Package ‘PhyloProfile’

May 25, 2024

Version 1.18.0
Date 2024-04-11
Title PhyloProfile

Description
PhyloProfile is a tool for exploring complex phylogenetic profiles. Phylogenetic profiles, presence/absence patterns of genes over a set of species, are commonly used to trace the functional and evolutionary history of genes across species and time. With PhyloProfile we can enrich regular phylogenetic profiles with further data like sequence/structure similarity, to make phylogenetic profiling more meaningful. Besides the interactive visualisation powered by R-Shiny, the package offers a set of further analysis features to gain insights like the gene age estimation or core gene identification.

URL https://github.com/BIONF/PhyloProfile/

BugReports https://github.com/BIONF/PhyloProfile/issues
License MIT + file LICENSE
Depends R (>= 4.3.0)
Encoding UTF-8

biocViews Software, Visualization, DataRepresentation, MultipleComparison, FunctionalPrediction

Imports ape, bioDist, BiocStyle, Biostrings, colourpicker, data.table, DT, energy, ExperimentHub, ggplot2, gridExtra, pbapply, RColorBrewer, RCurl, shiny, shinyBS, shinysseloaders, shinyFiles, shinyjs, stringr, OmaDB, plyr, xml2, zoo, yaml

RoxygenNote 7.2.3
Suggests knitr, rmarkdown, testthat

VignetteBuilder knitr
git_url https://git.bioconductor.org/packages/PhyloProfile
git_branch RELEASE_3_19
git_last_commit 7bb4956
git_last_commit_date 2024-04-30
Repository Bioconductor 3.19
Date/Publication 2024-05-24

Author  Vinh Tran [aut, cre] (<https://orcid.org/0000-0001-6772-7595>),
        Bastian Greshake Tzovaras [aut],
        Ingo Ebersberger [aut],
        Carla Mölbert [ctb]

Maintainer  Vinh Tran <tran@bio.uni-frankfurt.de>

Contents

addRankDivisionPlot .................................................. 4
calcPresSpec .......................................................... 5
checkInputValidity .................................................... 6
checkNewick ............................................................ 7
checkOmaID ............................................................. 8
clusterDataDend ......................................................... 8
compareMedianTaxonGroups ............................................. 9
compareTaxonGroups .................................................... 10
createArchiPlot ........................................................ 11
createGeneAgePlot ...................................................... 12
createLongMatrix ....................................................... 13
createPercentageDistributionData ................................... 14
createProfileFromOma .................................................. 15
createUnrootedTree ..................................................... 15
createVarDistPlot ...................................................... 16
createVariableDistributionData ....................................... 17
createVariableDistributionDataSubset ................................ 18
dataCustomizedPlot .................................................... 19
dataFeatureTaxGroup ................................................... 20
dataMainPlot ............................................................ 21
dataVarDistTaxGroup ................................................... 22
distributionTest ........................................................ 23
estimateGeneAge ........................................................ 23
fastaParser .............................................................. 25
featureDistTaxPlot ...................................................... 25
filteredProfile .......................................................... 27
filterProfileData ........................................................ 28
finalProcessedProfile ................................................... 30
fromInputToProfile ..................................................... 31
fullProcessedProfile ................................................... 33
geneAgePlotDf ........................................................... 33
generateSinglePlot ...................................................... 34
getAllDomainsOma ........................................................ 35
getAllFastaOma .......................................................... 36
getCommonAncestor ...................................................... 37
coreGene ................................................................. 37
dataClustering .......................................................... 39
dataForOneOma ........................................................... 40
<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>getDendrogram</td>
<td>40</td>
</tr>
<tr>
<td>getDistanceMatrix</td>
<td>41</td>
</tr>
<tr>
<td>getDomainFolder</td>
<td>42</td>
</tr>
<tr>
<td>getFastaFromFasInput</td>
<td>43</td>
</tr>
<tr>
<td>getFastaFromFile</td>
<td>43</td>
</tr>
<tr>
<td>getFastaFromFolder</td>
<td>44</td>
</tr>
<tr>
<td>getIDsRank</td>
<td>45</td>
</tr>
<tr>
<td>getInputTaxaID</td>
<td>46</td>
</tr>
<tr>
<td>getInputTaxaName</td>
<td>47</td>
</tr>
<tr>
<td>getNameList</td>
<td>47</td>
</tr>
<tr>
<td>getOmaDataForOneOrtholog</td>
<td>48</td>
</tr>
<tr>
<td>getOmaDomainFromURL</td>
<td>49</td>
</tr>
<tr>
<td>getOmaMembers</td>
<td>49</td>
</tr>
<tr>
<td>getQualColForVector</td>
<td>50</td>
</tr>
<tr>
<td>getSelectedFastaOma</td>
<td>51</td>
</tr>
<tr>
<td>getSelectedTaxonNames</td>
<td>51</td>
</tr>
<tr>
<td>getTaxHierarchy</td>
<td>52</td>
</tr>
<tr>
<td>getTaxonomyInfo</td>
<td>53</td>
</tr>
<tr>
<td>getTaxonomyMatrix</td>
<td>54</td>
</tr>
<tr>
<td>getTaxonomyRanks</td>
<td>55</td>
</tr>
<tr>
<td>gridArrangeSharedLegend</td>
<td>55</td>
</tr>
<tr>
<td>heatmapPlotting</td>
<td>56</td>
</tr>
<tr>
<td>highlightProfilePlot</td>
<td>58</td>
</tr>
<tr>
<td>id2name</td>
<td>59</td>
</tr>
<tr>
<td>idList</td>
<td>60</td>
</tr>
<tr>
<td>mainLongRaw</td>
<td>60</td>
</tr>
<tr>
<td>mainTaxonomyRank</td>
<td>61</td>
</tr>
<tr>
<td>modifyFeatureName</td>
<td>61</td>
</tr>
<tr>
<td>pairDomainPlotting</td>
<td>62</td>
</tr>
<tr>
<td>parseDomainInput</td>
<td>63</td>
</tr>
<tr>
<td>parseInfoProfile</td>
<td>64</td>
</tr>
<tr>
<td>ppTaxonomyMatrix</td>
<td>65</td>
</tr>
<tr>
<td>ppTree</td>
<td>66</td>
</tr>
<tr>
<td>processNcbiTaxonomy</td>
<td>66</td>
</tr>
<tr>
<td>processOrthoID</td>
<td>67</td>
</tr>
<tr>
<td>profileWithTaxonomy</td>
<td>68</td>
</tr>
<tr>
<td>qualitativeColours</td>
<td>69</td>
</tr>
<tr>
<td>rankIndexing</td>
<td>69</td>
</tr>
<tr>
<td>rankList</td>
<td>70</td>
</tr>
<tr>
<td>reduceProfile</td>
<td>70</td>
</tr>
<tr>
<td>runPhyloProfile</td>
<td>71</td>
</tr>
<tr>
<td>singleDomainPlotting</td>
<td>72</td>
</tr>
<tr>
<td>sortDomains</td>
<td>73</td>
</tr>
<tr>
<td>sortInputTaxa</td>
<td>74</td>
</tr>
<tr>
<td>sortTaxaFromTree</td>
<td>75</td>
</tr>
<tr>
<td>taxa2dist</td>
<td>76</td>
</tr>
<tr>
<td>taxonNamesReduced</td>
<td>77</td>
</tr>
<tr>
<td>taxonomyMatrix</td>
<td>77</td>
</tr>
</tbody>
</table>
addRankDivisionPlot

Add taxonomy rank division lines to the heatmap plot

Description

Add taxonomy rank division lines to the heatmap plot

Usage

addRankDivisionPlot(profilePlot = NULL, plotDf = NULL, 
taxDB = NULL, workingRank = NULL, superRank = NULL, xAxis = "taxa", 
groupLabelSize = 14, groupLabelDist = 2, groupLabelAngle = 90)

Arguments

profilePlot initial (highlighted) profile plot
plotDf dataframe for plotting the heatmap phylogentic profile
taxDB path to taxonomy database (taxonomyMatrix.txt file required!)
workingRank working taxonomy rank (e.g. species)
superRank taxonomy rank for division lines (e.g. superkingdom)
xAxis type of x-axis (either "genes" or "taxa")
groupLabelSize size of rank labels
groupLabelDist size of the plot area for rank labels
groupLabelAngle angle of rank labels

Value

A profile heatmap plot with highlighted gene and/or taxon of interest as ggplot object.

Author(s)

Vinh Tran tran@bio.uni-frankfurt.de

See Also

heatmapPlotting, highlightProfilePlot, getTaxonomyMatrix
Examples

```r
data("finalProcessedProfile", package="PhyloProfile")
plotDf <- dataMainPlot(finalProcessedProfile)
plotParameter <- list(
  "xAxis" = "taxa",
  "var1ID" = "FAS_FW",
  "var2ID" = "FAS_BW",
  "midVar1" = 0.5,
  "midColorVar1" = "#FFFFFF",
  "lowColorVar1" = "#FF8C00",
  "highColorVar1" = "#4682B4",
  "midVar2" = 1,
  "midColorVar2" = "#FFFFFF",
  "lowColorVar2" = "#CB4C4E",
  "highColorVar2" = "#3E436F",
  "paraColor" = "#07D000",
  "xSize" = 8,
  "ySize" = 8,
  "legendSize" = 8,
  "mainLegend" = "top",
  "dotZoom" = 0,
  "xAngle" = 60,
  "guideline" = 0,
  "colorByGroup" = FALSE,
  "colorByOrthoID" = FALSE
)
profilePlot <- heatmapPlotting(plotDf, plotParameter)
workingRank <- "class"
superRank <- "superkingdom"
addRankDivisionPlot(
  profilePlot, plotDf, NULL, workingRank, superRank, "taxa"
)
```

calcPresSpec

*Calculate percentage of present species in each super taxon*

Description

Calculate percentage of present species in each super taxon

Usage

```r
calcPresSpec(profileWithTax, taxaCount)
```

Arguments

- **profileWithTax** : data frame of main PhyloProfile input together with their taxonomy info (see `?profileWithTaxonomy`)
- **taxaCount** : number of species occur in each supertaxon (e.g. phylum or kingdom)
checkInputValidity

Description
Check if input file has one of the following format: orthoXML, multiple FASTA, tab-delimited matrix (wide or long), or list of OMA IDs.

Usage
checkInputValidity(filein)

Arguments
filein input file

Value
The format of the input file format, or type of error

Author(s)
Vinh Tran tran@bio.uni-frankfurt.de

See Also
checkOmaID
checkNewick

Examples

```r
filein <- system.file(
  "extdata", "test.main.wide", package = "PhyloProfile", mustWork = TRUE
)
checkInputValidity(filein)
```

data("ppTree", package="PhyloProfile")
checkNewick(ppTree, c("ncbi3702", "ncbi3711", "ncbi7029"))

Description

Check the validity of input newick tree

Usage

```r
checkNewick(tree, inputTaxonID = NULL)
```

Arguments

- `tree`: input newick tree
- `inputTaxonID`: list of all input taxon IDs for the phylogenetic profiles

Value

Possible formatting error of input tree. 0 = suitable tree for using with PhyloProfile, 1 = missing parenthesis; 2 = missing comma; 3 = tree has singleton; or a list of taxa that do not exist in the input phylogenetic profile.

Author(s)

Vinh Tran tran@bio.uni-frankfurt.de

See Also

- `getInputTaxaID` for getting input taxon IDs, `ppTree` for an example of input tree

Examples

```r
data("ppTree", package="PhyloProfile")
checkNewick(ppTree, c("ncbi3702", "ncbi3711", "ncbi7029"))
```
checkOmaID  
*Check the validity of input OMA IDs*

**Description**
Check if input IDs are valid OMA IDs for OMA Browser

**Usage**
checkOmaID(ids)

**Arguments**
ids  
list of ids to be checked

**Value**
List of invalid IDs (not readable for OMA)

**Author(s)**
Vinh Tran tran@bio.uni-frankfurt.de

**Examples**

```r
print("Uncomment the following line to run the function")
# checkOmaID("HUMAN29398")
```

---

clusterDataDend  
*Create a hclust object from the distance matrix*

**Description**
Create a hclust object from the distance matrix

**Usage**
clusterDataDend(distanceMatrix = NULL, clusterMethod = "complete")

**Arguments**
distanceMatrix  
calculated distance matrix (see ?getDistanceMatrix)

clusterMethod  
clustering method ("single", "complete", "average" for UPGMA, "mcquitty" for WPGMA, "median" for WPGMC, or "centroid" for UPGMC). Default = "complete".
Value

An object class hclust generated based on input distance matrix and a selected clustering method.

