Package ‘DDPNA’

July 29, 2021

Type Package

Title Disease-Driven Differential Proteins Co-Expression Network Analysis

Version 0.3.0

Date 2021-7-29

Author Kefu Liu [aut, cre]

Maintainer Kefu Liu <liukefu19@163.com>

URL https://github.com/liukf10/DDPNA

BugReports https://github.com/liukf10/DDPNA/issues

Description Functions designed to connect disease-related differential proteins and co-expression network. It provides the basic statics analysis included t test, ANOVA analysis. The network construction is not offered by the package, you can used `WGCNA` package which you can learn in Peter et al. (2008) <doi:10.1186/1471-2105-9-559>. It also provides module analysis included PCA analysis, two enrichment analysis, Planner maximally filtered graph extraction and hub analysis.

Imports stats, ggplot2, ggalt, MEGENA, igraph, Hmisc, utils, grDevices, plyr, scales, grid, VennDiagram

Suggests WGCNA, Biostrings, impute, ggfortify

License GPL-2

Encoding UTF-8

Depends R (>= 3.5)

LazyData true

RoxygenNote 6.1.1

NeedsCompilation no

Repository CRAN

Date/Publication 2021-07-29 06:50:02 UTC
R topics documented:

DDPNA-package ................................................................. 2
anova_p ........................................................................ 3
changedID ............................................................... 4
dataStatInf .............................................................. 5
Data_impute ................................................................ 6
DEPsets .......................................................................... 8
DEP_Mod_HeatMap ...................................................... 9
DEP_Mod_net_plot ......................................................... 10
fc.pos ........................................................................ 12
FCSenrichplot ............................................................ 13
getmoduleHub ............................................................. 14
groupparmean ................................................................ 15
ID_match ....................................................................... 16
MaxQdataconvert ........................................................ 17
MaxQprotein ............................................................... 20
ME_inf ........................................................................ 21
modpcomp ..................................................................... 22
moduleID ...................................................................... 23
Module_Enrich ............................................................. 24
Module_inf ..................................................................... 25
multi.t.test .................................................................... 26
P.G.extract ..................................................................... 27
rename_dupnewID ........................................................ 27
single_mod_enrichplot ................................................ 28
SoftThresholdScaleGraph ......................................... 29
wgcnatest ..................................................................... 31

Index 34

DDPNA-package Disease-driven Differential Proteins And Proteomic Co-expression Network Associated Analysis

Description
disease driven proteins associated network in different species crosstalk. The package is used to analysis differential proteomics consensus network in two or more datasets. The function Data_impute need impute package from Bioconductor, the function ID_match and the function MaxQdataconvert need Biostrings package from Bioconductor.
Details

Package: DDPNA
Type: Package
Version: 0.2.5
Create Date: 2019-03-18
Date: 2020-06-26
License: GPL (>= 2)

~~ An overview of how to use the package, including the most important ~~ ~~ functions ~~

Author(s)

Kefu Liu
Maintainer: Kefu Liu <lkf1013@gmail.com>

Description

anova analysis in proteomic data.

Usage

anova_p(data, group)

Arguments

data protein quantification data. column is sample. row is protein ID.
group sample group information

Author(s)

Kefu Liu

Examples

data(imputedData)
data <- imputedData
logD <- data$log2_value
rownames(logD) <- data$inf$ori.ID
group <- gsub("[0-9]+", "", colnames(logD))
anova_P <- anova_p(logD[1:100,], group)
Description

extract significant differential protein

Usage

changedID(relative_value, group, vs.set2, vs.set1 = "WT",
rank = "none", anova = TRUE, anova.cutoff = 0.05,
T.cutoff = 0.05, Padj = "fdr",
cutoff = 1.5, datatype = c("none","log2"), fctype = "all",...)

Arguments

relative_value  protein quantification data

group           sample group information

vs.set2         compared group 2 name

vs.set1         compared group 1 name

rank            order by which type. This must be (an abbreviation of) one of the strings
                "none","foldchange","anova","t"

anova           a logical value indicating whether do anova analysis.

anova.cutoff    a numeric value indicated that anova test p value upper limit.

T.cutoff        a numeric value indicated that t.test p value upper limit.

Padj            p adjust methods of multiple comparisons. it can seen in p.adjust.methods.

cutoff          a numeric value indicated that foldchange lower limit.

datatype        The quantification data is normal data or log2 data.

fctype          foldchange is ordered by up-regulated or down-regulated or changed

...             Other arguments.

Details

extract significant differential protein ID based on foldchange, t.test p value, anova p value.

Value

a vector of protein ID information.

Author(s)

Kefu Liu
dataStatInf

Examples

```r
data(imputedData)
data <- imputedData
logD <- data$log2_value
rownames(logD) <- data$inf$ori.ID
group <- gsub("[0-9]","", colnames(logD))
up <- changedID(logD[201:260,,], group, vs.set2 = "ad", vs.set1 = "ctl",
    rank = "foldchange", anova = FALSE, Padj = "none", cutoff = 1,
    datatype = "log2", fctype = "up")
```

Description

summarize the statistics information of data

Usage

```r
dataStatInf(prodata, group, intensity = "intensity",
Egrp = NULL, Cgrp = "ctl",
meanmethod = "mean", datatype = c("none", "log2"),
anova = TRUE, T.test = c("pairwise", "student", "none"),
Aadj = "none", Tadj = "none", cutoff = FALSE, ...)
```

