Package 'PhyloProfile'

November 23, 2024

Version 1.21.3

Date 2024-11-07

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 (>= 4.4.0)

Encoding UTF-8

biocViews Software, Visualization, DataRepresentation, MultipleComparison, FunctionalPrediction

Imports ape, bioDist, BiocStyle, Biostrings, colourpicker, data.table, dplyr, DT, energy, ExperimentHub, extrafont, ggplot2, gridExtra, pbapply, plotly, RColorBrewer, RCurl, scattermore, shiny, shinyBS, shinycssloaders, shinyFiles, shinyjs, stringr, umap, xml2, zoo, yaml

RoxygenNote 7.3.1

Suggests knitr, rmarkdown, testthat, OmaDB

VignetteBuilder knitr

git_url https://git.bioconductor.org/packages/PhyloProfile

git_branch devel

git_last_commit 5c948ba

git_last_commit_date 2024-11-07

Repository Bioconductor 3.21

Date/Publication 2024-11-22

Author Vinh Tran [aut, cre] (ORCID: <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

`addFeatureColors` *Add colors for each feature/domain*

Description

Add colors to features/domains of 2 domain dataframes. Users can choose to color only the shared features, unique features, all features (default) or based on feature types. Default color pallete is "Paired", but it can be changed.

Usage

```
addFeatureColors(
  seedDf = NULL,
  orthoDf = NULL,
  colorType = "all",
  colorPalette = "Paired",
  ignoreInstanceNo = FALSE
)
```

Arguments

seedDf	Domain dataframe of seed protein (protein 1)
orthoDf	Domain dataframe of orthologs protein (protein 2)
colorType	Choose to color "all", "shared", "unique" features or color by "Feature type". Default: "all"
colorPalette	Choose between "Paired", "Set1", "Set2", "Set3", "Accent", "Dark2" for the color pallete
ignoreInstanceNo	Ignore number of feature instances while identifying shared or unique features. Default: FALSE

Value

2 dataframes (seedDf and orthoDf) with an additional column for the assigned color to each feature instance

Author(s)

Vinh Tran tran@bio.uni-frankfurt.de

Examples

```
# get domain data
seedID <- "101621at6656"
domainFile <- system.file(
    "extdata", "domainFiles/101621at6656.domains",
    package = "PhyloProfile", 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",]
# add colors to features
PhyloProfile:::addFeatureColors(seedDf, orthoDf)
```

---

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",
    font = "Arial", groupLabelSize = 14, groupLabelDist = 2,
    groupLabelAngle = 90, refLine = TRUE)
```

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")
font	font of text. Default = Arial"
groupLabelSize	size of rank labels
groupLabelDist	size of the plot area for rank labels
groupLabelAngle	angle of rank labels
refLine	add vertical line to separate reference taxon

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

```
data("finalProcessedProfile", package="PhyloProfile")
plotDf <- dataMainPlot(finalProcessedProfile)
plotParameter <- list(
    "xAxis" = "taxa",
    "geneIdType" = "geneID",
    "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", font = "sans"
)
```

`addUmapTaxaColors` *Add colors for taxa in UMAP plot*

Description

Add colors for taxa in UMAP plot

Usage

```
addUmapTaxaColors(plotDf = NULL, colorPalette = "Set2",
    highlightTaxa = NULL)
```

Arguments

`plotDf`	data for UMAP plot
`colorPalette`	color palette. Default: "Set2"
`highlightTaxa`	list of taxa to be highlighted

Value

A dataframe for UMAP plot with an additional column for the assigned color to each taxon

Author(s)

Vinh Tran tran@bio.uni-frankfurt.de

See Also

`prepareUmapData`, `umapClustering`, `createUmapPlotData`

Examples

```
rawInput <- system.file(
   "extdata", "test.main.long", package = "PhyloProfile", mustWork = TRUE
)
longDf <- createLongMatrix(rawInput)
umapData <- prepareUmapData(longDf, "phylum")
data.umap <- umapClustering(umapData)
plotDf <- createUmapPlotData(data.umap, umapData)
PhyloProfile:::addUmapTaxaColors(plotDf, colorPalette = "Set2")
```

`calcPresSpec` *Calculate percentage of present species in each super taxon*

Description

Calculate percentage of present species in each super taxon

Usage

```
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)

Value

A data frame with

Author(s)

Vinh Tran tran@bio.uni-frankfurt.de

See Also

`profileWithTaxonomy` for a demo input data

Examples

```
# NOTE: for internal testing only
library(dplyr)
data("profileWithTaxonomy", package="PhyloProfile")
taxaCount <- profileWithTaxonomy %>% dplyr::count(supertaxon)
taxaCount$n <- 1
calcPresSpec(profileWithTaxonomy, taxaCount)
```

checkColorPalette *Check if a color pallete has enough colors for a list of items*

Description

Check if a color pallete has enough colors for a list of items

Usage

```
checkColorPalette(items, pallete = "Paired")
```

Arguments

items	vector contains list of items
pallete	name of color palette

Value

TRUE if color pallete has enough colors, otherwise FALSE

Author(s)

Vinh Tran tran@bio.uni-frankfurt.de

Examples

```
myItems <- rep("a",3)
checkColorPalette(myItems, "Set1")
```

checkInputValidity *Check the validity of the input phylogenetic profile file*

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

Examples

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

---

checkNewick *Check the validity of input newick tree*

Description

Check the validity of input newick tree

Usage

```
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

```
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 needs to be checked

Value

List of invalid IDs (not readable for OMA)

Author(s)

Vinh Tran tran@bio.uni-frankfurt.de

Examples

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

---

checkOverlapDomains *Identify feature type(s) containing overlapped domains/features*

