dcAlgoPredictMain

December 23, 2014

dcAlgoPredictMain	*Function to predict ontology terms given an input file containing domain architectures (including individual domains)*

Description

dcAlgoPredictMain is supposed to predict ontology terms given an input file containing domain architectures (including individual domains).

Usage

```
dcAlgoPredictMain(input.file, output.file = NULL, 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

input.file	an input file containing domain architectures (including individual domains). For example, a file containing UniProt ID and domain architectures for human proteins can be found in http://dcgor.r-forge.r-project.org/data/Feature/hs.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). 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
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)

`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 merging. The sequential merging 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 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)

See Also

dcRDataLoader, dcAlgoPropagate, dcAlgoPredict

Examples

```
# 1) Prepare an input file containing domain architectures
input.file <- "http://dcgor.r-forge.r-project.org/data/Feature/hs.txt"

# 2) Do prediction using built-in data
output <- dcAlgoPredictMain(input.file, RData.HIS="Feature2GOMF.sf",
parallel=FALSE)
output[1:5,]

# 3) Advanced usage: using customised data
x <-
base::load(base::url("http://dcgor.r-forge.r-project.org/data/Feature2GOMF.sf.RData"))
RData.HIS.customised <- Feature2GOMF.sf.RData
base::save(list=x, file=RData.HIS.customised)
#list.files(pattern=*.RData)
## you will see an RData file Feature2GOMF.sf.RData in local directory
output <- dcAlgoPredictMain(input.file, parallel=FALSE,
RData.HIS.customised=RData.HIS.customised)
output[1:5,]
```