dcAlgoPropagate

April 23, 2015

`dcAlgoPropagate` *Function to propagate ontology annotations according to an input file*

Description

`dcAlgoPropagate` is supposed to propagate ontology annotations, given an input file. This input file contains original annotations between domains/features and ontology terms, along with the hypergeometric scores (hscore) in support for their annotations. The annotations are propagated to the ontology root (either retaining the maximum hscore or additively accumulating the hscore). After the propogation, the ontology terms of increasing levels are determined based on the concept of Information Content (IC) to product a slim version of ontology. It returns an object of S3 class "HIS" with three components: "hscore", "ic" and "slim".

Usage

```
dcAlgoPropagate(input.file, ontology = c(NA, "GOBP", "GOMF", "GOCC",
"DO",
"HPPA", "HPMI", "HPON", "MP", "EC", "KW", "UP"), propagation = c("max",
"sum"), output.file = "HIS.RData", verbose = T,
RData.ontology.customised = NULL,
RData.location = "http://dcgor.r-forge.r-project.org/data")
```

Arguments

`input.file`	an input file used to build the object. This input file contains original annotations between domains/features and ontology terms, along with the hypergeometric scores (hscore) in support for their annotations. For example, a file containing original annotations between SCOP domain architectures and GO terms can be found in http://dcgor.r-forge.r-project.org/data/Feature/Feature2GO.sf.txt. As seen in this example, the input file must contain the header (in the first row) and three columns: 1st column for 'Feature_id' (here SCOP domain architectures), 2nd column for 'Term_id' (GO terms), and 3rd column for 'Score' (hscore). Alternatively, the input.file can be a matrix or data frame, assuming that input file has been read. Note: the file should use the tab delimiter as the field separator between columns
`ontology`	the ontology identity. It can be "GOBP" for Gene Ontology Biological Process, "GOMF" for Gene Ontology Molecular Function, "GOCC" for Gene Ontology

Cellular Component, "DO" for Disease Ontology, "HPPA" for Human Phenotype Phenotypic Abnormality, "HPMI" for Human Phenotype Mode of Inheritance, "HPON" for Human Phenotype ONset and clinical course, "MP" for Mammalian Phenotype, "EC" for Enzyme Commission, "KW" for UniProtKB KeyWords, "UP" for UniProtKB UniPathway. For details on the eligibility for pairs of input domain and ontology, please refer to the online Documentations at http://supfam.org/dcGOR/docs.html. If NA, then the user has to input a customised RData-formatted file (see `RData.ontology.customised` below)

`propagation` how to propagate the score. It can be "max" for retaining the maximum hscore (by default), "sum" for additively accumulating the hscore

`output.file` an output file used to save the `HIS` object as an RData-formatted file (see 'Value' for details). If NULL, this file will be saved into "HIS.RData" in the current working local directory. If NA, there will be no output file

`verbose` logical to indicate whether the messages will be displayed in the screen. By default, it sets to TRUE for display

`RData.ontology.customised`
a file name for RData-formatted file containing an object of S4 class 'Onto' (i.e. ontology). By default, it is NULL. It is only needed when the user wants to perform customised analysis using their own ontology. See `dcBuildOnto` for how to creat this object

`RData.location` the characters to tell the location of built-in RData files. By default, it remotely locates at "http://supfam.org/dcGOR/data" or "http://dcgor.r-forge.r-project.org/data". 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: $RData.location = "."$. If RData to load is already part of package itself, this parameter can be ignored (since this function will try to load it via function `data` first)

Value

an object of S3 class `HIS`, with following components:

- `hscore`: a list of features, each with a term-named vector containing hscore
- `ic`: a term-named vector containing information content (IC). Terms are ordered first by IC and then by longest-path level, making sure that for terms with the same IC, parental terms always come first
- `slim`: a list of four slims, each with a term-named vector containing information content (IC). Slim '1' for very general terms, '2' for general terms, '3' for specific terms, '4' for very specific terms

Note

None

See Also

`dcRDataLoader`, `dcConverter`, `dcAlgo`, `dcList2Matrix`

Examples

```
# build an "HIS" object for GO Molecular Function
input.file <-
"http://dcgor.r-forge.r-project.org/data/Feature/Feature2GO.sf.txt"
Feature2GOMF.sf <- dcAlgoPropagate(input.file=input.file,
ontology="GOMF", output.file="Feature2GOMF.sf.RData")
names(Feature2GOMF.sf)
Feature2GOMF.sf$hscore[1]
Feature2GOMF.sf$ic[1:10]
Feature2GOMF.sf$slim[1]

# extract hscore as a matrix with 3 columns (Feature_id, Term_id, Score)
hscore <- Feature2GOMF.sf$hscore
hscore_mat <- dcList2Matrix(hscore)
colnames(hscore_mat) <- c("Feature_id", "Term_id", "Score")
dim(hscore_mat)
hscore_mat[1:10,]
```