>

See Also

cepa.all

Examples

```
## Not run:

data(PID.db)

# ORA extension
data(gene.list)
# will spend about 20 min
res.ora = cepa(dif = gene.list$dif, bk = gene.list$bk, pc = PID.db$NCI, id = 2)

# GSA extension
# P53_symbol.gct and P53_cls can be downloaded from
# http://mcube.nju.edu.cn/jwang/lab/soft/cepa/
eset = read.gct("P53_symbol.gct")
label = read.cls("P53.cls", treatment="MUT", control="WT")
# will take about 45 min
```

```
res.gsa = cepa(mat = eset, label = label, pc = PID.db$NCI, id = 2)

## End(Not run)
```

`cepa.all` *Apply CePa algorithm on a list of pathways under multiple centralities*

Description

Apply CePa algorithm on a list of pathways under multiple centralities

Usage

```
cepa.all(dif = NULL, bk = NULL, mat = NULL, label = NULL, pc, cen = default.centralities,
   cen.name = sapply(cen, function(x) ifelse(mode(x) == "name", deparse(x), x)),
    nlevel = "tvalue_abs", plevel = "mean", iter = 1000)
```

Arguments

`dif`	differential gene list
`bk`	background gene list. If background gene list are not specified, use whole human genes
`mat`	expression matrix in which rows are genes and columns are samples
`label`	a `sampleLabel` object identify the design of the microarray experiment
`pc`	a `pathway.catalogue` object storing information of pathways
`cen`	centrality measuments, it can ce a string, or a function
`cen.name`	centrality measurement names. By default it is parsed from `cen` argument
`nlevel`	node level transformation, should be one of "tvalue", "tvalue_sq", "tvalue_abs". Also self-defined functions are allowed, see `cepa.univariate.all` for detail.
`plevel`	pathway level transformation, should be one of "max", "min", "median", "sum", "mean", "rank". Also, self-defined functions are allowed, see `cepa.univariate.all` for detail.
`iter`	number of simulations

Details

All the calculation can be achieved by this function. The function is wrapper of both ORA extension and GSA extension. It chooses corresponding procedure according to the arguments specified. If the arguments contain gene lists, then the calculation is sent to functions doing ORA extension. While if the arguments contain an expression matrix and a phenotype label, the GSA extension is evoked.

The function is a wrapper of `cepa.ora.all` and `cepa.univariate.all`.

This is the core function of the package. User can refer to the vignette to find how to use it (`vignette("CePa")`).

If `dif`, `bk`, `pc`, `cen`, `cen.name` and `iter` are specified, the arguments are passed to `cepa.ora.all`. The centrality-extension of over-representation analysis (ORA) will be applied on the list of differential genes.

If `mat`, `label`, `pc`, `cen`, `cen.name`, `nlevel`, `plevel` and `iter` are specified, the arguments are passed to `cepa.univariate.all`. The centrality-extension of gene-set analysis (GSA) will be applied on the whole gene expressions.

There is a parallel version of the function: `cepa.all.parallel`.

Value

A `cepa.all` class object

Author(s)

Zuguang Gu <z.gu@dkfz.de>

References

Gu Z, Liu J, Cao K, Zhang J, Wang J. Centrality-based pathway enrichment: a systematic approach for finding significant pathways dominated by key genes. BMC Syst Biol. 2012 Jun 6;6(1):56.

See Also

`cepa`, `cepa.ora.all`, `cepa.univariate.all`, `cepa.all.parallel`

Examples

```
## Not run:

data(PID.db)

# ORA extension
data(gene.list)
# will spend about 20 min
res.ora = cepa.all(dif = gene.list$dif, bk = gene.list$bk, pc = PID.db$NCI)

# GSA extension
# P53_symbol.gct and P53_cls can be downloaded from
# http://mcube.nju.edu.cn/jwang/lab/soft/cepa/
eset = read.gct("http://mcube.nju.edu.cn/jwang/lab/soft/cepa/P53_symbol.gct")
label = read.cls("http://mcube.nju.edu.cn/jwang/lab/soft/cepa/P53.cls",
    treatment="MUT", control="WT")
# will spend about 45 min
res.gsa = cepa.all(mat = eset, label = label, pc = PID.db$NCI)

## End(Not run)
```

`cepa.all.parallel` *use CePa package through parallel computing*

Description

use CePa package through parallel computing

Usage

```
cepa.all.parallel(dif = NULL, bk = NULL, mat = NULL, label = NULL,
    pc, cen = default.centralities,
   cen.name = sapply(cen, function(x) ifelse(mode(x) == "name", deparse(x), x)),
    nlevel = "tvalue_abs", plevel = "mean", iter = 1000, ncores = 2)
```

Arguments

`dif`	differential gene list
`bk`	background gene list. If background gene list are not specified, use whole human genes
`mat`	expression matrix in which rows are genes and columns are samples
`label`	a `sampleLabel` object identify the design of the microarray experiment
`pc`	a `pathway.catalogue` object storing information of pathways
`cen`	centrality measuments, it can ce a string, or a function
`cen.name`	centrality measurement names. By default it is parsed from `cen` argument
`nlevel`	node level transformation, should be one of "tvalue", "tvalue_sq", "tvalue_abs". Also self-defined functions are allowed, see `cepa.univariate.all` for detail.
`plevel`	pathway level transformation, should be one of "max", "min", "median", "sum", "mean", "rank". Also, self-defined functions are allowed, see `cepa.univariate.all` for detail.
`iter`	number of simulations
`ncores`	number of cores for parallel computing

Details

The function divides the pathway list into several parts and each part is sent to a core for parallel computing.