Description

Identify feature type(s) containing overlapped domains/features

Usage

```
checkOverlapDomains(domainDf)
```

Arguments

domainDf	input domain dataframe

Value

List of feature types that have overlapped domains

Author(s)

Vinh Tran tran@bio.uni-frankfurt.de

Examples

```
# get domain data
seedID <- "101621at6656"
domainFile <- system.file(
    "extdata", "domainFiles/101621at6656.domains",
    package = "PhyloProfile", 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",]
# check overlap features
PhyloProfile:::checkOverlapDomains(orthoDf)
```

---

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

getDataClustering, 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

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 architecure 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, domainDf, labelArchiSize, titleArchiSize,
    showScore, showWeight, namePosition, firstDist, nameType, nameSize,
    segmentSize, nameColor, labelPos, colorType, ignoreInstanceNo,
    currentNCBIinfo, featureClassSort, featureClassOrder, colorPalette,
    resolveOverlap, font)
```

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	Lable size (in px). Default = 12.
titleArchiSize	Title size (in px). Default = 12.
showScore	Show/hide E-values and Bit-scores. Default = NULL (hide)
showWeight	Show/hide feature weights. Default = NULL (hide)
namePosition	list of positions for domain names, choose from "plot", "legend" or "axis". Default: "plot"
firstDist	Distance of the first domain to plot title. Default = 0.5
nameType	Type of domain names, either "Texts" or "Labels" (default)

nameSize	Size of domain names. Default = 3
segmentSize	Height of domain segment. Default = 5
nameColor	Color of domain names (for Texts only). Default = "black"
labelPos	Position of domain names (for Labels only). Choose from
colorType	Choose to color "all", "shared", "unique" features or color by "Feature type". Default = "all"
ignoreInstanceNo	Ignore number of feature instances while identifying shared or unique features. Default = FALSE
currentNCBIinfo	Dataframe of the pre-processed NCBI taxonomy data. Default = NULL (will be automatically retrieved from PhyloProfile app)
featureClassSort	Choose to sort features. Default = "Yes"
featureClassOrder	vector of ordered feature classes
colorPalette	Choose between "Paired", "Set1", "Set2", "Set3", "Accent", "Dark2" for the color pallete
resolveOverlap	Choose to merge non-overlapped features of a feature type into one line. Default = "Yes"
font	font of text. Default = Arial"

Value

A domain plot as arrangeGrob object. Use grid::grid.draw(plot) to render.

Author(s)

Vinh Tran tran@bio.uni-frankfurt.de

See Also

singleDomainPlotting,pairDomainPlotting, sortDomains, parseDomainInput

Examples

```
seedID <- "101621at6656"
orthoID <- "101621at6656|AGRPL@224129@0|224129_0:001955|1"
info <- c(seedID, orthoID)
domainFile <- system.file(
    "extdata", "domainFiles/101621at6656.domains",
    package = "PhyloProfile", mustWork = TRUE
)
domainDf <- parseDomainInput(seedID, domainFile, "file")
domainDf$feature_id_mod <- domainDf$feature_id
domainDf$feature_id_mod <- gsub("SINGLE", "LCR", domainDf$feature_id_mod)
domainDf$feature_id_mod[domainDf$feature_type == "coils"] <- "Coils"
domainDf$feature_id_mod[domainDf$feature_type == "seg"] <- "LCR"
```

```
domainDf$feature_id_mod[domainDf$feature_type == "tmhmm"] <- "TM"
plot <- createArchiPlot(info, domainDf, font = "sans")
grid::grid.draw(plot)
```

`createGeneAgePlot` *Create gene age plot*

Description

Create gene age plot

Usage

```
createGeneAgePlot(geneAgePlotDf, textFactor = 1, font = "Arial")
```

Arguments

`geneAgePlotDf`	data frame required for plotting gene age (see ?geneAgePlotDf)
`textFactor`	increase factor of text size
`font`	font of text. Default = Arial"

Value

A gene age distribution plot as a ggplot2 object

Author(s)

Vinh Tran tran@bio.uni-frankfurt.de

See Also

`estimateGeneAge` and `geneAgePlotDf`

Examples

```
geneAgePlotDf <- data.frame(
    name = c("Streptophyta (Phylum)", "Bikonta", "Eukaryota (Superkingdom)"),
    count = c(7, 1, 30),
    percentage = c(18, 3, 79)
)
createGeneAgePlot(geneAgePlotDf, 1, "sans")
```

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

Description

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

```
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 taxononmy IDs of the orthologs.

Author(s)

Vinh Tran tran@bio.uni-frankfurt.de

See Also

getDataForOneOma

Examples

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

createUmapPlotData *Create UMAP cluster plot*

Description

Create UMAP cluster plot

Usage

```
createUmapPlotData(umapData = NULL, data4umap = NULL,
    freqCutoff = c(0,200), excludeTaxa = "None", currentNCBIinfo = NULL)
```

Arguments

umapData	data contains UMAP cluster (output from umapClustering())
data4umap	data for UMAP clustering (output from prepareUmapData())
freqCutoff	gene/taxon frequency cutoff range. Any labels that are outside of this range will be assigned as [Other]
excludeTaxa	hide taxa from plot. Default: "None"
currentNCBIinfo	table/dataframe of the pre-processed NCBI taxonomy data (/PhyloProfile/data/preProcessedTaxonomy.txt

Value

A plot as ggplot object

Author(s)

Vinh Tran tran@bio.uni-frankfurt.de

See Also

prepareUmapData, umapClustering

Examples

```
rawInput <- system.file(
   "extdata", "test.main.long", package = "PhyloProfile", mustWork = TRUE
)
longDf <- createLongMatrix(rawInput)
data4umap <- prepareUmapData(longDf, "phylum")
umapData <- umapClustering(data4umap)
createUmapPlotData(umapData, data4umap)
```

---

createUnrootedTree *Create unrooted tree from a taxonomy matrix*

Description

Create unrooted tree from a taxonomy matrix

Usage

```
createUnrootedTree(df)
```

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

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).