Author(s)

Vinh Tran tran@bio.uni-frankfurt.de

See Also

ggetDataClustering, getDistanceMatrix, hclust

Examples

data("finalProcessedProfile", package="PhyloProfile")
data <- finalProcessedProfile
profileType <- "binary"
profiles <- getDataClustering(
    data, profileType, var1AggregateBy, var2AggregateBy)
distMethod <- "mutualInformation"
distanceMatrix <- getDistanceMatrix(profiles, distMethod)
clusterMethod <- "complete"
clusterDataDend(distanceMatrix, clusterMethod)

compareMedianTaxonGroups

*Compare the median values of a variable between 2 taxon groups*

Description

Given the phylogenetic profiles that contains up to 2 additional variables besides the presence/absence information of the orthologous proteins. This function will compare the median scores of those variables between 2 different taxon groups (e.g. parasitic species vs non-parasitic species), which are defined as in-group and out-group. In-group is identified by the user. Out-group contains all taxa in the input phylogenetic profiles that are not part of the in-group.

Usage

`compareMedianTaxonGroups(data, inGroup, useCommonAncestor, variable, taxDB)`

Arguments

data input phylogenetic profile in long format (see ?mainLongRaw and ?createLongMatrix)
inGroup ID list of in-group taxa (e.g. "ncbi1234")
useCommonAncestor TRUE/FALSE if using all taxa that share the same common ancestor with the pre-selected in-group as the in-group taxa. Default = TRUE.
variable  name of the variable that need to be compared

taxDB  Path to the taxonomy DB files

Value

List of genes that have a difference in the variable’s median scores between the in-group and out-group taxa and their corresponding delta-median.

Author(s)

Vinh Tran (tran@bio.uni-frankfurt.de)

Examples

data("mainLongRaw", package="PhyloProfile")
data <- mainLongRaw
inGroup <- c("ncbi9606", "ncbi10116")
variable <- colnames(data)[4]
compareMedianTaxonGroups(data, inGroup, TRUE, variable)

compareTaxonGroups  Compare the score distributions between 2 taxon groups

Description

Given the phylogenetic profiles that contains up to 2 additional variables besides the presence/absence information of the orthologous proteins. This function will compare the distribution of those variables between 2 different taxon groups (e.g. parasitic species vs non-parasitic species), which are defined as in-group and out-group. In-group is identified by the user. Out-group contains all taxa in the input phylogenetic profiles that are not part of the in-group.

Usage

compareTaxonGroups(data, inGroup, useCommonAncestor, variable, significanceLevel, taxDB)

Arguments

data  input phylogenetic profile in long format (see ?mainLongRaw and ?createLongMatrix)
inGroup  ID list of in-group taxa (e.g. "ncbi1234")
useCommonAncestor  TRUE/FALSE if using all taxa that share the same common ancestor with the pre-selected in-group as the in-group taxa. Default = TRUE.
variable  name of the variable that need to be compared
significanceLevel  significant cutoff for the statistic test (between 0 and 1). Default = 0.05.
taxDB  Path to the taxonomy DB files
**createArchiPlot**

**Value**

list of genes that have a significant difference in the variable distributions between the in-group and out-group taxa and their corresponding p-values.

**Author(s)**

Vinh Tran (tran@bio.uni-frankfurt.de)

**Examples**

data("mainLongRaw", package="PhyloProfile")
data <- mainLongRaw
inGroup <- c("ncbi9606", "ncbi10116")
variable <- colnames(data)[4]
compareTaxonGroups(data, inGroup, TRUE, variable, 0.05)

**createArchiPlot**  
Create protein's domain architecture plot

**Description**

Create architecture plot for both seed and orthologous protein. If domains of ortholog are missing, only architecture of seed protein will be plotted. NOTE: seed protein ID is the one being shown in the profile plot, which normally is also the orthologous group ID.

**Usage**

createArchiPlot(info = NULL, domainDf = NULL, labelArchiSize = 12, titleArchiSize = 12, showFeature = "all", seqIdFormat = "unknown", currentNCBIinfo = NULL)

**Arguments**

- **info**: a list contains seed and ortholog's IDs
- **domainDf**: dataframe contains domain info for the seed and ortholog. This including the seed ID, orthologs IDs, sequence lengths, feature names, start and end positions, feature weights (optional) and the status to determine if that feature is important for comparison the architecture between 2 proteins* (e.g. seed protein vs ortholog) (optional).
- **labelArchiSize**: label size (in px). Default = 12.
- **titleArchiSize**: title size (in px). Default = 12.
- **showFeature**: choose to show all, common or unique features. Default = "all"
- **seqIdFormat**: sequence ID format (either bionf or unknown). Default = "unknown"
- **currentNCBIinfo**: dataframe of the pre-processed NCBI taxonomy data. Default = NULL (will be automatically retrieved from PhyloProfile app)
createGeneAgePlot

Create gene age plot

description
Create gene age plot

Usage
createGeneAgePlot(geneAgePlotDf, textFactor = 1)

Arguments

  geneAgePlotDf  data frame required for plotting gene age (see ?geneAgePlotDf)
  textFactor     increase factor of text size

Value
A gene age distribution plot as a ggplot2 object

Author(s)
Vinh Tran tran@bio.uni-frankfurt.de
createLongMatrix

Create a long matrix format for all kinds of input phylogenetic profiles

Usage

createLongMatrix(inputFile = NULL)

Arguments

inputFile input profile file in orthoXML, multiple FASTA, tab-delimited matrix format (wide or long).

Value

A data frame of input data in long-format containing seed gene IDs (or orthologous group IDs), their orthologous proteins together with the corresponding taxonomy IDs and values of (up to) two additional variables.

Author(s)

Vinh Tran tran@bio.uni-frankfurt.de

See Also

xmlParser, fastaParser, wideToLong

Examples

inputFile <- system.file(
  "extdata", "test.main.wide", package = "PhyloProfile", mustWork = TRUE
)
createLongMatrix(inputFile)
createPercentageDistributionData

Create data for percentage present taxa distribution

Description

Create data for percentage present taxa distribution

Usage

createPercentageDistributionData(inputData = NULL, rankName = NULL, taxDB = NULL)

Arguments

inputData dataframe contains raw input data in long format (see ?mainLongRaw)
rankName name of the working taxonomy rank (e.g. "species", "family")
taxDB Path to the taxonomy DB files

Value

A dataframe for analysing the distribution of the percentage of species in the selected supertaxa, containing the seed protein IDs, percentage of their orthologs in each supertaxon and the corresponding supertaxon names.

Author(s)

Vinh Tran tran@bio.uni-frankfurt.de

See Also

mainLongRaw

Examples

data("mainLongRaw", package="PhyloProfile")
createPercentageDistributionData(mainLongRaw, "class")
**createProfileFromOma**

Create a phylogenetic profile from a raw OMA dataframe

### Description
Create a phylogenetic profile from a raw OMA dataframe

### Usage
```r
createProfileFromOma(finalOmaDf = NULL)
```

### Arguments
- `finalOmaDf` raw OMA data for a list of proteins (see `getDataForOneOma`)

### Value
Dataframe of the phylogenetic profiles in long format, which contains the seed protein IDs, their orthologous proteins and the corresponding taxonomy IDs of the orthologs.

### Author(s)
Vinh Tran tran@bio.uni-frankfurt.de

### See Also
- `getDataForOneOma`

### Examples
```r
print("Uncomment the following lines to run the function")
# omaData <- getDataForOneOma("HUMAN29397", "OG")
# createProfileFromOma(omaData)
```

**createUnrootedTree**

Create unrooted tree from a taxonomy matrix

### Description
Create unrooted tree from a taxonomy matrix

### Usage
```r
createUnrootedTree(df)
```
createVarDistPlot

Arguments
  df       data frame contains taxonomy matrix used for generating tree

Value
  A unrooted taxonomy tree as an object of class "phylo".

Author(s)
  Vinh Tran tran@bio.uni-frankfurt.de

See Also
  taxa2dist for distance matrix generation from a taxonomy matrix, getTaxonomyMatrix for getting taxonomy matrix, ppTaxonomyMatrix for a demo taxonomy matrix data

Examples
  data("ppTaxonomyMatrix", package = "PhyloProfile")
  createUnrootedTree(ppTaxonomyMatrix)

createVarDistPlot

Create distribution plot

Description
  Create distribution plot for one of the additional variable or the percentage of the species present in the supertaxa.

Usage
  createVarDistPlot(data, varName = "var", varType = "var1", percent = c(0, 1), textSize = 12)

Arguments
  data       dataframe contains data for plotting (see ?createVariableDistributionData, ?createVariableDistributionDataSubset or ?createPercentageDistributionData)
  varName    name of the variable that need to be analyzed (either name of variable 1 or variable 2 or "percentage of present taxa"). Default = "var".
  varType    type of variable (either "var1", "var2" or "presSpec"). Default = "var1".
  percent    range of percentage cutoff (between 0 and 1). Default = c(0,1)
  textSize   text size of the distribution plot (in px). Default = 12.

Value
  A distribution plot for the selected variable as a ggplot object
createVariableDistributionData

Author(s)
Vinh Tran tran@bio.uni-frankfurt.de

See Also
mainLongRaw, createVariableDistributionData, createVariableDistributionDataSubset, createPercentageDistributionData

Examples

data("mainLongRaw", package="PhyloProfile")
data <- createVariableDistributionData(
  mainLongRaw, c(0, 1), c(0.5, 1)
)
varName <- "Variable abc"
varType <- "var1"
percent <- c(0,1)
textSize <- 12
createVarDistPlot(
  data,
  varName,
  varType,
  percent,
  textSize
)

createVariableDistributionData

Create data for additional variable distribution

Description
Create data for additional variable distribution

Usage

createVariableDistributionData(inputData, var1Cutoff = c(0 ,1),
  var2Cutoff = c(0, 1))

Arguments

inputData    dataframe contains raw input data in long format (see ?mainLongRaw)
var1Cutoff   min and max cutoff for var1. Default = c(0, 1).
var2Cutoff   min and max cutoff for var2. Default = c(0, 1).

Value
A dataframe for analysing the distribution of the additional variable(s) containing the protein (ortholog) IDs and the values of their variables (var1 and var2).
createVariableDistributionDataSubset

Create data for additional variable distribution (for a subset data)

Description

Create data for additional variable distribution (for a subset data)

Usage

createVariableDistributionDataSubset(fullProfileData, distributionData, selectedGenes, selectedTaxa)

Arguments

fullProfileData
dataframe contains the full processed profiles (see `?fullProcessedProfile`, `?filterProfileData` or `?fromInputToProfile`)
distributionData
dataframe contains the full distribution data (see `?createVariableDistributionData`)
selectedGenes  list of genes of interest. Default = "all".
selectedTaxa  list of taxa of interest Default = "all".

Value

A dataframe for analysing the distribution of the additional variable(s) for a subset of genes and/or taxa containing the protein (ortholog) IDs and the values of their variables (var1 and var2).

Author(s)

Vinh Tran tran@bio.uni-frankfurt.de
**dataCustomizedPlot**

Create data for customized profile plot

**Description**

Create data for customized profile plot based on a selected list of genes and/or taxa, containing seed protein IDs (geneID), ortholog IDs (orthoID) together with their ncbi taxonomy IDs (ncbiID and abbrName), full names (fullName), indexed supertaxa (supertaxon), values for additional variables (var1, var2) and the aggregated values of those additional variables for each supertaxon (mVar1, mVar2), number of original and filtered co-orthologs in each supertaxon (paralog and paralogNew), number of species in each supertaxon (numberSpec) and the each supertaxon (presSpec).