The package for parallel computing is `snow`.

Note: there may be warnings saying connections not closed. In fact I have closed connections after the parallel computing is done. I don't know why this happens. Maybe you breaked the computing ahead manually. However it does not matter unless you have obsessive compulsive disorder.

Value

A `cepa.all` class object

Author(s)

Zuguang Gu <z.gu@dkfz.de>

References

Gu Z, Liu J, Cao K, Zhang J, Wang J. Centrality-based pathway enrichment: a systematic approach for finding significant pathways dominated by key genes. BMC Syst Biol. 2012 Jun 6;6(1):56.

See Also

cepa.all

Examples

```
## Not run:
data(PID.db)
# ORA extension
data(gene.list)
res.ora = cepa.all.parallel(dif = gene.list$dif, bk = gene.list$bk, pc = PID.db$NCI, ncores = 4)
# GSA extension
# P53_symbol.gct and P53_cls can be downloaded from
# http://mcube.nju.edu.cn/jwang/lab/soft/cepa/
eset = read.gct("http://mcube.nju.edu.cn/jwang/lab/soft/cepa/P53_symbol.gct")
label = read.cls("http://mcube.nju.edu.cn/jwang/lab/soft/cepa/P53.cls",
    treatment="MUT", control="WT")
res.gsa = cepa.all.parallel(mat = eset, label = label, pc = PID.db$NCI, ncores = 4)

## End(Not run)
```

---

cepa.ora *Apply centrality-extended ORA on a single pathway*

Description

Apply centrality-extended ORA on a single pathway

Usage

```
cepa.ora(dif, pc, bk = NULL, pathway = NULL, id = NULL, cen = "equal.weight",
    cen.name = if(is.function(cen)) deparse(substitute(cen))
    else if(mode(cen) == "name") deparse(cen)
    else cen,
    iter = 1000)
```

Arguments

dif	differential gene list
pc	a pathway.catalogue class object
bk	background gene list. If background gene list are not specified, use whole human genes
pathway	igraph object or edge list
id	identify which pathway in the catalogue
cen	centrality measuments, it can ce a string, function, or function that has been quoted
cen.name	centrality measurement names. This argument should be set if the cen is a function.
iter	number of simulations

Details

The function is always called by cepa.ora.all. But you can still use it if you realy want to analysis just one pathway under one centrality.

Value

A cepa class object

Author(s)

Zuguang Gu <z.gu@dkfz.de>

See Also

cepa.all

Examples

```
## Not run:
data(PID.db)

# ORA extension
data(gene.list)
# will spend about 20 min
res.ora = cepa(dif = gene.list$dif, bk = gene.list$bk, pc = PID.db$NCI, id = 2)

## End(Not run)
```

`cepa.ora.all` *Apply centrality-extented ORA on a list of pathways*

Description

Apply centrality-extented ORA on a list of pathways

Usage

```
cepa.ora.all(dif, pc, bk = NULL, cen = default.centralities,
   cen.name = sapply(cen, function(x) ifelse(mode(x) == "name", deparse(x), x)),
    iter = 1000)
```

Arguments

`dif`	differential gene list
`pc`	a `pathway.catalogue` class object
`bk`	background gene list. If background gene list are not specified, use whole human genes
`cen`	centrality measuments, it can ce a string, or a function
`cen.name`	centrality measurement names. By default it is parsed from `cen` argument
`iter`	number of simulations

Details

The traditional over-representation analysis (ORA) to find significant pathways uses a 2x2 contingency table to test the independency of genes belonging to a functional category and these genes being differentially expressed, usually by Fisher's exact test. The ORA only consider the number of genes and the function extend traditional ORA with network centralities.

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. If users use the `PID.db` data, all genes should be formatted in gene symbol.

If the centrality measurement is set as a string, only pre-defined "equal.weight", "in.degree", "out.degree", "degree", "betweenness", "in.reach", "out.reach", "reach", "in.spread", "out.spread" and "spread" are allowed. More centrality measurements can be used by setting it as a function (such as closeness, cluster coefficient). In the function, we recommand users choose at least two centrality measurements. The default centralities are "equal.weight", "in.degree", "out.degree", "betweenness", "in.reach" and "out.reach".

However, in most circumstance, the function is called by `cepa.all`.