Author(s)

Vinh Tran tran@bio.uni-frankfurt.de

See Also

mainLongRaw

Examples

```
data("mainLongRaw", package="PhyloProfile")
createVariableDistributionData(
    mainLongRaw, c(0, 1), c(0.5, 1)
)
```

---

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

See Also

parseInfoProfile, createVariableDistributionData, fullProcessedProfile, mainLongRaw

Examples

```
data("fullProcessedProfile", package="PhyloProfile")
data("mainLongRaw", package="PhyloProfile")
distributionData <- createVariableDistributionData(
    mainLongRaw, c(0, 1), c(0.5, 1)
)
selectedGenes <- "100136at6656"
selectedTaxa <- c("Mammalia", "Saccharomycetes", "Insecta")
createVariableDistributionDataSubset(
    fullProcessedProfile,
    distributionData,
    selectedGenes,
    selectedTaxa
)
```

---

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

```
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)
```

---

dataFeatureTaxGroup *Create data for feature distribution comparison plot*

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)
```

`dataMainPlot` *Create data for main profile plot*

Description

Create data for main profile plot

Usage

```
dataMainPlot(dataHeat = NULL)
```

Arguments

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

Value

A dataframe for plotting the phylogenetic profile, 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 species that have orthologs in each supertaxon (presSpec).

Author(s)

Vinh Tran tran@bio.uni-frankfurt.de

See Also

`filterProfileData`

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 tess. 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

```
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

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

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

```
library(dplyr)
data("fullProcessedProfile", package="PhyloProfile")
rankName <- "class"
refTaxon <- "Mammalia"
processedProfileData <- fullProcessedProfile
taxonIDs <- levels(as.factor(processedProfileData$ncbiID))
sortedInputTaxa <- sortInputTaxa(
    taxonIDs, rankName, refTaxon, NULL, NULL
)
taxaCount <- sortedInputTaxa %>% dplyr::count(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/Inp 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)
```

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)
```

Format

Dataframe

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. "100136at6656|HUMAN@9606@1|Q9UNQ2|1"
- 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 phylogentic 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!
library(dplyr)
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 <- sortedInputTaxa %>% dplyr::count(supertaxon)
filterProfileData(
    fullProcessedProfile,
    taxaCount,
    refTaxon,
    percentCutoff,
    coorthologCutoffMax,
    var1Cutoff,
    var2Cutoff,
    var1Relation,
    var2Relation,
    groupByCat,
```

```
    catDt,
    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)
```

Format

Dataframe

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. "100136at6656|RAT@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),
    coortholog CutoffMax = 9999, var1Cutoff = c(0, 1), var2Cutoff = c(0, 1),
    var1Relation = "protein", var2Relation = "protein", groupByCat = FALSE,
    catDt = NULL, taxDB = NULL)
```

Arguments

rawInput	input file (in long, wide, multi-fasta or orthoxml format)
rankName	taxonomy rank (e.g. "species","phylum",...)
refTaxon	selected reference taxon name (used for sorting and will be protected from filtering). Default = NULL.
taxaTree	input taxonomy tree for taxa in input profiles (optional). Default = NULL.
sortedTaxonList	list of sorted taxa (optional). Default = NULL.
var1AggregateBy	aggregate method for var1 (min, max, mean or median). Default = "max".
var2AggregateBy	aggregate method for VAR2 (min, max, mean or median). Default = "max".
percentCutoff	min and max cutoffs for percentage of species present in a supertaxon. Default = c(0, 1).
coortholog CutoffMax	maximum number of co-orthologs allowed. Default = 9999.
var1Cutoff	min and max cutoffs for var1. Default = c(0, 1).
var2Cutoff	min and max cutoffs for var2. Default = c(0, 1).
var1Relation	relation of var1 ("protein" for protein-protein or "species" for protein-species). Default = "protein".
var2Relation	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. Default = NULL
taxDB	Path to the taxonomy DB files

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)
```

Format

Dataframe

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. "100136at6656|HUMAN@9606@1|Q9UNQ2|1"
- 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)
```

Arguments

geneAgeDf	data frame containing estimated gene ages for seed proteins

Value

A dataframe for plotting gene age plot containing the absolute number and percentage of genes for each calculated evolutionary ages and the corresponding position for writting those number on the plot.

Author(s)

Vinh Tran tran@bio.uni-frankfurt.de

See Also

estimateGeneAge

Examples

```
geneAgeDf <- data.frame(
geneID = c("100136at6656", "100265at6656", "101621at6656", "103479at6656"),
cath = c("0000001", "0000011", "0000001", "0000011"),
age = c("07_LUCA", "06_Eukaryota", "07_LUCA", "06_Eukaryota")
)
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).

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

```
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

See Also

getDataForOneOma

Examples

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

getAllFastaOma *Get all fasta sequences from a raw OMA dataframe*

Description

Get all fasta sequences from a raw OMA dataframe

Usage

```
getAllFastaOma(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

getDataForOneOma

Examples

```
### Uncomment the following line to run the function
# omaData <- getDataForOneOma("HUMAN29397", "OG")
# getAllFastaOma(omaData)
```

---

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", "ncbi9606", "ncbi374847", "ncbi123851",
    "ncbi5664", "ncbi189518", "ncbi418459", "ncbi10116", "ncbi284812",
    "ncbi35128", "ncbi7070")
inGroup <-  c("ncbi9606", "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

```
getCoreGene(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

