Package 'pwOmics'

April 12, 2018

Type Package

Title Pathway-based data integration of omics data

Version 1.10.1

Date 2014-12-29

Author Astrid Wachter <Astrid.Wachter@med.uni-goettingen.de>

Maintainer Maren Sitte <Maren.Sitte@med.uni-goettingen.de>

Description pwOmics performs pathway-based level-specific data comparison of matching omics data sets based on pre-analysed user-specified lists of differential genes/transcripts and phosphoproteins. A separate downstream analysis of phosphoproteomic data including pathway identification, transcription factor identification and target gene identification is opposed to the upstream analysis starting with gene or transcript information as basis for identification of upstream transcription factors and potential proteomic regulators. The cross-platform comparative analysis allows for comprehensive analysis of single time point experiments and time-series experiments by providing static and dynamic analysis tools for data integration. In addition, it provides functions to identify individual signaling axes based on data integration.

License GPL (>= 2)

Depends R (>= 3.2)

Imports data.table, rBiopaxParser, igraph, STRINGdb, graphics, gplots, Biobase, BiocGenerics, AnnotationDbi, biomaRt, AnnotationHub, GenomicRanges, graph, grDevices, stats, utils

Suggests ebdbNet, longitudinal, Mfuzz

biocViews SystemsBiology, Transcription, GeneTarget, GeneSignaling

NeedsCompilation no

RoxygenNote 6.0.1

R topics documented:

`pwIntOmics-package` *Pathway-based data integration of omics data*

Description

pwIntOmics performs pathway-based level-specific data comparison of matching omics data sets based on pre-analysed user-specified lists of differential genes/transcripts and proteins. A separate downstream analysis of proteomic data including pathway identification and enrichment analysis, transcription factor identification and target gene identification is opposed to the upstream analysis starting with gene or transcript information as basis for identification of upstream transcription factors and regulators. The cross-platform comparative analysis allows for comprehensive analysis of single time point experiments and time-series experiments by providing static and dynamic analysis tools for data integration.

Details

Package:	pwIntOmics
Type:	Package
Version:	1.0
Date:	2014-12-29
License:	GLP-3

Author(s)

Astrid Wachter Maintainer: Astrid Wachter <Astrid.Wachter@med.uni-goettingen.de>

addFeedbackLoops *Add feedback loops from target genes to proteins/TFs if present.*

Description

Add feedback loops from target genes to proteins/TFs if present.

Usage

```
addFeedbackLoops(ST_net_targets)
```

Arguments

ST_net_targets	full consensus graph.

Value

igraph object with feedback loops added.

clusterTimeProfiles *Clustering of time profiles.*

Description

Soft clustering of time series data with Mfuzz R package [1]. Filtering of genes with low expression changes possible via min.std parameter. Expression values are standardized and undergo fuzzy c-means clustering based on minimization of weighted square error function (see [2]). Fuzzifier parameter m is estimated via mestimate function of [1] based on a relation proposed by Schwaemmle and Jansen [3]. The optimal number of clusters is determined via the minimum distance between cluster centroid using Dmin function of [3]. Be aware that the cluster number determination might be difficult especially for short time series measurements.

Usage

```
clusterTimeProfiles(dynConsensusNet, min.std = 0, ncenters = 12)
```

Arguments

dynConsensusNet	result of dynamic analysis: inferred net generated by consDynamicNet function.
min.std	threshold parameter to exclude genes with a low standard deviation. All genes with an expression smaller than min.std will be excluded from clustering. Default value is 0.
ncenters	integer specifying the maximum number of centers which should be tested in minimum distance between cluster centroid test; this number is used as initial number to determine the data-specific maximal cluster number based on number of distinct data points.

Value

output dataframe of mfuzz function.

References

1. L. Kumar and M. Futschik, Mfuzz: a software package for soft clustering of microarray data, Bioinformation, 2(1) 5-7, 2007.

2. Bezdak JC, Pattern Recognition with Fuzzy Objective Function Algorithms, Plenum Press, New York, 1981.

3. Schwaemmle and Jensen, Bioinformatics, Vol. 26 (22), 2841-2848, 2010.

Examples

```
## Not run:
data(OmicsExampleData)
data_omics = readOmics(tp_prots = c(0.25, 1, 4, 8, 13, 18, 24),
tp_genes = c(1, 4, 8, 13, 18, 24), OmicsExampleData,
PWdatabase = c("biocarta", "kegg", "nci", "reactome"),
TFtargetdatabase = c("userspec"))
data_omics = readPhosphodata(data_omics,
phosphoreg = system.file("extdata", "phospho_reg_table.txt",
package = "pwOmics.newupdown"))
data_omics = readTFdata(data_omics,
TF_target_path = system.file("extdata", "TF_targets.txt",
package = "pwOmics.newupdown"))
data_omics_plus = readPWdata(data_omics,
loadgenelists = system.file("extdata/Genelists", package = "pwOmics.newupdown"))

## End(Not run)
## Not run:
data_omics_plus = identifyPR(data_omics_plus)
setwd(system.file("extdata/Genelists", package = "pwOmics.newupdown"))
data_omics = identifyPWs(data_omics_plus)
data_omics = identifyTFs(data_omics)
data_omics = identifyRsofTFs(data_omics,
noTFs_inPW = 1, order_neighbors = 10)
data_omics = identifyPWTFTGs(data_omics)
statConsNet = staticConsensusNet(data_omics)
consDynNet = consDynamicNet(data_omics, statConsNet)
clusterTimeProfiles(consDynNet)

## End(Not run)
```

`consDynamicNet` *Dynamic analysis.*

Description

Generates continous data for dynamic analysis of protein, TF and gene data via smoothing splines. 50 time points are generated this way. The following nodes are considered: Nodes which are part of the static consensus graphs from corresponding time points of the two different measurement types. In case a node is not significantly changed at a certain point in time its FC is assumed to

remain constant at this time point. Calculation of the consensus-based dynamic net parameters are based on the ebdbNet R package [1]. The number of time points generated via smoothing splines (50) is based on their results for median AUCs of ROC curves. The number of forward time units a node is assumed to influence other nodes can be specified via the laghankel parameter. The cutoff determining the percent of total variance explained by the singular values generated by singular value decomposition (SVD) of the block-Hankel matrix H in order to specify the hidden state dimension K (for further details see [1]).

Usage

```
consDynamicNet(data_omics, consensusGraphs, laghankel = 3,
  cutoffhankel = 0.9, conv.1 = 0.15, conv.2 = 0.05, conv.3 = 0.05,
  verbose = TRUE, max.iter = 100, max.subiter = 200)
```

Arguments

`data_omics`	OmicsData object.
`consensusGraphs`	result from static analysis: consensus graph generated by staticConsensusNet function.
`laghankel`	integer specifying the maximum relevant time lag to be used in constructing the block-Hankel matrix.
`cutoffhankel`	cutoff to determine desired percent of total variance explained; default = 0.9 as in [1].
`conv.1`	value of convergence criterion 1; default value is 0.15 (for further details see [1]).
`conv.2`	value of convergence criterion 2; default value is 0.05 (for further details see [1]).
`conv.3`	value of convergence criterion 3; default value is 0.05 (for further details see [1]).
`verbose`	boolean value, verbose output TRUE or FALSE
`max.iter`	maximum overall iterations; default value is 100 (for further details see [1]).
`max.subiter`	maximum iterations for hyperparameter updates; default value is 200 (for further details see [1]).

Value

list of 2 elements: 1) output parameters of dynamic network inference with ebdbNet package 2) splines data generated.

References

1. A. Rau, F. Jaffrezic, J.-L. Foulley, R. W. Doerge (2010). An empirical Bayesian method for estimating biological networks from temporal microarray data. Statistical Applications in Genetics and Molecular Biology, vol. 9, iss. 1, article 9.

Examples

```
## Not run:
data(OmicsExampleData)
data_omics = readOmics(tp_prots = c(0.25, 1, 4, 8, 13, 18, 24),
```

```
tp_genes = c(1, 4, 8, 13, 18, 24), OmicsExampleData,
PWdatabase = c("biocarta", "kegg", "nci", "reactome"),
TFtargetdatabase = c("userspec"))
data_omics = readPhosphodata(data_omics,
phosphoreg = system.file("extdata", "phospho_reg_table.txt",
package = "pwOmics.newupdown"))
data_omics = readTFdata(data_omics,
TF_target_path = system.file("extdata", "TF_targets.txt",
package = "pwOmics.newupdown"))
data_omics_plus = readPWdata(data_omics,
loadgenelists = system.file("extdata/Genelists", package = "pwOmics.newupdown"))

## End(Not run)
## Not run:
data_omics_plus = identifyPR(data_omics_plus)
setwd(system.file("extdata/Genelists", package = "pwOmics.newupdown"))
data_omics = identifyPWs(data_omics_plus)
data_omics = identifyTFs(data_omics)
data_omics = identifyRsofTFs(data_omics,
noTFs_inPW = 1, order_neighbors = 10)
data_omics = identifyPWTFTGs(data_omics)
statConsNet = staticConsensusNet(data_omics)
dynConsNet = consDynamicNet(data_omics, statConsNet)

## End(Not run)
```

`createBiopaxnew` *Create a new Biopax model containing all database information.*

Description

This function creates a new biopax model depending on which pathway databases are chosen for analysis.

Usage

```
createBiopaxnew(intIDs, pwdatabases)
```

Arguments

`intIDs`	output list of genintIDs function.
`pwdatabases`	vector indicating with which pathway database the pathways should be determined; possible choices are "biocarta", "kegg", "nci", "reactome".

Value

biopax object generated from the specified pathway databases.

createIntIDs *Create internal IDs.*

Description

Create new internal ids in biopax$df:

Usage

```
createIntIDs(data_omics, PWinfo)
```

Arguments

data_omics	OmicsData object.
PWinfo	pathway database information from chosen pathway databases as from loadPWs.

Value

list of internal IDs from specified pathway databases.

findSignalingAxes *Find downstream signaling axis.*

Description

This function determines the regulated downstream structures of a selected phosphoprotein at a specified time point.

Usage

```
findSignalingAxes(data_omics, phosphoprot, tpDS)
```

Arguments

data_omics	OmicsData object.
phosphoprot	character specifying the name of the phosphoprotein that is selected as the starting point for downstream analysis.
tpDS	integer specifying the time point considered for downstream analysis of phosphoprotein data .