Value

A `cepa.all` class object

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
## Not run:
data(PID.db)
# ORA extension
data(gene.list)
# will spend about 20 min
res.ora = cepa.ora.all(dif = gene.list$dif, bk = gene.list$bk, pc = PID.db$NCI)

## End(Not run)
```

`cepa.univariate` *Apply centrality-extended GSA on a single pathway*

Description

Apply centrality-extended GSA on a single pathway

Usage

```
cepa.univariate(mat, label, pc, pathway = NULL, id = NULL, cen = "equal.weight",
    cen.name = if(is.function(cen)) deparse(substitute(cen))
    else if(mode(cen) == "name") deparse(cen)
    else cen,
    iter = 1000, nlevel = "tvalue_abs", plevel = "mean",
    node.level.from.expr = NULL, node.level.t.value = NULL,
    r.node.level.from.expr = NULL)
```

Arguments

`mat`	expression matrix in which rows are genes and columns are samples
`label`	a `sampleLabel` object identify the design of the microarray experiment
`pc`	a `pathway.catalogue` object storing information of pathways
`pathway`	`igraph` object or edge list
`id`	identify the number of the pathway in the catalogue
`cen`	centrality measuments, it can ce a string, or function has been quote
`cen.name`	centrality measurement names
`nlevel`	node level transformation, should be one of "tvalue", "tvalue_sq", "tvalue_abs". Also self-defined functions are allowed, see `cepa.univariate.all` for detail.
`plevel`	pathway level transformation, should be one of "max", "min", "median", "sum", "mean", "rank". Also, self-defined functions are allowed, see `cepa.univariate.all` for detail.

`node.level.from.expr`
: for simplicity of computing

`node.level.t.value`
: for simplicity of computing

`r.node.level.from.expr`
: for simplicity of computing

`iter`
: number of simulations

Details

The function is always called by `cepa.univariate.all`. But you can still use it if you realy want to analysis just one pathway under one centrality.

Value

A `cepa` class object

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
## Not run:

data(PID.db)

# GSA extension
# P53_symbol.gct and P53_cls can be downloaded from
# http://mcube.nju.edu.cn/jwang/lab/soft/cepa/
eset = read.gct("P53_symbol.gct")
label = read.cls("P53.cls", treatment="MUT", control="WT")
# will spend about 45 min
res.gsa = cepa.univariate(mat = eset, label = label, pc = PID.db$NCI, id = 2)

## End(Not run)
```

`cepa.univariate.all` *Apply centrality-extented GSA on a list of pathways*

Description

Apply centrality-extented GSA on a list of pathways

Usage

```
cepa.univariate.all(mat, label, pc, cen = default.centralities,
    cen.name = sapply(cen, function(x) ifelse(mode(x) == "name", deparse(x), x)),
     nlevel = "tvalue_abs", plevel = "mean", iter = 1000)
```

Arguments

mat	expression matrix in which rows are genes and columns are samples
label	a sampleLabel object identify the design of the microarray experiment
pc	a pathway.catalogue object storing information of pathways
cen	centrality measuments, it can ce a string, or a function
cen.name	centrality measurement names. By default it is parsed from cen argument
nlevel	node level transformation, should be one of "tvalue", "tvalue_sq", "tvalue_abs". Also self-defined functions are allowed
plevel	pathway level transformation, should be one of "max", "min", "median", "sum", "mean", "rank". Also, self-defined functions are allowed
iter	number of simulations

Details

The traditional gene-set analysis (GSA) to find significant pathways uses the whole expression matrix. GSA methods are implemented via either a univariate or a multivariate procedure. In univariate analysis, node 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. 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 or MANOVA models. The function implement univariate procedure of GSA with network centralities.

If users use the PID.db data, all genes should be formatted in gene symbol.

If the centrality measurement is set as a string, only pre-defined "equal.weight", "in.degree", "out.degree", "degree", "betweenness", "in.reach", "out.reach", "reach", "in.spread", "out.spread" and "spread" are allowed. More centrality measurements can be used by setting it as a function (such as closeness, cluster coefficient). In the function, we recommand users choose at least two centrality measurements. Note that the self-defined function should only contain one argument which is an igraph object. The default centralities are "equal.weight", "in.degree", "out.degree", "betweenness", "in.reach" and "out.reach".

The node level statistic can be self-defined. The self-defined function should contain two arguments: a vector for expression value in treatment class and a vector for expression value in control class.

The pathway level statistic can be self-defined. The self-defined function should only contain one argument: the vector of node-level statistic.

However, in most circumstance, the function is called by cepa.all.

We are sorry that only the univariate procedures in GSA are extended. We are still trying to figure out the extension for the multivariate procedures in GSA.