Arguments

- **prodata**: proteome data. a list Vector which contain two data.frame: ID information and quantification data
- **intensity**: the data.frame name only contain quantification data
- **group**: sample group information
- **Egrp**: experiment group name. It must be assigned when use Student T.test.
- **Cgrp**: control group name. It must be assigned. The default value is "ctl".
- **meanmethod**: Arithmetic mean of sample group or median of sample group. This must be (an abbreviation of) one of the strings "mean","median".
- **datatype**: The quantification data is normal data or log2 data.
- **anova**: a logical value indicating whether do anova analysis.
- **T.test**: T.test method. "none" means not running t.test. "pairwise" means calculate pairwise comparisons between group levels with corrections for multiple testing "student" means student t test. This must be (an abbreviation of) one of the strings "pairwise","student" and "none".
- **Aadj**: anova P value adjust methods. it can seen in p.adjust.methods.
- **Tadj**: T test P value adjust methods. it can seen in p.adjust.methods.
**Description**

Data clean process: detect and remove outlier sample and impute missing value. The process is following: 1. Remove some genes which the number of missing value larger than maxNAratio. 2. Outlier sample detect and remove these sample. 3. Repeat Steps 1-2 untile meet the iteration times or no outlier sample can be detected. 4. impute the missing value. The function also can only do gene filter or remove outlier or impute missing value.

**Usage**

```r
data_impute(data, inf = "inf", intensity = "LFQ", miss.value = NA, splINExt = TRUE, maxNAratio = 0.5, removeOutlier = TRUE, outlierdata = "intensity", iteration = NA, sdout = 2, distmethod = "manhattan", A.IAC = FALSE, dohclust = FALSE, treelabels = NA, plot = TRUE, filename = NULL, text.cex = 0.7, text.col = "red", text.pos = 1, text.labels = NA, abline.col = "red", abline.lwd = 2, impute = TRUE, verbose = 1, ...)
```

**Examples**

```r
data(imputedData)
group <- gsub("[0-9]+", "", colnames(imputedData$intensity))
data <- imputedData
data$inf <- data$inf[1:100,]
data$intensity <- data$intensity[1:100,]
stat <- dataStatInf(data, group, meanmethod = "median", T.test = "pairwise", Adj = "fdr", Tadj = "fdr", cutoff = FALSE)
```
**Data_impute**

**Arguments**

- **data**
  MaxQconvert data or a list Vector which contain two data.frame: ID information and quantification data.
- **inf**
  the data.frame name contain protein ID information.
- **intensity**
  the data.frame name only contain quantification data.
- **miss.value**
  the type of miss.value showed in quantification data. The default value is NA. The miss.value usually can be NA or 0.
- **splNExt**
  a logical value whether extract sample name.(suited for MaxQuant quantification data)
- **maxNAratio**
  The maximum percent missing data allowed in any row (default 50%). For any rows with more than maxNAratio% missing will deleted.
- **removeOutlier**
  a logical value indicated whether remove outlier sample.
- **outlierdata**
  The value is deprecated. which data will be used to analysis outlier sample detect. This must be (an abbreviation of) one of the strings "intensity","relative_value","log2_value".
- **iteration**
  a numeric value indicating how many times it go through the outlier sample detect and remove loop. NA means do loops until no outlier sample.
- **sdout**
  a numeric value indicating the threshold to judge the outlier sample. The default 2 means 0.95 confidence intervals.
- **distmethod**
  The distance measure to be used. This must be (an abbreviation of) one of the strings "manhattan","euclidean","canberra","correlation".
- **A.IAC**
  a logical value indicated whether decreasing correlation variance.
- **dohclust**
  a logical value indicated whether doing hierarchical clustering and plot dendrograms.
- **treelabels**
  labels of dendrograms.
- **plot**
  a logical value indicated whether plot numbersd scatter diagrams.
- **filename**
  the filename of plot. The number and plot type information will added automatically. The default value is NULL which means no file saving. all the plot will be saved to "plot" folder and saved in pdf format.
- **text.cex**
  outlier sample annotation text size (scatter diagrams parameters).
- **text.col**
  outlier sample annotation color (scatter diagrams parameters).
- **text.pos**
  outlier sample annotation position (scatter diagrams parameters).
- **text.labels**
  outlier sample annotation (scatter diagrams parameters).
- **abline.col**
  the threshold line color (scatter diagrams parameters).
- **abline.lwd**
  the threshold line width (scatter diagrams parameters).
- **impute**
  a logical value indicated whether do knn imputation.
- **verbose**
  integer level of verbosity. Zero means silent, 1 means have some Diagnostic Messages.

**Details**

detect and remove outlier sample and impute missing value.
**Value**

- a list of proteomic data.
  - inf: Protein information included protein IDs and other information.
  - intensity: Quantification information.
  - relative_value: intensity divided by geometric mean
  - log2_value: log2 of relative_value

**Author(s)**

Kefu Liu

**Examples**

```r
data(Dforimpute)
data <- Data_impute(Dforimpute,distmethod="manhattan")
```

**Description**

extract two or more IDsets intersection set and complementary set and define the colors.

**Usage**

```r
DEPsets(datalist, colors = c("red", "green", "blue"))
```

**Arguments**

- datalist: a list contains more than two ID sets.
- colors: define each ID sets color.

**Value**

- a list contains intersection set and complementary set information and colors.
  - gene.set: a list of each set ID information.
  - color.code: the colors of each set

**Author(s)**

Kefu Liu
Examples

data(net)
data(imputedData)
Module <- Module_inf(net, imputedData$inf)
group <- gsub("[0-9]+","", colnames(imputedData$intensity))
data <- imputedData
data$inf <- data$inf[1:100,]
data$intensity <- data$intensity[1:100,]
stat <- dataStatInf(data, group, meanmethod = "median",
T.test = "pairwise", Aadj = "fdr",
Tadj = "fdr", cutoff = FALSE)
stat <- rename_dupnewID(stat, Module, DEPfromMod = TRUE)
stat1 <- stat$new.ID[stat$ad > 1]
stat2 <- stat$new.ID[stat$asym > 1]
datalist <- list(stat1 = stat1, stat2 = stat2)
sets <- DEPsets(datalist)

DEP_Mod_HeatMap

Description

get the DEP enrich fold in Module and plot a HeatMap

Usage

DEP_Mod_HeatMap(DEP_Mod, xlab = "DEP", filter = c("p","p.adj"),
cutoff = 0.05, filename = NULL, ...)