Value

list of downstream pathways identified for this time point of phosphoprotein measurement with sublists containing information about the transcription factor, the regulation, the target genes of these transcription factors and the matching transcripts.

Examples

```
## Not run:
data(OmicsExampleData)
data_omics = readOmics(tp_prots = c(0.25, 1, 4, 8, 13, 18, 24),
tp_genes = c(1, 4, 8, 13, 18, 24), OmicsExampleData,
PWdatabase = c("biocarta", "kegg", "nci", "reactome"),
TFtargetdatabase = c("userspec"))
data_omics = readPhosphodata(data_omics,
phosphoreg = system.file("extdata", "phospho_reg_table.txt",
package = "pwOmics.newupdown"))
data_omics = readTFdata(data_omics,
TF_target_path = system.file("extdata", "TF_targets.txt",
package = "pwOmics.newupdown"))
data_omics_plus = readPWdata(data_omics,
loadgenelists = system.file("extdata/Genelists", package = "pwOmics.newupdown"))
setwd(system.file("extdata/Genelists", package = "pwOmics.newupdown"))

## End(Not run)
## Not run:
data_omics_plus = identifyPR(data_omics_plus)
data_omics = identifyPWs(data_omics_plus)
data_omics = identifyTFs(data_omics)
data_omics = identifyPWTFTGs(data_omics)
data_omics = identifyRsofTFs(data_omics, noTFs_inPW = 1, order_neighbors = 10)
SYK_axis = findSignalingAxes(data_omics,phosphoprot = "SYK", tpDS = 2)

## End(Not run)
```

`findxneighborsoverlap` *Find overlap of next neighbors of transcription factors in identified pathways.*

Description

Find the overlap of x next neighbors of transcription factors in identified pathways. Writes the overlap into a given list called 'regulators'.

Usage

```
findxneighborsoverlap(neighbors, noTFs_inPW, regul)
```

Arguments

`neighbors`	list of x next neighbors for each transcription factor in the pathway as provided by findxnextneighbors function.
`noTFs_inPW`	numeric value specifying number of TFs being at least part of the pathway.
`regul`	list element of regulators list for current pathway.

Value

list of regulators identified in x next neighbors of TFs.

`findxnextneighbors` *Find next neighbors of transcription factors in identified pathways. Produces a list of x next neighbors for each transcription factor in the pathway.*

Description

Find next neighbors of transcription factors in identified pathways.

Produces a list of x next neighbors for each transcription factor in the pathway.

Usage

```
findxnextneighbors(data_omics, pws_morex_TFs, pwxTFs, order_neighbors)
```

Arguments

`data_omics`	OmicsData object.
`pws_morex_TFs`	list of transcription factors in identified pathways.
`pwxTFs`	numeric variable of pathway currently investigated (from pws_morexTFs).
`order_neighbors`	integer specifiying the order of the neighborhood: order 1 is TF plus its immediate neighbors.

Value

list of x next neighbors for each TF in the pathway.

`generate_DSSignalingBase` *Generate a folder with downstream information about all phosphoproteins.*

Description

This function generates a folder structure with an RData object and csv tables for each timepoint sepcified for each phosphoprotein considered in the data_omics object.

Usage

```
generate_DSSignalingBase(data_omics, timepoints = c(0.25, 1, 4, 8, 13, 18,
  24))
```

Arguments

`data_omics`	OmicsData object.
`timepoints`	integer vector specifying the timepoints of interest for downstream analysis.

Value

Folder structure in working directory, containing phosphoprotein downstream information in individual folders.

Examples

```
## Not run:
data(OmicsExampleData)
data_omics = readOmics(tp_prots = c(0.25, 1, 4, 8, 13, 18, 24),
tp_genes = c(1, 4, 8, 13, 18, 24), OmicsExampleData,
PWdatabase = c("biocarta", "kegg", "nci", "reactome"),
TFtargetdatabase = c("userspec"))
data_omics = readPhosphodata(data_omics,
phosphoreg = system.file("extdata", "phospho_reg_table.txt",
package = "pwOmics.newupdown"))
data_omics = readTFdata(data_omics,
TF_target_path = system.file("extdata", "TF_targets.txt",
package = "pwOmics.newupdown"))
data_omics_plus = readPWdata(data_omics,
loadgenelists = system.file("extdata/Genelists", package = "pwOmics.newupdown"))

## End(Not run)
## Not run:
data_omics_plus = identifyPR(data_omics_plus)
setwd(system.file("extdata/Genelists", package = "pwOmics.newupdown"))
data_omics = identifyPWs(data_omics_plus)
data_omics = identifyTFs(data_omics)
data_omics = identifyPWTFTGs(data_omics)
data_omics = identifyRsofTFs(data_omics, noTFs_inPW = 1, order_neighbors = 10)
setwd("~/Signaling_axes/")
generate_DSSignalingBase(data_omics, timepoints = c(0.25, 1, 4, 8, 13, 18, 24))

## End(Not run)
```

---

genfullConsensusGraph *Combine SteinerNet with bipartite graph to get full consensus network.*

Description

Combine SteinerNet with bipartite graph to get full consensus network.

Usage

```
genfullConsensusGraph(ST_net, ST_TFTG)
```

Arguments

ST_net	steiner tree graph generated by SteinerTree_cons function.
ST_TFTG	steiner tree graph extended with consensus target genes and the edges between TFs and target genes.

Value

igraph object of network comprising steiner tree graph and TF - target gene interactions.

`genGenelists` *Generate genelists from pathway databases.*

Description

This function generates genelists from the chosen pathway databases for further processing in detPathways function.

Usage

```
genGenelists(intIDs, pwdatabases)
```

Arguments

`intIDs`	list containing Biopax model with newly generated internal IDs as processed with genintIDs function. The components of the list are biopax models for "biocarta", "kegg", "nci", "reactome". In case a database was not chosen the list entry contains a NA.
`pwdatabases`	vector indicating with which pathway database the pathways should be determined; possible choices are "biocarta", "kegg", "nci", "reactome".

Value

list of genelists of specified pathway databases.

`genGenelistssub` *Generate internally genelists from pathway databases.*

Description

This function generates genelists for a particular pathway database for further processing in detPathways function.

Usage

```
genGenelistssub(intIDs, database_int, PWDB_name)
```

Arguments

`intIDs`	list containing Biopax model with newly generated internal IDs as processed with genintIDs function. The components of the list are biopax models for "biocarta", "kegg", "nci", "reactome". In case a database was not chosen the list entry contains a NA.
`database_int`	integer indicating database entry in indIDs (output of genintIDs); biocarta = 1, kegg = 2, nci = 4, reactome = 4.
`PWDB_name`	character; pathway database name.

Value

data.table of genelist of particular pathway database.

`genIntIDs` *Internal function for generation of pathway database specific internal IDs.*

Description

Generates new internal ids (database-specific) in biopax$df.

Usage

```
genIntIDs(data_omics, PWinfo, PWinfo_ind, PWDBname)
```

Arguments

`data_omics`	OmicsData object.
`PWinfo`	pathway database information from chosen pathway databases as from loadPWs.
`PWinfo_ind`	integer specifying element of loadPWs output matching the chosen pathway database.
`PWDBname`	character; pathway database name.

Value

data.table with newly generated internal IDs of biopax model.

`getAliasfromSTRINGIDs` *Map alias names to STRING IDs of consensus graph.*

Description

Map alias names to STRING IDs of consensus graph.

Usage

```
getAliasfromSTRINGIDs(data_omics, ST_net, updown = FALSE, phospho = TRUE,
  consSTRINGIDs, tps, string_db)
```

Arguments

data_omics	OmicsData object.
ST_net	steiner tree graph generated by SteinerTree_cons function.
updown	boolean value; TRUE in case up- and downregulation should be checked individually for intersection. Type of checking is defined with parameter 'phospho'.
phospho	boolean value; TRUE in case up- and downregulation should be checked based on provided downstream phosphoprotein influence from identifyPR function; FALSE in case up- and downregulation should be checked for without phosphoprotein database knowledge. Default is TRUE.
consSTRINGIDs	first element of list generated by getConsensusSTRINGIDs function; a data.frame including the proteins to be considered as terminal nodes in Steiner tree with colnames ST_proteins and the corresponding STRING IDs in column 'STRING_id'.
tps	integer specifying current timepoint under consideration.
string_db	second element of list generated by getConsensusSTRINGIDs function; species table (for human) of STRING database.

Value

igraph object with alias name annotation.

---

getAlias_Ensemble *Get Gene Symbols from Ensemble Gene Accession IDs.*

Description

Get Gene Symbols from Ensemble Gene Accession IDs.

Usage

```
getAlias_Ensemble(ids)
```

Arguments

ids	vector of Ensemble Gene Accession IDs.

Value

ids character vector of gene symbols.

getBiopaxModel *Get upstream regulators of identified transcription factors.*

Description

Get upstream regulators of identified transcription factors.

Usage

```
getBiopaxModel(data_omics)
```

Arguments

data_omics	OmicsData object.

Value

biopax model generated as consensus biopax models from all pathway databases selected for analysis.

Examples

```
## Not run:
data(OmicsExampleData)
data_omics = readOmics(tp_prots = c(0.25, 1, 4, 8, 13, 18, 24),
tp_genes = c(1, 4, 8, 13, 18, 24), OmicsExampleData,
PWdatabase = c("biocarta", "kegg", "nci", "reactome"),
TFtargetdatabase = c("userspec"))
data_omics = readPhosphodata(data_omics,
phosphoreg = system.file("extdata", "phospho_reg_table.txt",
package = "pwOmics.newupdown"))
data_omics = readTFdata(data_omics,
TF_target_path = system.file("extdata", "TF_targets.txt",
package = "pwOmics.newupdown"))
data_omics_plus = readPWdata(data_omics,
loadgenelists = system.file("extdata/Genelists", package = "pwOmics.newupdown"))

## End(Not run)
## Not run:
data_omics_plus = identifyPR(data_omics_plus)
setwd(system.file("extdata/Genelists", package = "pwOmics.newupdown"))
data_omics = identifyPWs(data_omics_plus)
getBiopaxModel(data_omics)

## End(Not run)
```

getbipartitegraphInfo *Get TF-target gene information for the consensus graph.*

Description

Get TF-target gene information for the consensus graph.

