dcAlgoPredict

December 22, 2014

dcAlgoPredict	*Function to predict ontology terms given domain architectures (including individual domains)*

Description

dcAlgoPredict is supposed to predict ontology terms given domain architectures (including individual domains). It involves 3 steps: 1) splitting an architecture into individual domains and all possible consecutive domain combinations (viewed as component features); 2) merging hscores among component features; 3) scaling merged hscores into predictive scores across terms.

Usage

```
dcAlgoPredict(data, RData.HIS = c(NA, "Feature2GOBP.sf",
"Feature2GOMF.sf",
"Feature2GOCC.sf", "Feature2HPPA.sf", "Feature2GOBP.pfam",
"Feature2GOMF.pfam", "Feature2GOCC.pfam", "Feature2HPPA.pfam",
"Feature2GOBP.interpro", "Feature2GOMF.interpro",
"Feature2GOCC.interpro",
"Feature2HPPA.interpro"), merge.method = c("sum", "max", "sequential"),
scale.method = c("log", "linear", "none"), feature.mode = c("supra",
"individual", "comb"), slim.level = NULL, max.num = NULL,
parallel = TRUE, multicores = NULL, verbose = T,
RData.HIS.customised = NULL,
RData.location = "http://dcgor.r-forge.r-project.org/data")
```

Arguments

data	an input data vector containing domain architectures. An architecture is represented in the form of comma-separated domains
RData.HIS	RData to load. This RData conveys two bits of information: 1) feature (domain) type; 2) ontology. It stores the hypergeometric scores (hscore) between features (individual domains or consecutive domain combinations) and ontology terms. The RData name tells which domain type and which ontology to use. It can be: SCOP sf domains/combinations (including "Feature2GOBP.sf", "Feature2GOMF.sf", "Feature2GOCC.sf", "Feature2HPPA.sf"), Pfam domains/combinations (including "Feature2GOBP.pfam", "Feature2GOMF.pfam", "Feature2GOCC.pfam", "Feature2HPPA.pfam"), InterPro domains (including "Feature2GOBP.interpro",

"Feature2GOMF.interpro", "Feature2GOCC.interpro", "Feature2HPPA.interpro").
If NA, then the user has to input a customised RData-formatted file (see `RData.HIS.customised` below)

`merge.method` the method used to merge predictions for each component feature (individual domains and their combinations derived from domain architecture). It can be one of "sum" for summing up, "max" for the maximum, and "sequential" for the sequential weighting. The sequential weighting is done via: $\sum_{i=1} \frac{R_i}{i}$, where R_i is the i^{th} ranked highest hscore

`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

`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)

`slim.level` whether only slim terms are returned. By defaut, it is NULL and all predicted terms will be reported. If it is specified as a vector containing any values from 1 to 4, then only slim terms at these levels will be reported. Here is the meaning of these values: '1' for very general terms, '2' for general terms, '3' for specific terms, and '4' for very specific terms

`max.num` whether only top terms per sequence are returned. By defaut, it is NULL and no constraint is imposed. If an integer is specified, then all predicted terms (with scores in a decreasing order) beyond this number will be discarded. Notably, this parameter works after the preceding parameter `slim.level`

`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.HIS.customised`
a file name for RData-formatted file containing an object of S3 class 'HIS'. By default, it is NULL. It is only needed when the user wants to perform customised analysis. See `dcAlgoPropagate` on how this object is created

`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 named list of architectures, each containing predictive scores

Note

none

See Also

dcRDataLoader, dcSplitArch, dcConverter, dcAlgoPropagate, dcAlgoPredictMain, dcAlgoPredictGenome

Examples

```
# 1) randomly generate 5 domains and/or domain architectures
x <- dcRDataLoader(RData="Feature2GOMF.sf")
data <- sample(names(x$hscore), 5)

# 2) get predictive scores of all predicted terms for this domain architecture
## using sequential method (by default)
pscore <- dcAlgoPredict(data=data, RData.HIS="Feature2GOMF.sf",
parallel=FALSE)
## using max method
pscore_max <- dcAlgoPredict(data=data, RData.HIS="Feature2GOMF.sf",
merge.method="max", parallel=FALSE)
## using sum method
pscore_sum <- dcAlgoPredict(data=data, RData.HIS="Feature2GOMF.sf",
merge.method="sum", parallel=FALSE)

# 3) advanced usage
## a) focus on those terms at the 2nd level (general)
pscore <- dcAlgoPredict(data=data, RData.HIS="Feature2GOMF.sf",
slim.level=2, parallel=FALSE)
## b) visualise predictive scores in the ontology hierarchy
### load the ontology
g <- dcRDataLoader("onto.GOMF", verbose=FALSE)
ig <- dcConverter(g, from=Onto, to=igraph, verbose=FALSE)
### do visualisation for the 1st architecture
data <- pscore[[1]]
subg <- dnet::dDAGinduce(ig, nodes_query=names(data),
path.mode="shortest_paths")
dnet::visDAG(g=subg, data=data, node.info="term_id")
```