```
library(dplyr)
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 <- sortedInputTaxa %>% dplyr::count(supertaxon)
var1Cutoff <- c(0.75, 1.0)
var2Cutoff <- c(0.75, 1.0)
percentCutoff <- c(0.0, 1.0)
coreCoverage <- 100
getCoreGene(
    rankName,
    taxaCore,
    profileDt,
    taxaCount,
    var1Cutoff, var2Cutoff,
    percentCutoff, coreCoverage
)
```

---

getDataClustering *Get data for calculating distance matrix from phylogenetic profiles*

Description

Get data for calculating distance matrix from phylogenetic profiles

Usage

```
getDataClustering(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 variables 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)
```

---

getDataForOneOma *Get OMA info for a query protein and its orthologs*

Description

Get taxonomy IDs, sequences, length and annotations for an OMA orthologous group (or OMA HOG).

Usage

```
getDataForOneOma(seedID = NULL, orthoType = "OG")
```

Arguments

seedID	OMA protein ID
orthoType	type of OMA orthologs ("OG" or "HOG"). Default = "OG".

Value

Data frame contains info for all sequences of the input OMA group (or HOG). That info contains the protein IDs, taxonomy IDs, sequences, lengths, domain annotations (tab delimited) and the corresponding seed ID.

Author(s)

Vinh Tran tran@bio.uni-frankfurt.de

Examples

```
### Uncomment the following line to run the function
# getDataForOneOma("HUMAN29397", "OG")
```

---

getDendrogram *Plot dendrogram tree*

Description

Plot dendrogram tree

Usage

```
getDendrogram(dd = NULL)
```

Arguments

dd	dendrogram object (see ?clusterDataDend)

Value

A dendrogram plot for the genes in the input phylogenetic profiles.

Author(s)

Vinh Tran tran@bio.uni-frankfurt.de

See Also

clusterDataDend

Examples

```
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)
```

`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.

Author(s)

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

See Also

`getDataClustering`

Examples

```
data("finalProcessedProfile", package="PhyloProfile")
data <- finalProcessedProfile
profileType <- "binary"
profiles <- getDataClustering(
    data, profileType, var1AggregateBy, var2AggregateBy)
method <- "mutualInformation"
getDistanceMatrix(profiles, method)
```

getDomainFolder *Get domain file from a folder for a seed protein*

Description

Get domain file from a folder for a seed protein

Usage

```
getDomainFolder(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

```
domainPath <- paste0(
    path.package("PhyloProfile", quiet = FALSE), "/extdata/domainFiles"
)
PhyloProfile:::getDomainFolder("101621at6656", domainPath)
```

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)
```

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

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

getFastaFromFolder *Get fasta sequences*

Description

Get fasta sequences for the input phylogenetic profiles.

Usage

```
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.

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)
```

getIDsRank *Get taxonomy info for a list of taxa*

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

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)
```

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

```
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

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

---

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

Description

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

Usage

```
getNameList(taxDB = NULL)
```

Arguments

taxDB	Path to the taxonomy DB files

Value

List of taxon IDs, their full names, taxonomy ranks and parent IDs obtained from "PhyloProfile/data/taxonNamesReduced.txt"

Author(s)

Vinh Tran tran@bio.uni-frankfurt.de

Examples

```
getNameList()
```

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

```
### 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

```
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

```
### 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

```
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".

Value

List of OMA orthologs for an input seed protein.

Author(s)

Carla Mölbert carla.moelbert@gmx.de

Examples

```
### 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

```
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

```
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

```
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

```
### 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

```
getSelectedTaxonNames(inputTaxonIDs = NULL, rank = NULL,
    higherRank = NULL, higherID = NULL, higherName = NULL, taxDB = NULL)
```

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 neccessary to give both
higherName	supertaxon name (e.g. "Chordata"). NOTE: either supertaxon ID or name is required, not neccessary 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

```
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

```
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

Value

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

Author(s)

Vinh Tran tran@bio.uni-frankfurt.de

Examples

```
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

```
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

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 taxanomy ranks*

Description

Create a list containing all main taxanomy ranks

Usage

```
getTaxonomyRanks()
```

Value

A list of all main 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`	Gird 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

```
## Not run:
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
)

## End(Not run)
```

`groupLabelUmapData` *Reduce the number of labels for UMAP plot based on the gene/taxon frequency*

Description

Reduce the number of labels for UMAP plot based on the gene/taxon frequency

Usage

```
groupLabelUmapData(data4umap = NULL, freqCutoff = c(0,200))
```

Arguments

`data4umap`	data for UMAP clustering (output from prepareUmapData)
`freqCutoff`	gene/taxon frequency cutoff range. Any labels that are outside of this range will be assigned as [Other]

Value

A dataframe similar to input data4umap, but with modified Label column, where less frequent labels are grouped together as "Other"

Author(s)

Vinh Tran tran@bio.uni-frankfurt.de

See Also

`prepareUmapData`

Examples

```
rawInput <- system.file(
   "extdata", "test.main.long", package = "PhyloProfile", mustWork = TRUE
)
longDf <- createLongMatrix(rawInput)
data4umap <- prepareUmapData(longDf, "phylum")
groupLabelUmapData(data4umap, freqCutoff = c(3,5))
```