**Usage**

```r
dataCustomizedPlot(dataHeat = NULL, selectedTaxa = "all", selectedSeq = "all")
```

**Arguments**

- `dataHeat`: a data frame contains processed profiles (see `?fullProcessedProfile, ?filterProfileData`)
- `selectedTaxa`: selected subset of taxa. Default = "all".
- `selectedSeq`: selected subset of genes. Default = "all".

**Value**

A dataframe contains data for plotting the customized profile.
Author(s)

Vinh Tran tran@bio.uni-frankfurt.de

See Also

filterProfileData

Examples

data("finalProcessedProfile", package="PhyloProfile")
selectedTaxa <- c("Mammalia", "Saccharomycetes", "Insecta")
selectedSeq <- "all"
dataCustomizedPlot(finalProcessedProfile, selectedTaxa, selectedSeq)

Description

Create data for plotting the distribution of the protein domain features between 2 group of taxa for a selected gene (average number of feature occurrency per protein/ortholog).

Usage

dataFeatureTaxGroup(mainDf, domainDf, inGroup, gene)

Arguments

mainDf          input phylogenetic profile in long format (see ?mainLongRaw and ?createLongMatrix)
domainDf        dataframe contains domain info for the seed and ortholog. This including the seed ID, orthologs IDs, sequence lengths, feature names, start and end positions, feature weights (optional) and the status to determine if that feature is important for comparison the architecture between 2 proteins* (e.g. seed protein vs ortholog) (optional). (see ?parseDomainInput)
inGroup          ID list of in-group taxa (e.g. "ncbi1234")
gene            ID of gene that need to be plotted the feature distribution comparison between in- and out-group taxa.

Value

Dataframe containing all feature names, their frequencies (absolute count and the average instances per protein - IPP) in each taxon group and the corresponding taxa group type (in- or out-group).

Author(s)

Vinh Tran (tran@bio.uni-frankfurt.de)
See Also

createLongMatrix, parseDomainInput

Examples

data(mainLongRaw, package="PhyloProfile")
mainDf <- mainLongRaw
gene <- "101621at6656"
inputFile <- system.file(
  "extdata", "domainFiles/101621at6656.domains",
  package = "PhyloProfile", mustWork = TRUE
)
type <- "file"
domainDf <- parseDomainInput(gene, inputFile, type)
inGroup <- c("ncbi9606", "ncbi10116")
dataFeatureTaxGroup(mainDf, domainDf, inGroup, gene)
Examples

data("finalProcessedProfile", package="PhyloProfile")
dataMainPlot(finalProcessedProfile)

---
dataVarDistTaxGroup  Create data for variable distribution comparison plot

Description

Create data for plotting the distribution comparison between 2 groups of taxa for a selected gene.

Usage

dataVarDistTaxGroup(data, inGroup, gene, variable)

Arguments

data  input phylogenetic profile in long format (see ?mainLongRaw and ?createLongMatrix)
inGroup  ID list of in-group taxa (e.g. "ncbi1234")
gene  ID of gene that need to be plotted the distribution comparison between in- and out-group taxa.
variable  var1 or c(var1, var2)

Value

Dataframe containing list of values for all available variables for the selected genes in in-group and out-group taxa (max. 3 columns).

Author(s)

Vinh Tran (tran@bio.uni-frankfurt.de)

See Also

createLongMatrix

Examples

data("mainLongRaw", package="PhyloProfile")
data <- mainLongRaw
inGroup <- c("ncbi9606", "ncbi10116")
variable <- colnames(data)[c(4, 5)]
dataVarDistTaxGroup(data, inGroup, "101621at6656", variable)
**distributionTest**  
*Compare the distribution of 2 numeric vectors*

**Description**

This function tests the difference between the distributions of two input numeric samples using the statistical test. First the Kolmogorov-Smirnov is used to check if 2 samples have the same distribution. If yes, Wilcoxon-Mann-Whitney will be used to compare the distribution difference.

**Usage**

```r
distributionTest(varIn, varOut, significanceLevel)
```

**Arguments**

- `varIn`: first numeric vector
- `varOut`: second numeric vector
- `significanceLevel`: significant cutoff of the Kolmogorov-Smirnov test. Default = 0.05.

**Value**

p-value of the comparison test.

**Author(s)**

Carla Mölbert (carla.moelbert@gmx.de)

**estimateGeneAge**  
*Calculate the phylogenetic gene age from the phylogenetic profiles*

**Description**

Calculate the phylogenetic gene age from the phylogenetic profiles.

**Usage**

```r
estimateGeneAge(processedProfileData, taxaCount, rankName, refTaxon, var1CO, var2CO, percentCO, taxDB = NULL)
```
estimateGeneAge

Arguments

- **processedProfileData**: dataframe contains the full processed phylogenetic profiles (see `?fullProcessedProfile` or `?parseInfoProfile`)
- **taxaCount**: dataframe counting present taxa in each supertaxon
- **rankName**: working taxonomy rank (e.g. "species", "genus", "family")
- **refTaxon**: reference taxon name (e.g. "Homo sapiens", "Homo" or "Hominidae")
- **var1CO**: cutoff for var1. Default: c(0, 1)
- **var2CO**: cutoff for var2. Default: c(0, 1)
- **percentCO**: cutoff for percentage of species present in each supertaxon. Default: c(0, 1)
- **taxDB**: Path to the taxonomy DB files

Value

A dataframe contains estimated gene ages for the seed proteins.

Author(s)

Vinh Tran tran@bio.uni-frankfurt.de

See Also

- `parseInfoProfile` for creating a full processed profile dataframe; `getNameList` and `getTaxonomyMatrix` for getting taxonomy info, `fullProcessedProfile` for a demo input dataframe

Examples

```r
data("fullProcessedProfile", package="PhyloProfile")
rankName <- "class"
refTaxon <- "Mammalia"
processedProfileData <- fullProcessedProfile
taxonIDs <- levels(as.factor(processedProfileData$ncbiID))
sortedInputTaxa <- sortInputTaxa(
  taxonIDs, rankName, refTaxon, NULL, NULL
)
taxaCount <- plyr::count(sortedInputTaxa, "supertaxon")
var1Cutoff <- c(0, 1)
var2Cutoff <- c(0, 1)
percentCutoff <- c(0, 1)
estimateGeneAge(
  processedProfileData,
taxaCount,
  rankName,
  refTaxon,
  var1Cutoff, var2Cutoff, percentCutoff
)
```
fastaParser  Parse multi-fasta input file

Description
Parse multi-fasta input file

Usage
fastaParser(inputFile = NULL)

Arguments
inputFile  input multiple fasta file. Check extdata/test.main.fasta or https://github.com/BIONF/PhyloProfile/wiki/Input-Data#multi-fasta-format for the supported FASTA header.

Value
A data frame of input data in long-format containing seed gene IDs (or orthologous group IDs), their orthologous proteins together with the corresponding taxonomy IDs and values of (up to) two additional variables.

Author(s)
Vinh Tran tran@bio.uni-frankfurt.de

Examples
inputFile <- system.file(
  "extdata", "test.main.fasta", package = "PhyloProfile", mustWork = TRUE
)
fastaParser(inputFile)

featureDistTaxPlot  Create feature distribution comparison plot

Description
Create protein feature distribution plots between 2 groups of taxa for a selected gene.

Usage
featureDistTaxPlot(data, plotParameters)
featureDistTaxPlot

Arguments

data dataframe for plotting (see ?dataFeatureTaxGroup)
plotParameters plot parameters, including size of x-axis, y-axis, legend and title; position of legend ("right", "bottom" or "none"); names of in-group and out-group; flip the plot coordinate ("Yes" or "No"). NOTE: Leave blank or NULL to use default values.

Value

Distribution plots as a ggplot2 object.

Author(s)

Vinh Tran tran@bio.uni-frankfurt.de

See Also

dataFeatureTaxGroup

Examples

data("mainLongRaw", package="PhyloProfile")
data <- mainLongRaw
gene <- "101621at6656"
inputFile <- system.file("extdata", "domainFiles/101621at6656.domains",
  package = "PhyloProfile", mustWork = TRUE
)
type <- "file"
domainDf <- parseDomainInput(gene, inputFile, type)
inGroup <- c("ncbi9606", "ncbi10116")
plotDf <- dataFeatureTaxGroup(data, domainDf, inGroup, gene)
plotParameters <- list(
  "xSize" = 12,
  "ySize" = 12,
  "angle" = 15,
  "legendSize" = 12,
  "inGroupName" = "In-group",
  "outGroupName" = "Out-group",
  "flipPlot" = "No"
)
featureDistTaxPlot(plotDf, plotParameters)
filteredProfile  

An example of a filtered phylogenetic profile.

Description

An example of a filtered phylogenetic profile.

Usage

data(filteredProfile)

Value

A data frame with 168 rows and 20 variables:

- geneID Seed or ortholog group ID, e.g. "100136at6656"
- supertaxon Supertaxon name together with its ordered index, e.g. "1001_Mammalia"
- ncbiID Taxon ID, e.g. "ncbi10116"
- orthoID Ortholog ID, e.g. "100136at6656lHUMAN@9606@1Q9UNQ211"
- var1 First additional variable
- var2 Second additional variable
- paralog Number of co-orthologs in the current taxon
- abbrName NCBI ID of the ortholog, e.g. "ncbi9606"
- taxonID Taxon ID of the ortholog, in this case: "0"
- fullName Full taxon name of the ortholog, e.g. "Homo sapiens"
- supertaxonID Supertaxon ID (only different than ncbiID in case working with higher taxonomy rank than input's). e.g. "40674"
- rank Rank of the supertaxon, e.g. "class"
- category "cat"
- numberSpec Total number of species in each supertaxon
- taxonMod Name of supersupertaxon w/o its index, e.g. "Mammalia"
- presSpec Percentage of taxa having orthologs in each supertaxon
- presentTaxa Number of taxa that have ortho in each supertaxon
- totalTaxa Total number of taxa in each supertaxon
- mVar1 Value of the 1. variable after grouping into supertaxon
- mVar2 Value of the 2. variable after grouping into supertaxon
filterProfileData  

Filter phylogenetic profiles

Description

Create a filtered data needed for plotting or clustering phylogenetic profiles. NOTE: this function require some intermediate steps using the results from other functions. If you would like to get a full processed data from the raw input, please use the function fromInputToProfile() instead!

Usage

filterProfileData(DF, taxaCount, refTaxon = NULL, percentCO = c(0, 1), coorthoCOMax = 9999, var1CO = c(0, 1), var2CO = c(0, 1), var1Rel = "protein", var2Rel = "protein", groupByCat = FALSE, catDt = NULL, var1AggregateBy = "max", var2AggregateBy = "max")

Arguments

DF a reduced dataframe contains info for all phylogenetic profiles in the selected taxonomy rank.
taxaCount dataframe counting present taxa in each supertaxon
refTaxon selected reference taxon. NOTE: This taxon will not be affected by the filtering. If you want to filter all, set refTaxon <- NULL. Default = NULL.
percentCO min and max cutoffs for percentage of species present in a supertaxon. Default = c(0, 1).
coorthoCOMax maximum number of co-orthologs allowed. Default = 9999.
var1CO min and max cutoffs for var1. Default = c(0, 1).
var2CO min anc max cutoffs for var2. Default = c(0, 1).
var1Rel relation of var1 ("protein" for protein-protein or "species" for protein-species). Default = "protein".
var2Rel relation of var2 ("protein" for protein-protein or "species" for protein-species). Default = "protein".
groupByCat group genes by their categories (TRUE or FALSE). Default = FALSE.
catDt dataframe contains gene categories (optional, NULL if groupByCat = FALSE or no info provided). Default = NULL.
var1AggregateBy aggregate method for VAR1 (max, min, mean or median), applied for calculating var1 of supertaxa. Default = "max".
var2AggregateBy aggregate method for VAR2 (max, min, mean or median), applied for calculating var2 of supertaxa. Default = "max".
Value

A filtered dataframe for generating profile plot including seed gene IDs (or orthologous group IDs), their ortholog IDs and the corresponding (super)taxa, (super)taxon IDs, number of co-orthologs in each (super)taxon, values for two additional variables var1, var2, supertaxon, and the categories of seed genes (or ortholog groups).