Usage

```
getbipartitegraphInfo(data_omics, tps, updown = FALSE, phospho = TRUE)
```

Arguments

data_omics	OmicsData object.
tps	integer specifying current timepoint under consideration.
updown	boolean value; TRUE in case up- and downregulation should be checked individually for intersection. Type of checking is defined with parameter 'phospho'.
phospho	boolean value; TRUE in case up- and downregulation should be checked based on provided downstream phosphoprotein influence from identifyPR function; FALSE in case up- and downregulation should be checked for without phosphoprotein database knowledge. Default is TRUE.

Value

list of transcription factor target gene interactions.

getConsensusSTRINGIDs *Get consensus graph in STRING IDs.*

Description

Get consensus graph in STRING IDs.

Usage

```
getConsensusSTRINGIDs(data_omics, tps, string_db, updown = FALSE,
  phospho = TRUE)
```

Arguments

data_omics	OmicsData object.
tps	integer specifying current timepoint under consideration.
string_db	STRING_db object.
updown	boolean value; TRUE in case up- and downregulation should be checked individually for intersection. Type of checking is defined with parameter 'phospho'.
phospho	boolean value; TRUE in case up- and downregulation should be checked based on provided downstream phosphoprotein influence from identifyPR function; FALSE in case up- and downregulation should be checked for without phosphoprotein database knowledge. Default is TRUE.

Value

igraph object consensus graph with STRING IDs (only including proteins and transcription factors).

`getDS_PWs` *Get downstream analysis pathways.*

Description

This function returns pathways identified in the downstream analysis containing the significantly abundant proteins.

Usage

```
getDS_PWs(data_omics)
```

Arguments

`data_omics`	OmicsData object.

Value

list of length = number of protein time points, each element containing a data frame with the pathway IDs in the generated biopax model and corresponding pathway names.

Examples

```
## Not run:
data(OmicsExampleData)
data_omics = readOmics(tp_prots = c(0.25, 1, 4, 8, 13, 18, 24),
tp_genes = c(1, 4, 8, 13, 18, 24), OmicsExampleData,
PWdatabase = c("biocarta", "kegg", "nci", "reactome"),
TFtargetdatabase = c("userspec"))
data_omics = readPhosphodata(data_omics,
phosphoreg = system.file("extdata", "phospho_reg_table.txt",
package = "pwOmics.newupdown"))
data_omics = readTFdata(data_omics,
TF_target_path = system.file("extdata", "TF_targets.txt",
package = "pwOmics.newupdown"))
data_omics_plus = readPWdata(data_omics,
loadgenelists = system.file("extdata/Genelists", package = "pwOmics.newupdown"))

## End(Not run)
## Not run:
data_omics_plus = identifyPR(data_omics_plus)
setwd(system.file("extdata/Genelists", package = "pwOmics.newupdown"))
data_omics = identifyPWs(data_omics_plus)
data_omics = identifyTFs(data_omics)
data_omics = identifyPWTFTGs(data_omics)
getDS_PWs(data_omics)

## End(Not run)
```

getDS_TFs *Get downstream analysis transcription factors in pathways.*

Description

This function returns the genes identified in the downstream analysis and a column indicating if the genes are transcription factors.

Usage

```
getDS_TFs(data_omics)
```

Arguments

data_omics	OmicsData object.

Value

list of length = number of protein time points, each element containing a character vector with identified transcription factors.

Examples

```
## Not run:
data(OmicsExampleData)
data_omics = readOmics(tp_prots = c(0.25, 1, 4, 8, 13, 18, 24),
tp_genes = c(1, 4, 8, 13, 18, 24), OmicsExampleData,
PWdatabase = c("biocarta", "kegg", "nci", "reactome"),
TFtargetdatabase = c("userspec"))
data_omics = readPhosphodata(data_omics,
phosphoreg = system.file("extdata", "phospho_reg_table.txt",
package = "pwOmics.newupdown"))
data_omics = readTFdata(data_omics,
TF_target_path = system.file("extdata", "TF_targets.txt",
package = "pwOmics.newupdown"))
data_omics_plus = readPWdata(data_omics,
loadgenelists = system.file("extdata/Genelists", package = "pwOmics.newupdown"))

## End(Not run)
## Not run:
data_omics_plus = identifyPR(data_omics_plus)
setwd(system.file("extdata/Genelists", package = "pwOmics.newupdown"))
data_omics = identifyPWs(data_omics_plus)
data_omics = identifyTFs(data_omics)
data_omics = identifyPWTFTGs(data_omics)
getDS_TFs(data_omics)

## End(Not run)
```

getDS_TGs *Get downstream analysis target genes of TFs found in pathways.*

Description

Get downstream analysis target genes of TFs found in pathways.

Usage

```
getDS_TGs(data_omics)
```

Arguments

data_omics	OmicsData object.

Value

list of length = number of protein time points, each element containing a character vector with identified target genes.

Examples

```
## Not run:
data(OmicsExampleData)
data_omics = readOmics(tp_prots = c(0.25, 1, 4, 8, 13, 18, 24),
tp_genes = c(1, 4, 8, 13, 18, 24), OmicsExampleData,
PWdatabase = c("biocarta", "kegg", "nci", "reactome"),
TFtargetdatabase = c("userspec"))
data_omics = readPhosphodata(data_omics,
phosphoreg = system.file("extdata", "phospho_reg_table.txt",
package = "pwOmics.newupdown"))
data_omics = readTFdata(data_omics,
TF_target_path = system.file("extdata", "TF_targets.txt",
package = "pwOmics.newupdown"))
data_omics_plus = readPWdata(data_omics,
loadgenelists = system.file("extdata/Genelists", package = "pwOmics.newupdown"))

## End(Not run)
## Not run:
data_omics_plus = identifyPR(data_omics_plus)
setwd(system.file("extdata/Genelists", package = "pwOmics.newupdown"))
data_omics = identifyPWs(data_omics_plus)
data_omics = identifyTFs(data_omics)
data_omics = identifyPWTFTGs(data_omics)
getDS_TGs(data_omics)

## End(Not run)
```

getFCsplines *Get fold change splines.*

Description

Calculate the splines used for the dynamic analysis.

Usage

```
getFCsplines(data_omics, nodes, nodetype)
```

Arguments

data_omics	OmicsData object.
nodes	character vector of nodes the fold change splines should be calculated for.
nodetype	character indicating to calculate splines for "proteins" or "genes".

Value

splines values used in dynamic analysis.

getGenesIntersection *Get genes intersection for the omics data on the different time points.*

Description

Get genes intersection for the omics data on the different time points.

Usage

```
getGenesIntersection(data_omics, tp_prot, tp_genes, updown = FALSE,
  phospho = TRUE)
```

Arguments

data_omics	OmicsData object.
tp_prot	numeric integer defining protein timepoint measurement chosen for comparison.
tp_genes	numeric integer defining gene/transcript timepoint measurement chosen for comparison.
updown	boolean value; TRUE in case up- and downregulation should be checked individually for intersection. Type of checking is defined with parameter 'phospho'.
phospho	boolean value; TRUE in case up- and downregulation should be checked based on provided downstream phosphoprotein influence from identifyPR function; FALSE in case up- and downregulation should be checked for without phosphoprotein database knowledge. Default is TRUE.

Value

list with three elements: 1) character vector of gene IDs identified in both upstream and downstream analysis 2) protein time point 3) gene/transcript time point.

Examples

```
## Not run:
data(OmicsExampleData)
data_omics = readOmics(tp_prots = c(0.25, 1, 4, 8, 13, 18, 24),
tp_genes = c(1, 4, 8, 13, 18, 24), OmicsExampleData,
PWdatabase = c("biocarta", "kegg", "nci", "reactome"),
TFtargetdatabase = c("userspec"))
data_omics = readPhosphodata(data_omics,
phosphoreg = system.file("extdata", "phospho_reg_table.txt",
package = "pwOmics.newupdown"))
data_omics = readTFdata(data_omics,
TF_target_path = system.file("extdata", "TF_targets.txt",
package = "pwOmics.newupdown"))
data_omics_plus = readPWdata(data_omics,
loadgenelists = system.file("extdata/Genelists", package = "pwOmics.newupdown"))

## End(Not run)
## Not run:
data_omics_plus = identifyPR(data_omics_plus)
setwd(system.file("extdata/Genelists", package = "pwOmics.newupdown"))
data_omics = identifyPWs(data_omics_plus)
data_omics = identifyTFs(data_omics)
data_omics = identifyPWTFTGs(data_omics)
data_omics = identifyRsofTFs(data_omics, noTFs_inPW = 1, order_neighbors = 10)
getGenesIntersection(data_omics, tp_prot = 4, tp_genes = 4, updown = FALSE,
phospho = TRUE)

## End(Not run)
```

`getOmicsallGeneIDs` *Get all gene IDs.*

Description

This function returns the gene IDs of all genes (transcripts) measured.

Usage

```
getOmicsallGeneIDs(data_omics)
```

Arguments

`data_omics`	OmicsData object.

Value

all gene IDs.

Examples

```
## Not run:
data(OmicsExampleData)
data_omics = readOmics(tp_prots = c(0.25, 1, 4, 8, 13, 18, 24),
tp_genes = c(1, 4, 8, 13, 18, 24), OmicsExampleData,
PWdatabase = c("biocarta", "kegg", "nci", "reactome"),
TFtargetdatabase = c("userspec"))
getOmicsallGeneIDs(data_omics)

## End(Not run)
```

`getOmicsallProteinIDs` *Get all protein IDs.*

Description

This function returns the protein IDs of all proteins measured.

Usage

```
getOmicsallProteinIDs(data_omics)
```

Arguments

`data_omics`	OmicsData object.

Value

all protein IDs.

Examples

```
## Not run:
data(OmicsExampleData)
data_omics = readOmics(tp_prots = c(0.25, 1, 4, 8, 13, 18, 24),
tp_genes = c(1, 4, 8, 13, 18, 24), OmicsExampleData,
PWdatabase = c("biocarta", "kegg", "nci", "reactome"),
TFtargetdatabase = c("userspec"))
getOmicsallProteinIDs(data_omics)

## End(Not run)
```

getOmicsDataset *Get Omics dataset.*

Description

This function returns the omics datasets of the experiment.