`heatmapPlotting` *Create profile heatmap plot*

Description

Create profile heatmap plot

Usage

```
heatmapPlotting(data = NULL, parm = NULL)
```

Arguments

`data`	dataframe for plotting the heatmap phylogentic profile (either full or subset profiles)
`parm`	plot parameters, including (1) type of x-axis "taxa" or "genes" - default = "taxa"; (2) display gene IDs (default) or gene names; (3+4) names of 2 variables var1ID and var2ID - default = "var1" & "var2"; (5+6) mid value and color for mid value of var1 - default is 0.5 and #FFFFFF; (7) color for lowest var1 - default = "#FF8C00"; (8) color for highest var1 - default = "#4682B4"; (9+10) mid value and color for mid value of var2 - default is 1 and #FFFFFF;(11) color for lowest var2 - default = "#FFFFFF", (12) color for highest var2 - default = "#F0E68C", (13) color of co-orthologs - default = "#07D000"; (14+15+16) text sizes for x, y axis and legend - default = 9 for each; (17) legend position "top", "bottom", "right", "left" or "none" - default = "top"; (18) zoom ratio of the co-ortholog dots from -1 to 3 - default = 0; (19) angle of x-axis from 0 to 90 - default = 60; (20) show/hide separate line for reference taxon 1/0 - default = 0; (21) enable/disable coloring gene categories TRUE/FALSE - default = FALSE; (22) 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",
    "geneIdType" = "geneID",
    "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,
    "catColors" = NULL,
    "colorByOrthoID" = FALSE
)

heatmapPlotting(plotDf, plotParameter)
```

---

heatmapPlottingFast *Create profile heatmap plot using scattermore*

Description

Create profile heatmap plot using scattermore

Usage

```
heatmapPlottingFast(data = NULL, parm = NULL)
```

Arguments

data	dataframe for plotting the heatmap phylogentic profile (either full or subset profiles)
parm	plot parameters, including (1) type of x-axis "taxa" or "genes" - default = "taxa"; (2) display gene IDs (default) or gene names; (3+4) names of 2 variables var1ID and var2ID - default = "var1" & "var2"; (5+6) mid value and color for mid value of var1 - default is 0.5 and #FFFFFF; (7) color for lowest var1 - default = "#FF8C00"; (8) color for highest var1 - default = "#4682B4"; (9+10) mid value and color for mid value of var2 - default is 1 and #FFFFFF;(11) color for lowest var2 - default = "#FFFFFF", (12) color for highest var2 - default = "#F0E68C", (13) color of co-orthologs - default = "#07D000"; (14+15+16) text sizes for x, y axis and legend - default = 9 for each; (17) legend position "top", "bottom", "right", "left" or "none" - default = "top"; (18) zoom ratio of the co-ortholog dots from -1 to 3 - default = 0; (19) color dots based on either "var1" or "var2". 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",
    "geneIdType" = "geneID",
    "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,
    "colorVar" = "var1"
)

heatmapPlottingFast(plotDf, plotParameter)
```

---

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 interst. 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

Examples

```
data("finalProcessedProfile", package="PhyloProfile")
plotDf <- dataMainPlot(finalProcessedProfile)
plotParameter <- list(
    "xAxis" = "taxa",
    "geneIdType" = "geneID",
    "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)
taxonHighlight <- "none"
workingRank <- "class"
geneHighlight <- "100265at6656"
highlightProfilePlot(
    profilePlot, plotDf, taxonHighlight, workingRank, geneHighlight,
    NULL, plotParameter$xAxis
)
```

---

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

```
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)
```

---

idList *NCBI ID list for experimental data sets*

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)
```

Format

Dataframe

Value

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

`joinPlotMergeLegends` *Join multiple plots and merge legends*

Description

Join multiple plots and merge legends

Usage

```
joinPlotMergeLegends(
  df1 = NULL,
  df2 = NULL,
  plot1 = NULL,
  plot2 = NULL,
  position = c("bottom", "right"),
  font = "Arial"
)
```

Arguments

`df1`	Data frame for plot 1
`df2`	Data frame for plot 2
`plot1`	ggplot object of plot 1
`plot2`	ggplot object of plot 2
`position`	position of legend (bottom or right)
`font`	font of text

Value

joined plots with merged legend as a grid object

Author(s)

Vinh Tran tran@bio.uni-frankfurt.de

Examples

```
seed <- "101621at6656"
ortho <- "101621at6656|AGRPL@224129@0|224129_0:001955|1"
ortho <- gsub("\\|", ":", ortho)
grepID <- paste(seed, "#", ortho, sep = "")
domainFile <- system.file(
    "extdata", "domainFiles/101621at6656.domains",
    package = "PhyloProfile", mustWork = TRUE
)
domainDf <- parseDomainInput(seed, domainFile, "file")
domainDf$feature_id_mod <- domainDf$feature_id
```

```
subdomainDf <- domainDf[grep(grepID, domainDf$seedID), ]
subdomainDf$feature <- as.character(subdomainDf$feature)
orthoDf <- subdomainDf[subdomainDf$orthoID == ortho,]
seedDf <- subdomainDf[subdomainDf$orthoID != ortho,]
minStart <- min(subdomainDf$start)
maxEnd <- max(c(subdomainDf$end, subdomainDf$length))
# resolve overlapping domains
seedDf <- PhyloProfile:::resolveOverlapFeatures(seedDf)
orthoDf <- PhyloProfile:::resolveOverlapFeatures(orthoDf)
# add feature colors
featureColorDf <- PhyloProfile:::addFeatureColors(seedDf, orthoDf)
seedDf <- featureColorDf[[1]]
orthoDf <- featureColorDf[[2]]
# generate plots
plotSeed <- PhyloProfile:::singleDomainPlotting(
    seedDf, seed, minStart = minStart, maxEnd = maxEnd, font = "sans"
)
plotOrtho <- PhyloProfile:::singleDomainPlotting(
    orthoDf, ortho, minStart = minStart, maxEnd = maxEnd, font = "sans"
)
# merge plots
PhyloProfile:::joinPlotMergeLegends(
    seedDf, orthoDf, plotSeed, plotOrtho, "bottom", font = "sans")
```

`linearizeArchitecture` *Linearize PFAM/SMART annotations by best e-value/bitscore*

Description

Linearize PFAM/SMART annotations by best e-value/bitscore

Usage