Author(s)

Vinh Tran tran@bio.uni-frankfurt.de

See Also

parseInfoProfile and reduceProfile for generating input dataframe, fullProcessedProfile for a demo full processed profile dataframe, fromInputToProfile for generating fully processed data from raw input.

Examples

# NOTE: this function require some intermediate steps using the results from # other functions. If you would like to get a full processed data from the # raw input, please use the function fromInputToProfile() instead!
data("fullProcessedProfile", package="PhyloProfile")
rankName <- "class"
refTaxon <- "Mammalia"
percentCutoff <- c(0.0, 1.0)
coorthologCutoffMax <- 10
var1Cutoff <- c(0.75, 1.0)
var2Cutoff <- c(0.5, 1.0)
var1Relation <- "protein"
var2Relation <- "species"
groupByCat <- FALSE
catDt <- NULL
var1AggregateBy <- "max"
var2AggregateBy <- "max"
taxonIDs <- levels(as.factor(fullProcessedProfile$ncbiID))
sortedInputTaxa <- sortInputTaxa(
    taxonIDs, rankName, refTaxon, NULL, NULL)
taxaCount <- plyr::count(sortedInputTaxa, "supertaxon")
filterProfileData(
    fullProcessedProfile, taxaCount, refTaxon, percentCutoff, coorthologCutoffMax, var1Cutoff, var2Cutoff, var1Relation, var2Relation, groupByCat, catDt,
finalProcessedProfile

```
var1AggregateBy,  
var2AggregateBy
```

finalProcessedProfile  An example of a final processed & filtered phylogenetic profile.

**Description**

An example of a final processed & filtered phylogenetic profile.

**Usage**

data(finalProcessedProfile)

**Value**

A data frame with 88 rows and 11 variables:

- `geneID` Seed or ortholog group ID, e.g. "100136at6656"
- `supertaxon` Supertaxon name together with its ordered index, e.g. "1001_Mammalia"
- `supertaxonID` Supertaxon ID (only different than ncbiID in case working with higher taxonomy rank than input's), e.g. "40674"
- `var1` First additional variable
- `presSpec` The percentage of species presenting in each supertaxon
- `category "cat"`
- `orthoID` Ortholog ID, e.g. "100136at6656lRAT@10116@1|G3V7R8@1"
- `var2` Second additional variable
- `paralog` Number of co-orthologs in the current taxon
- `presentTaxa` Number of taxa that have ortho in each supertaxon
- `totalTaxa` Total number of taxa in each supertaxon
fromInputToProfile

Complete processing of raw input phylogenetic profiles

Description

Create a processed and filtered data for plotting or analysing phylogenetic profiles from raw input file (from raw input to final filtered dataframe)

Usage

fromInputToProfile(rawInput, rankName, refTaxon = NULL, taxaTree = NULL, sortedTaxonList = NULL, var1AggregateBy = "max", var2AggregateBy = "max", percentCutoff = c(0, 1), coorthologCutoffMax = 9999, var1Cutoff = c(0, 1), var2Cutoff = c(0, 1), var1Relation = "protein", var2Relation = "protein", groupByCat = FALSE, catDt = NULL, taxDB = NULL)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rawInput</td>
<td>input file (in long, wide, multi-fasta or orthoxml format)</td>
</tr>
<tr>
<td>rankName</td>
<td>taxonomy rank (e.g. &quot;species&quot;,&quot;phylum&quot;,...)</td>
</tr>
<tr>
<td>refTaxon</td>
<td>selected reference taxon name (used for sorting and will be protected from filtering). Default = NULL.</td>
</tr>
<tr>
<td>taxaTree</td>
<td>input taxonomy tree for taxa in input profiles (optional). Default = NULL.</td>
</tr>
<tr>
<td>sortedTaxonList</td>
<td>list of sorted taxa (optional). Default = NULL.</td>
</tr>
<tr>
<td>var1AggregateBy</td>
<td>aggregate method for var1 (min, max, mean or median). Default = &quot;max&quot;.</td>
</tr>
<tr>
<td>var2AggregateBy</td>
<td>aggregate method for VAR2 (min, max, mean or median). Default = &quot;max&quot;.</td>
</tr>
<tr>
<td>percentCutoff</td>
<td>min and max cutoffs for percentage of species present in a supertaxon. Default = c(0, 1).</td>
</tr>
<tr>
<td>coorthologCutoffMax</td>
<td>maximum number of co-orthologs allowed. Default = 9999.</td>
</tr>
<tr>
<td>var1Cutoff</td>
<td>min and max cutoffs for var1. Default = c(0, 1).</td>
</tr>
<tr>
<td>var2Cutoff</td>
<td>min and max cutoffs for var2. Default = c(0, 1).</td>
</tr>
<tr>
<td>var1Relation</td>
<td>relation of var1 (&quot;protein&quot; for protein-protein or &quot;species&quot; for protein-species). Default = &quot;protein&quot;.</td>
</tr>
<tr>
<td>var2Relation</td>
<td>relation of var2 (&quot;protein&quot; for protein-protein or &quot;species&quot; for protein-species). Default = &quot;protein&quot;.</td>
</tr>
<tr>
<td>groupByCat</td>
<td>group genes by their categories (TRUE or FALSE). Default = FALSE.</td>
</tr>
<tr>
<td>catDt</td>
<td>dataframe contains gene categories. Default = NULL</td>
</tr>
<tr>
<td>taxDB</td>
<td>Path to the taxonomy DB files</td>
</tr>
</tbody>
</table>
fromInputToProfile

Value

Dataframe required for generating phylogenetic profile plot or clustering analysis. It contains seed gene IDs (or orthologous group IDs), their ortholog IDs and the corresponding (super)taxa, (super)taxon IDs, number of co-orthologs in each (super)taxon, values for two additional variables var1, var2, categories of seed genes (or ortholog groups).

Author(s)

Vinh Tran tran@bio.uni-frankfurt.de

See Also

createLongMatrix, getInputTaxaID, getInputTaxaName, sortInputTaxa, parseInfoProfile, reduceProfile, filterProfileData

Examples

rawInput <- system.file(
  "extdata", "test.main.long", package = "PhyloProfile", mustWork = TRUE
)
rankName <- "class"
refTaxon <- "Mammalia"
taxaTree <- NULL
sortedTaxonList <- NULL
var1AggregateBy <- "max"
var2AggregateBy <- "mean"
percentCutoff <- c(0.0, 1.0)
coorthologCutoffMax <- 10
var1Cutoff <- c(0.75, 1.0)
var2Cutoff <- c(0.5, 1.0)
var1Relation <- "protein"
var2Relation <- "species"
groupByCat <- FALSE
catDt <- NULL
fromInputToProfile(
  rawInput, rankName, refTaxon, taxaTree, sortedTaxonList, var1AggregateBy, var2AggregateBy, percentCutoff, coorthologCutoffMax, var1Cutoff, var2Cutoff, var1Relation, var2Relation, groupByCat, catDt
)
**fullProcessedProfile**  
An example of a fully processed phylogenetic profile.

**Description**  
An example of a fully processed phylogenetic profile.

**Usage**  
data(fullProcessedProfile)

**Value**  
A data frame with 168 rows and 14 variables:

- supertaxon Supertaxon name together with its ordered index, e.g. "1001_Mammalia"
- ncbiID Taxon ID, e.g. "ncbi10116"
- geneID Seed or ortholog group ID, e.g. "100136at6656"
- orthoID Ortholog ID, e.g. "100136at6656iHUMAN@9606@1@Q9UNQ2@l"
- var1 First additional variable
- var2 Second additional variable
- paralog Number of co-orthologs in the current taxon
- abbrName NCBI ID of the ortholog, e.g. "ncbi9606"
- taxonID Taxon ID of the ortholog, in this case: "0"
- fullName Full taxon name of the ortholog, e.g. "Homo sapiens"
- supertaxonID Supertaxon ID (only different than ncbiID in case working with higher taxonomy rank than input’s). e.g. "40674"
- rank Rank of the supertaxon, e.g. "class"
- category "cat"
- numberSpec Total number of species in each supertaxon

---

**geneAgePlotDf**  
Create data for plotting gene ages

**Description**  
Create data for plotting gene ages

**Usage**  
geneAgePlotDf(geneAgeDf)
generateSinglePlot

Create a single violin distribution plot

Description

Create a single violin distribution plot

Usage

generateSinglePlot(plotDf, parameters, variable)

Arguments

plotDf dataframe for plotting containing values for each variable in in-group and out-group.

parameters plot parameters, including size of x-axis, y-axis, legend and title; position of legend ("right", "bottom" or "none"); mean/median point; names of in-group and out-group; and plot title. NOTE: Leave blank or NULL to use default values.

variable name of variable that need to be plotted (one of the column names of input dataframe plotDf).
getAllDomainsOma

Value

A violin plot as a ggplot object.

Author(s)

Vinh Tran tran@bio.uni-frankfurt.de

Examples

data("mainLongRaw", package="PhyloProfile")
data <- mainLongRaw
inGroup <- c("ncbi9606", "ncbi10116")
varNames <- colnames(data)[c(4, 5)]
plotDf <- dataVarDistTaxGroup(data, inGroup, "101621at6656", varNames)
plotParameters <- list(
  "xSize" = 12,
  "ySize" = 12,
  "titleSize" = 15,
  "legendSize" = 12,
  "legendPosition" = "right",
  "mValue" = "mean",
  "inGroupName" = "In-group",
  "outGroupName" = "Out-group",
  "title" = "101621at6656"
)
generateSinglePlot(plotDf, plotParameters, colnames(plotDf)[1])

getAllDomainsOma

Create domain annotation dataframe from a raw OMA dataframe

Description

Create domain annotation dataframe from a raw OMA dataframe

Usage

g getAllDomainsOma(finalOmaDf = NULL)

Arguments

finalOmaDf raw OMA data for a list of proteins (see ?getDataForOneOma)

Value

Dataframe of the domain annotation used for PhyloProfile, which contains seed IDs, ortholog IDs, ortholog lengths, annotated features, start and end positions of those features.

Author(s)

Vinh Tran tran@bio.uni-frankfurt.de
getAllFastaOma

Get all fasta sequences from a raw OMA dataframe

Description

Get all fasta sequences from a raw OMA dataframe

Usage

ggetAllFastaOma(finalOmaDf = NULL)

Arguments

finalOmaDf raw OMA data for a list of proteins (see ?getDataForOneOma)

Value

A list contains all protein sequences in fasta format.

Author(s)

Vinh Tran tran@bio.uni-frankfurt.de

See Also

ggetDataForOneOma

to change the following line to run the function"
# omaData <- getDataForOneOma("HUMAN29397", "OG")
# getAllDomainsOma(omaData)

Examples
getCommonAncestor

Get all taxa that share a common ancestor

Description

Identify the common ancestor for a selected taxa and return a list of all taxa that have that common ancestor from an large input taxa set.

Usage

getCommonAncestor(inputTaxa = NULL, inGroup = NULL, taxDB = NULL)

Arguments

inputTaxa ID list of all input taxa (e.g. "ncbi12345")
inGroup ID list of selected taxa used for identify the common ancestor (e.g.: "ncbi55555")
taxDB Path to the taxonomy DB files

Value

A list containing the taxonomy rank and name of the common ancestor, together with a dataframe storing the full taxonomy info of all taxa that share that corresponding common ancestor.

Author(s)

Vinh Tran (tran@bio.uni-frankfurt.de)

Examples

```
inputTaxa <- c("ncbi34740", "ncbi9006", "ncbi374847", "ncbi123851",
              "ncbi5664", "ncbi189518", "ncbi418459", "ncbi10116", "ncbi1284812",
              "ncbi35128", "ncbi7070")
inGroup <- c("ncbi9006", "ncbi10116")
getCommonAncestor(inputTaxa, inGroup)
```

getCoreGene

Identify core genes for a list of selected taxa

Description

Identify core genes for a list of selected (super)taxa. The identified core genes must be present in at least a certain proportion of species in each selected (super)taxon (identified via percentCutoff) and that criteria must be fullfilled for a certain percentage of selected taxa or all of them (determined via coreCoverage).
Usage

getcogene(rankName, taxaCore = c("none"), profileDt, taxaCount,
  var1Cutoff = c(0, 1), var2Cutoff = c(0, 1), percentCutoff = c(0, 1),
  coreCoverage = 100, taxDB = NULL)

Arguments

rankName working taxonomy rank (e.g. "species", "genus", "family")
taxaCore list of selected taxon names
profileDt dataframe contains the full processed phylogenetic profiles (see ?fullProcessedProfile or ?parseInfoProfile)
taxaCount dataframe counting present taxa in each supertaxon
var1Cutoff cutoff for var1. Default = c(0, 1).
var2Cutoff cutoff for var2. Default = c(0, 1).
percentCutoff cutoff for percentage of species present in each supertaxon. Default = c(0, 1).
coreCoverage the least percentage of selected taxa should be considered. Default = 1.
taxDB Path to the taxonomy DB files

Value

A list of identified core genes.

