dcAlgoPredictGenome

July 23, 2015

`dcAlgoPredictGenome` *Function to predict ontology terms for genomes with domain architectures (including individual domains)*

Description

`dcAlgoPredictGenome` is supposed to predict ontology terms for genomes with domain architectures (including individual domains).

Usage

```
dcAlgoPredictGenome(input.file, RData.HIS = c(NULL, "Feature2GOBP.sf",
"Feature2GOMF.sf", "Feature2GOCC.sf", "Feature2HPPA.sf",
"Feature2GOBP.pfam",
"Feature2GOMF.pfam", "Feature2GOCC.pfam", "Feature2HPPA.pfam",
"Feature2GOBP.interpro", "Feature2GOMF.interpro",
"Feature2GOCC.interpro",
"Feature2HPPA.interpro"), weight.method = c("none", "copynum", "ic",
"both"), 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 =
"https://github.com/hfang-bristol/RDataCentre/blob/master/dcGOR")
```

Arguments

`input.file`	an input file containing genomes and their domain architectures (including individual domains). For example, a file containing Hominidae genomes and their domain architectures can be found in http://dcgor.r-forge.r-project.org/data/Feature/Hominidae.txt. As seen in this example, the input file must contain the header (in the first row) and two columns: 1st column for 'Genome' (a genome like a container), 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

`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)

`weight.method` the method used how to weight predictions. It can be one of "none" (no weighting; by default), "copynum" for weighting copynumber of architectures, and "ic" for weighting information content (ic) of the term, "both" for weighting both copynumber and ic

`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 "https://github.com/hfang-bristol/RDataCentre/blob/master/dcGOR" and "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). Here is the UNIX command for downloading all RData files (preserving the directory structure): $wget - r - l2 - A" * .RData" - np - nH - -cut - dirs = 0"http : //dcgor.r - forge.r - project.org/data"$

Value

a matrix of terms X genomes, containing the predicted scores (per genome) as a whole

Note

none

See Also

dcRDataLoader, dcAlgoPropagate, dcAlgoPredict

Examples

```
## Not run:
# 1) Prepare an input file containing domain architectures
input.file <-
"http://dcgor.r-forge.r-project.org/data/Feature/Hominidae.txt"

# 2) Do prediction using built-in data
output <- dcAlgoPredictGenome(input.file, RData.HIS="Feature2GOMF.sf",
parallel=FALSE)
dim(output)
output[1:10,]

# 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 <- dcAlgoPredictGenome(input.file, parallel=FALSE,
RData.HIS.customised=RData.HIS.customised)
dim(output)
output[1:10,]
```

```
## End(Not run)
```