dcAlgo

November 27, 2014

`dcAlgo` *Function to apply dcGO algorithm to infer domain-centric ontology*

Description

`dcAlgo` is supposed to apply dcGO algorithm to infer domain-centric ontology from input files. It requires two input files: 1) an annotation file containing annotations between proteins/genes and ontology terms; 2) an architecture file containing domain architectures for proteins/genes.

Usage

```
dcAlgo(anno.file, architecture.file, output.file = NULL, ontology =
c(NA,
"GOBP", "GOMF", "GOCC", "DO", "HPPA", "HPMI", "HPON", "MP", "EC", "KW",
"UP"),
feature.mode = c("supra", "individual", "comb"), min.overlap = 3,
fdr.cutoff = 0.001, parallel = TRUE, multicores = NULL, verbose = T,
RData.ontology.customised = NULL,
RData.location = "http://dcgor.r-forge.r-project.org/data")
```

Arguments

`anno.file` an annotation file containing annotations between proteins/genes and ontology terms. For example, a file containing annotations between human genes and HP terms can be found in http://dcgor.r-forge.r-project.org/data/Algo/HP_anno.txt. As seen in this example, the input file must contain the header (in the first row) and two columns: 1st column for 'SeqID' (actually these IDs can be anything), 2nd column for 'termID' (HP terms)

`architecture.file` an architecture file containing domain architectures (including individual domains) for proteins/genes. For example, a file containing human genes and domain architectures can be found in http://dcgor.r-forge.r-project.org/data/Algo/SCOP_architecture.txt. As seen in this example, the input file must contain the header (in the first row) and two columns: 1st column for 'SeqID' (actually these IDs can be anything), 2nd column for 'Architecture' (SCOP domain architectures, each represented as comma-separated domains)

`output.file` an output file containing results. If not NULL, a tab-delimited text file will be also written out, with 1st column 'Feature_id' for features/domains, 2nd column 'Term_id' for ontology terms, 3rd column 'Score' for hypergeometric scores (indicative of strength for feature-term associations). Otherwise, there is no output file (by default)

`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)

`feature.mode` the mode of how to define the features thereof. It can be: "supra" for combinations of one or two successive domains (including individual domains; considering the order), "individual" for individual domains only, and "comb" for all possible combinations (including individual domains; ignoring the order)

`min.overlap` the minimum number of overlaps with each term in consideration. By default, it sets to a minimum of 3

`fdr.cutoff` the fdr cutoff to call the significant associations between features and terms. By default, it sets to 1e-3

`parallel` logical to indicate whether parallel computation with multicores is used. By default, it sets to true, but not necessarily does so. Partly because parallel backends available will be system-specific (now only Linux or Mac OS). Also, it will depend on whether these two packages "foreach" and "doMC" have been installed. It can be installed via: `source("http://bioconductor.org/biocLite.R"); biocLite(c("foreach","doMC"))`. If not yet installed, this option will be disabled

`multicores` an integer to specify how many cores will be registered as the multicore parallel backend to the 'foreach' package. If NULL, it will use a half of cores available in a user's computer. This option only works when parallel computation is enabled

`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

a data frame containing three columns: 1st column 'Feature_id' for features, 2nd 'Term_id' for terms, and 3rd 'Score' for the hypergeometric score indicative of strength of associations beteen features and terms

Note

When 'output.file' is specified, a tab-delimited text file is output, with the column names: 1st column 'Feature_id' for features, 2nd 'Term_id' for terms, and 3rd 'Score' for the hypergeometric score indicative of strength of associations beteen features and terms

See Also

dcRDataLoader, dcSplitArch, dcConverter, dcDuplicated, dcAlgoPropagate

Examples

```
# 1) Prepare input file: anno.file and architecture.file
anno.file <- "http://dcgor.r-forge.r-project.org/data/Algo/HP_anno.txt"
architecture.file <-
"http://dcgor.r-forge.r-project.org/data/Algo/SCOP_architecture.txt"

# 2) Do inference using built-in data
res <- dcAlgo(anno.file, architecture.file, ontology="HPPA",
feature.mode="supra", parallel=FALSE)
res[1:5,]

# 3) Advanced usage: using customised data
x <-
base::load(base::url("http://dcgor.r-forge.r-project.org/data/onto.HPPA.RData"))
RData.ontology.customised <- onto.HPPA.RData
base::save(list=x, file=RData.ontology.customised)
## you will see a RData file onto.HPPA.RData in local directory: list.files(pattern=*.RData)
res <- dcAlgo(anno.file, architecture.file, feature.mode="supra",
parallel=FALSE, RData.ontology.customised=RData.ontology.customised)
res[1:5,]
```