Author(s)

Vinh Tran tran@bio.uni-frankfurt.de

See Also

parseInfoProfile for creating a full processed profile dataframe

Examples

data("fullProcessedProfile", package="PhyloProfile")
rankName <- "class"
refTaxon <- "Mammalia"
taxaCore <- c("Mammalia", "Saccharomycetes", "Insecta")
profileDt <- fullProcessedProfile
taxonIDs <- levels(as.factor(fullProcessedProfile$ncbiID))
sortedInputTaxa <- sortInputTaxa(
  taxonIDs, rankName, refTaxon, NULL, NULL)
taxaCount <- plyr::count(sortedInputTaxa, "supertaxon")
var1Cutoff <- c(0.75, 1.0)
var2Cutoff <- c(0.75, 1.0)
percentCutoff <- c(0.0, 1.0)
coreCoverage <- 100
getCoreGene(
  rankName,
getDataClustering

Get data for calculating distance matrix from phylogenetic profiles

Description

Get data for calculating distance matrix from phylogenetic profiles

Usage

ggetDataClustering(data, profileType = "binary", var1AggBy = "max", var2AggBy = "max")

Arguments

data: a data frame contains processed and filtered profiles (see ?fullProcessedProfile
and ?filterProfileData, ?fromInputToProfile)

profileType: type of data used for calculating the distance matrix. Either "binary" (consider
only the presence/absence status of orthlogs), "orthoID" (consider ortholog IDs
as values for clustering), "var1"/"var2" for taking values of the additional vari-
ables into account. Default = "binary".

var1AggBy: aggregate method for VAR1 (min, max, mean or median). Default = "max".

var2AggBy: aggregate method for VAR2 (min, max, mean or median). Default = "max".

Value

A wide dataframe contains values for calculating distance matrix.

Author(s)

Carla Mölbert (carla.moelbert@gmx.de), Vinh Tran (tran@bio.uni-frankfurt.de)

See Also

fromInputToProfile

Examples

data("finalProcessedProfile", package="PhyloProfile")
data <- finalProcessedProfile
profileType <- "binary"
var1AggregateBy <- "max"
var2AggregateBy <- "mean"
getDataClustering(data, profileType, var1AggregateBy, var2AggregateBy)
getDendrogram

getDendrogram

Plot dendrogram tree

Description
Plot dendrogram tree

Usage

getDendrogram(dd = NULL)

Arguments

dd dendrogram object (see ?clusterDataDend)

getDendrogram

Plot dendrogram tree

Description
Plot dendrogram tree

Usage

getDendrogram(dd = NULL)

Arguments

dd dendrogram object (see ?clusterDataDend)

getDendrogram

Plot dendrogram tree

Description
Plot dendrogram tree

Usage

getDendrogram(dd = NULL)

Arguments

dd dendrogram object (see ?clusterDataDend)
getDistanceMatrix

Calculate the distance matrix

Description
Calculate the distance matrix

Usage
getDistanceMatrix(profiles = NULL, method = "mutualInformation")

Arguments
- profiles: dataframe contains profile data for distance calculating (see \?getDataClustering)
- method: distance calculation method ("euclidean", "maximum", "manhattan", "canberra", "binary", "distanceCorrelation", "mutualInformation" or "pearson" for binary data; "distanceCorrelation" or "mutualInformation" for non-binary data). Default = "mutualInformation".

Value
A calculated distance matrix for input phylogenetic profiles.

Examples
```r
data("finalProcessedProfile", package="PhyloProfile")
data <- finalProcessedProfile
profileType <- "binary"
profiles <- getDataClustering(
  data, profileType, var1AggregateBy, var2AggregateBy)
distMethod <- "mutualInformation"
distanceMatrix <- getDistanceMatrix(profiles, distMethod)
clusterMethod <- "complete"
 dd <- clusterDataDend(distanceMatrix, clusterMethod)
getDendrogram(dd)
```
getDomainFolder

Get domain file from a folder for a seed protein

Description

Get domain file from a folder for a seed protein

Usage

generateDomainFolder(seed, domainPath)

Arguments

seed seed protein ID
domainPath path to domain folder

Value

Domain file and its complete directory path for the selected protein.

Author(s)

Vinh Tran tran@bio.uni-frankfurt.de

Examples

# Not run:
domainPath <- paste0(
  path.package("PhyloProfile", quiet = FALSE), "/extdata/domainFiles"
)
generateDomainFolder("OG_1009", domainPath)

# End(Not run)
getFastaFromFasInput  Get fasta sequences from main input file in multi-fasta format

Description
Get fasta sequences from main input file in multi-fasta format

Usage
getFastaFromFasInput(seqIDs = NULL, file = NULL)

Arguments
seqIDs  list of sequences IDs. Set seqIDs = "all" if you want to get all fasta sequences from the input file.
file  raw phylogenetic profile input file in multi-fasta format.

Value
A dataframe with one column contains sequences in fasta format.

Author(s)
Vinh Tran tran@bio.uni-frankfurt.de

Examples
file <- system.file(
  "extdata", "test.main.fasta",
  package = "PhyloProfile", mustWork = TRUE
)
getFastaFromFasInput("all", file)

getFastaFromFile  Get fasta sequences from main input file in multi-fasta format

Description
Get fasta sequences from main input file in multi-fasta format

Usage
getFastaFromFile(seqIDs = NULL, concatFasta = NULL)
**getFastaFromFolder**

**Arguments**

- **seqIDs**
  - list of sequences IDs. Set seqIDs = "all" if you want to get all fasta sequences from the concatenated input fasta file.
- **concatFasta**
  - input concatenated fasta file.

**Value**

A dataframe with one column contains sequences in fasta format.

**Author(s)**

Vinh Tran tran@bio.uni-frankfurt.de

**Examples**

```r
concatFasta <- system.file(
  "extdata", "fastaFiles/concatenatedFile.fa",
  package = "PhyloProfile", mustWork = TRUE
)
getFastaFromFasInput("all", concatFasta)
```

---

**Description**

Get fasta sequences for the input phylogenetic profiles.

**Usage**

```r
gETFastaFromFolder(seqIDs = NULL, path = NULL, dirFormat = NULL, fileExt = NULL, idFormat = NULL)
```

**Arguments**

- **seqIDs**
  - list of sequences IDs.
- **path**
  - path to fasta folder.
- **dirFormat**
  - directory format (either 1 for "path/speciesID.fa*" or 2 for "path/speciesID/speciesID.fa*")
- **fileExt**
  - fasta file extension ("fa", "fasta", "fas" or "txt")
- **idFormat**
  - fasta header format (1 for ">speciesID:seqID", 2 for ">speciesID@seqID", 3 for ">speciesID|seqID" or 4 for "seqID")

**Value**

A dataframe with one column contains sequences in fasta format.
getIDsRank

Author(s)

Vinh Tran tran@bio.uni-frankfurt.de

See Also

mainLongRaw

Examples

seqIDs <- "RAT@10116@1|D3ZUE4"
path <- system.file(
  "extdata", "fastaFiles", package = "PhyloProfile", mustWork = TRUE
)
dirFormat <- 1
fileExt <- "fa"
idFormat <- 3
getFastaFromFolder(seqIDs, path, dirFormat, fileExt, idFormat)

Description

Get NCBI taxonomy IDs, ranks and names for an input taxon list.

Usage

getIDsRank(inputTaxa = NULL, currentNCBIinfo = NULL)

Arguments

inputTaxa NCBI ID list of input taxa.
currentNCBIinfo table/dataframe of the pre-processed NCBI taxonomy data (/PhyloProfile/data/preProcessedTaxonomy.txt)

Value

A list of 3 dataframes: idList, rankList and reducedInfoList. The "rankList" contains taxon names and all taxonomy ranks of the input taxa including also the noranks from the input rank to the taxonomy root. The "idList" contains input taxon IDs, taxon names, all the ranks from current rank to the taxonomy root together with their IDs (with the format "id#rank"). The reducedInfoList is a subset of preProcessedTaxonomy.txt file, containing the NCBI IDs, taxon fullnames, their current rank and their direct parent ID.

Author(s)

Vinh Tran tran@bio.uni-frankfurt.de
getInputTaxaID

Examples

inputTaxa <- c("272557", "176299")
ncbiFilein <- system.file(  
    "extdata", "data/preProcessedTaxonomy.txt",  
    package = "PhyloProfile", mustWork = TRUE  
)
currentNCBIinfo <- as.data.frame(data.table::fread(ncbiFilein))
getIDsRank(inputTaxa, currentNCBIinfo)

g getInputTaxaID  

Get ID list of input taxa from the main input

Description

Get ID list of input taxa from the main input

Usage

getInputTaxaID(rawProfile = NULL)

Arguments

rawProfile  

A dataframe of input phylogenetic profile in long format

Value

List of all input taxon IDs (e.g. ncbi1234). Default = NULL.

Author(s)

Vinh Tran tran@bio.uni-frankfurt.de

See Also

createLongMatrix, mainLongRaw

Examples

data("mainLongRaw", package="PhyloProfile")
getInputTaxaID(mainLongRaw)
getInputTaxaName

**Get NCBI taxon names for a selected list of taxa**

**Description**

Get NCBI taxon names from "PhyloProfile/data/taxonNamesReduced.txt" for a list of input taxa.

**Usage**

```r
g getInputTaxaName(rankName, taxonIDs = NULL, taxDB = NULL)
```

**Arguments**

- `rankName`: taxonomy rank (e.g. "species", "phylum", ...)
- `taxonIDs`: list of taxon IDs (e.g. ncbi1234). Default = NULL
- `taxDB`: Path to the taxonomy DB files

**Value**

Data frame contains a list of full names, taxonomy ranks and parent IDs for the input taxa.

**Author(s)**

Vinh Tran tran@bio.uni-frankfurt.de

**See Also**

- `getInputTaxaID` for getting input taxon IDs, `getNameList` for getting the full taxon name list

**Examples**

```r
taxonIDs <- c("ncbi9606", "ncbi10116")
g getInputTaxaName("species", taxonIDs)
```

---

getNameList

**Get list of pre-installed NCBI taxon names**

**Description**

Get all NCBI taxon names from "PhyloProfile/data/taxonNamesReduced.txt"

**Usage**

```r
g getNameList(taxDB = NULL)
```
getOmaDataForOneOrtholog

Get taxonomy ID, sequence and annotation for one OMA protein

Description

Get taxonomy ID, sequence and annotation for one OMA protein

Usage

getOmaDataForOneOrtholog(id = NULL)

Arguments

id oma ID of one protein

Value

Data frame contains the input protein ID with its taxonomy ID, sequence, length and domain annotations (tab delimited) for input OMA protein

Author(s)

Vinh Tran tran@bio.uni-frankfurt.de

Examples

print("Uncomment the following line to run the function")
# getOmaDataForOneOrtholog("HUMAN29397")
getOmaDomainFromURL  
*Get domain annotation from OMA Browser*

**Description**
Get domain annotation from OMA Browser based on a URL or a raw data frame contains annotation info from OMA

**Usage**
```r
getOmaDomainFromURL(domainURL = NULL)
```

**Arguments**
- `domainURL` URL address for domain annotation of ONE OMA id or a raw data frame contains annotation info from OMA

**Value**
Data frame contains feature names with their start and end positions

**Author(s)**
Vinh Tran tran@bio.uni-frankfurt.de

**Examples**
```r
print("Uncomment the following line to run the function")
# getOmaDomainFromURL("https://omabrowser.org/api/protein/7916808/domains/")
```

---

getOmaMembers  
*Get OMA members*

**Description**
Get OMA ortholog group, OMA HOG or OMA pair’s members for a seed protein from OMA Browser.

**Usage**
```r
getoMaMembers(id = NULL, orthoType = "OG")
```

**Arguments**
- `id` ID of the seed protein (OMA or UniProt ID)
- `orthoType` type of OMA orthologs: either "HOG", "OG" (orthologous group) or "PAIR" (orthologous pair - CURRENTLY NOT WORKING). Default = "OG". 
getQualColForVector

**Value**

List of OMA orthologs for an input seed protein.