```
linearizeArchitecture(domainDf = NULL, orthoID = NULL, value = "evalue")
```

Arguments

`domainDf`	input domain dataframe
`orthoID`	ID of protein that needs to be linearized
`value`	type of values that will be used for linearized, either evalue (default) or bitscore

Value

Domain dataframe of the selected protein after linearization

Author(s)

Vinh Tran tran@bio.uni-frankfurt.de

Examples

```
demoDomainDf <- data.frame(
    orthoID = rep("protID", 4),
    start = c(1, 5, 100, 80),
    end = c(30, 40, 130, 110),
    evalue = c(0.001, 0.0005, 0.2, 0.004),
    feature_type = c(rep("pfam", 2), rep("smart", 2)),
    feature_id = c("pf1", "pf2", "sm1", "sm2")
)
linearizeArchitecture(demoDomainDf, "protID", "evalue")
```

`mainLongRaw` *An example of a raw long input file*

Description

An example of a raw long input file

Usage

```
data(mainLongRaw)
```

Format

Dataframe

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@1|Q8ILT8|1"
- FAS_F First additional variable
- FAS_B Second additional variable

mainTaxonomyRank *Get all NCBI taxonomy rank names*

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)
```

Arguments

domainDf	domain data as a dataframe object

Value

Dataframe contains simlified domain names in yLabel column

Author(s)

Vinh Tran tran@bio.uni-frankfurt.de

Examples

```
domainFile <- system.file(
    "extdata", "domainFiles/101621at6656.domains",
    package = "PhyloProfile", mustWork = TRUE
)
seedID <- "101621at6656"
domainDf <- parseDomainInput(seedID, domainFile, "file")
PhyloProfile:::modifyFeatureName(domainDf)
```

`pairDomainPlotting` *Create architecure plot for a pair of seed and ortholog protein*

Description

Create architecure plot for a pair of seed and ortholog protein

Usage

```
pairDomainPlotting(seed, ortho, seedDf, orthoDf, minStart, maxEnd,
    labelSize, titleSize, showScore, showWeight, namePosition, firstDist,
    nameType, nameSize, segmentSize, nameColor, labelPos, colorPalette, font)
```

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
`labelSize`	lable size. Default = 12
`titleSize`	title size. Default = 12
`showScore`	show/hide E-values and Bit-scores. Default = NULL (hide)
`showWeight`	Show/hide feature weights. Default = NULL (hide)
`namePosition`	list of positions for domain names, choose from "plot", "legend" or "axis". Default: "plot"
`firstDist`	distance of the first domain to plot title. Default = 0.5
`nameType`	type of domain names, either "Texts" or "Labels" (default)
`nameSize`	Size of domain names. Default = 3
`segmentSize`	Height of domain segment. Default = 5

nameColor	color of domain names (for Texts only). Default = "black"
labelPos	position of domain names (for Labels only). Choose from "Above" (default), "Below" or "Inside" the domain bar
colorPalette	color pallete. Default = Paired"
font	font of text. Default = Arial"

Value

Domain plot of a pair proteins as a arrangeGrob object.

Author(s)

Vinh Tran tran@bio.uni-frankfurt.de

See Also

singleDomainPlotting, sortDomains, parseDomainInput

Examples

```
seed <- "101621at6656"
ortho <- "101621at6656|AGRPL@224129@0|224129_0:001955|1"
ortho <- gsub("\\|", ":", ortho)
grepID <- paste(seed, "#", ortho, sep = "")
domainFile <- system.file(
    "extdata", "domainFiles/101621at6656.domains",
    package = "PhyloProfile", mustWork = TRUE
)
domainDf <- parseDomainInput(seed, domainFile, "file")
domainDf$feature_id_mod <- domainDf$feature_id
subdomainDf <- domainDf[grep(grepID, domainDf$seedID), ]
subdomainDf$feature <- as.character(subdomainDf$feature)
orthoDf <- subdomainDf[subdomainDf$orthoID == ortho,]
seedDf <- subdomainDf[subdomainDf$orthoID != ortho,]
minStart <- min(subdomainDf$start)
maxEnd <- max(c(subdomainDf$end, subdomainDf$length))
# resolve overlapping domains
seedDf <- PhyloProfile:::resolveOverlapFeatures(seedDf)
orthoDf <- PhyloProfile:::resolveOverlapFeatures(orthoDf)
# add feature colors
featureColorDf <- PhyloProfile:::addFeatureColors(seedDf, orthoDf)
seedDf <- featureColorDf[[1]]
orthoDf <- featureColorDf[[2]]
# do plot
g <- PhyloProfile:::pairDomainPlotting(
   seed,ortho,seedDf,orthoDf,minStart,maxEnd, font = "sans"
)
grid::grid.draw(g)
```

`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)
```

parseInfoProfile *Parsing info for phylogenetic profiles*

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
coorthoCOMax	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 (?estimateGeneAge) or identification of core gene (?getCoreGene).

Author(s)

Vinh Tran tran@bio.uni-frankfurt.de

See Also

createLongMatrix, sortInputTaxa, calcPresSpec, mainLongRaw

Examples

```
library(dplyr)
data("mainLongRaw", package="PhyloProfile")
taxonIDs <- getInputTaxaID(mainLongRaw)
sortedInputTaxa <- sortInputTaxa(
    taxonIDs, "class", "Mammalia", NULL, NULL
)
taxaCount <- sortedInputTaxa %>% dplyr::count(supertaxon)
coorthoCOMax <- 999
parseInfoProfile(
    mainLongRaw, sortedInputTaxa, taxaCount, coorthoCOMax
)
```

plotUmap *Create UMAP cluster plot*

Description

Create UMAP cluster plot

Usage