Arguments

DEP_Mod a list of DEP_Mod enrichment information. data.frame in list is get from Module_Enrich function.

xlab it indicate x value in heatmap. it must be a value between "DEP" and "MOD".

filter p value or p.adjust value used to filter the enrich significant module.

cutoff a numeric value is the cutoff of p value. Larger than the value will remove to show in plot.

filename plot filename. If filename is null, it will print the plot.

... other argument.

Value

a list of enrich fold heatmap information.

enrichFold enrichFold of DEP in Modules.

textMatrix significant enrichment module information.
Author(s)

Kefu Liu

Examples

```r
# Author and examples start here

data(net)
data(imputedData)
data <- imputedData
logD <- data$log2_value
rownames(logD) <- data$inf$ori.ID
group <- gsub("[0-9]+\", "", colnames(logD))
Module <- Module_inf(net, data$inf)

# define 2 DEP ID data: a and b
a <- Module$ori.ID[1:100]
b <- Module$ori.ID[50:100]
a <- Module_Enrich(Module, a, coln="ori.ID", enrichtype = "ORA")
b <- Module_Enrich(Module, b, coln="ori.ID", enrichtype = "ORA")
rowname <- a$module$name;
a <- data.frame(Counts = a$Counts, module.size = a$module.size, precent = a$precent, p = a$p, p.adj = a$p.adj, Z.score = a$Z.score, stringsAsFactors = FALSE)
rownames(a) <- rowname;
rowname <- b$module$name;
b <- data.frame(Counts = b$Counts, module.size = b$module.size, precent = b$precent, p = b$p, p.adj = b$p.adj, Z.score = b$Z.score, stringsAsFactors = FALSE)
rownames(b) <- rowname;
DEP_Mod <- list(a = a, b = b)
heatMapInf <- DEP_Mod_HeatMap(DEP_Mod)
```

---

**Description**

remove hubs which is not in the IDsets and replot the PFG network

**Usage**

```r
DEP_Mod_net_plot(ModNet, IDsets = NULL, data = NULL, module = NULL, plot = TRUE, filename = NULL, filetype = "pdf", OnlyPlotLast = TRUE, BranchCut = TRUE, reconstructNet = TRUE, iteration = Inf, label.hubs.only = TRUE, node.default.color = "grey", hubLabel.col = "black", ...)
```
Arguments

- **ModNet**: data contains network information which get from `getmoduleHub`
- **IDsets**: ID sets information which get from `DEPsets`
- **data**: the value should be defined only when `reconstructNet` is `TRUE`. The value is proteomic quantification data, which is same as the input in `getmoduleHub`.
- **module**: the value should be defined only when `reconstructNet` is `TRUE`. The value is module information which is same as the input in `getmoduleHub`.
- **plot**: a logical value whether plot a picture.
- **filename**: the filename of plot. The default value is `NULL` which means no file saving. The function is use `ggsave` to achieve.
- **filetype**: the file type of plot. The type should be one of "eps", "ps", "tex" (pictex), "pdf", "jpeg", "tiff", "png", "bmp", "svg" or "wmf" (windows only).
- **OnlyPlotLast**: a logical value whether plot the final network.
- **BranchCut**: a logical value whether remove unhub proteins which have no connection to DEPs.
- **reconstructNet**: a logical value whether reconstruct network.
- **iteration**: iteration times when reconstruct network.
- **label.hubs.only**: a logical value whether show labels for hubs only.
- **node.default.color**: Default node colors for those that do not intersect with signatures in gene.set.
- **hubLabel.col**: Label color for hubs.

Value

- a list contains network information
  - **netgene**: all IDs in network.
  - **hub**: hub IDs
  - **PMFG**: PMFG graph data frame information

Author(s)

- Kefu Liu

Examples

```r
data(net)
data(imputedData)
Module <- Module_inf(net, imputedData$inf)
group <- gsub("[0-9]+","", colnames(imputedData$intensity))
data <- imputedData
data$inf <- data$inf[1:100,]
```
data$intensity <- data$intensity[1:100,]
stat <- dataStatInf(data, group, meanmethod = "median",
                    T.test = "pairwise", Aadj = "fdr",
                    Tadj = "fdr", cutoff = FALSE)
stat1 <- stat$ori.ID[stat$sad > 1]
stat2 <- stat$ori.ID[stat$sasym > 1]
datalist <- list(stat1 = stat1, stat2 = stat2)
sets <- DEPsets(datalist)

logD <- imputedData$log2_value
rownames(logD) <- imputedData$inf$ori.ID
Mod3 <- getmoduleHub(logD, Module, 3, coln = "ori.ID", adjustp = FALSE)

newnet <- DEP_Mod_net_plot(Mod3, sets,
                            data = logD, module = Module,
                            plot = FALSE, filename = NULL, filetype = "pdf",
                            OnlyPlotLast = FALSE,reconstructNet = FALSE)

fc.pos

Description

Pick up proteins based on foldchange and return proteins position in data.

Usage

fc.pos(fc, vs.set2, vs.set1 = "WT",
       cutoff = 1, datatype = c("none", "log2"),
       fctype = "all", order = TRUE)

Arguments

fc  proteomic data of mean value in groups.
vs.set2 compared group 2 name
vs.set1 compared group 1 name
cutoff a numeric value indicated foldchange threshold.
datatype The quantification data is normal data or log2 data. This must be (an abbreviation of) one of the strings "none", "log2".
fctype foldchange is ordered by up-regulated or down-regulated or changed
order a logical value indicated that whether ordered by foldchange.