**Author(s)**

Carla Mölbert carla.moelbert@gmx.de

**Examples**

```r
print("Uncomment the following line to run the function")
# getOmaMembers("HUMAN29397", "OG")
```

---

getQualColForVector  Get color for a list of items

**Description**

Get color for a list of items

**Usage**

```r
getQualColForVector(x = NULL)
```

**Arguments**

- `x` input list

**Value**

list of colors for each element (same elements will have the same color)

**Author(s)**

Vinh Tran tran@bio.uni-frankfurt.de

**See Also**

- `qualitativeColours`

**Examples**

```r
items <- c("a", "b", "c")
getQualColForVector(items)
```
**getSelectedFastaOma**  
Get selected fasta sequences from a raw OMA dataframe

**Description**  
Get selected fasta sequences from a raw OMA dataframe

**Usage**  
```r  
getSelectedFastaOma(finalOmaDf = NULL, seqID = NULL)  
```

**Arguments**  
- `finalOmaDf`: raw OMA data for a list of proteins (see `getDataForOneOma`)  
- `seqID`: OMA ID of selected protein

**Value**  
Required protein sequence in fasta format.

**Author(s)**  
Vinh Tran tran@bio.uni-frankfurt.de

**See Also**  
`getDataForOneOma`

**Examples**  
```r  
print("Uncomment the following line to run the function")  
# omaData <- getDataForOneOma("HUMAN29397", "OG")  
# getSelectedFastaOma(omaData, "HUMAN29397")  
```

**getSelectedTaxonNames**  
Get a subset of input taxa based on a selected taxonomy rank

**Description**  
Get a subset of taxon ncbi IDs and names from an input list of taxa based on a selected supertaxon (identified by its taxonomy rank and supertaxon name or supertaxon ID).

**Usage**  
```r  
getSelectedTaxonNames(inputTaxonIDs = NULL, rank = NULL,  
                       higherRank = NULL, higherID = NULL, higherName = NULL, taxDB = NULL)  
```
getTaxHierarchy

Get taxonomy hierarchy for a list of taxon IDs

Arguments

- **inputTaxonIDs**: list of input taxon IDs (e.g. `c("10116", "122586")`)
- **rank**: taxonomy rank of input taxa (e.g. "species")
- **higherRank**: selected taxonomy rank (e.g. "phylum")
- **higherID**: supertaxon ID (e.g. 7711). NOTE: either supertaxon ID or name is required, not necessary to give both
- **higherName**: supertaxon name (e.g. "Chordata"). NOTE: either supertaxon ID or name is required, not necessary to give both
- **taxDB**: Path to the taxonomy DB files

Value

A data frame contains ncbi IDs and names of taxa from the input taxon list that belong to the selected supertaxon.

Author(s)

Vinh Tran tran@bio.uni-frankfurt.de

Examples

```r
inputTaxonIDs <- c("10116", "122586", "123851", "13616", "188937", "189518", "208964", "224129", "224324", "237631", "243230")
rank <- "species"
higherRank <- "phylum"
higherID <- 7711
getSelectedTaxonNames(inputTaxonIDs, rank, higherRank, higherID, NULL)
higherName <- "Chordata"
getSelectedTaxonNames(inputTaxonIDs, rank, higherRank, NULL, higherName, NULL)
```

getTaxHierarchy

Get taxonomy hierarchy for a list of taxon IDs

Description

Get NCBI taxonomy hierarchy and URLs for an input taxon list.

Usage

```r
getTaxHierarchy(inputTaxa = NULL, currentNCBIinfo = NULL)
```

Arguments

- **inputTaxa**: NCBI ID list of input taxa.
- **currentNCBIinfo**: table/dataframe of the pre-processed NCBI taxonomy data (/PhyloProfile/data/preProcessedTaxonomy.txt)
getTaxonomyInfo

Value

A list of dataframes containing taxonomy hierarchy and its URL to NCBI database for input taxon IDs

Author(s)

Vinh Tran tran@bio.uni-frankfurt.de

Examples

```r
inputTaxa <- c("272557", "176299")
ncbiFilein <- system.file("extdata", "data/preProcessedTaxonomy.txt", package = "PhyloProfile", mustWork = TRUE)
currentNCBIinfo <- as.data.frame(data.table::fread(ncbiFilein))
PhyloProfile:::getTaxHierarchy(inputTaxa, currentNCBIinfo)
```

---

getTaxonomyInfo Get taxonomy info for a list of input taxa

Description

Get taxonomy info for a list of input taxa

Usage

```r
getTaxonomyInfo(inputTaxa = NULL, currentNCBIinfo = NULL)
```

Arguments

- `inputTaxa`: NCBI taxonomy IDs of input taxa.
- `currentNCBIinfo`: table/dataframe of the pre-processed NCBI taxonomy data (/PhyloProfile/data/preProcessedTaxonomy.txt)

Value

A list of NCBI taxonomy info for input taxa, including the taxonomy IDs, full scientific names, taxonomy ranks and the parent IDs.

Author(s)

Vinh Tran tran@bio.uni-frankfurt.de
getTaxonomyMatrix

Examples

inputTaxa <- c("272557", "176299")
ncbiFilein <- system.file("extdata", "data/preProcessedTaxonomy.txt", package = "PhyloProfile", mustWork = TRUE)
currentNCBIinfo <- as.data.frame(data.table::fread(ncbiFilein))
getTaxonomyInfo(inputTaxa, currentNCBIinfo)

getTaxonomyMatrix Get taxonomy matrix

Description
Get the (full or subset) taxonomy matrix from "data/taxonomyMatrix.txt" based on an input taxon list

Usage
getTaxonomyMatrix(taxDB = NULL, subsetTaxaCheck = FALSE, taxonIDs = NULL)

Arguments
taxDB Path to the taxonomy DB files
subsetTaxaCheck TRUE/FALSE subset taxonomy matrix based on input taxon IDs. Default = FALSE
taxonIDs list of input taxon IDs (e.g. ncbi1234). Default = NULL

Value
Data frame contains the (subset of) taxonomy matrix for list of input taxa.

Author(s)
Vinh Tran tran@bio.uni-frankfurt.de

Examples
# get full pre-installed taxonomy matrix
getTaxonomyMatrix()
# get taxonomy matrix for a list of taxon IDs
taxonIDs <- c("ncbi9606", "ncbi10116")
getTaxonomyMatrix(NULL, TRUE, taxonIDs)
**getTaxonomyRanks**

Create a list containing all main taxonomy ranks

**Description**

Create a list containing all main taxonomy ranks

**Usage**

`getTaxonomyRanks()`

**Value**

A list of all ranks (from strain to superkingdom)

**Author(s)**

Carla Mölbert (carla.moelbert@gmx.de)

**Examples**

`getTaxonomyRanks()`

---

**gridArrangeSharedLegend**

*Plot Multiple Graphs with Shared Legend in a Grid*

**Description**

Plot Multiple Graphs with Shared Legend in a Grid

**Usage**

`gridArrangeSharedLegend(..., ncol = length(list(...)), nrow = 1, position = c("bottom", "right"), title = NA, titleSize = 12)`

**Arguments**

- `...` Plots to be arranged in grid
- `ncol` Number of columns in grid
- `nrow` Number of rows in grid
- `position` Grid position (bottom or right)
- `title` Title of grid
- `titleSize` Size of grid title
Value

Grid of plots with common legend

Note

adapted from https://rdrr.io/github/PhilBoileau/CLSAR/src/R/gridArrangeSharedLegend.R

Author(s)

Phil Boileau, <philippe.boileau@rimuhc.ca>

Examples

data("mainLongRaw", package="PhyloProfile")
data <- mainLongRaw
inGroup <- c("ncbi9606", "ncbi10116")
varNames <- colnames(data)[c(4, 5)]
plotDf <- dataVarDistTaxGroup(data, inGroup, "101621at6656", varNames)
plotParameters <- list(
  "xSize" = 12,
  "ySize" = 12,
  "titleSize" = 15,
  "legendSize" = 12,
  "legendPosition" = "right",
  "mValue" = "mean",
  "inGroupName" = "In-group",
  "outGroupName" = "Out-group",
  "title" = "101621at6656"
)
plotVar1 <- generateSinglePlot(plotDf, plotParameters, colnames(plotDf)[1])
plotVar2 <- generateSinglePlot(plotDf, plotParameters, colnames(plotDf)[2])
g <- gridArrangeSharedLegend(
  plotVar1, plotVar2,
  position = plotParameters$legendPosition,
  title = plotParameters$title,
  size = plotParameters$titleSize
)

heatmapPlotting

Create profile heatmap plot

Description

Create profile heatmap plot

Usage

heatmapPlotting(data = NULL, parm = NULL)
Arguments

data dataframe for plotting the heatmap phylogenetic profile (either full or subset profiles)

parm plot parameters, including (1) type of x-axis "taxa" or "genes" - default = "taxa";
(2+3) names of 2 variables var1ID and var2ID - default = "var1" & "var2"; (4+5)
mid value and color for mid value of var1 - default is 0.5 and #FFFFFF; (6)
mid value and color for mid value of var2 - default is 1 and #FFFFFF;
(7) color for lowest var1 - default = "#FF8C00"; (8) color for highest var1 - default
is 0 and #07D000; (12) color of co-orthologs - default = "#FF8C00"; (13+14+15) mid value and color for mid value of var2 - default is 1 and #FFFFFF;
(10) color for lowest var2 - default = "#FF8C00"; (11) color
(13+14+15) text sizes for x, y axis and legend - default = 9 for each; (16) legend position "top", "bottom", "right", "left" or "none" - default = "top";
(17) zoom ratio of the co-ortholog dots from -1 to 3 - default = 0; (18)
angle of x-axis from 0 to 90 - default = 60; (19) show/hide separate line for reference taxa - default = 0; (20) enable/disable coloring gene categories
TRUE/FALSE - default = FALSE; (21) enable/disable coloring duplicated ortholog IDs TRUE/FALSE - default=FALSE. NOTE: Leave blank or NULL to use default values.

Value

A profile heatmap plot as a ggplot object.

Author(s)

Vinh Tran tran@bio.uni-frankfurt.de

See Also

dataMainPlot, dataCustomizedPlot

Examples

data("finalProcessedProfile", package="PhyloProfile")
plotDf <- dataMainPlot(finalProcessedProfile)
plotParameter <- list(
  "xAxis" = "taxa",
  "var1ID" = "FAS_FW",
  "var2ID" = "FAS_BW",
  "midVar1" = 0.5,
  "midColorVar1" = "#FFFFFF",
  "lowColorVar1" = "#FF8C00",
  "highColorVar1" = "#4682B4",
  "midVar2" = 1,
  "midColorVar2" = "#FFFFFF",
  "lowColorVar2" = "#CB4C4E",
  "highColorVar2" = "#3E436F",
  "paraColor" = "#07D000",
  "xSize" = 8,
  "ySize" = 8,
highlightProfilePlot

Highlight gene and/or taxon of interest on the phylogenetic profile plot

Description
Highlight gene and/or taxon of interest on the phylogenetic profile plot

Usage
highlightProfilePlot(profilePlot = NULL, plotDf = NULL,
taxonHighlight = "none", workingRank = "none", geneHighlight = NULL,
taxDB = NULL, xAxis = "taxa")

Arguments

profilePlot initial (highlighted) profile plot
plotDf dataframe for plotting the heatmap phylogentic profile
taxonHighlight taxon of interest. Default = "none".
workingRank working taxonomy rank (needed only for highlight taxon).
geneHighlight gene of interest. Default = NULL.
taxDB Path to the taxonomy DB files
xAxis type of x-axis (either "genes" or "taxa")

Value
A profile heatmap plot with highlighted gene and/or taxon of interest as ggplot object.