Value

A cepa.all class object

Author(s)

Zuguang Gu <z.gu@dkfz.de>

See Also

cepa

Examples

```
## Not run:
data(PID.db)
# GSA extension
# P53_symbol.gct and P53.cls can be downloaded from
# http://mcube.nju.edu.cn/jwang/lab/soft/cepa/
eset = read.gct("http://mcube.nju.edu.cn/jwang/lab/soft/cepa/P53_symbol.gct")
label = read.cls("http://mcube.nju.edu.cn/jwang/lab/soft/cepa/P53.cls",
    treatment="MUT", control="WT")
# will spend about 45 min
res.gsa = cepa.univariate.all(mat = eset, label = label, pc = PID.db$NCI)

## End(Not run)
```

---

gene.list *Differential gene list and background gene list*

Description

Differential gene list and background gene list

Usage

```
data(gene.list)
```

Details

Differential gene list and background gene list was extracted from microarray data from GEO database. The accession number for the data set is GSE22058. The t-test was applied to find differentially expressed genes. Top 2000 genes were selected as the gene list.

Value

A list containing two componets:

`bk` background gene list, gene symbol

`dif` differentially expressed gene list, gene symbol

Examples

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

generate.pathway *Generate igraph object from edge list*

Description

Generate igraph object from edge list

Usage

```
generate.pathway(el)
```

Arguments

el	edge list, matrix with two columns. The first column is the input node and the second column is the output node.

Details

The function is a wrapper of graph.edgelist and it generates a directed graph.

In the function, repeated edged for two nodes will be eliminated.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

See Also

cepa, graph.edgelist

Examples

```
edgelist = rbind(c("a", "b"), c("a", "b"), c("a", "c"))
g = generate.pathway(edgelist)
```

get.cepa *get single cepa object from cepa.all object*

Description

get single cepa object from cepa.all object

Usage

```
get.cepa(x, id = NULL, cen = 1)
```

Arguments

x	a cepa.all object
id	index or the name of the pathway
cen	index or the name of the centrality

Details

The 'cepa.all object contains the result for pathways under several centrality measurements. In cepa.all object, each pathway under a specific centrality is a single cepa object. The get.cepa function is used to get the cepa object from the cepa.all object.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

See Also

cepa, cepa.all

Examples

```
## Not run:
data(PID.db)

# ORA extension
data(gene.list)
# will spend about 20 min
res.ora = cepa.all(dif = gene.list$dif, bk = gene.list$bk, pc = PID.db$NCI)
ora = get.cepa(res.ora, id = 5, cen = 3)

# GSA extension
# P53_symbol.gct and P53_cls can be downloaded from
# http://mcube.nju.edu.cn/jwang/lab/soft/cepa/
eset = read.gct("P53_symbol.gct")
label = read.cls("P53.cls", treatment="MUT", control="WT")
# will spend about 45 min
res.gsa = cepa.all(mat = eset, label = label, pc = PID.db$NCI)
gsa = get.cepa(res.gsa, id = 5, cen = 3)

## End(Not run)
```

---

p.table *Table of p-values of pathways*

Description

Table of p-values of pathways

Usage

```
p.table(x, adj.method = NA, cutoff = ifelse(adj.method == "none", 0.01, 0.05))
```

Arguments

x	a cepa.all object
adj.method	methods in p.adjust, available methods are "holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr", "none"
cutoff	cutoff for significance

Details

Since the p-values for each pathway are calculated for several centralities, the whole p-values are represented as a table.

Also it can extract significant pathways only.

Value

A data matrix where rows are pathways and columns are centralities.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

See Also

cepa.all

Examples

```
## Not run:
data(PID.db)

# ORA extension
data(gene.list)
# will spend about 20 min
res.ora = cepa.all(dif = gene.list$dif, bk = gene.list$bk, pc = PID.db$NCI)
p.table(res.ora)
p.table(res.ora, adj.method = "BH")

# GSA extension
# P53_symbol.gct and P53_cls can be downloaded from
# http://mcube.nju.edu.cn/jwang/lab/soft/cepa/
eset = read.gct("P53_symbol.gct")
label = read.cls("P53.cls", treatment="MUT", control="WT")
# will spend about 45 min
res.gsa = cepa.all(mat = eset, label = label, pc = PID.db$NCI)
p.table(res.gsa)

## End(Not run)
```

pathway.nodes *names of the pathway nodes*

Description

names of the pathway nodes

Usage

```
pathway.nodes(pathway)
```

Arguments

pathway	an igraph object

Details

If nodes in the pathway have names, then it returns a vector of nodes names. If nodes in the pathway have no name, it just returns the index of nodes (start from 1, after igraph version 0.6).