Author(s)

Kefu Liu
Examples

```r
data(imputedData)
data <- imputedData
relative <- data$relative_value
rownames(relative) <- data$inf$ori.ID
group <- gsub("[0-9]+", ",", colnames(relative))
datamean <- groupmean(relative, group, name = FALSE)
fc_1vs2 <- fc.pos(datamean, vs.set2 = "ad", vs.set1 = "ctl",
cutoff = 1, datatype = "none",
fctype = "up", order = TRUE)
fcs_ID <- rownames(relative)[fc_1vs2]
```

Description

Plot of FCS enrichment analysis

Usage

```r
FCSenrichplot(FCSenrich, count = 1, p = 0.05, filter = "p",
plot = TRUE, filename = NULL, filetype = "pdf", ...)
```

Arguments

- **FCSenrich**: FCS enrichment information which is gotten in `module_enrich` function.
- **count**: a numeric value. Module will choosed when count number is larger than count value.
- **p**: a numeric value. Module will choosed when any Fisher's extract test p value is less than count value.
- **filter**: filter methods. This must be (an abbreviation of) one of the strings "p", "p.adj", "none".
- **plot**: a logical value indicating whether draw enrichment variation trend plot.
- **filename**: the filename of plot. The default value is `NULL` which means no file saving. The plot will be saved to "plot" folder.
- **filetype**: the file type of plot. The type should be one of "eps", "ps", "tex" (pictex), "pdf", "jpeg", "tiff", "png", "bmp", "svg" or "wmf" (windows only).
- **...**: Other arguments.

Author(s)

Kefu Liu
Examples

```r
data(imputedData)
data(net)
data <- imputedData
logD <- data$log2_value
rownames(logD) <- data$inf$ori.ID
group <- gsub("[0-9]+,"", colnames(logD))
Module <- Module_inf(net, data$inf)
pos<-which(Module$moduleNum %in% c(11:13))
up <- changedID(logD[pos,], group, vs.set2 = "ad", vs.set1 = "ctl",
                rank = "foldchange",anova = FALSE, Padj = "none", cutoff = 1,
                datatype = "log2", fctype = "up")
FCSenrich <- Module_Enrich(Module[pos,], up, coln="ori.ID")
FCSenrich <- FCSenrichplot(FCSenrich)
```

Description

extract PMFG information and get Module hub proteins.

Usage

```r
getmoduleHub(data, module, mod_num, coln = "new.ID",
cor.sig = 0.05, cor.r = 0, cor.adj="none",
adjustp = TRUE, hub.p = 0.05)
```

Arguments

data        proteomic quantification data.
module      module information which is getted in Module_inf function.
mod_num     the module name which module will be calculate.
coln        column name of module contains protein IDs. it could be matched with "classifiedID"
cor.sig     a numeric value indicated that correlation p value less than cor.sig will be picked.
cor.r       a numeric value indicated that correlation r value larger than cor.r will be picked.
cor.adj     P value correction method. method information can see in p.adjust.method
adjustp     a logical value indicating whether pick hub protein by FDR methods.
hub.p       a numeric value indicated that hub proteins are p value less than hub.p.
Value

A list contains PMFG network information. list(hub = hubgene, degreeStat = Stat, graph = g, PMFG = gg)

- **hub**: hub information.
- **degreeStat**: degree statistics information
- **graph**: the original graph data frame
- **PMFG**: PMFG graph data frame

Author(s)

Kefu Liu

Examples

data(net)
data(imputedData)
data <- imputedData
logD <- data$log2_value
rownames(logD) <- data$inf$ori.ID
group <- gsub("[0-9]+", ".", colnames(logD))
Module <- Module_inf(net, data$inf)
Mod10 <- getmoduleHub(logD, Module, 10, coln = "ori.ID", adjustp = FALSE)
if (requireNamespace("MEGENA", quietly = TRUE)) {
library(MEGENA)
plot_subgraph(module = Mod10$degreeStat$gene, hub = Mod10$hub, PMFG = Mod10$PMFG,
node.default.color = "black", gene.set = NULL, color.code = c("grey"), show.legend = TRUE,
label.hubs.only = TRUE, hubLabel.col = "red", hubLabel.sizeProp = 0.5,
show.topn.hubs = 10, node.sizeProp = 13, label.sizeProp = 13,
label.scaleFactor = 10, layout = "kamada.kawai")
}

Description

Mean of sample group

Usage

groupmean(data, group, method = c("mean", "median"), name = TRUE)
Arguments

- **data**: protein quantification data. column is sample, row is protein ID.
- **group**: sample group information
- **method**: Arithmetic mean of sample group or median of sample group. This must be (an abbreviation of) one of the strings "mean", "median".
- **name**: a logical value indicated whether add "mean" or "median" in sample group name.

Author(s)

- Kefu Liu

Examples

```r
data(imputedData)
data <- imputedData
logD <- data$log2_value
group <- gsub("[0-9]+","", colnames(logD))
datamean <- groupmean(logD, group, name = FALSE)
```

### Description

homolog protein Uniprot ID match

### Usage

```r
ID_match(data, db1.path = NULL, db2.path = NULL, out.folder = NULL,
         blast.path = NULL, evalue = 0.1, verbose = 1)
```

### Arguments

- **data**: dataset of protein information. Column Names should contain "ori.ID" and "ENTRY.NAME". "ori.ID" is Uniprot ID
- **db1.path**: fasta file, database of transferred species
- **db2.path**: fasta file, database of original species
- **out.folder**: blast result output folder, the folder path should be the same with db1.path
- **blast.path**: blast+ software install path
- **evalue**: blast threshold, the lower means more rigorous
- **verbose**: integer level of verbosity. Zero means silent, 1 means have Diagnostic Messages.
Details

homolog protein Uniprot ID match is based on the ENTRY.NAME, gene name and sequence homophyly in two different species or different version of database.

Value

da.data.frame included 4 columns: ori.ID, ENTRY.NAME, new.ID, match.type.