Usage

```
getOmicsDataset(data_omics, writeData = FALSE)
```

Arguments

data_omics	OmicsData object.
writeData	boolean value indicating if datasets should be written into csv file.

Value

list with protein data set as first element and gene data set as second element; both elements contain matrices with significant proteins/ genes/transcripts at the given time points.

Examples

```
## Not run:
data(OmicsExampleData)
data_omics = readOmics(tp_prots = c(0.25, 1, 4, 8, 13, 18, 24),
tp_genes = c(1, 4, 8, 13, 18, 24), OmicsExampleData,
PWdatabase = c("biocarta", "kegg", "nci", "reactome"),
TFtargetdatabase = c("userspec"))
getOmicsDataset(data_omics)

## End(Not run)
```

getOmicsTimepoints *Get Omics timepoints.*

Description

This function returns the timepoints of the OmicsData.

Usage

```
getOmicsTimepoints(data_omics)
```

Arguments

data_omics	OmicsData object.

Value

list of protein time points and gene time points; in case of single time point measurement experiment number entered in OmicsData object.

Examples

```
## Not run:
data(OmicsExampleData)
data_omics = readOmics(tp_prots = c(0.25, 1, 4, 8, 13, 18, 24),
tp_genes = c(1, 4, 8, 13, 18, 24), OmicsExampleData,
PWdatabase = c("biocarta", "kegg", "nci", "reactome"),
TFtargetdatabase = c("userspec"))
getOmicsTimepoints(data_omics)

## End(Not run)
```

`getProteinIntersection`

Get protein intersection for the omics data on the different time points. The timepoints or measurement names for comparison have to be defined in tp_prot and tp_genes as given in the readOmics function.

Description

Get protein intersection for the omics data on the different time points.

The timepoints or measurement names for comparison have to be defined in tp_prot and tp_genes as given in the readOmics function.

Usage

```
getProteinIntersection(data_omics, tp_prot, tp_genes, updown = FALSE,
  phospho = TRUE)
```

Arguments

`data_omics`	OmicsData object.
`tp_prot`	numeric integer defining protein timepoint measurement chosen for comparison.
`tp_genes`	numeric integer defining gene/transcript timepoint measurement chosen for comparison.
`updown`	boolean value; TRUE in case up- and downregulation should be checked individually for intersection. Type of checking is defined with parameter 'phospho'.
`phospho`	boolean value; TRUE in case up- and downregulation should be checked based on provided downstream phosphoprotein influence from identifyPR function; FALSE in case up- and downregulation should be checked for without phosphoprotein database knowledge. Default is TRUE.

Value

list with three elements: 1) character vector of protein IDs identified in both upstream and downstream analysis 2) protein time point 3) gene/transcript time point.

Examples

```
## Not run:
data(OmicsExampleData)
data_omics = readOmics(tp_prots = c(0.25, 1, 4, 8, 13, 18, 24),
tp_genes = c(1, 4, 8, 13, 18, 24), OmicsExampleData,
PWdatabase = c("biocarta", "kegg", "nci", "reactome"),
TFtargetdatabase = c("userspec"))
data_omics = readPhosphodata(data_omics,
phosphoreg = system.file("extdata", "phospho_reg_table.txt",
package = "pwOmics.newupdown"))
data_omics = readTFdata(data_omics,
TF_target_path = system.file("extdata", "TF_targets.txt",
package = "pwOmics.newupdown"))
data_omics_plus = readPWdata(data_omics,
loadgenelists = system.file("extdata/Genelists", package = "pwOmics.newupdown"))

## End(Not run)
## Not run:
data_omics_plus = identifyPR(data_omics_plus)
setwd(system.file("extdata/Genelists", package = "pwOmics.newupdown"))
data_omics = identifyPWs(data_omics_plus)
data_omics = identifyTFs(data_omics)
data_omics = identifyPWTFTGs(data_omics)
data_omics = identifyRsofTFs(data_omics, noTFs_inPW = 1, order_neighbors = 10)
getProteinIntersection(data_omics, tp_prot = 4, tp_genes = 4,
updown = FALSE, phospho = TRUE)

## End(Not run)
```

`getSTRING_graph` *Generate STRING PPI graph.*

Description

Generates connected graph with undirected edges from STRING PPI-database.

Usage

```
getSTRING_graph(string_db)
```

Arguments

`string_db`	STRING_db object generated by getConsensusSTRINGIDs function.

Value

igraph object connected graph from STRING PPI database.

`getTFIntersection` *Get TF intersection for the omics data on the different time points.*

Description

Get TF intersection for the omics data on the different time points.

Usage

```
getTFIntersection(data_omics, tp_prot, tp_genes, updown = FALSE,
  phospho = TRUE)
```

Arguments

data_omics	OmicsData object.
tp_prot	numeric integer defining protein timepoint measurement chosen for comparison.
tp_genes	numeric integer defining gene/transcript timepoint measurement chosen for comparison.
updown	boolean value; TRUE in case up- and downregulation should be checked individually for intersection. Type of checking is defined with parameter 'phospho'.
phospho	boolean value; TRUE in case up- and downregulation should be checked based on provided downstream phosphoprotein influence from identifyPR function; FALSE in case up- and downregulation should be checked for without phosphoprotein database knowledge. Default is TRUE.

Value

list with three elements: 1) character vector of transcription factor IDs identified in both upstream and downstream analysis 2) protein time point 3) gene/transcript time point.

Examples

```
## Not run:
data(OmicsExampleData)
data_omics = readOmics(tp_prots = c(0.25, 1, 4, 8, 13, 18, 24),
tp_genes = c(1, 4, 8, 13, 18, 24), OmicsExampleData,
PWdatabase = c("biocarta", "kegg", "nci", "reactome"),
TFtargetdatabase = c("userspec"))
data_omics = readPhosphodata(data_omics,
phosphoreg = system.file("extdata", "phospho_reg_table.txt",
package = "pwOmics.newupdown"))
data_omics = readTFdata(data_omics,
TF_target_path = system.file("extdata", "TF_targets.txt",
package = "pwOmics.newupdown"))
data_omics_plus = readPWdata(data_omics,
loadgenelists = system.file("extdata/Genelists", package = "pwOmics.newupdown"))

## End(Not run)
## Not run:
data_omics_plus = identifyPR(data_omics_plus)
setwd(system.file("extdata/Genelists", package = "pwOmics.newupdown"))
data_omics = identifyPWs(data_omics_plus)
```

```
data_omics = identifyTFs(data_omics)
data_omics = identifyPWTFTGs(data_omics)
data_omics = identifyRsofTFs(data_omics, noTFs_inPW = 1, order_neighbors = 10)
getTFIntersection(data_omics, tp_prot = 4, tp_genes = 4, updown = FALSE,
phospho = TRUE)

## End(Not run)
```

`gettpIntersection` *Get omics data intersection on the three levels. Get intersection for the omics data on all three levels (proteins, TFs, genes) on corresponding time points.*

Description

Get omics data intersection on the three levels.

Get intersection for the omics data on all three levels (proteins, TFs, genes) on corresponding time points.

Usage

```
gettpIntersection(data_omics, updown = FALSE, phospho = TRUE)
```

Arguments

`data_omics`	OmicsData object.
`updown`	boolean value; TRUE in case up- and downregulation should be checked individually for intersection. Type of checking is defined with parameter 'phospho'.
`phospho`	boolean value; TRUE in case up- and downregulation should be checked based on provided downstream phosphoprotein influence from identifyPR function; FALSE in case up- and downregulation should be checked for without phosphoprotein database knowledge. Default is TRUE.

Value

list with three elements: 1) protein intersection 2) transcription factor intersection 3) gene intersection each element contains a list with overlapping time points of both upstream and downstream analyses.

Examples

```
## Not run:
data(OmicsExampleData)
data_omics = readOmics(tp_prots = c(0.25, 1, 4, 8, 13, 18, 24),
tp_genes = c(1, 4, 8, 13, 18, 24), OmicsExampleData,
PWdatabase = c("biocarta", "kegg", "nci", "reactome"),
TFtargetdatabase = c("userspec"))
data_omics = readPhosphodata(data_omics,
phosphoreg = system.file("extdata", "phospho_reg_table.txt",
package = "pwOmics.newupdown"))
data_omics = readTFdata(data_omics,
TF_target_path = system.file("extdata", "TF_targets.txt",
```

```
package = "pwOmics.newupdown"))
data_omics_plus = readPWdata(data_omics,
loadgenelists = system.file("extdata/Genelists", package = "pwOmics.newupdown"))

## End(Not run)
## Not run:
data_omics_plus = identifyPR(data_omics_plus)
setwd(system.file("extdata/Genelists", package = "pwOmics.newupdown"))
data_omics = identifyPWs(data_omics_plus)
data_omics = identifyTFs(data_omics)
data_omics = identifyPWTFTGs(data_omics)
data_omics = identifyRsofTFs(data_omics, noTFs_inPW = 1, order_neighbors = 10)
gettpIntersection(data_omics, updown = FALSE, phospho = TRUE)

## End(Not run)
```

`getUS_PWs` *Get upstream pathways of identified transcription factors.*

Description

Get upstream pathways of identified transcription factors.

Usage

```
getUS_PWs(data_omics)
```

Arguments

`data_omics`	OmicsData object.

Value

list of length = number of gene/transcript time points, each element containing a list of transcription factors; these transcription factor elements contain data frame with pathway IDs and pathway names.

Examples

```
## Not run:
data(OmicsExampleData)
data_omics = readOmics(tp_prots = c(0.25, 1, 4, 8, 13, 18, 24),
tp_genes = c(1, 4, 8, 13, 18, 24), OmicsExampleData,
PWdatabase = c("biocarta", "kegg", "nci", "reactome"),
TFtargetdatabase = c("userspec"))
data_omics = readPhosphodata(data_omics,
phosphoreg = system.file("extdata", "phospho_reg_table.txt",
package = "pwOmics.newupdown"))
data_omics = readTFdata(data_omics,
TF_target_path = system.file("extdata", "TF_targets.txt",
package = "pwOmics.newupdown"))
data_omics_plus = readPWdata(data_omics,
loadgenelists = system.file("extdata/Genelists", package = "pwOmics.newupdown"))
data_omics_plus = identifyPR(data_omics_plus)
```

```
## End(Not run)
## Not run:
setwd(system.file("extdata/Genelists", package = "pwOmics.newupdown"))
data_omics = identifyPWs(data_omics_plus)
data_omics = identifyTFs(data_omics)
data_omics = identifyRsofTFs(data_omics,
noTFs_inPW = 1, order_neighbors = 10)
getUS_PWs(data_omics)

## End(Not run)
```

`getUS_regulators` *Get upstream regulators of identified transcription factors.*

Description

Get upstream regulators of identified transcription factors.