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
interaction = rbind(c("a", "b"),
                    c("a", "c"))
g = generate.pathway(interaction)
pathway.nodes(g)
```

PID.db *pathway catalogues from Pathway Interaction Database(PID)*

Description

pathway catalogues from Pathway Interaction Database(PID)

Usage

```
data(PID.db)
```

Details

The pathway data is parsed from XML format file provided by PID FTP site.

There are four pathway catalogues which are NCI_Nature, BioCarta, KEGG and Reactome.

Each catalogue contains at least three members: pathway list (pathList), interaction list (interactionList) and mappings from node id to gene id (mapping). The pathway list contains a list of pathways in which each pathway is represented by a list of interaction id. The interactions can be queried from the interaction list by the interaction id. The interaction list data is represented as a data frame 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. In real biological pathways, a node in the pathway can be proteins, complex, families and none-gene nodes, so the mapping from node ids to gene ids is also provided. It must be noted that in this package, gene symbol is selected as the primary gene id, so if users apply the PID.db data, they should pay attension to the gene ids they transform.

Besides, in each catalogue, there also a node.name, node.type and version data. The node.name provides the custumed name for each node. The node.type provides the type for each node (e.g. it is a complex or compound). The version provides the version of the catalogue data.

Data has been updated to the lastest version by the day the package released (at 2012_07_19 09:34::20).

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.

Value

A list containing four component:

NCI NCI_Nature-curated pathway catalogue

BioCarta BioCarta pathway catalogue

KEGG KEGG pathway catalogue

Reactome Reactome pathway catalogue

Each pathway catalogue is a `pathway.catalogue` class object. Each pathway catalogue can be used directly in `cepa.all` and `cepa`

Examples

```
data(PID.db)
names(PID.db)
PID.db$NCI
plot(PID.db$NCI)
```

plot.cepa *Plot the cepa object*

Description

Plot the cepa object

Usage

```
## S3 method for class 'cepa'
plot(x, type = c("graph", "null"), ...)
```

Arguments

x	a cepa object
type	identify the type for the plot
...	arguments passed to plotGraph

Details

The function is wrapper of plotGraph and plotNull. If type is specified to "graph", the graph of the network will be plotted (see plotGraph for details). If type is specified to "null", the null distribution of the pathway score in the pathway will be plotted (see plotNull for details).

Value

if type is set to "graph", the function will return a igraph object or a graphML object of the pathway. Else it is NULL.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

See Also

cepa, plotNull, plotGraph

Examples

```
## Not run:

data(PID.db)

# ORA extension
data(gene.list)
# will spend about 20 min
res.ora = cepa(dif = gene.list$dif, bk = gene.list$bk, pc = PID.db$NCI, id = 2)
plot(res.ora)
```

```
plot(res.ora, type = "null")

# GSA extension
# P53_symbol.gct and P53_cls can be downloaded from
# http://mcube.nju.edu.cn/jwang/lab/soft/cepa/
eset = read.gct("P53_symbol.gct")
label = read.cls("P53.cls", treatment="MUT", control="WT")
# will spend about 45 min
res.gsa = cepa(mat = eset, label = label, pc = PID.db$NCI, id = 2)
plot(res.gsa, type = "null")

## End(Not run)
```

`plot.cepa.all` *plot the cepa.all object*

Description

plot the cepa.all object

Usage

```
## S3 method for class 'cepa.all'
plot(x, id = NULL, cen = 1, type = c("graph", "null"), tool = c("igraph", "Rgraphviz"),
    node.name = NULL, node.type = NULL,
    adj.method = "none", only.sig = FALSE,
    cutoff = ifelse(adj.method == "none", 0.01, 0.05), ...)
```

Arguments

`x`	a `cepa.all` object
`id`	index or the name for the pathway
`cen`	index or the name for the centrality
`type`	If the aim is to plot single pathway, then this argument is to identify the kind of the plotting.
`tool`	Use which tool to visualize the graph. Choices are 'igraph' and 'Rgraphviz'
`node.name`	node.name for each node
`node.type`	node.type for each node
`adj.method`	method of `p.adjust`, available methods are "holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr", "none"
`only.sig`	whether to show all pathways. If just show significant pathways, the names for each significant pathway will be draw.
`cutoff`	cutoff for significance
`...`	other arguments

Details

This function has two applications. First, it can draw heatmaps of p-values of all pathways under different centrality measurements. To do it, users should set `x`, `adj.method`, `only.sig`, `cutoff` arguments.

Second, it can draw figures of single pathway under specific centrality measurement. Under this circumstance, this function is just a wrapper of `plot.cepa`. To do it, users should set `x`, `id`, `cen`, `type`, `tool`, `node.name` and `node.type` arguments. The `id` and `cen` arguments are used to get single `cepa` object that sent to the plot function.

It must be noted that these two kinds of arguments should not be mixed.

There is also another popular method `qvalue` to adjust p-values. However, errors may occur when adjusting some kind of p-value list by `qvalue`. So `qvalue` was not implemented into CePa. But still users can override the default p.adjust to support qvalue by themselves, see the vignette.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

See Also

`cepa.all`

Examples

```
## Not run:
data(PID.db)

# ORA extension
data(gene.list)
# will spend about 20 min
res.ora = cepa.all(dif = gene.list$dif, bk = gene.list$bk, pc = PID.db$NCI)
plot(res.ora)
plot(res.ora, id = 3)
plot(res.ora, id = 3, type = "null")