Note

This function should install 'blast+' software, Version 2.7.1. 'blast+' download website:https://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/LATEST/
If unstall 'blast+' software, it could use R function replaced, but it will take a lot of time. db1.path, db2.path, out.folder are both need the complete path. Out.folder and db1.path should be in the same folder. Path should have no special character. data should have colname: ori.ID, ENTRY.NAME.

Author(s)

Kefu Liu

Examples

# suggested to install blast+ software
# it will take a long time without blast+ software
data(Sample_ID_data)
if(requireNamespace("Biostrings", quietly = TRUE)){
  out.folder = tempdir();
  write.table(Sample_ID_data$db1,file.path(out.folder,"db1.fasta"),
    quote = FALSE,row.names = FALSE, col.names = FALSE);
  write.table(Sample_ID_data$db2,file.path(out.folder,"db2.fasta"),
    quote = FALSE,row.names = FALSE, col.names = FALSE);
  data <- ID_match(Sample_ID_data$ID_match_data,
    db1.path = file.path(out.folder,"db1.fasta"),
    db2.path = file.path(out.folder,"db2.fasta"),
    out.folder = out.folder,
    blast.path = NULL,
    evalue = 0.1, verbose = 1)
  file.remove( file.path(out.folder,"db1.fasta"),
              file.path(out.folder,"db2.fasta"))
}

MaxQdataconvert  one-step to extract 'Maxquant' quantification data and convert

Description

'Maxquant' quantification data extract and homolog protein Uniprot ID match.
Usage

MaxQdataconvert(pgfilename, IDname = "Majority.protein.IDs", IDtype = c("MaxQ","none"), CONremove = TRUE, justID = TRUE, status1 = TRUE, ENTRY1 = TRUE, db1.path = NULL, db2.path = NULL, out.folder = NULL, blast.path = NULL, savecsvpath = NULL, csvfilename = NULL, verbose = 1, ...)

Arguments

pgfilename 'Maxquant' quantification file "protein groups.txt"
IDname The column name of uniprot ID. The default value is "Majority.protein.IDs" which is the column name in MaxQuant data.
IDtype "MaxQ" means proteinGroups is 'Maxquant' quantification data, "none" means other type data. This must be (an abbreviation of) one of the strings: "MaxQ","none".
CONremove a logical value indicated whether remove contaminant IDs. When IDtype is "none", it will remove unmatch ID compared with database2.
justID a logical value indicated whether only extract ID when IDtype is "MaxQ".
status1 a logical value indicated whether extract the first ID status when IDtype is "MaxQ".
ENTRY1 a logical value indicated whether extract the first ID ENTRY NAME when IDtype is "MaxQ".
db1.path fasta file, database of transferred species
db2.path fasta file, database of original species
out.folder blast result output folder, the folder path should be the same with db1.path
blast.path blast+ software install path
savecsvpath the information of csv file name output path. The default value means don’t save csv file.
csvfilename the name of csv file which the data are to be output. The default value means don’t save csv file.
verbose integer level of verbosity. Zero means silent, higher values make the output progressively more and more verbose.
... Other arguments.

Details

one-step to extract MaxQuant or other quantification data and convert. The function contain ID_match function.
Value

a list of proteomic information.

protein_IDs Portein IDs which is IDname column information.

intensity Quantification intensity information. When IDtype is "none", it is the QuanCol columns information.

iBAQ Quantification iBAQ intensity information.(only for IDtype is "MaxQ")

LFQ Quantification LFQ intensity information.(only for IDtype is "MaxQ")

Note

db1.path, db2.path, out.folder are both need the complete path. Out.folder and db1.path should be in the same folder. Path should have no special character.

Author(s)

Kefu Liu

See Also

ID_match

Examples

# suggested to install blast+ software

data(Sample_ID_data)
if(requireNamespace("Biostrings", quietly = TRUE)){
  out.folder = tempdir();
  write.table(Sample_ID_data$db1,file.path(out.folder,"db1.fasta"),
              quote = FALSE,row.names = FALSE, col.names = FALSE);
  write.table(Sample_ID_data$db2,file.path(out.folder,"db2.fasta"),
              quote = FALSE,row.names = FALSE, col.names = FALSE);
  write.table(Sample_ID_data$pginf,
              file = file.path(out.folder,"proteingroups.txt"),
              quote = FALSE,
              sep = "\t",dec = ".", row.names = FALSE, col.names = TRUE );
  Maxdata <- MaxQdataconvert(file.path(out.folder,"proteingroups.txt"),
                              IDtype = "MaxQ",
                              db1.path = file.path(out.folder,"db1.fasta"),
                              db2.path = file.path(out.folder,"db2.fasta"),
                              out.folder = out.folder,
                              blast.path = NULL)
  file.remove( file.path(out.folder,"db1.fasta"),
              file.path(out.folder,"db2.fasta"),
              file.path(out.folder,"proteingroups.txt") )
}

MaxQprotein

MaxQprotein reads proteomic quantification data and separate the protein information and quantification information.

Description

The function will separate data into 4 parts: protein information, intensity, iBAQ and LFQ (iBAQ and LFQ only fit for 'MaxQuant' software result). For MaxQ data, it can remove the contaminant and reverse protein.

Usage

MaxQprotein(proteinGroups, IDname = "Majority.protein.IDs",
IDtype = "MaxQ", remove = TRUE, QuanCol = NULL,
verbose = 1)

Arguments

proteinGroups  the proteomic quantification data
IDname  The column name of uniprot ID. The default value is "Majority.protein.IDs" which is the column name in MaxQuant data.
IDtype  "MaxQ" means proteinGroups is Maxquant quantification data, "none" means other type data. This must be (an abbreviation of) one of the strings: "MaxQ", "none".
remove  a logical value indicated whether remove contaminant and reverse ID.
QuanCol  The quantification data columns. It's only needed when IDtype is "none". When IDtype is "none" and QuanCol is not given, the intensity will auto extract all columns except IDname as quantification data. It may have error in next analysis.
verbose  integer level of verbosity. Zero means silent, 1 means have Diagnostic Messages.