Usage

```
getUS_regulators(data_omics)
```

Arguments

`data_omics`	OmicsData object.

Value

list of length = number of gene/transcript time points, each element containing a vector of protein regulator IDs.

Examples

```
## Not run:
data(OmicsExampleData)
data_omics = readOmics(tp_prots = c(0.25, 1, 4, 8, 13, 18, 24),
tp_genes = c(1, 4, 8, 13, 18, 24), OmicsExampleData,
PWdatabase = c("biocarta", "kegg", "nci", "reactome"),
TFtargetdatabase = c("userspec"))
data_omics = readPhosphodata(data_omics,
phosphoreg = system.file("extdata", "phospho_reg_table.txt",
package = "pwOmics.newupdown"))
data_omics = readTFdata(data_omics,
TF_target_path = system.file("extdata", "TF_targets.txt",
package = "pwOmics.newupdown"))
data_omics_plus = readPWdata(data_omics,
loadgenelists = system.file("extdata/Genelists", package = "pwOmics.newupdown"))
data_omics_plus = identifyPR(data_omics_plus)

## End(Not run)
## Not run:
setwd(system.file("extdata/Genelists", package = "pwOmics.newupdown"))
data_omics = identifyPWs(data_omics_plus)
```

```
data_omics = identifyTFs(data_omics)
data_omics = identifyRsofTFs(data_omics,
noTFs_inPW = 1, order_neighbors = 10)
getUS_regulators(data_omics)

## End(Not run)
```

`getUS_TFs` *Get upstream TFs.*

Description

Get upstream TFs.

Usage

```
getUS_TFs(data_omics)
```

Arguments

`data_omics`	OmicsData object.

Value

list of length = number of gene/transcript time points, each element containing a data frame with upstream transcription factors.

Examples

```
## Not run:
data(OmicsExampleData)
data_omics = readOmics(tp_prots = c(0.25, 1, 4, 8, 13, 18, 24),
tp_genes = c(1, 4, 8, 13, 18, 24), OmicsExampleData,
PWdatabase = c("biocarta", "kegg", "nci", "reactome"),
TFtargetdatabase = c("userspec"))
data_omics = readPhosphodata(data_omics,
phosphoreg = system.file("extdata", "phospho_reg_table.txt",
package = "pwOmics.newupdown"))
data_omics = readTFdata(data_omics,
TF_target_path = system.file("extdata", "TF_targets.txt",
package = "pwOmics.newupdown"))
data_omics_plus = readPWdata(data_omics,
loadgenelists = system.file("extdata/Genelists", package = "pwOmics.newupdown"))
data_omics_plus = identifyPR(data_omics_plus)

## End(Not run)
## Not run:
setwd(system.file("extdata/Genelists", package = "pwOmics.newupdown"))
data_omics = identifyPWs(data_omics_plus)
data_omics = identifyTFs(data_omics)
data_omics = identifyRsofTFs(data_omics,
noTFs_inPW = 1, order_neighbors = 10)
getUS_TFs(data_omics)

## End(Not run)
```

get_matching_transcripts *Summarize table of matching downstream transcripts.*

Description

This function returns the summarized table of matching transcripts information based on the result list from the findSignalingAxes function.

Usage

```
get_matching_transcripts(data_omics, axis)
```

Arguments

data_omics	OmicsData object.
axis	list output of findSignalingAxes.

Value

dataframe containing the target genes of the axis matching the transcript data in at least one of the time points in the first column, the direction of their regulation inferred from the phosphoproteome data in the second column, the transcript regulation in the next columns (per time point) and the summarized matched information in the following columns (per time point).

Examples

```
## Not run:
data(OmicsExampleData)
data_omics = readOmics(tp_prots = c(0.25, 1, 4, 8, 13, 18, 24),
tp_genes = c(1, 4, 8, 13, 18, 24), OmicsExampleData,
PWdatabase = c("biocarta", "kegg", "nci", "reactome"),
TFtargetdatabase = c("userspec"))
data_omics = readPhosphodata(data_omics,
phosphoreg = system.file("extdata", "phospho_reg_table.txt",
package = "pwOmics.newupdown"))
data_omics = readTFdata(data_omics,
TF_target_path = system.file("extdata", "TF_targets.txt",
package = "pwOmics.newupdown"))
data_omics_plus = readPWdata(data_omics,
loadgenelists = system.file("extdata/Genelists", package = "pwOmics.newupdown"))

## End(Not run)
## Not run:
data_omics_plus = identifyPR(data_omics_plus)
setwd(system.file("extdata/Genelists", package = "pwOmics.newupdown"))
data_omics = identifyPWs(data_omics_plus)
data_omics = identifyTFs(data_omics)
data_omics = identifyPWTFTGs(data_omics)
data_omics = identifyRsofTFs(data_omics, noTFs_inPW = 1, order_neighbors = 10)
SYK_axis = findSignalingAxes(data_omics,phosphoprot = "SYK", tpDS = 2)
SYK_transcripts = get_matching_transcripts(data_omics, SYK_axis)

## End(Not run)
```

identifyPR *Identify phosphorylation regulation influence downstream*

Description

This function identifies the downstream regulation influence of phosphoprotein regulation for further downstream analysis steps.

Usage

```
identifyPR(data_omics_plus)
```

Arguments

data_omics_plus	output list of readPWdata function; first element contains an OmicsData object, secons element the genelist data corresponding to the selected pathway database.

Value

OmicsData object: list of 4 elements (OmicsD, PathwayD, TFtargetsD, Status); OmicsD containing omics data set + results (after analysis); PathwayD containing selected pathway databases + biopax model; TFtargetsD containing selected TF target gene databases + TF target gene data.

Examples

```
## Not run:
data(OmicsExampleData)
data_omics = readOmics(tp_prots = c(0.25, 1, 4, 8, 13, 18, 24),
tp_genes = c(1, 4, 8, 13, 18, 24), OmicsExampleData,
PWdatabase = c("biocarta", "kegg", "nci", "reactome"),
TFtargetdatabase = c("userspec"))
data_omics = readPhosphodata(data_omics,
phosphoreg = system.file("extdata", "phospho_reg_table.txt",
package = "pwOmics.newupdown"))
data_omics = readTFdata(data_omics,
TF_target_path = system.file("extdata", "TF_targets.txt",
package = "pwOmics.newupdown"))
data_omics_plus = readPWdata(data_omics,
loadgenelists = system.file("extdata/Genelists", package = "pwOmics.newupdown"))
data_omics_plus = identifyPR(data_omics_plus)

## End(Not run)
```

identifyPWs *Identify pathway IDs and pathway names of differentially abundant proteins*

Description

This function identifies the pathways of the differentially abundant phosphoproteins dependent on the chosen database. Requires rBiopaxParser package. Takes a lot of time for a high number of proteins and/or if all databases are chosen. First, chosen databases are loaded, then new internal pathway IDs are generated. Afterwards the genelists of the different databases are loaded or generated, depending on the loadgenelists option. After pathway identification for the reference time point, also pathway identification for different time points is performed. Pathway ID mapping takes some time, especially for such big databases as reactome, so use savegenelists and loadgenelists for easier and faster usage...

Usage

```
identifyPWs(data_omics_plus)
```

Arguments

data_omics_plus
: output list of readPWdata function; first element contains an OmicsData object, secons element the genelist data corresponding to the selected pathway database.

Value

OmicsData object: list of 4 elements (OmicsD, PathwayD, TFtargetsD, Status); OmicsD containing omics data set + results (after analysis); PathwayD containing selected pathway databases + biopax model; TFtargetsD containing selected TF target gene databases + TF target gene data.

Examples

```
## Not run:
data(OmicsExampleData)
data_omics = readOmics(tp_prots = c(0.25, 1, 4, 8, 13, 18, 24),
tp_genes = c(1, 4, 8, 13, 18, 24), OmicsExampleData,
PWdatabase = c("biocarta", "kegg", "nci", "reactome"),
TFtargetdatabase = c("userspec"))
data_omics = readPhosphodata(data_omics,
phosphoreg = system.file("extdata", "phospho_reg_table.txt",
package = "pwOmics.newupdown"))
data_omics = readTFdata(data_omics,
TF_target_path = system.file("extdata", "TF_targets.txt",
package = "pwOmics.newupdown"))
data_omics_plus = readPWdata(data_omics,
loadgenelists = system.file("extdata/Genelists", package = "pwOmics.newupdown"))

## End(Not run)
## Not run:
data_omics_plus = identifyPR(data_omics_plus)
setwd(system.file("extdata/Genelists", package = "pwOmics.newupdown"))
data_omics = identifyPWs(data_omics_plus)
```

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

`identifyPWTFTGs` *Identify TFs in pathways and their target genes - downstream analysis.*

Description

This function identifies the transcription factors being part of the pathways of downstream analysis. Subsequently it finds the target genes of these transcription factors from the selected TF-target gene database.

Usage

```
identifyPWTFTGs(data_omics, updown = FALSE)
```

Arguments

`data_omics`	OmicsData object.
`updown`	boolean value; TRUE in case up- and downregulation should be checked individually for intersection; FALSE = default, if only deregulation should be checked for.

Value

OmicsData object: list of 4 elements (OmicsD, PathwayD, TFtargetsD, Status); OmicsD containing omics data set + results (after analysis); PathwayD containing selected pathway databases + biopax model; TFtargetsD containing selected TF target gene databases + TF target gene data.