```
plotUmap(plotDf = NULL, legendPos = "bottom", colorPalette = "Set2",
    transparent = 0, textSize = 12, font = "Arial", highlightTaxa = NULL,
    dotZoom = 0)
```

Arguments

plotDf	data for UMAP plot
legendPos	position of legend. Default: "right"
colorPalette	color palette. Default: "Set2"
transparent	transparent level (from 0 to 1). Default: 0
textSize	size of axis and legend text. Default: 12
font	font of text. Default = Arial"
highlightTaxa	list of taxa to be highlighted
dotZoom	dot size zooming factor. Default: 0

Value

A plot as ggplot object

Author(s)

Vinh Tran tran@bio.uni-frankfurt.de

See Also

prepareUmapData, umapClustering, createUmapPlotData

Examples

```
rawInput <- system.file(
   "extdata", "test.main.long", package = "PhyloProfile", mustWork = TRUE
)
longDf <- createLongMatrix(rawInput)
umapData <- prepareUmapData(longDf, "phylum")
data.umap <- umapClustering(umapData)
plotDf <- createUmapPlotData(data.umap, umapData)
plotUmap(plotDf, font = "sans")
```

plotUmap3D *Create UMAP cluster 3D plot*

Description

Create UMAP cluster 3D plot

Usage

```
plotUmap3D(plotDf = NULL, legendPos = "bottom",
    colorPalette = "Set2", transparent = 0,highlightTaxa = NULL,
    dotZoom = 0)
```

Arguments

plotDf	data for UMAP plot
legendPos	position of legend. Default: "right"
colorPalette	color palette. Default: "Set2"
transparent	transparent level (from 0 to 1). Default: 0
highlightTaxa	list of taxa to be highlighted
dotZoom	dot size zooming factor. Default: 0

Value

A plot as ggplot object

Author(s)

Vinh Tran tran@bio.uni-frankfurt.de

See Also

prepareUmapData, umapClustering, createUmapPlotData

Examples

```
rawInput <- system.file(
   "extdata", "test.main.long", package = "PhyloProfile", mustWork = TRUE
)
longDf <- createLongMatrix(rawInput)
umapData <- prepareUmapData(longDf, "phylum")
data.umap <- umapClustering3D(umapData)
plotDf <- createUmapPlotData(data.umap, umapData)
plotUmap3D(plotDf)
```

ppTaxonomyMatrix *An example of a taxonomy matrix*

Description

An example of a taxonomy matrix

Usage

```
data(ppTaxonomyMatrix)
```

Format

Dataframe

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" ...

ppTree *An example of a taxonomy tree in newick format*

Description

An example of a taxonomy tree in newick format

Usage

```
data(ppTree)
```

Format

Dataframe

Value

A data frame with only one entry

- V1 tree in newick format

prepareUmapData *Prepare data for UMAP*

Description

Prepare data for UMAP

Usage

```
prepareUmapData(longDf = NULL, taxonRank = NULL, type = "taxa",
    taxDB = NULL, filterVar = "both", cutoff = 0, groupLabelsBy = "taxa")
```

Arguments

longDf	input phyloprofile file in long format
taxonRank	taxonomy rank for labels (e.g. "phylum")
type	type of clustering, either "taxa" (default) or "genes"
taxDB	path to taxonomy database
filterVar	choose variable (either "var1", "var2" or "both") to filter the data. Default: "both"
cutoff	cutoff to filter data values. Default: 0
groupLabelsBy	group labels by the number of "taxa" (default) or "genes"

Value

A dataframe in wide format

Author(s)

Vinh Tran tran@bio.uni-frankfurt.de

Examples

```
rawInput <- system.file(
   "extdata", "test.main.long", package = "PhyloProfile", mustWork = TRUE
)
longDf <- createLongMatrix(rawInput)
prepareUmapData(longDf, "phylum")
```

`processNcbiTaxonomy` *Pre-processing NCBI taxonomy data*

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

```
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

Examples

```
## Not run:
?processNcbiTaxonomy
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

Examples

```
?processOrthoID
data("finalProcessedProfile", package="PhyloProfile")
processOrthoID(finalProcessedProfile)
```

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)
```

Format

Dataframe

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 taxonomy 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

```
PhyloProfile:::qualitativeColours(5)
```

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

```
rankListFile <- system.file(
    "extdata", "data/rankList.txt", package = "PhyloProfile", mustWork = TRUE
)
PhyloProfile:::rankIndexing(rankListFile)
```

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) upto the taxonomy root for one taxon in the experimental data sets.

Usage

```
data(rankList)
```

Format

Dataframe

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)
```

`resolveOverlapFeatures` *Modify domain dataframe to resolve overlapped domains/features*

Description

Modify domain dataframe to resolve overlapped domains/features

Usage

```
resolveOverlapFeatures(domainDf)
```

Arguments

`domainDf`	input domain dataframe

Value

Domain dataframe with modified feature names that join multiple domains of the same type that are not overlapped

Author(s)

Vinh Tran tran@bio.uni-frankfurt.de

Examples

```
# get domain data
seedID <- "101621at6656"
domainFile <- system.file(
    "extdata", "domainFiles/101621at6656.domains",
    package = "PhyloProfile", 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",]
# resolve overlapped featuers
PhyloProfile:::resolveOverlapFeatures(orthoDf)
```

runPhyloProfile *Run PhyloProfile app*

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

Examples

```
## Not run:
?runPhyloProfile
runPhyloProfile()

## End(Not run)
```

singleDomainPlotting *Create architecure plot for a single protein*

Description

Create architecure plot for a single protein

Usage