Value

a list of proteomic information.

protein_IDs  Protein IDs which is IDname column information.
intensity  Quantification intensity information. When IDtype is "none", it is the QuanCol columns information.
iBAQ  Quantification iBAQ intensity information (only for IDtype is "MaxQ")
LFQ  Quantification LFQ intensity information (only for IDtype is "MaxQ")

Author(s)

Kefu Liu
ME_inf

Examples

```r
data(ProteomicData)
# example for MaxQ Data
MaxQdata <- MaxQprotein(ProteomicData$MaxQ)
# example for other type Data
otherdata <- MaxQprotein(ProteomicData$none, IDname = "Protein",
                       IDtype = "none", QuanCol = 2:9)
```

---

**ME_inf**

*module eigengenes information*

---

**Description**

put sample names as rownames in WGCNA module eigenvalue data.frame.

**Usage**

```r
ME_inf(MEs, data, intensity.type = "LFQ", rowname = NULL)
```

**Arguments**

- **MEs**
  module eigenvalue which is calculated in WGCNA package.
- **data**
  protein quantification data. column is sample. row is protein ID.
- **intensity.type**
  quantification data type, which can help extract sample name. This must be (an abbreviation of) one of the strings "LFQ", "intensity", "iBAQ", "none".
- **rowname**
  sample names when "intensity.type" is "none", rowname will be used.

**Author(s)**

Kefu Liu

**Examples**

```r
data(net)
data(imputedData)
data <- imputedData
logD <- data$log2_value
MEs <- ME_inf(net$MEs, logD)
```
modpcomp

Description
extract module pca component

Usage
modpcomp(data, colors, nPC = 2,
plot = FALSE, filename = NULL, group = NULL)

Arguments
data protein quantification data. column is sample. row is protein ID.
colors protein and module information. which is calculated in WGCNA package.
nPC how many PCA component will saved.
plot a logical value indicating whether draw PCA plot. This function need load
ggfortify first.
filename The filename of plot. The default value is NULL which means no file saving. The
plot will be saved to "plot" folder and saved in pdf format.
group sample group information.

Author(s)
Kefu Liu

Examples
data(net)
data(imputedData)
data <- imputedData
logD <- data$log2_value
rownames(logD) <- data$inf$ori.ID
Module_PCA <- modpcomp(logD, net$colors)

# if plot PCA and plot module 6 PCA
group <- gsub("[0-9]+", ",", colnames(logD))
pos <- which(net$colors == 6)
if (requireNamespace("ggfortify", quietly = TRUE)){
require("ggfortify")
Module_PCA <- modpcomp(logD[pos,], net$colors[pos], plot = TRUE, group = group)
}
moduleID

extract intersection ID between dataset and module

Description

extract intersection ID between dataset and one of module

Usage

moduleID(inf, module, num, coln = "new.ID")

Arguments

inf dataset protein ID information. a vector of protein IDs.
module module information which is getted in Module_inf function.
num module number which will extract to compared with dataset ID information.
coln column names of module protein IDs.

Details

column coln information in module when module number is num intersect with inf.

Author(s)

Kefu Liu

Examples

data(net)
data(imputedData)
data <- imputedData
logD <- data$log2_value
rownames(logD) <- data$inf$ori.ID
group <- gsub("[0-9]+", "", colnames(logD))
Module <- Module_inf(net, data$inf)
up <- changedID(logD, group, vs.set2 = "ad", vs.set1 = "ctl",
        rank = "foldchange",anova = FALSE, Padj = "none",cutoff = 1,
        datatype = "log2",fc_type = "up")
intersection <- moduleID(up, Module, 5, coln = "ori.ID")
Module_Enrich

Description

Enrichment analysis of a sets of proteins in all modules. The function offered two enrichment methods: ORA and FCS.

Usage

```
Module_Enrich(module, classifiedID, enrichtype = "FCS",
              coln = "new.ID", datainf = NULL, p.adj.method = "BH")
```

Arguments

- **module**: module information which is getted in `Module_inf` function.
- **classifiedID**: a sets of protein IDs which is ordered by change value/ p value and so on.
- **enrichtype**: enrichment method. This must be (an abbreviation of) one of the strings "FCS", "ORA". "FCS" means analyzes step-by-step a proteins list which is ordered by change ratio/ p value and so on. "ORA" means analyzes a proteins list by Fisher's extract test.
- **coln**: column name of module contains protein IDs. It could be matched with "classifiedID"
- **datainf**: proteomic data protein ID information. The default value is "NULL". Which is means that the "classifiedID" come from proteomic information is the same as the module construction proteomic information. If they are different, proteomic data information should be given.
- **p.adj.method**: p adjust methods of multiple comparisons. It can seen in `p.adjust.methods`.

Value

A list contains classifiedID enrichment information.

- **Counts**: the counts of classifiedID in module.
- **module.size**: the number of module ID
- **module.name**: module name
- **precent**: counts divided module.size
- **p**: enrichment p value in each module
- **p.adj**: enrichment p.adj value in each module
- **Z.score**: Z score is -log2 P value.

Author(s)

Kefu Liu
Examples

data(net)
data(imputedData)
data <- imputedData
logD <- data$log2_value
rownames(logD) <- data$inf$ori.ID
group <- gsub("[0-9]+", ",", colnames(logD))
Module <- Module_inf(net, data$inf)
up <- changedID(logD, group, vs.set2 = "ad", vs.set1 = "ctl",
rank = "foldchange", anova = FALSE, Padj = "none", cutoff = 1,
datatype = "log2", fctype = "up")
FCSenrich <- Module_Enrich(Module, up, coln="ori.ID")

Module_inf

Module and protein information.