Examples

```
## Not run:
data(OmicsExampleData)
data_omics = readOmics(tp_prots = c(0.25, 1, 4, 8, 13, 18, 24),
tp_genes = c(1, 4, 8, 13, 18, 24), OmicsExampleData,
PWdatabase = c("biocarta", "kegg", "nci", "reactome"),
TFtargetdatabase = c("userspec"))
data_omics = readPhosphodata(data_omics,
phosphoreg = system.file("extdata", "phospho_reg_table.txt",
package = "pwOmics.newupdown"))
data_omics = readTFdata(data_omics,
TF_target_path = system.file("extdata", "TF_targets.txt",
package = "pwOmics.newupdown"))
data_omics_plus = readPWdata(data_omics,
loadgenelists = system.file("extdata/Genelists", package = "pwOmics.newupdown"))
data_omics_plus = identifyPR(data_omics_plus)
setwd(system.file("extdata/Genelists", package = "pwOmics.newupdown"))

## End(Not run)
## Not run:
data_omics = identifyPWs(data_omics_plus)
data_omics = identifyTFs(data_omics)
data_omics = identifyPWTFTGs(data_omics)

## End(Not run)
```

identifyRsofTFs *Identify regulators of transcription factors - upstream analysis.*

Description

This function identifies the regulators upstream of the identified transcription factors in upstream analysis. Converting the pathway information to a regulatory graph needs some time... Warnings regarding the skipping of edges in building the regulatory graph can be ignored.

Usage

```
identifyRsofTFs(data_omics, noTFs_inPW = 2, order_neighbors = 6)
```

Arguments

data_omics	OmicsData object.
noTFs_inPW	integer; only regulators in upstream pathways with more than this number of TFs are identified.
order_neighbors	integer specifiying the order of the neighborhood: order 1 is TF plus its immediate neighbors.

Value

OmicsData object: list of 4 elements (OmicsD, PathwayD, TFtargetsD, Status); OmicsD containing omics data set + results (after analysis); PathwayD containing selected pathway databases + biopax model; TFtargetsD containing selected TF target gene databases + TF target gene data.

Examples

```
## Not run:
data(OmicsExampleData)
data_omics = readOmics(tp_prots = c(0.25, 1, 4, 8, 13, 18, 24),
tp_genes = c(1, 4, 8, 13, 18, 24), OmicsExampleData,
PWdatabase = c("biocarta", "kegg", "nci", "reactome"),
TFtargetdatabase = c("userspec"))
data_omics = readPhosphodata(data_omics,
phosphoreg = system.file("extdata", "phospho_reg_table.txt",
package = "pwOmics.newupdown"))
data_omics = readTFdata(data_omics,
TF_target_path = system.file("extdata", "TF_targets.txt",
package = "pwOmics.newupdown"))
data_omics_plus = readPWdata(data_omics,
loadgenelists = system.file("extdata/Genelists", package = "pwOmics.newupdown"))

## End(Not run)
## Not run:
data_omics_plus = identifyPR(data_omics_plus)
setwd(system.file("extdata/Genelists", package = "pwOmics.newupdown"))
data_omics = identifyPWs(data_omics_plus)
data_omics = identifyTFs(data_omics)
setwd(system.file("extdata/Genelists", package = "pwOmics"))
data_omics = identifyRsofTFs(data_omics,
```

```
noTFs_inPW = 1, order_neighbors = 10)

## End(Not run)
```

`identifyTFs` *Transcription factor identification.*

Description

This function identifies the upstream transcription factors of the provided gene IDs.

Usage

```
identifyTFs(data_omics)
```

Arguments

`data_omics`	OmicsData object.

Value

OmicsData object: list of 4 elements (OmicsD, PathwayD, TFtargetsD, Status); OmicsD containing omics data set + results (after analysis); PathwayD containing selected pathway databases + biopax model; TFtargetsD containing selected TF target gene databases + TF target gene data.

Examples

```
## Not run:
data(OmicsExampleData)
data_omics = readOmics(tp_prots = c(0.25, 1, 4, 8, 13, 18, 24),
tp_genes = c(1, 4, 8, 13, 18, 24), OmicsExampleData,
PWdatabase = c("biocarta", "kegg", "nci", "reactome"),
TFtargetdatabase = c("userspec"))
data_omics = readPhosphodata(data_omics,
phosphoreg = system.file("extdata", "phospho_reg_table.txt",
package = "pwOmics.newupdown"))
data_omics = readTFdata(data_omics,
TF_target_path = system.file("extdata", "TF_targets.txt",
package = "pwOmics.newupdown"))
data_omics_plus = readPWdata(data_omics,
loadgenelists = system.file("extdata/Genelists", package = "pwOmics.newupdown"))

## End(Not run)
## Not run:
data_omics_plus = identifyPR(data_omics_plus)
setwd(system.file("extdata/Genelists", package = "pwOmics.newupdown"))
data_omics = identifyPWs(data_omics_plus)
data_omics = identifyTFs(data_omics)

## End(Not run)
```

identPWsofTFs *Identification of pathways containing the transcription factors identified in upstream analysis*

Description

Identification of pathways containing the transcription factors identified in upstream analysis

Usage

```
identPWsofTFs(genelists, tps_TFs)
```

Arguments

genelists	data.table as read/loaded by loadGenelist function.
tps_TFs	data.table of upstream transcription factors as returned from identTFs function.

Value

list with first element being a pathway list and second being a pathway dataframe of pathways including the TFs of the specified timepoint.

identRegulators *Identify overlapping upstream regulators of x transcription factors*

Description

Identify overlapping upstream regulators of x transcription factors

Usage

```
identRegulators(pws_morex_TFs, data_omics, order_neighbors, noTFs_inPW)
```

Arguments

pws_morex_TFs	list of transcription factors in identified pathways.
data_omics	OmicsData object.
order_neighbors	integer specifiying the order of the neighborhood: order 1 is TF plus its immediate neighbors.
noTFs_inPW	integer; only regulators in upstream pathways with more than this number of TFs are identified.

Value

list of possible proteomic regulators.

identTFs *This function provides a data.table of upstream transcription factors.*

Description

This function provides a data.table of upstream transcription factors.

Usage

```
identTFs(data_omics, glen)
```

Arguments

data_omics	OmicsData object.
glen	numeric value; identifier for current timepoint.

Value

data.table of upstream TFs.

identTFTGsinPWs *Prepare OmicsData object for pathway information.*

Description

This function identifies the TFs in the pathway genes and determines their target genes on basis of the given (chosen) TF-target database(s).

Usage

```
identTFTGsinPWs(data_omics, temp_genelist)
```

Arguments

data_omics	OmicsData object.
temp_genelist	dataframe of unique gene IDs in PWs.

Value

list with first element being a genelist of the pathways and second being a target gene list of TFs.

infoConsensusGraph *Add Consensus Graph information.*

Description

Adds phosphoprotein information based on phosphoprotein data table and redraws Consensus Graph edges

Usage

```
infoConsensusGraph(data_omics, consensusGraph, phosphotab)
```

Arguments

data_omics	OmicsData object.
consensusGraph	result from static analysis: consensus graph generated by staticConsensusNet function.
phosphotab	dataframe with phosphoprotein information annotated in columns 'Gene.names', 'Amino acid', 'Position' (of phosphosite).

Value

graph of igraph class containing complemented consensus graph information

Examples

```
## Not run:
data(OmicsExampleData)
data_omics = readOmics(tp_prots = c(0.25, 1, 4, 8, 13, 18, 24),
tp_genes = c(1, 4, 8, 13, 18, 24), OmicsExampleData,
PWdatabase = c("biocarta", "kegg", "nci", "reactome"),
TFtargetdatabase = c("userspec"))
data_omics = readPhosphodata(data_omics,
phosphoreg = system.file("extdata", "phospho_reg_table.txt",
package = "pwOmics.newupdown"))
data_omics = readTFdata(data_omics,
TF_target_path = system.file("extdata", "TF_targets.txt",
package = "pwOmics.newupdown"))
data_omics_plus = readPWdata(data_omics,
loadgenelists = system.file("extdata/Genelists", package = "pwOmics.newupdown"))

## End(Not run)
## Not run:
data_omics_plus = identifyPR(data_omics_plus)
setwd(system.file("extdata/Genelists", package = "pwOmics.newupdown"))
data_omics = identifyPWs(data_omics_plus)
data_omics = identifyTFs(data_omics)
data_omics = identifyRsofTFs(data_omics,
noTFs_inPW = 1, order_neighbors = 10)
data_omics = identifyPWTFTGs(data_omics)
statConsNet = staticConsensusNet(data_omics)

## End(Not run)
```

loadGenelists *Loading of genelists*

Description

This function automatically loads the genelists corresponding to the selected pathway databases stored as RData file in the current working directory.

Usage

```
loadGenelists()
```

Value

genelist of specified pathway database.

loadPWs *Load pathway database information.*

Description

This function loads the pathway information from pathway databases. Needed in the identifyPWs function.

Usage

```
loadPWs(pwdatabases, biopax_level)
```

Arguments

pwdatabases	vector indicating with which pathway database the pathways should be determined; possible choices are "biocarta", "kegg", "nci", "reactome".
biopax_level	integer indicating biopax level of pathway database information to be retrieved.

Value

list of biopax model corresponding to specified pathway databases.

OmicsExampleData *Omics example dataset.*

Description

A dataset as input example for readOmics: A list containing two input lists, one for protein data, one for gene data, both including a vector of all measured IDs as first element and a list as second element including for each tp a dataframe with IDs and log foldchanges per timepoint.

Usage

```
data(OmicsExampleData)
```

Format

A list with a 'P' sublist containing protein data and a 'G' sublist containing gene/transcript data. Each of the sublists has a first element with all measured protein/gene IDs and a second element with a list of the length of the number of measured time points. Each of these lists contains a dataframe with a first column of significant protein/gene IDs at that time point and a second column with the corresponding logFCs.

Value

List with 2 sublists.

plotConsDynNet *Plot inferred net based on analysis analysis.*

Description

Dynamic analysis result is plotted to pdf file stored in current working directory.

Usage

```
plotConsDynNet(dynConsensusNet, sig.level, clarify = "TRUE",
  layout = layout.fruchterman.reingold, ...)
```

Arguments

dynConsensusNet	result of dynamic analysis: inferred net generated by consDynamicNet function.
sig.level	significance level for plotting edges in network (between 0 and 1).
clarify	indicating if unconnected nodes should be removed; default = "TRUE".
layout	igraph layout parameter; default is layout.fruchterman.reingold.
...	further plotting/legend parameters.

Value

pdf file in current working directory.