Author(s)
Vinh Tran tran@bio.uni-frankfurt.de

See Also
dataMainPlot, dataCustomizedPlot, heatmapPlotting
id2name

Get taxon names for a list of taxon IDs

Description

Get taxon names for a list of taxon IDs

Usage

id2name(idList = NULL, currentNCBIinfo = NULL)

Arguments

idList

list of taxonomy IDs

currentNCBIinfo
table/dataframe of the pre-processed NCBI taxonomy data (/PhyloProfile/data/preProcessedTaxonomy.txt)
Value

A dataframe contains input taxon IDs and their full names.

Author(s)

Vinh Tran tran@bio.uni-frankfurt.de

Examples

```r
ncbiFilein <- system.file(
  "extdata", "data/preProcessedTaxonomy.txt",
  package = "PhyloProfile", mustWork = TRUE
)
currentNCBIinfo <- as.data.frame(data.table::fread(ncbiFilein))
idList <- c("9606", "5207", "40674", "4751")
id2name(idList, currentNCBIinfo)
```

<table>
<thead>
<tr>
<th>idList</th>
<th>NCBI ID list for experimental data sets</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Description

Data frame, in which each row contains the complete taxonomy ranks from the lowest systematic level (strain/species) upto the taxonomy root and the corresponding IDs for one taxon in the experimental data sets.

Usage

data(idList)

Value

A data frame with up to 41 columns and 95 rows corresponding to 95 taxa in the 2 experimental data sets.

<table>
<thead>
<tr>
<th>mainLongRaw</th>
<th>An example of a raw long input file.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Description

An example of a raw long input file.

Usage

data(mainLongRaw)
mainTaxonomyRank

Value

A data frame with 168 rows and 5 variables:

- geneID Seed or ortholog group ID, e.g. "100136at6656"
- ncbiID Taxon ID, e.g. "ncbi36329"
- orthoID Ortholog ID, e.g. "100136at6656|PLAF7@36329@1Q8ILT8|1"
- FAS_F First additional variable
- FAS_B Second additional variable

Description

Get all NCBI taxonomy rank names

Usage

mainTaxonomyRank()

Value

A list of all available NCBI taxonomy rank names.

Author(s)

Vinh Tran tran@bio.uni-frankfurt.de

Examples

mainTaxonomyRank()

modifyFeatureName

Modify feature names

Description

Simplify feature names (e.g. TM for transmembrane domain, LCR for low complexity regions, remove tool names from domain name) and add weight to feature names (if available)

Usage

modifyFeatureName(domainDf = NULL)
pairDomainPlotting

Create architecture plot for a pair of seed and ortholog protein

Description

Create architecture plot for a pair of seed and ortholog protein

Usage

pairDomainPlotting(seed, ortho, seedDf, orthoDf, minStart, maxEnd, labelSize, titleSize)

Arguments

seed                     Seed ID
ortho                    Ortho ID
seedDf                   domain dataframe for seed domains containing the seed ID, ortholog ID, sequence length, feature names, start and end positions, feature weights (optional) and the status to determine if that feature is important for comparison the architecture between 2 proteins* (e.g. seed protein vs ortholog) (optional).
orthoDf                  domain dataframe for ortholog domains (same format as seedDf).
minStart                 the smallest start position of all domains
maxEnd                   the highest stop position of all domains
parseDomainInput

Parse domain input file

Description

Get all domain annotations for one seed protein IDs.

Usage

parseDomainInput(seed = NULL, inputFile = NULL, type = "file")

Arguments

seed  
seed protein ID

inputFile  
name of input file (file name or path to folder contains individual domain files)

type  
type of data ("file" or "folder"). Default = "file".
Value

A dataframe for protein domains including seed ID, its orthologs IDs, sequence lengths, feature names, start and end positions, feature weights (optional) and the status to determine if that feature is important for comparison the architecture between 2 proteins* (e.g. seed protein vs ortholog) (optional).

Author(s)

Vinh Tran tran@bio.uni-frankfurt.de

See Also

getDomainFolder

Examples

seed <- "101621at6656"
inputFile <- system.file("extdata", "domainFiles/101621at6656.domains",
package = "PhyloProfile", mustWork = TRUE)
type <- "file"
parseDomainInput(seed, inputFile, type)

Description

Creating main dataframe for the input phylogenetic profiles based on selected input taxonomy level (e.g. strain, species) and reference taxon. The output contains the number of paralogs, the max/min/mean/median of VAR1 and VAR2.

Usage

parseInfoProfile(inputDf, sortedInputTaxa, taxaCount, coorthoCOMax)

Arguments

inputDf input profiles in long format
sortedInputTaxa sorted taxonomy data for the input taxa (check sortInputTaxa())
taxaCount dataframe counting present taxa in each supertaxon
corthoCOMax maximum number of co-orthologs allowed
Value
A dataframe contains all info for the input phylogenetic profiles. This full processed profile that is required for several profiling analyses e.g. estimation of gene age (\texttt{estimateGeneAge}) or identification of core gene (\texttt{getCoreGene}).

Author(s)
Vinh Tran tran@bio.uni-frankfurt.de

See Also
\texttt{createLongMatrix}, \texttt{sortInputTaxa}, \texttt{calcPresSpec}, \texttt{mainLongRaw}

Examples
\begin{verbatim}
data("mainLongRaw", package="PhyloProfile")
taxonIDs <- getInputTaxaID(mainLongRaw)
sortedInputTaxa <- sortInputTaxa(
    taxonIDs, "class", "Mammalia", NULL, NULL
)
taxaCount <- plyr::count(sortedInputTaxa, "supertaxon")
coorthoCOMax <- 999
parseInfoProfile(
    mainLongRaw, sortedInputTaxa, taxaCount, coorthoCOMax
)
\end{verbatim}

---

**ppTaxonomyMatrix**  
An example of a taxonomy matrix.

Description
An example of a taxonomy matrix.

Usage
```r
data(ppTaxonomyMatrix)
```

Value
A data frame with 10 rows and 162 variables:

- abbrName e.g. "ncbi10090"
- ncbiID e.g. "10090"
- fullName e.g. "Mus musculus"
- strain e.g. "10090" ...
### processNcbiTaxonomy

**Description**

Download NCBI taxonomy database and parse information that are needed for PhyloProfile, including taxon IDs, their scientific names, systematic ranks, and parent (next higher) rank IDs.

**Usage**

```r
processNcbiTaxonomy()
```

**Value**

A dataframe contains NCBI taxon IDs, taxon names, taxon ranks and the next higher taxon IDs (parent’s IDs) of all taxa in the NCBI taxonomy database.

**Author(s)**

Vinh Tran tran@bio.uni-frankfurt.de
processOrthoID

Examples

?processNcbiTaxonomy
## Not run:
preProcessedTaxonomy <- PhyloProfile:::processNcbiTaxonomy()
# save to text (tab-delimited) file
write.table(
  preProcessedTaxonomy,
  file = "preProcessedTaxonomy.txt",
  col.names = TRUE,
  row.names = FALSE,
  quote = FALSE,
  sep = "\t"
)
# save to rdata file
save(
  preProcessedTaxonomy, file = "preProcessedTaxonomy.RData", compress='xz'
)
## End(Not run)

__processOrthoID__  __Process ortholog IDs__

Description

Process ortholog IDs to identify duplicated IDs

Usage

processOrthoID(dataHeat = NULL)

Arguments

dataHeat  a data frame contains processed profiles (see ?fullProcessedProfile, ?filterProfileData)

Value

the same dataframe as input, but the ortholog IDs are changed into <taxID:orthoID>. New column orthoFreq specifies if the ortholog IDs are single or duplicated

Author(s)

Vinh Tran tran@bio.uni-frankfurt.de
profileWithTaxonomy

An example of a raw long input file together with the taxonomy info.

Description

An example of a raw long input file together with the taxonomy info.

Usage

data(profileWithTaxonomy)

Value

A data frame with 20 rows and 12 variables:

- geneID Seed or ortholog group ID, e.g. "OG_1017"
- ncbiID Taxon ID, e.g. "ncbi176299"
- orthoID Ortholog ID, e.g. "A.fabrum@176299@1582"
- var1 First additional variable
- var2 Second additional variable
- paralog Number of co-orthologs in the current taxon
- abbrName e.g. "ncbi176299"
- taxonID Taxon ID, e.g. "176299"
- fullName Full taxon name, e.g. "Agrobacterium fabrum str. C58"
- supertaxonID Supertaxon ID (only different than ncbiID in case working with higher taxon-
  omy rank than input's)
- supertaxon Name of the corresponding supertaxon
- rank Rank of the supertaxon
qualitativeColours

Create qualitative colours

Description
Create qualitative colours

Usage
qualitativeColours(n, light = FALSE)

Arguments
- n number of colors
- light light colors TRUE or FALSE

Value
list of n different colors

Source
Modified based on https://gist.github.com/peterk87/6011397

Examples
## Not run:
PhyloProfile:::qualitativeColours(5)

## End(Not run)

rankIndexing

Indexing all available ranks (including norank)

Description
Indexing all available ranks (including norank)

Usage
rankIndexing(rankListFile = NULL)

Arguments
- rankListFile Input file, where each row is a rank list of a taxon (see rankListFile in example)
Value
A dataframe containing a list of all possible ranks and their indexed values.

Author(s)
Vinh Tran tran@bio.uni-frankfurt.de

Examples
## Not run:
rankListFile <- system.file(
    "extdata", "data/rankList.txt", package = "PhyloProfile", mustWork = TRUE
)
PhyloProfile:::rankIndexing(rankListFile)
## End(Not run)

rankList | NCBI rank list for experimental data sets

Description
Data frame, in which each row contains the complete taxonomy ranks from the lowest systematic level (strain/species) up to the taxonomy root for one taxon in the experimental data sets.

Usage
data(rankList)

Value
A data frame with up to 41 columns and 95 rows corresponding to 95 taxa in the 2 experimental data sets

reduceProfile | Reduce the filtered profile data into supertaxon level

Description
Reduce data of the processed phylogenetic profiles from input taxonomy rank into supertaxon level (e.g. from species to phylum)

Usage
reduceProfile(filteredProfile)
Arguments

filteredProfile
dataframe contains the filtered profiles (see ?parseInfoProfile, ?filterProfileData
and ?filteredProfile)

Value

A reduced dataframe contains only profile data for the selected supertaxon rank. This dataframe
contains only supertaxa and their value (mVar1 & mVar2) for each gene.

Author(s)

Vinh Tran tran@bio.uni-frankfurt.de

See Also

parseInfoProfile for creating a full processed profile dataframe, filterProfileData for filter
processed profile and filteredProfile for a demo filtered profile dataframe

Examples

data("filteredProfile", package="PhyloProfile")
reduceProfile(filteredProfile)

Description

Run PhyloProfile app

Usage

runPhyloProfile(configFile = NULL, host = NULL, port = NULL)

Arguments

configFile Configuration file for specifying path to input files, taxonomy rank and reference
taxon, and some other settings
host IP adress (e.g. host = "127.0.0.1")
port Port (e.g. port = 8888)

Value

A shiny application - GUI version of PhyloProfile
**singleDomainPlotting**

Create architecture plot for a single protein

**Description**

Create architecture plot for a single protein

**Usage**

```r
singleDomainPlotting(df, geneID = "GeneID", sep = "|", labelSize = 12,
                      titleSize = 12, minStart = NULL, maxEnd = NULL, colorScheme)
```

**Arguments**

- `df`: domain dataframe for plotting containing the seed ID, ortholog ID, ortholog sequence length, feature names, start and end positions, feature weights (optional) and the status to determine if that feature is important for comparison the architecture between 2 proteins* (e.g. seed protein vs ortholog) (optional).
- `geneID`: ID of seed or orthologous protein
- `sep`: separate indicator for title. Default = "|".
- `labelSize`: label size. Default = 12.
- `minStart`: the smallest start position of all domains
- `maxEnd`: the highest stop position of all domains
- `colorScheme`: color scheme for all domain types

**Value**

Domain plot of a single protein as a ggplot object.