Description

module and protein information match

Usage

Module_inf(net, inf, infType = "Convert", IDname = NULL, ...)

Arguments

net module network which is calculated in WGCNA package.
inf proteome quantification data information which contains protein IDs.
inftype data information type. This must be (an abbreviation of) one of the strings "Convert", "MaxQ", "none". "Convert" means protein ID is converted by MaxquantDataconvert function. "MaxQ" means original Maxquant software quantification data information.
IDname IDname is "inf" column names which will extract.
... other argument.

Author(s)

Kefu Liu

Examples

data(net)
data(imputedData)
data <- imputedData
Module <- Module_inf(net, data$inf)
multi.t.test

Description

multiple comparisons t test and choose significant proteins in proteomic data.

Usage

```r
multi.t.test(data, group,
  sig = 0.05, Adj.sig = TRUE,
  grpAdj = "bonferroni",
  geneAdj = "fdr", ...)
```

Arguments

data protein quantification data. column is sample, row is protein ID.
group sample group information
sig significance P value threshold. The default is 0.05.
Adj.sig a logical value indicated that whether adjust P-values for multiple proteins comparisons in each two groups.
grpAdj adjust multiple groups comparisons P-value in each two groups. The default is "bonferroni". it can seen in p.adjust.methods.
geneAdj adjust multiple proteins comparisons P-value in each group. The default is "fdr". it can seen in p.adjust.methods.
...
Other arguments.

Author(s)

Kefu Liu

Examples

```r
data(imputedData)
data <- imputedData
logD <- data$log2_value
rownames(logD) <- data$inf$ori.ID
group <- gsub("[0-9]+", ",", colnames(logD))
Tsig_P <- multi.t.test(logD[1:100,], group, Adj.sig = FALSE, geneAdj = "fdr")
```
**P.G.extract**

_Protein Groups information extract._

**Description**

uniprot ID, ENTRYNAME and status information extract (only fit for 'MaxQuant' data.)

**Usage**

\[
P.G.extract(inf, ncol = 4, justID = FALSE, status1 = FALSE, ENTRY1 = FALSE, verbose = 0)
\]

**Arguments**

- **inf**: protein groups IDs information.
- **ncol**: column numbers of output result.
- **justID**: a logical value indicated whether only extract uniprot ID.
- **status1**: a logical value indicated whether extract the first ID status.
- **ENTRY1**: a logical value indicated whether extract the first ID ENTRY NAME.
- **verbose**: integer level of verbosity. Zero means silent, 1 means have Diagnostic Messages.

**Author(s)**

Kefu Liu

**Examples**

\[
\begin{align*}
\text{data(ProteomicData)} \\
\text{MaxQdata <- MaxQprotein(ProteomicData$MaxQ)} \\
\text{inf <- P.G.extract(MaxQdata$protein_IDs, justID = TRUE, status = TRUE, ENTRY = TRUE)}
\end{align*}
\]

---

**rename_dupnewID**

rename the duplicated newID in moduleinf and renew the ID in DEPstat

**Usage**

\[
\text{rename_dupnewID(DEPstat, moduleinf, DEPfromMod = FALSE)}
\]
**Arguments**

- **DEPstat** a dataframe contains columns: "new.ID" and "ori.ID". it can get from `dataStatInf`.
- **moduleinf** a dataframe contains columns: "new.ID" and "ori.ID". it can get from `Module_inf`.
- **DEPfromMod** a logical value indicated that whether DEPstat and moduleinf is getted from the same datasets. The default value is FALSE.

**Value**

a data.frame contains DEPstat information and renewed the new.ID column.

**Author(s)**

Kefu Liu

**Examples**

```r
data(net)
data(imputedData)
Module <- Module_inf(net, imputedData$inf)
group <- gsub("[0-9]+","", colnames(imputedData$intensity))
data <- imputedData
data$inf <- data$inf[1:100,]
data$intensity <- data$intensity[1:100,]
stat <- dataStatInf(data, group, meanmethod = "median",
                    T.test = "pairwise", Aadj = "fdr",
                    Tadj = "fdr", cutoff = FALSE)
stat <- renameDupnewID(stat, Module, DEPfromMod = TRUE)
```

**Description**

FCS enrichment analysis of a sets of proteins in one module.

**Usage**

```r
single_mod_enrichplot(module, Mod_Nam, classifiedID,
                      coln = "new.ID", datainf = NULL,
                      plot = TRUE, filename = NULL, ...)
```
Arguments

- **module**: module information which is getted in `Module_inf` function.
- **Mod_Nam**: the module name which module will be calculate.
- **classifiedID**: a sets of protein IDs which is ordered by change value/ p value and so on.
- **coln**: column name of module contains protein IDs. it could be matched with "classifiedID"
- **datainf**: proteomic data protein ID information. The default value is "NULL". which is means that the "classifiedID" come from proteomic information is the same as the module construction proteomic information. If they are different, proteomic data information should be given.
- **plot**: a logical value indicating whether draw enrichment variation trend plot.
- **filename**: the filename of plot. The default value is NULL which means no file saving. The plot will be saved to 'plot' folder and saved in pdf format.

Author(s)

Kefu Liu

Examples

```r
# Example code

data(net)
data(imputedData)
data <- imputedData
logD <- data$log2_value
rownames(logD) <- data$inf$ori.ID
group <- gsub("[0-9]+", ",", colnames(logD))
Module <- Module_inf(net, data$inf)
up <- changedID(logD, group, vs.set2 = "ad", vs.set1 = "ctl",
rank = "foldchange", anova = FALSE, Padj = "none", cutoff = 1,
datatype = "log2", fctype = "up")
m5enrich <- single_mod_enrichplot(Module, 5, up, coln="ori.ID")
```

Description

pick soft thresholding powers for WGCNA analysis and plot

Usage

```r
SoftThresholdScaleGraph(data,
xlab = "Soft Threshold (power)",
ylab = "Scale Free Topology Model Fit, signed R^2",
main = "Scale independence",
filename = NULL)
```
**Arguments**

- `data`: protein quantification data. row is sample, column is protein ID.
- `xlab`: x axis label
- `ylab`: y axis label
- `main`: plot title
- `filename`: the filename of plot. The default value is NULL which means no file saving. The plot will be saved to "plot" folder and saved in pdf format.

