dEnricher

July 21, 2015

dEnricher	*Function to conduct enrichment analysis given the input data and the ontology in query*

Description

dEnricher is supposed to conduct enrichment analysis given the input data and the ontology in query. It returns an object of class "eTerm". Enrichment analysis is based on either Fisher's exact test or Hypergeometric test. The test can respect the hierarchy of the ontology.

Usage

```
dEnricher(data, identity = c("symbol", "entrez"),
check.symbol.identity = FALSE, genome = c("Hs", "Mm", "Rn", "Gg", "Ce",
"Dm", "Da", "At"), ontology = c("GOBP", "GOMF", "GOCC", "PS", "PS2",
"SF",
"DO", "HPPA", "HPMI", "HPCM", "HPMA", "MP", "MsigdbH", "MsigdbC1",
"MsigdbC2CGP", "MsigdbC2CP", "MsigdbC2KEGG", "MsigdbC2REACTOME",
"MsigdbC2BIOCARTA", "MsigdbC3TFT", "MsigdbC3MIR", "MsigdbC4CGN",
"MsigdbC4CM",
"MsigdbC5BP", "MsigdbC5MF", "MsigdbC5CC", "MsigdbC6", "MsigdbC7",
"DGIdb"),
sizeRange = c(10, 1000), min.overlap = 3, which_distance = NULL,
test = c("HypergeoTest", "FisherTest", "BinomialTest"),
p.adjust.method = c("BH", "BY", "bonferroni", "holm", "hochberg",
"hommel"),
ontology.algorithm = c("none", "pc", "elim", "lea"), elim.pvalue =
0.01,
lea.depth = 2, verbose = T,
RData.location = "http://supfam.org/dnet/RData/1.0.7")
```

Arguments

data	an input vector. It contains either Entrez Gene ID or Symbol
identity	the type of gene identity (i.e. row names of input data), either "symbol" for gene symbols (by default) or "entrez" for Entrez Gene ID. The option "symbol" is preferred as it is relatively stable from one update to another; also it is possible to search against synonyms (see the next parameter)

`check.symbol.identity`
: logical to indicate whether synonyms will be searched against when gene symbols cannot be matched. By default, it sets to FALSE since it may take a while to do such check using all possible synoyms

`genome`
: the genome identity. It can be one of "Hs" for human, "Mm" for mouse, "Rn" for rat, "Gg" for chicken, "Ce" for c.elegans, "Dm" for fruitfly, "Da" for zebrafish, and "At" for arabidopsis

`ontology`
: the ontology supported currently. It can be "GOBP" for Gene Ontology Biological Process, "GOMF" for Gene Ontology Molecular Function, "GOCC" for Gene Ontology Cellular Component, "PS" for phylostratific age information, "PS2" for the collapsed PS version (inferred ancestors being collapsed into one with the known taxonomy information), "SF" for domain superfamily assignments, "DO" for Disease Ontology, "HPPA" for Human Phenotype Phenotypic Abnormality, "HPMI" for Human Phenotype Mode of Inheritance, "HPCM" for Human Phenotype Clinical Modifier, "HPMA" for Human Phenotype Mortality Aging, "MP" for Mammalian Phenotype, and Drug-Gene Interaction database (DGIdb) and the molecular signatures database (Msigdb) only in human (including "MsigdbH", "MsigdbC1", "MsigdbC2CGP", "MsigdbC2CP", "MsigdbC2KEGG", "MsigdbC2REACTOME", "MsigdbC2BIOCARTA", "MsigdbC3TFT", "MsigdbC3MIR", "MsigdbC4CGN", "MsigdbC4CM", "MsigdbC5BP", "MsigdbC5MF", "MsigdbC5CC", "MsigdbC6", "MsigdbC7"). Note: These four ("GOBP", "GOMF", "GOCC" and "PS") are availble for all genomes/species; for "Hs" and "Mm", these six ("DO", "HPPA", "HPMI", "HPCM", "HPMA" and "MP") are also supported; all "Msigdb" are only supported in "Hs". For details on the eligibility for pairs of input genome and ontology, please refer to the online Documentations at http://supfam.org/dnet/docs.html

`sizeRange`
: the minimum and maximum size of members of each gene set in consideration. By default, it sets to a minimum of 10 but no more than 1000

`min.overlap`
: the minimum number of overlaps. Only those gene sets that overlap with input data at least min.overlap (3 by default) will be processed

`which_distance`
: which distance of terms in the ontology is used to restrict terms in consideration. By default, it sets to 'NULL' to consider all distances

`test`
: the statistic test used. It can be "FisherTest" for using fisher's exact test, "HypergeoTest" for using hypergeometric test, or "BinomialTest" for using binomial test. Fisher's exact test is to test the independence between gene group (genes belonging to a group or not) and gene annotation (genes annotated by a term or not), and thus compare sampling to the left part of background (after sampling without replacement). Hypergeometric test is to sample at random (without replacement) from the background containing annotated and non-annotated genes, and thus compare sampling to background. Unlike hypergeometric test, binomial test is to sample at random (with replacement) from the background with the constant probability. In terms of the ease of finding the significance, they are in order: hypergeometric test > binomial test > fisher's exact test. In other words, in terms of the calculated p-value, hypergeometric test < binomial test < fisher's exact test