Examples

```
## Not run:
data(OmicsExampleData)
data_omics = readOmics(tp_prots = c(0.25, 1, 4, 8, 13, 18, 24),
tp_genes = c(1, 4, 8, 13, 18, 24), OmicsExampleData,
PWdatabase = c("biocarta", "kegg", "nci", "reactome"),
TFtargetdatabase = c("userspec"))
data_omics = readPhosphodata(data_omics,
phosphoreg = system.file("extdata", "phospho_reg_table.txt",
package = "pwOmics.newupdown"))
data_omics = readTFdata(data_omics,
TF_target_path = system.file("extdata", "TF_targets.txt",
package = "pwOmics.newupdown"))
data_omics_plus = readPWdata(data_omics,
loadgenelists = system.file("extdata/Genelists", package = "pwOmics.newupdown"))

## End(Not run)
## Not run:
data_omics_plus = identifyPR(data_omics_plus)
setwd(system.file("extdata/Genelists", package = "pwOmics.newupdown"))
data_omics = identifyPWs(data_omics_plus)
data_omics = identifyTFs(data_omics)
data_omics = identifyRsofTFs(data_omics,
noTFs_inPW = 1, order_neighbors = 10)
data_omics = identifyPWTFTGs(data_omics)
statConsNet = staticConsensusNet(data_omics)
dynConsNet = consDynamicNet(data_omics, statConsNet)
plotConsDynNet(dynConsNet, sig.level = 0.8)

## End(Not run)
```

`plotConsensusGraph` *Plot consensus graph(s) from static analysis.*

Description

Consensus graph(s) plotted to pdf file stored in current working directory.

Usage

```
plotConsensusGraph(consensusGraphs, data_omics, ...)
```

Arguments

`consensusGraphs`	result from static analysis: consensus graph generated by staticConsensusNet function.
`data_omics`	OmicsData object.
`...`	further plotting/legend parameters.

Value

pdf file in current working directory.

Examples

```
## Not run:
data(OmicsExampleData)
data_omics = readOmics(tp_prots = c(0.25, 1, 4, 8, 13, 18, 24),
tp_genes = c(1, 4, 8, 13, 18, 24), OmicsExampleData,
PWdatabase = c("biocarta", "kegg", "nci", "reactome"),
TFtargetdatabase = c("userspec"))
data_omics = readPhosphodata(data_omics,
phosphoreg = system.file("extdata", "phospho_reg_table.txt",
package = "pwOmics.newupdown"))
data_omics = readTFdata(data_omics,
TF_target_path = system.file("extdata", "TF_targets.txt",
package = "pwOmics.newupdown"))
data_omics_plus = readPWdata(data_omics,
loadgenelists = system.file("extdata/Genelists", package = "pwOmics.newupdown"))

## End(Not run)
## Not run:
data_omics_plus = identifyPR(data_omics_plus)
setwd(system.file("extdata/Genelists", package = "pwOmics.newupdown"))
data_omics = identifyPWs(data_omics_plus)
data_omics = identifyTFs(data_omics)
data_omics = identifyRsofTFs(data_omics,
noTFs_inPW = 1, order_neighbors = 10)
data_omics = identifyPWTFTGs(data_omics)
statConsNet = staticConsensusNet(data_omics)
plot(ConsensusGraph, data_omics)

## End(Not run)
```

---

plotConsensusProfiles *Plot consensus graph profiles of static consensus molecules.*

Description

Consensus graph profiles of static consensus molecules plotted as heatmap to pdf file stored in current working directory.

Usage

```
plotConsensusProfiles(consensusGraphs, data_omics, subsel = TRUE, ...)
```

Arguments

consensusGraphs	result from static analysis: consensus graph generated by staticConsensusNet function.
data_omics	OmicsData object.
subsel	character vector of selected consensus molecules for plotting; if TRUE all consensus molecules are plotted
...	further plotting/legend parameters.

Value

pdf file in current working directory.

Examples

```
## Not run:
data(OmicsExampleData)
data_omics = readOmics(tp_prots = c(0.25, 1, 4, 8, 13, 18, 24),
tp_genes = c(1, 4, 8, 13, 18, 24), OmicsExampleData,
PWdatabase = c("biocarta", "kegg", "nci", "reactome"),
TFtargetdatabase = c("userspec"))
data_omics = readPhosphodata(data_omics,
phosphoreg = system.file("extdata", "phospho_reg_table.txt",
package = "pwOmics.newupdown"))
data_omics = readTFdata(data_omics,
TF_target_path = system.file("extdata", "TF_targets.txt",
package = "pwOmics.newupdown"))
data_omics_plus = readPWdata(data_omics,
loadgenelists = system.file("extdata/Genelists", package = "pwOmics.newupdown"))

## End(Not run)
## Not run:
data_omics_plus = identifyPR(data_omics_plus)
setwd(system.file("extdata/Genelists", package = "pwOmics.newupdown"))
data_omics = identifyPWs(data_omics_plus)
data_omics = identifyTFs(data_omics)
data_omics = identifyRsofTFs(data_omics,
noTFs_inPW = 1, order_neighbors = 10)
data_omics = identifyPWTFTGs(data_omics)
statConsNet = staticConsensusNet(data_omics)
plotConsensusProfiles(statConsNet, data_omics, subsel = TRUE)

## End(Not run)
```

---

plotTimeProfileClusters

Plot time profile clusters of dynamic analysis result.

Description

Plot time profile clusters of dynamic analysis result.

Usage

```
plotTimeProfileClusters(fuzzed_matsplines)
```

Arguments

fuzzed_matsplines	result of dynamic analysis: inferred net generated by consDynamicNet function.
...	further plotting/legend parameters.

Value

pdf file in current working directory.

Examples

```
## Not run:
data(OmicsExampleData)
data_omics = readOmics(tp_prots = c(0.25, 1, 4, 8, 13, 18, 24),
tp_genes = c(1, 4, 8, 13, 18, 24), OmicsExampleData,
PWdatabase = c("biocarta", "kegg", "nci", "reactome"),
TFtargetdatabase = c("userspec"))
data_omics = readPhosphodata(data_omics,
phosphoreg = system.file("extdata", "phospho_reg_table.txt",
package = "pwOmics.newupdown"))
data_omics = readTFdata(data_omics,
TF_target_path = system.file("extdata", "TF_targets.txt",
package = "pwOmics.newupdown"))
data_omics_plus = readPWdata(data_omics,
loadgenelists = system.file("extdata/Genelists", package = "pwOmics.newupdown"))

## End(Not run)
## Not run:
data_omics_plus = identifyPR(data_omics_plus)
setwd(system.file("extdata/Genelists", package = "pwOmics.newupdown"))
data_omics = identifyPWs(data_omics_plus)
data_omics = identifyTFs(data_omics)
data_omics = identifyRsofTFs(data_omics,
noTFs_inPW = 1, order_neighbors = 10)
data_omics = identifyPWTFTGs(data_omics)
statConsNet = staticConsensusNet(data_omics)
consDynNet = consDynamicNet(data_omics, statConsNet)
timeprof = clusterTimeProfiles(consDynNet)
plotTimeProfileClusters(timeprof)

## End(Not run)
```

---

predictFCvals *Prediction of continous data points via smoothing splines.*

Description

Prediction of continous data points via smoothing splines.

Usage

```
predictFCvals(data_omics, nopredpoints, splineslist, title)
```

Arguments

data_omics	OmicsData object.
nopredpoints	integer number; how many timpoints should be predicted?
splineslist	list of protein or gene nodes for which splines were generated: output of getFC-splines function.
title	character vector specifying name of title.

Value

list of splines matrix values and calculated times.

`preparePWinfo` *Prepare OmicsData object for pathway information.*

Description

This function prepares the OmicsData object for the identified pathway information.

Usage

```
preparePWinfo(data_omics, plen)
```

Arguments

`data_omics`	OmicsData object.
`plen`	integer indicating the timepoint currently investigated.

Value

list of OmicsData object, current timepoint and pathways of interest.

`print.OmicsData` *Print an OmicsData object.*

Description

Print an OmicsData object.

Usage

```
## S3 method for class 'OmicsData'
print(x, ...)
```

Arguments

`x`	an OmicsData object to print.
`...`	further arguments to be passed to print.

Value

prints OmicsData object.

PWidentallprots *Identification of pathwayIDs and pathway names for all proteins.*

Description

Identification of pathwayIDs and pathway names for all proteins.

Usage

```
PWidentallprots(data_omics, genelists)
```

Arguments

data_omics	OmicsData object.
genelists	lists of genelists from chosen pathway databases.

Value

OmicsData object with identified pathway IDs for list of all proteins.

PWidentallprotssub *Internal subfunction for all protein pathway identification.*

Description

Internal subfunction for all protein pathway identification.

Usage

```
PWidentallprotssub(data_omics, genelists, genelist_ind, datab)
```

Arguments

data_omics	OmicsData object.
genelists	lists of genelists from chosen pathway databases
genelist_ind	integer specifying pathway database genelist matching; 1 = biocarta, 2 = kegg, 3 = nci, 4 = reactome.
datab	character vector indicating database name for message.

Value

OmicsData object with identified pathways for each protein.

PWidenttps *Identification of pathwayIDs and pathway names for proteins at individual timepoints.*

Description

Take the identified pathways from the list of all proteins and transfer this information for the individual timepoints.

Usage

```
PWidenttps(data_omics)
```

Arguments

data_omics	OmicsData object.

Value

data_omics OmicsData object with all pathways identified for the individual timepoints.

readOmics *Read in omics data.*

Description

This function reads omics data: differentially expressed genes and relatively differentially abundant proteins with corresponding fold changes for each time point.

Usage

```
readOmics(tp_prots, tp_genes, omics, PWdatabase, TFtargetdatabase)
```

Arguments

tp_prots	numeric vector of protein timepoints used in experiment; in case of single time point experiments simply assign an experiment number (e.g. 1).
tp_genes	numeric vector of gene/transcript timepoints used in experiment; in case of single time point experiments simply assign an experiment number (e.g. 1).
omics	list containing protein and gene IDs and fold changes: Input are two lists, one for protein data, one for gene data, both including a vector of all measured IDs as first element and a list as second element including for each tp a dataframe with IDs and foldchanges per timepoint.
PWdatabase	character vector of pathway database names which should be used for pathway identification, possible choices are "biocarta", "kegg", "nci", "reactome".
TFtargetdatabase	character vector of TF target database names which should be used for transcription factor/target gene identification, possible choices are "chea", "pazar", "userspec". In case the user is able to provide an own list of transcription factor target gene matches, he can indicate this via "userspec".