**Details**

pick soft thresholding powers for WGCNA analysis and plot. The function is also can replaced by "pickSoftThreshold" function in WGCNA package.

**Value**

A list with the following components:

- `powerEstimate`: the lowest power fit for scale free topology.
- `fitIndices`: a data frame containing the fit indices for scale free topology.

**Author(s)**

Kefu Liu

**See Also**

pickSoftThreshold in WGCNA package.

**Examples**

```r
# it will take some times
data(imputedData)
data <- imputedData
logD <- data$log2_value
rownames(logD) <- data$inf$ori.ID
if (requireNamespace("WGCNA", quietly = TRUE))
  sft <- SoftThresholdScaleGraph(t(logD))
```
Description

The major parameter optimization in function blockwiseModules in WGCNA package. The function will do a series of network construction by change various parameter in blockwiseModules and record the result. (it will take a long time)

Usage

\[
\text{wgcnatest}(\text{data, power = NULL, TOMType = "unsigned", detectCutHeight = NULL, maxBlockSize = 5000, deepSplit = TRUE, minModSize = TRUE, pamRespectsDendro = FALSE, minKMEtoStay = TRUE, minCoreKME = FALSE, reassignThreshold = FALSE, mergeCutHeight = FALSE, maxModNum = 30, minModNum = 8, MaxMod0ratio = 0.3)}
\]

Arguments

data  protein quantification data used in network construction. Row is sample. Column is protein ID. More information can get from blockwiseModules in WGCNA package.

power  Soft-thresholding power for network construction. The default value is NULL. it will run pickSoftThreshold function in WGCNA package to pick the lowest appropriate power. More information can get from blockwiseModules in WGCNA package.

TOMType  one of "none", "unsigned", "signed". More information can get from blockwiseModules in WGCNA package.

detectCutHeight  dendrogram cut height for module detection. The default value is NULL, which means it will calculate the cutheight through correlation r when p value is 0.05. When the value is larger than 0.995, it will set to detectCutHeight or 0.995. More information can get from blockwiseModules in WGCNA package.

maxBlockSize  integer giving maximum block size for module detection. More information can get from blockwiseModules in WGCNA package.

deepSplit  The default value is TRUE, which means the function will test deepSplit from 0 to 4. If the value is FALSE, deepSplit is 2. You also can setting integer value between 0 and 4 by yourself. integer value between 0 and 4. More information can get from blockwiseModules in WGCNA package.
minModSize minimum module size for module detection. The default value is TRUE, which means the function will test 15, 20, 30, 50. If the value is FALSE, minModSize is 20. You also can setting integer value by yourself. More information can get from blockwiseModules in WGCNA package.

pamRespectsDendro a logical value indicated that whether do pamStage or not. More information can get from blockwiseModules in WGCNA package.

minKMEtoStay The default value is TRUE, which means the function will test 0.1, 0.2, 0.3. If the value is FALSE, minKMEtoStay is 0.3. You also can setting value by yourself. Value between 0 to 1. More information can get from blockwiseModules in WGCNA package.

minCoreKME The default value is FALSE, minCoreKME is 0.5. If the value is TRUE, which means the function will test 0.4 and 0.5. You also can setting value by yourself. Value between 0 to 1. More information can get from blockwiseModules in WGCNA package.

reassignThreshold p-value ratio threshold for reassigning genes between modules. The default value is FALSE, reassignThreshold is 1e-6. If the value is TRUE, which means the function will test 0.01 and 0.05. You also can setting value by yourself. More information can get from blockwiseModules in WGCNA package.

mergeCutHeight dendrogram cut height for module merging. The default value is FALSE, mergeCutHeight is 0.15. If the value is TRUE, which means the function will test 0.15, 0.3 and 0.45. You also can setting value by yourself. More information can get from blockwiseModules in WGCNA package.

maxModNum The maximum module number. If network construction make more than maxModNum of modules. The result will not record.

minModNum The minimum module number. If network construction make less than minModNum of modules. The result will not record.

MaxMod0ratio The maximum Mod0 protein numbers ratio in total proteins. If network construction make more than MaxMod0ratio in module 0. The result will not record.

Details
More information can get from blockwiseModules in WGCNA package.

Value
a data.frame contains protein number in each module and the parameter information.

Author(s)
Kefu Liu
Examples

data(imputedData)
wgcndata <- t(imputedData$intensity)
sft <- SoftThresholdScaleGraph(wgcndata)
# It will take a lot of time
if (requireNamespace("WGCNA", quietly = TRUE)){
  require("WGCNA")
  WGCNAadjust <- wgcnatest(wgcndata, power = sft$powerEstimate)
}
Index

* package
  DDPNA-package, 2
  anova_p, 3
  changedID, 4
  Data_impute, 6
  dataStatInf, 5
  DDPNA (DDPNA-package), 2
  DDPNA-package, 2
  DEP_Mod_HeatMap, 9
  DEP_Mod_net_plot, 10
  DEPsets, 8
  fc.pos, 12
  FCSenrichplot, 13
  getmoduleHub, 14
  groupmean, 15
  ID_match, 16, 18, 19
  MaxQdataconvert, 17
  MaxQprotein, 20
  ME_inf, 21
  modpcomp, 22
  Module_Enrich, 24
  Module_inf, 25
  moduleID, 23
  multi.t.test, 26
  P.G.extract, 27
  rename_dupnewID, 27
  single_mod_enrichplot, 28
  SoftThresholdScaleGraph, 29
  wgcnatest, 31