**Author(s)**

Vinh Tran tran@bio.uni-frankfurt.de

**See Also**

`getQualColForVector, parseDomainInput`
sortDomains

Sort one domain dataframe based on the other domain dataframe

Description

Sort domain dataframe of one protein (either seed or ortholog) based on the dataframe of its paired protein, in order to bring the common domain feature in the same order which make it easy for comparing.

Usage

sortDomains(seedDf, orthoDf)
Arguments

seedDf     data of seed protein
orthoDf    data of ortholog protein

Value

Dataframe contains sorted domain list.

Author(s)

Vinh Tran tran@bio.uni-frankfurt.de

Examples

```r
## Not run:
# get domain data
domainFile <- system.file(
  "extdata", "domainFiles/101621at6656.domains",
  package = "PhylProfile", mustWork = TRUE
)
domainDF <- parseDomainInput(seedID, domainFile, "file")
# get seedDf and orthoDf
subDF <- domainDF[
  domainDF$seedID ==
  "101621at6656#101621at6656:AGRPL@224129@0:224129_0:001955:1",]
orthoDF <- subDF[subDF$orthoID == "101621at6656:DROME@7227@1:Q9VG04",]
seedDF <- subDF[subDF$orthoID != "101621at6656:DROME@7227@1:Q9VG04",]
# sort
PhylProfile:::sortDomains(seedDF, orthoDF)
## End(Not run)
```

---

**sortInputTaxa**

Sort list of (super)taxa based on a selected reference (super)taxon

Description

Sort list of (super)taxa based on a selected reference (super)taxon

Usage

```r
sortInputTaxa(taxonIDs = NULL, rankName, refTaxon = NULL,
taxaTree = NULL, sortedTaxonList = NULL, taxDB = NULL)
```
sortTaxaFromTree

**Arguments**

- **taxonIDs**: list of taxon IDs (e.g.: ncbi1234, ncbi9999,...). Default = NULL
- **rankName**: working taxonomy rank (e.g. "species", "phylum",...)
- **refTaxon**: selected reference taxon. Default = NULL
- **taxaTree**: taxonomy tree for the input taxa (optional). Default = NULL
- **sortedTaxonList**: list of sorted taxa (optional). Default = NULL
- **taxDB**: Path to the taxonomy DB files

**Value**

A taxonomy matrix for the input taxa ordered by the selected reference taxon. This matrix is sorted either based on the NCBI taxonomy info, or based on an user-defined taxonomy tree (if provided).

**Author(s)**

Vinh Tran tran@bio.uni-frankfurt.de

**See Also**

getNameList, getTaxonomyMatrix, createUnrootedTree, sortTaxaFromTree, getInputTaxaName, getInputTaxaID, createLongMatrix

**Examples**

taxonIDs <- c(
  "ncbi10116", "ncbi123851", "ncbi3702", "ncbi13616", "ncbi9606"
)
sortInputTaxa(taxonIDs, "species", "Homo sapiens", NULL, NULL)

---

sortTaxaFromTree **Get sorted supertaxon list based on a rooted taxonomy tree**

**Description**

Get sorted supertaxon list based on a rooted taxonomy tree

**Usage**

sortTaxaFromTree(tree)

**Arguments**

- **tree**: an "phylo" object for a rooted taxonomy tree
Value

A list of sorted taxa obtained the input taxonomy tree.

Author(s)

Vinh Tran tran@bio.uni-frankfurt.de

See Also

ppTaxonomyMatrix for a demo taxonomy matrix data

Examples

data("ppTaxonomyMatrix", package = "PhyloProfile")
# create taxonomy tree rooted by ncbi10090
tree <- createUnrootedTree(ppTaxonomyMatrix)
rootedTree <- ape::root(tree, outgroup = "ncbi10090", resolve.root = TRUE)
# get taxon list sorted from tree
sortTaxaFromTree(rootedTree)

Description

taxa2dist

Usage

taxa2dist(x, varstep = FALSE, check = TRUE, labels)

Arguments

x       taxa matrix
varstep   var-step
check     check
labels    labels

Value

a distance matrix

Author(s)

function from taxize library
**taxonNamesReduced**

**NCBI Taxonomy reduced data set**

**Description**

A list of NCBI taxonomy info (including taxon IDs, taxon names, their systematic taxonomy rank and IDs of their next rank - parent IDs) for 95 taxa in two experimental sets included in PhyloProfileData package.

**Usage**

```r
data(taxonNamesReduced)
```

**Value**

A data frame with 4 columns:

- **ncbiID** e.g. "10090"
- **fullName** e.g. "Mus musculus"
- **rank** e.g. "species"
- **parentID** e.g. "862507"

---

**taxonomyMatrix**

**Taxonomy matrix for experimental data sets**

**Description**

Data frame containing the fully aligned taxonomy IDs of 95 taxa in the experimental data sets. By taking into account both the defined ranks (e.g. strain), this data is used for clustering and then creating a taxon tree. It is used also for cross-linking between different taxonomy ranks within a taxon.

**Usage**

```r
data(taxonomyMatrix)
```

**Value**

A data frame with up to 149 columns and 95 rows corresponding to 95 taxa in the 2 experimental data sets
 taxonomyTableCreator  
Align NCBI taxonomy IDs of list of taxa into a sorted rank list.

Description
Align NCBI taxonomy IDs of list of taxa into a sorted rank list.

Usage
taxonomyTableCreator(idListFile = NULL, rankListFile = NULL)

Arguments
idListFile  a text file whose each row is a rank+ID list of a taxon (see idListFile in example)
rankListFile a text file whose each row is a rank list of a taxon (see rankListFile in example)

Value
An aligned taxonomy dataframe which contains all the available taxonomy ranks from the id and rank list file. This dataframe can be used for creating a well resolved taxonomy tree (see ?createUnrootedTree) and sorting taxa based on a selected reference taxon (see ?sortInputTaxa).

Author(s)
Vinh Tran tran@bio.uni-frankfurt.de

See Also
rankIndexing, createUnrootedTree, sortInputTaxa

Examples
idListFile <- system.file(  
  "extdata", "data/idList.txt", package = "PhyloProfile", mustWork = TRUE  
)
rankListFile <- system.file(  
  "extdata", "data/rankList.txt", package = "PhyloProfile", mustWork = TRUE  
)
taxonomyTableCreator(idListFile, rankListFile)
varDistTaxPlot  

Create variable distribution comparison plot

Description

Create variable distribution plots between 2 groups of taxa for a selected gene.

Usage

varDistTaxPlot(data, plotParameters)

Arguments

data  
dataframe for plotting. Last column indicates what type of taxon group (in- or out-group). The first (or first 2) column contains values of the variables. See ?dataVarDistTaxGroup

plotParameters  
plot parameters, including size of x-axis, y-axis, legend and title; position of legend ("right", "bottom" or "none"); mean/median point; names of in-group and out-group; and plot title. NOTE: Leave blank or NULL to use default values.

Value

Distribution plots as a grob (gtable) object. Use grid.draw to plot.

Author(s)

Vinh Tran tran@bio.uni-frankfurt.de

See Also

dataVarDistTaxGroup

Examples

data("mainLongRaw", package="PhyloProfile")
data <- mainLongRaw
inGroup <- c("ncbi9606", "ncbi10116")
variable <- colnames(data)[c(4, 5)]
plotDf <- dataVarDistTaxGroup(data, inGroup, "101621at6656", variable)
plotParameters <- list(
  "xSize" = 12,
  "ySize" = 12,
  "titleSize" = 15,
  "legendSize" = 12,
  "legendPosition" = "right",
  "mValue" = "mean",
  "inGroupName" = "In-group",
  "outGroupName" = "Out-group",
  "title" = "101621at6656"
wideToLong

Transform input file in wide matrix into long matrix format

Description

Transform input file in wide matrix into long matrix format

Usage

wideToLong(inputFile = NULL)

Arguments

inputFile input file in wide matrix format

Value

A data frame of input data in long-format containing seed gene IDs (or orthologous group IDs), their orthologous proteins together with the corresponding taxonomy IDs and values of (up to) two additional variables.

Author(s)

Vinh Tran tran@bio.uni-frankfurt.de

Examples

inputFile <- system.file(
  "extdata", "test.main.wide", package = "PhyloProfile", mustWork = TRUE
)
wideToLong(inputFile)
xmlParser

Parse orthoXML input file

Description
Parse orthoXML input file

Usage
xmlParser(inputFile = NULL)

Arguments
inputFile input file in xml format

Value
A data frame of input data in long-format containing seed gene IDs (or orthologous group IDs), their orthologous proteins together with the corresponding taxonomy IDs and values of (up to) two additional variables.

Author(s)
Vinh Tran tran@bio.uni-frankfurt.de

Examples
inputFile <- system.file(
  "extdata", "test.main.xml", package = "PhyloProfile", mustWork = TRUE
)
xmlParser(inputFile)
Index

addRankDivisionPlot, 4
calcPresSpec, 5, 65
checkInputValidity, 6
checkNewick, 7
checkOmaID, 6, 8
clusterDataDend, 8, 41
compareMedianTaxonGroups, 9
compareTaxonGroups, 10
createArchiPlot, 11
createGeneAgePlot, 12
createLongMatrix, 13, 21, 22, 46, 65, 75
createPercentageDistributionData, 14, 17
createProfileFromOma, 15
createUnrootedTree, 15, 73, 78
createVarDistPlot, 16
createVariableDistributionData, 17, 17, 19
createVariableDistributionDataSubset, 17, 18
dataCustomizedPlot, 19, 57, 58
dataFeatureTaxGroup, 20, 26
dataMainPlot, 21, 57, 58
dataVarDistTaxGroup, 22, 79
distributionTest, 23
estimateGeneAge, 13, 23, 34
fastaParser, 13, 25
featureDistTaxPlot, 25
filteredProfile, 27, 71
filterProfileData, 20, 21, 28, 32, 71
finalProcessedProfile, 30
fromInputToProfile, 29, 31, 39
fullProcessedProfile, 19, 24, 29, 33
geneAgePlotDf, 13, 33
generateSinglePlot, 34
getAllDomainsOma, 35
getAllFastaOma, 36
getCommonAncestor, 37
getCoreGene, 37
dataTransferClustering, 9, 39, 42
dataForOneOma, 15, 36, 40, 51
dEndrogram, 40
dataDistanceMatrix, 9, 41
dataDomainFolder, 42, 64
dataFastaFromFasInput, 43
dataFastaFromFile, 43
dataFastaFromFolder, 44
dataIDsRank, 45
dataInputTaxaID, 7, 32, 46, 47, 75
dataInputTaxaName, 32, 47, 75
dataNameList, 24, 47, 47, 75
dataOmaDataForOneOrtholog, 48
dataOmaDomainFromURL, 49
dataOmaMembers, 49
dataQualColForVector, 12, 50, 72
dataSelectedFastaOma, 51
dataSelectedTaxonNames, 51
dataTaxHierarchy, 52
getTaxonomyInfo, 53
dataTaxonomyMatrix, 4, 16, 24, 54, 75
getDataRanks, 55
dataGridArrangeSharedLegend, 55
hclust, 9
heatmapPlotting, 4, 56, 58
highlightProfilePlot, 4, 58
id2name, 59
idList, 60
mainLongRaw, 14, 17–19, 45, 46, 60, 65
mainTaxonomyRank, 61
modifyFeatureName, 61
pairDomainPlotting, 62
parseDomainInput, 12, 21, 63, 72
parseInfoProfile, 19, 24, 29, 32, 38, 64, 71
ppTaxonomyMatrix, 16, 65, 76
ppTree, 7, 66
processNcbiTaxonomy, 66
processOrthoID, 67
profileWithTaxonomy, 6, 68

qualitativeColours, 50, 69

rankIndexing, 69, 78
rankList, 70
reduceProfile, 29, 32, 70
runPhyloProfile, 71

singleDomainPlotting, 12, 72
sortDomains, 12, 73
sortInputTaxa, 32, 65, 74, 78
sortTaxaFromTree, 75, 75

taxa2dist, 16, 76
taxonNamesReduced, 77
taxonomyMatrix, 77
taxonomyTableCreator, 78

varDistTaxPlot, 79

wideToLong, 13, 80

xmlParser, 13, 81