dcRWRpredict

December 23, 2014

`dcRWRpredict` *Function to perform RWR-based ontology term predictions from input known annotations and an input graph*

Description

`dcRWRpredict` is supposed to perform ontology term predictions based on Random Walk with Restart (RWR) from input known annotations and an input graph.

Usage

```
dcRWRpredict(data, g, output.file = NULL, ontology = c(NA, "GOBP",
"GOMF",
"GOCC", "DO", "HPPA", "HPMI", "HPON", "MP", "EC", "KW", "UP"),
method = c("indirect", "direct"), normalise = c("laplacian", "row",
"column", "none"), restart = 0.75, normalise.affinity.matrix =
c("none",
"quantile"), leave.one.out = T, propagation = c("max", "sum"),
scale.method = c("log", "linear", "none"), parallel = TRUE,
multicores = NULL, verbose = T, RData.ontology.customised = NULL,
RData.location = "http://dcgor.r-forge.r-project.org/data")
```

Arguments

`data`	an input gene-term data matrix containing known annotations used for seeds. Each value in input matrix does not necessarily have to be binary (non-zeros will be used as a weight, but should be non-negative for easy interpretation). Also, data can be a list, each containing the known annotated genes
`g`	an object of class "igraph" or `Dnetwork`
`output.file`	an output file containing predicted results. If not NULL, a tab-delimited text file will be also written out; 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)

`method` the method used to calculate RWR. It can be 'direct' for directly applying RWR, 'indirect' for indirectly applying RWR (first pre-compute affinity matrix and then derive the affinity score)

`normalise` the way to normalise the adjacency matrix of the input graph. It can be 'laplacian' for laplacian normalisation, 'row' for row-wise normalisation, 'column' for column-wise normalisation, or 'none'

`restart` the restart probability used for RWR. The restart probability takes the value from 0 to 1, controlling the range from the starting nodes/seeds that the walker will explore. The higher the value, the more likely the walker is to visit the nodes centered on the starting nodes. At the extreme when the restart probability is zero, the walker moves freely to the neighbors at each step without restarting from seeds, i.e., following a random walk (RW)

`normalise.affinity.matrix` the way to normalise the output affinity matrix. It can be 'none' for no normalisation, 'quantile' for quantile normalisation to ensure that columns (if multiple) of the output affinity matrix have the same quantiles

`leave.one.out` logical to indicate whether the leave-one-out test is used for predictions. By default, it sets to true for doing leave-one-out test (that is, known seeds are removed)

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

`scale.method` the method used to scale the predictive scores. It can be: "none" for no scaling, "linear" for being linearily scaled into the range between 0 and 1, "log" for the same as "linear" but being first log-transformed before being scaled. The scaling between 0 and 1 is done via: $\frac{S-S_{min}}{S_{max}-S_{min}}$, where S_{min} and S_{max} are the minimum and maximum values for S

`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 the same as the input file (e.g. 'SeqID'), 2nd for 'Term' (predicted ontology terms), 3rd for 'Score' (along with predicted scores)

Note

When 'output.file' is specified, a tab-delimited text file is written out, with the column names: 1st column the same as the input file (e.g. 'SeqID'), 2nd for 'Term' (predicted ontology terms), 3rd for 'Score' (along with predicted scores). The choice of which method to use RWR depends on the number of seed sets and whether using leave-one-out test. If the total product of both numbers are huge, it is better to use 'indrect' method (for a single run). Also, when using leave-one-out test, it has to be use 'indrect' method.

See Also

dcRDataLoader, dcAlgoPropagate, dcList2Matrix

Examples

```
# 1) define an input network
## 1a) an igraph object that contains a functional protein association network in human.
### The network is extracted from the STRING database (version 9.1).
### Only those associations with medium confidence (score>=400) are retained
org.Hs.string <- dnet::dRDataLoader(RData=org.Hs.string)
## 1b) restrict to those edges with confidence score>=999
### keep the largest connected component
network <- igraph::subgraph.edges(org.Hs.string,
eids=E(org.Hs.string)[combined_score>=999])
g <- dnet::dNetInduce(g=network, nodes_query=V(network)$name,
largest.comp=TRUE)
## Notably, in reality, 1b) can be replaced by:
#g <- igraph::subgraph.edges(org.Hs.string, eids=E(org.Hs.string)[combined_score>=400])
## 1c) make sure there is a weight edge attribute
E(g)$weight <- E(g)$combined_score
### use EntrezGene ID as default name node attribute
V(g)$name <- V(g)$geneid
g

# 2) define the known annotations as seeds
anno.file <- "http://dcgor.r-forge.r-project.org/data/Algo/HP_anno.txt"
data <- dcSparseMatrix(anno.file)

# 3) perform RWR-based ontology term predictions
res <- dcRWRpredict(data=data, g=g, ontology="HPPA", parallel=FALSE)
```

```
res[1:10,]
# 4) calculate Precision and Recall
GSP.file <- anno.file
prediction.file <- res
res_PR <- dcAlgoPredictPR(GSP.file=GSP.file,
prediction.file=prediction.file, ontology="HPPA")
res_PR

# 5) Plot PR-curve
plot(res_PR[,2], res_PR[,1], xlim=c(0,1), ylim=c(0,1), type="b",
xlab="Recall", ylab="Precision")
```