# GSA extension
# P53_symbol.gct and P53_cls can be downloaded from
# http://mcube.nju.edu.cn/jwang/lab/soft/cepa/
eset = read.gct("P53_symbol.gct")
label = read.cls("P53.cls", treatment="MUT", control="WT")
# will spend about 45 min
res.gsa = cepa.all(mat = eset, label = label, pc = PID.db$NCI)
plot(res.gsa)
plot(res.gsa, id = 3, cen = 2)
plot(res.gsa, id = 3, cen = 2, type = "null")

## End(Not run)
```

`plot.pathway.catalogue` *plot pathway.catalogue object*

Description

plot pathway.catalogue object

Usage

```
## S3 method for class 'pathway.catalogue'
plot(x, ...)
```

Arguments

`x`	a `pathway.catalogue` object
`...`	other arguments

Details

There are three fugures: 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.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

See Also

`set.pathway.catalogue`

Examples

```
data(PID.db)
NCI = PID.db$NCI
plot(NCI)
```

plotGraph *Plot graph for the pathway network*

Description

Plot graph for the pathway network

Usage

```
plotGraph(x, node.name = NULL, node.type = NULL, draw = TRUE,
    tool = c("igraph", "Rgraphviz"), graph.node.max.size = 20,
    graph.node.min.size = 3, graph.layout.method = NULL)
```

Arguments

x	a cepa object
node.name	node.name for each node
node.type	node.type for each node
draw	Whether to draw the graph
tool	Use which tool to visualize the graph. Choices are 'igraph' and 'Rgraphviz'
graph.node.max.size	max size of the node in the graph
graph.node.min.size	min size of the node in the graph
graph.layout.method	function of the layout method. For the list of available methods, see layout

Details

Graph view of the pathway where the size of node is proportional to centrality value of the node.

By default, the layout for the pathway tree-like. If the number of pathway nodes is large, the layout would be a random layout.

Two packages can be selected to visualize the graph: igraph and Rgraphviz. Default package is igraph (in fact, this package just uses the data generated from the layout function in igraph package, which are the coordinate of nodes and edges. And the I re-wrote the plotting function to generate the graph). From my personal view, Rgraphviz package generated more beautiful graphs.

If the tool is set as igraph, the function returns a igraph object. And if the tool is set as Rgraphviz, the function returns a graphAM class object. So if users don't satisfy, they can draw graphs of the network with their own settings.

The function is always called through plot.cepa.all and plot.cepa.

Value

A igraph object of the pathway

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
## Not run:
data(PID.db)
# ORA extension
data(gene.list)
# will spend about 20 min
res.ora = cepa.all(dif = gene.list$dif, bk = gene.list$bk, pc = PID.db$NCI)
ora = get.cepa(res.ora, id = 5, cen = 3)
plotGraph(ora)
# GSA extension
# P53_symbol.gct and P53_cls can be downloaded from
# http://mcube.nju.edu.cn/jwang/lab/soft/cepa/
eset = read.gct("P53_symbol.gct")
label = read.cls("P53.cls", treatment="MUT", control="WT")
# will spend about 45 min
res.gsa = cepa.all(mat = eset, label = label, pc = PID.db$NCI)
gsa = get.cepa(res.gsa, id = 5, cen = 3)
plotGraph(gsa)

## End(Not run)
```

---

plotNull *Plot the null distribution of the pathway score*

Description

Plot the null distribution of the pathway score

Usage

```
plotNull(x)
```

Arguments

x	a cepa object

Details

There are two figures in the plotting.

- A) Distribution of node score in the pathway under simulation. Since a pathway contains a list of nodes. The distribution of node score for the pathway in each simulation is measures by maximum value, the 75th quantile, median value and minimum value. The distribution of node score for the pathway in the real data is highlighted.

- B) Histogram of simulated pathway scores.

The function is always called through plot.cepa.all and plot.cepa.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

See Also

cepa, plot.cepa

Examples

```
## Not run:
data(PID.db)

# ORA extension
data(gene.list)
# will spend about 20 min
res.ora = cepa.all(dif = gene.list$dif, bk = gene.list$bk, pc = PID.db$NCI)
ora = get.cepa(res.ora, id = 5, cen = 3)
plotNull(ora)

# GSA extension
# P53_symbol.gct and P53_cls can be downloaded from
# http://mcube.nju.edu.cn/jwang/lab/soft/cepa/
eset = read.gct("P53_symbol.gct")
label = read.cls("P53.cls", treatment="MUT", control="WT")
# will spend about 45 min
res.gsa = cepa.all(mat = eset, label = label, pc = PID.db$NCI)
gsa = get.cepa(res.gsa, id = 5, cen = 3)
plotNull(gsa)

## End(Not run)
```

---

print.cepa *print the cepa object*

Description

print the cepa object

Usage

```
## S3 method for class 'cepa'
print(x, ...)
```

Arguments

x	a cepa object
...	other arguments

Details

The function print procedure of the analysis, the centrality and the p-value for the pathway.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