```
singleDomainPlotting(df, geneID, sep, labelSize, titleSize, minStart,
    maxEnd, colorPalette, showScore, showWeight, namePosition, firstDist,
    nameType, nameSize, segmentSize, nameColor, labelPos, font)
```

Arguments

df	Domain dataframe for ploting 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	Lable size. Default = 12
titleSize	Title size. Default = 12
minStart	The smallest start position of all domains
maxEnd	The highest stop position of all domains
colorPalette	Color pallete. Default = Paired"
showScore	Show/hide E-values and Bit-scores. Default = NULL (hide)
showWeight	Show/hide feature weights. Default = NULL (hide)
namePosition	List of positions for domain names, choose from "plot", "legend" or "axis". Default: "plot"
firstDist	Distance of the first domain to plot title. Default = 0.5
nameType	Type of domain names, either "Texts" or "Labels" (default)
nameSize	Size of domain names. Default = 3
segmentSize	Height of domain segment. Default = 5
nameColor	Color of domain names (for Texts only). Default = "black"
labelPos	Position of domain names (for Labels only). Choose from "Above" (default), "Below" or "Inside" the domain bar
font	font of text. Default = Arial"

Value

Domain plot of a single protein as a ggplot object.

Author(s)

Vinh Tran tran@bio.uni-frankfurt.de

See Also

parseDomainInput

Examples

```
seed <- "101621at6656"
ortho <- "101621at6656|AGRPL@224129@0|224129_0:001955|1"
ortho <- gsub("\\|", ":", ortho)
grepID <- paste(seed, "#", ortho, sep = "")
domainFile <- system.file(
```

```
    "extdata", "domainFiles/101621at6656.domains",
    package = "PhyloProfile", mustWork = TRUE
)
domainDf <- parseDomainInput(seed, domainFile, "file")
domainDf$feature_id_mod <- domainDf$feature_id
subdomainDf <- domainDf[grep(grepID, domainDf$seedID), ]
subdomainDf$feature <- as.character(subdomainDf$feature)
orthoDf <- subdomainDf[subdomainDf$orthoID == ortho,]
seedDf <- subdomainDf[subdomainDf$orthoID != ortho,]
minStart <- min(subdomainDf$start)
maxEnd <- max(c(subdomainDf$end, subdomainDf$length))
# resolve overlapping domains
seedDf <- PhyloProfile:::resolveOverlapFeatures(seedDf)
orthoDf <- PhyloProfile:::resolveOverlapFeatures(orthoDf)
# add feature colors
featureColorDf <- PhyloProfile:::addFeatureColors(seedDf, orthoDf)
seedDf <- featureColorDf[[1]]
orthoDf <- featureColorDf[[2]]
# do plot
g <- PhyloProfile:::singleDomainPlotting(
   seedDf, seed, minStart = minStart, maxEnd = maxEnd, font = "sans"
)
grid::grid.draw(g)
```

`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 the 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

```
# get domain data
seedID <- "101621at6656"
domainFile <- system.file(
    "extdata", "domainFiles/101621at6656.domains",
    package = "PhyloProfile", 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
PhyloProfile:::sortDomains(seedDf, orthoDf)
```

`sortDomainsByList` *Sort one domain dataframe based on list of ordered feature types*

Description

Sort domain dataframe of one protein based on a given list of ordered feature types

Usage

```
sortDomainsByList(domainDf = NULL, featureClassOrder = NULL)
```

Arguments

`domainDf`	domain dataframe
`featureClassOrder`	vector of ordered feature classes

Value

Dataframe contains sorted domain list.

Author(s)

Vinh Tran tran@bio.uni-frankfurt.de

Examples

```
# get domain data
seedID <- "101621at6656"
domainFile <- system.file(
    "extdata", "domainFiles/101621at6656.domains",
    package = "PhyloProfile", 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",]
featureClassOrder <- c("pfam", "smart", "tmhmm", "coils", "signalp", "seg",
    "flps")
# sort
PhyloProfile:::sortDomainsByList(orthoDf, featureClassOrder)
```

`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

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

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)
```

taxa2dist *taxa2dist*

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 PhyloProfilData package.

Usage

```
data(taxonNamesReduced)
```

Format

Dataframe

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 talking 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

```
data(taxonomyMatrix)
```

Format

Dataframe

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)
```

---

umapClustering *Perform UMAP clustering 2D*

Description

Perform UMAP clustering 2D

Usage

```
umapClustering(data4umap = NULL, by = "taxa", type = "binary",
    randomSeed = 123)
```

Arguments

data4umap	data for UMAP clustering (output from prepareUmapData)
by	cluster data by "taxa" (default) or "genes"
type	type of data, either "binary" (default) or "non-binary"
randomSeed	random seed. Default: 123

Value

A list contain UMAP cluster objects

Author(s)

Vinh Tran tran@bio.uni-frankfurt.de

See Also

prepareUmapData

Examples

```
rawInput <- system.file(
   "extdata", "test.main.long", package = "PhyloProfile", mustWork = TRUE
)
longDf <- createLongMatrix(rawInput)
data4umap <- prepareUmapData(longDf, "phylum")
umapClustering(data4umap)
```

---

umapClustering3D *Perform UMAP clustering 3D*

Description

Perform UMAP clustering 3D

Usage

```
umapClustering3D(data4umap = NULL, by = "taxa", type = "binary",
    randomSeed = 123)
```

Arguments

data4umap	data for UMAP clustering (output from prepareUmapData)
by	cluster data by "taxa" (default) or "genes"
type	type of data, either "binary" (default) or "non-binary"
randomSeed	random seed. Default: 123

Value

A list contain UMAP cluster objects

Author(s)

Vinh Tran tran@bio.uni-frankfurt.de

See Also

prepareUmapData

Examples

```
rawInput <- system.file(
   "extdata", "test.main.long", package = "PhyloProfile", mustWork = TRUE
)
longDf <- createLongMatrix(rawInput)
data4umap <- prepareUmapData(longDf, "phylum")
umapClustering3D(data4umap)
```

`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"
)
g <- varDistTaxPlot(plotDf, plotParameters)
grid::grid.draw(g)
```

`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