`p.adjust.method`
: the method used to adjust p-values. It can be one of "BH", "BY", "bonferroni", "holm", "hochberg" and "hommel". The first two methods "BH" (widely used) and "BY" control the false discovery rate (FDR: the expected proportion of false discoveries amongst the rejected hypotheses); the last four methods "bonferroni", "holm", "hochberg" and "hommel" are designed to give strong control of

the family-wise error rate (FWER). Notes: FDR is a less stringent condition than FWER

`ontology.algorithm`
: the algorithm used to account for the hierarchy of the ontology. It can be one of "none", "pc", "elim" and "lea". For details, please see 'Note'

`elim.pvalue`
: the parameter only used when "ontology.algorithm" is "elim". It is used to control how to declare a signficantly enriched term (and subsequently all genes in this term are eliminated from all its ancestors)

`lea.depth`
: the parameter only used when "ontology.algorithm" is "lea". It is used to control how many maximum depth is uded to consider the children of a term (and subsequently all genes in these children term are eliminated from the use for the recalculation of the signifance at this term)

`verbose`
: logical to indicate whether the messages will be displayed in the screen. By default, it sets to false for no display

`RData.location`
: the characters to tell the location of built-in RData files. By default, it remotely locates at http://dnet.r-forge.r-project.org/RData. Be aware of several versions and the latest one is matched to the current package version. For the user equipped with fast internet connection, this option can be just left as default. But it is always advisable to download these files locally. Especially when the user needs to run this function many times, there is no need to ask the function to remotely download every time (also it will unnecessarily increase the runtime). For examples, these files (as a whole or part of them) can be first downloaded into your current working directory, and then set this option as: $RData.location = "."$. Surely, the location can be anywhere as long as the user provides the correct path pointing to (otherwise, the script will have to remotely download each time). Here is the UNIX command for downloading all RData files (preserving the directory structure): $wget - r - l2 - A" * .RData" - np - nH - -cut - dirs = 0"http : //dnet.r - forge.r - project.org/RData"$

Value

an object of class "eTerm", a list with following components:

- `set_info`: a matrix of nSet X 4 containing gene set information, where nSet is the number of gene set in consideration, and the 4 columns are "setID" (i.e. "Term ID"), "name" (i.e. "Term Name"), "namespace" and "distance"
- `gs`: a list of gene sets, each storing gene members. Always, gene sets are identified by "setID" and gene members identified by "Entrez ID"
- `data`: a vector containing input data in consideration. It is not always the same as the input data as only those mappable are retained
- `overlap`: a list of overlapped gene sets, each storing genes overlapped between a gene set and the given input data (i.e. the genes of interest). Always, gene sets are identified by "setID" and gene members identified by "Entrez ID"
- `zscore`: a vector containing z-scores
- `pvalue`: a vector containing p-values
- `adjp`: a vector containing adjusted p-values. It is the p value but after being adjusted for multiple comparisons
- `call`: the call that produced this result

Note

The interpretation of the algorithms used to account for the hierarchy of the ontology is:

- "none": does not consider the ontology hierarchy at all.
- "lea": computers the significance of a term in terms of the significance of its children at the maximum depth (e.g. 2). Precisely, once genes are already annotated to any children terms with a more signficance than itself, then all these genes are eliminated from the use for the recalculation of the signifance at that term. The final p-values takes the maximum of the original p-value and the recalculated p-value.
- "elim": computers the significance of a term in terms of the significance of its all children. Precisely, once genes are already annotated to a signficantly enriched term under the cutoff of e.g. pvalue<1e-2, all these genes are eliminated from the ancestors of that term).
- "pc": requires the significance of a term not only using the whole genes as background but also using genes annotated to all its direct parents/ancestors as background. The final p-value takes the maximum of both p-values in these two calculations.
- "Notes": the order of the number of significant terms is: "none" > "lea" > "elim" > "pc".

See Also

dEnricherView

Examples

```
# load data
library(Biobase)
TCGA_mutations <- dRDataLoader(RData='TCGA_mutations')
symbols <- as.character(fData(TCGA_mutations)$Symbol)

# Enrichment analysis using Disease Ontology (DO)
data <- symbols[1:100] # select the first 100 human genes
eTerm <- dEnricher(data, identity="symbol", genome="Hs", ontology="DO")

# visualise the top significant terms in the ontology hierarchy
ig.DO <- dRDataLoader(RData='ig.DO')
g <- ig.DO
nodes_query <- names(sort(eTerm$adjp)[1:5])
nodes.highlight <- rep("red", length(nodes_query))
names(nodes.highlight) <- nodes_query
subg <- dDAGinduce(g, nodes_query)
# color-code terms according to the adjust p-values (taking the form of 10-based negative logarithm)
visDAG(g=subg, data=-1*log10(eTerm$adjp[V(subg)$name]),
node.info="both", zlim=c(0,2), node.attrs=list(color=nodes.highlight))
# color-code terms according to the z-scores
visDAG(g=subg, data=eTerm$zscore[V(subg)$name], node.info="both",
colormap="darkblue-white-darkorange",
node.attrs=list(color=nodes.highlight))
```