See Also

cepa

Examples

```
## Not run:

data(PID.db)

# ORA extension
data(gene.list)
# will spend about 20 min
res.ora = cepa(dif = gene.list$dif, bk = gene.list$bk, pc = PID.db$NCI, id = 2)
res.ora

# GSA extension
# P53_symbol.gct and P53_cls can be downloaded from
# http://mcube.nju.edu.cn/jwang/lab/soft/cepa/
eset = read.gct("P53_symbol.gct")
label = read.cls("P53.cls", treatment="MUT", control="WT")
# will spend about 45 min
res.gsa = cepa(mat = eset, label = label, pc = PID.db$NCI, id = 2)
res.gsa

## End(Not run)
```

---

print.cepa.all *print the cepa.all object*

Description

print the cepa.all object

Usage

```
## S3 method for class 'cepa.all'
print(x, ...)
```

Arguments

x	a cepa.all object
...	other arguments

Details

The function print the number of significant pathways under various centrality measures at p-value <= 0.01.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

See Also

cepa.all

Examples

```
## Not run:
data(PID.db)

# ORA extension
data(gene.list)
# will spend about 20 min
res.ora = cepa.all(dif = gene.list$dif, bk = gene.list$bk, pc = PID.db$NCI)
res.ora

# GSA extension
# P53_symbol.gct and P53_cls can be downloaded from
# http://mcube.nju.edu.cn/jwang/lab/soft/cepa/
eset = read.gct("P53_symbol.gct")
label = read.cls("P53.cls", treatment="MUT", control="WT")
# will spend about 45 min
res.gsa = cepa.all(mat = eset, label = label, pc = PID.db$NCI)
res.gsa

## End(Not run)
```

---

print.pathway.catalogue *print pathway.catalogue object*

Description

print pathway.catalogue object

Usage

```
## S3 method for class 'pathway.catalogue'
print(x, ...)
```

Arguments

x	a pathway.catalogue object
...	other arguments

Details

Simply print the number of pathways in the catalogue

Author(s)

Zuguang Gu <z.gu@dkfz.de>

See Also

set.pathway.catalogue

Examples

```
data(PID.db)
NCI = PID.db$NCI
NCI
```

---

radiality *Calculate radiality centrality*

Description

Calculate radiality centrality

Usage

```
radiality(graph, mode = c("all", "in", "out"))
```

Arguments

graph	an igraph object
mode	mode of the centrality

Details

The radiality is defined as sum(d_G + 1 - d(v, w))/(n - 1). where d(w, v) is the length of the shortest path from node w to node v, d_G is the diameter of the network, n is the size of the network.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
require(igraph)
pathway = barabasi.game(200)
radiality(pathway)
```

`reach` *Calculate largest reach centrality*

Description

Calculate largest reach centrality

Usage

```
reach(graph, weights=E(graph)$weight, mode=c("all", "in", "out"))
```

Arguments

`graph`	an `igraph` object
`mode`	mode of the centrality
`weights`	If the edges in the graph have weight, then by default, the weight is used to calculate the length of the shortest path. Set it to NULL to supress the weight.

Details

The largest reach centrality measures how far a node can send or receive the information in the network. It is defined as the largest length of the shortest path from all the other nodes in the network.

Examples

```
# There is no example
NULL
```

`read.cls` *Read CLS file which stores the phenotype data*

Description

Read CLS file which stores the phenotype data

Usage

```
read.cls(file, treatment, control)
```

Arguments

`file`	cls file path
`treatment`	string of treatment label in cls file
`control`	string of control label in cls file

Details

The CLS file format defines the phenotype data of microarray experiments. The first line is the number of samples, number of classes and the third number always be 1. These three numbers are seperated by spaces or tabs. The second line begins with #. The next two strings usually are the label of the phenotype. The third line is the label of each samples where same label represents the same class.

The first and the second line is ignored by this function and class labels are taken from the factor of the vector parsed from the third line.

Value

A `sampleLabel` class object

Author(s)

Zuguang Gu <z.gu@dkfz.de>

See Also

`read.gct`, `sampleLabel`

Examples

```
## Not run:
# P53.cls can be downloaded from
# http://mcube.nju.edu.cn/jwang/lab/soft/cepa/
label = read.cls("http://mcube.nju.edu.cn/jwang/lab/soft/cepa/P53.cls",
    treatment="MUT", control="WT")

## End(Not run)
```

read.gct *Read GCT format file which stores the expression values*

Description

Read GCT format file which stores the expression values

Usage

```
read.gct(file)
```

Arguments

file	gct file path

Details

The GCT format is a tab delimited file format that stores the expression value matrix. The first line of the file is the version number which always be #1.2. The second line is the number of the size of genes and samples, seperated by space, usually for the initiation of reading the expression matrix. The third line contains a list of identifiers for the samples associated with each of the columns in the remainder of the file. From the fourth line will be the expression value of each gene.

GCT file is used together with CLS file.

Value

A matrix of the expression values, with rows correponding to genes and cols to samples.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
## Not run:
# expression data stored in a gct format file
# P53_symbol.gct can be downloaded from
# http://mcube.nju.edu.cn/jwang/lab/soft/cepa/
eset = read.gct("http://mcube.nju.edu.cn/jwang/lab/soft/cepa/P53_symbol.gct")
head(eset)

## End(Not run)
```

`report` *Generate report for CePa analysis*

Description

Generate report for CePa analysis

Usage

```
report(x, adj.method = "none", cutoff = ifelse(adj.method == "none", 0.01, 0.05),
    template.file = system.file(package = "CePa", "extdata", "cepa.template"),
    only.sig = TRUE, dir.path = NULL, ...)
```

Arguments

`x`	a `cepa.all` object
`adj.method`	methods in `p.adjust`, available methods are "holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr", "none"
`cutoff`	cutoff for significance
`template.file`	path of the template file
`only.sig`	whether to generate detailed report for every pathway. If it is set to FALSE, the page for every pathway under every centrality would be generated (there would be so many images!).
`dir.path`	dir name
`...`	other arguments

Details

The report is in HTML format that you can view it in you web browser. Networks for pathways can be visualized interactively (by using Cytoscape Web, in which you can drag the network, zoom in and zoom out the network). To load Flash Player successful in you browser, you may need to set the Flash security settings on your machine.

The report would locate at the current working directory. View the report by clicking `index.html` in the report directory.

There is also another popular method qvalue to adjust p-values. Turn to `plot.cepa.all` to find out how to use qvalue.

Source

https://cytoscapeweb.cytoscape.org/

Author(s)

Zuguang Gu <z.gu@dkfz.de>

See Also

cepa.all

Examples

```
## Not run:
data(PID.db)

# ORA extension
data(gene.list)
# will spend about 20 min
res.ora = cepa.all(dif = gene.list$dif, bk = gene.list$bk, pc = PID.db$NCI)
report(res.ora)

## End(Not run)
```

---

sampleLabel *Generate data structure of sample labels*

Description

Generate data structure of sample labels

Usage

```
sampleLabel(label, treatment, control)
```

Arguments

label	sample label vector
treatment	treatment label
control	control label

Details

Since sample label will not be modified in the analysis, this function is used to integrate all the label information in one single data object.

Value

A sampleLabel class object

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
sampleLabel(c("A", "B", "B", "A", "A", "A", "B", "B"), treatment = "A", control = "B")
```

`set.pathway.catalogue` *store pathway data and pre-processing*

Description

store pathway data and pre-processing

Usage

```
set.pathway.catalogue(pathList, interactionList, mapping,
    min.node = 5, max.node = 500, min.gene = min.node, max.gene = max.node, ...)
```

Arguments

`pathList`	list of pathways
`interactionList`	list of interactions
`mapping`	a data frame or matrix providing mappings from gene id to pathway node id. The first column is node id and the second column is gene id.
`min.node`	minimum number of connected nodes in each pathway
`max.node`	maximum number of connected nodes in each pathway
`min.gene`	minimum number of genes in each pathway
`max.gene`	maximum number of genes in each pathway
`...`	other arguments, should have names, these data will be stored as a list member in the returned value from the function

Details

The pathway data will not be changed in the analysis, so the pathway data is integrated in one single data object by this function. Also, the function will do a little preprocess of the pathway data.

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.

Besides the pathList, interactionList and mapping arguments, more arguments can be added to the function. These new data will be stored as the member of the list that returned by the function. E.g., in the `PID.db` data, each catalogue is a `pathway.catalogue` object. Except the pathList, interactionList and mapping arguments, there are also node.name, node.type and version arguments.

The summary can be visualized by `plot.pathway.catalogue`.

Value

A pathway.catalogue class object

Author(s)

Zuguang Gu <z.gu@dkfz.de>

See Also

cepa.all

Examples

```
## Not run:
data(PID.db)
catalogue = set.pathway.catalogue(pathList = PID.db$NCI$pathList[1:20],
            interactionList = PID.db$NCI$intertionList, mapping = PID.db$NCI$mapping)

## End(Not run)
```

---

spread *Calculate radiality centrality*

Description

Calculate radiality centrality

Usage

```
spread(graph, mode = c("all", "in", "out"),
    weights = E(graph)$weight, f = function(x) 1/x)
```

Arguments

graph	an igraph object
mode	mode of the centrality
weights	If edges in the graph have weight, then by default, the weight is used to calculate the length of the shortest path. Set it to NULL to supress the weight
f	function for the weaken rate

Details

The spread centrality measures how wide the node can send or receive the information in the network. Like the water wave, the effect would be weakened with the increase of the distance to other nodes.

If the weaken function is defined as 1/x, then the spread centrality is calculated as sum(1/d(w, v)) where d(w, v) is the length of the shortest path of node w and node v.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

See Also

reach, radiality

Examples

```
require(igraph)
pathway = barabasi.game(200)
spread(pathway)
```

Index