Value

OmicsData object: list of 4 elements (OmicsD, PathwayD, TFtargetsD, Status); OmicsD containing omics data set + results (after analysis); PathwayD containing selected pathway databases + biopax model; TFtargetsD containing selected TF target gene databases + TF target gene data.

Examples

```
## Not run:
data(OmicsExampleData)
data_omics = readOmics(tp_prots = c(0.25, 1, 4, 8, 13, 18, 24),
tp_genes = c(1, 4, 8, 13, 18, 24), OmicsExampleData,
PWdatabase = c("biocarta", "kegg", "nci", "reactome"),
TFtargetdatabase = c("userspec"))

## End(Not run)
```

---

readPhosphodata *Reads in phosphoprotein downstream regulation information.*

Description

This function reads in phosphoprotein downstream regulation information from a txt file.

Usage

```
readPhosphodata(data_omics, phosphoreg)
```

Arguments

data_omics	OmicsData object.
phosphoreg	txt file with 2 columns, first providing gene names, second giving -1/1 annotation whether downstream regulation of phosphoprotein is inhibiting or activating, respectively.

Value

OmicsData object: list of 4 elements (OmicsD, PathwayD, TFtargetsD, Status); OmicsD containing omics data set + results (after analysis); PathwayD containing selected pathway databases + biopax model; TFtargetsD containing selected TF target gene databases + TF target gene data; PhosphoD containing optionally phosphoprotein downstream regulation annotation.

Examples

```
## Not run:
data(OmicsExampleData)
data_omics = readOmics(tp_prots = c(0.25, 1, 4, 8, 13, 18, 24),
tp_genes = c(1, 4, 8, 13, 18, 24), OmicsExampleData,
PWdatabase = c("biocarta", "kegg", "nci", "reactome"),
TFtargetdatabase = c("userspec"))
data_omics = readPhosphodata(data_omics,
phosphoreg = system.file("extdata", "phospho_reg_table.txt",
package = "pwOmics"))

## End(Not run)
```

`readPWdata` *Read in pathway database data needed for pathway identification.*

Description

This function reads pathway data of the chosen database(s) via the AnnotationHub [1] package and rBiopaxParser [2] package. Takes a lot of time for a high number of proteins and/or if all databases are chosen. First, chosen databases are retrieved, then new internal pathway IDs are generated. Afterwards the genelists of the different databases are loaded or generated, depending on the loadgenelists option. Pathway ID mapping takes some time, especially for such big databases as reactome, so the genelists are automatically stored in the current working folder and can be used via loadgenelists in case you use this function again for easier and faster usage... Biopax level of retrieved databases is 2 by default.

Usage

```
readPWdata(data_omics, loadgenelists, biopax_level = 2)
```

Arguments

`data_omics`	OmicsData object.
`loadgenelists`	path of genelist RData files stored previously; all genelists stored in this path are read in and used automatically if path is given; if loadgenelists = FALSE, then genelists from pathway databases have to be generated first.
`biopax_level`	integer indicating biopax level of pathway database information. default level is 2.

Value

list of OmicsData object and genelists for selected pathway databases.

References

1. Morgan M, Carlson M, Tenenbaum D and Arora S. AnnotationHub: Client to access AnnotationHub resources. R package version 1.99.75.

2. Kramer F, Bayerlova M, Klemm F, Bleckmann A and Beissbarth T. rBiopaxParser - an R package to parse, modify and visualize BioPAX data.

Examples

```
## Not run:
data(OmicsExampleData)
data_omics = readOmics(tp_prots = c(0.25, 1, 4, 8, 13, 18, 24),
tp_genes = c(1, 4, 8, 13, 18, 24), OmicsExampleData,
PWdatabase = c("biocarta", "kegg", "nci", "reactome"),
TFtargetdatabase = c("userspec"))
data_omics = readPhosphodata(data_omics,
phosphoreg = system.file("extdata", "phospho_reg_table.txt",
package = "pwOmics.newupdown"))
data_omics = readTFdata(data_omics,
TF_target_path = system.file("extdata", "TF_targets.txt",
package = "pwOmics.newupdown"))
```

```
data_omics_plus = readPWdata(data_omics,
loadgenelists = system.file("extdata/Genelists", package = "pwOmics.newupdown"))

## End(Not run)
```

`readTFdata` *Reads in chosen transcription factor target database information.*

Description

This function reads in transcription factor information given the selected transcription factor target gene database. The information is downloaded via the AnnotationHub package and merged, if necessary.

Usage

```
readTFdata(data_omics, TF_target_path, cell_match = 0,
  TF_filter_threshold = 0)
```

Arguments

`data_omics`	OmicsData object.
`TF_target_path`	character vector indicating path of the txt file of matching transcription factors and target genes; the file should be a txt file with first column transcription factors and second column target gene symbols without a header.
`cell_match`	character indicating the cell line/cells for which the TF target gene data should be extracted from the database; this is only possible for chea database. Available cell-specific data from chea for matching are "Hs578T", "Raji B cells and iDC", "MCF7", "THP-1", "Hela cells", "STHdh", "H3396 breast cancer cells","HL60","HESC","T-ALL", "HPC-7", "ovarian surface epithelium", "HaCaT", "HCT116","U2OS", "Wilms tumor-derived CCG99-9611", "HepG2","HUMAN INTESTINAL CELL LINE CACO-2", "HEK293T","K562", "AK7", "NEUROBLASTOMA","JURKAT","T-47D","LS174T", "MULTIPLE HUMAN CANCER CELL TYPES", "501MEL", "PC3", "CACO-2", "FETAL_BRAIN", "HELA", "U937_AND_SAOS2", "CD4_POS_T", "ERYTHROLEUKEMIA", "RHABDOMYOSARCOMA", "293T", "SW620", "LYMPHOBLASTOID", "VCAP", "SK-N-MC", "CADO-ES1", "MEDULLOBLASTOMA", "M12", "K562_HELA_HEPG2_GM12878", "NT2", "SHEP-21N", "LN229_GBM", "MCF-7", "MELANOMA", "MYOFIBROBLAST", "NTERA2", "MEGAKARYOCYTES", "HMVEC", "ZR75-1", "TREG", "TLL", "A2780", "MONOCYTES", "BEAS2B", "LNCAP PROSTATE CANCER CELL LINES", "MCF10A", "GC-B", "BL", "IMR90", "EOC", "PCA", "PROSTATE_CANCER", "OVCAR3", "MALME-3M", "HFKS", "HEK293", "HELA-AND-SCP4", "CD34+", "IB4-LCL", "MDA-MB-231", "U87", "T47D", "Z138-A519-JVM2", "DLD1", "ATHEROSCLEROTIC-FOAM", "LCL-AND-THP1", "NB4", "PFSK-1 AND SK-N-MC", "EP156T","GBM1-GSC","CD4+", "FIBROSARCOMA", "LGR5+ INTESTINAL STEM CELL","NEUROBLASTOMA BE2-C". If no tissue is given the data from all cells/cell lines are merged.
`TF_filter_threshold`	integer defining a threshold number to filter out those transcription factors having higher numbers of target genes than 'TF_filter_threshold' from the further analysis

Value

OmicsData object - a list containing information about the user data (timepoints, IDs, fold changes) and the selected databases chosen for the analysis.

OmicsData object: list of 4 elements (OmicsD, PathwayD, TFtargetsD, Status); OmicsD containing omics data set + results (after analysis); PathwayD containing selected pathway databases + biopax model; TFtargetsD containing selected TF target gene databases + TF target gene data.

Examples

```
## Not run:
data(OmicsExampleData)
data_omics = readOmics(tp_prots = c(0.25, 1, 4, 8, 13, 18, 24),
tp_genes = c(1, 4, 8, 13, 18, 24), OmicsExampleData,
PWdatabase = c("biocarta", "kegg", "nci", "reactome"),
TFtargetdatabase = c("userspec"))
data_omics = readPhosphodata(data_omics,
phosphoreg = system.file("extdata", "phospho_reg_table.txt",
package = "pwOmics"))
data_omics = readTFdata(data_omics,
TF_target_path = system.file("extdata", "TF_targets.txt",
package = "pwOmics"))

## End(Not run)
```

---

readTFtargets *Read in matching transcription factor target gene pairs.*

Description

In case the user is able to provide a file with transcription factor - target gene matches (e.g. from TRANSFAC database) this function can read in the information. The file needs to be a txt file with first column transcription factors and second column target gene symbols without a header.

Usage

```
readTFtargets(data_omics, TF_target_path)
```

Arguments

data_omics	OmicsData object.
TF_target_path	path of txt file containing the transcription factor target gene information as specified above.

Value

data frame with user-specified TF target gene information.

selectPWsofTFs *Select pathways with more than x TFs*

Description

Select pathways with more than x TFs

Usage

```
selectPWsofTFs(pathway_list, pathway_frame, noTFs_inPW)
```

Arguments

pathway_list	first element of list returned from identPWsofTFs function; contains a list of pathways.
pathway_frame	second element of list returned from identPWsofTFs function; contains a data.frame of pathways.
noTFs_inPW	numeric value specifying number of TFs being at least part of the pathway.

Value

list of pathways with more than x TFs.

staticConsensusNet *Static analysis.*

Description

Identify for each corresponding timepoint of the two datasets the consensus network. Protein intersection of the omics data and TF intersection are linked via SteinerTree algorithm applied on STRING protein-protein interaction database. The Steiner tree algorithm refers to the shortest path heuristic algorithm of [1,2]. Target genes of this consensus network are identified via the chosen TF-target gene database(s). Please note that the consensus graphs can be different as in the Steiner Tree algorithm the start terminal node is picked arbitrarily and there are always several shortest path distances. By default the same time points of both data sets are considered.

Usage

```
staticConsensusNet(data_omics, run_times = 3, updown = FALSE,
  tp_prot = NULL, tp_gene = NULL, phospho = TRUE)
```

Arguments

data_omics	OmicsData object.
run_times	integer specifying number of times to run SP Steiner tree algorithm to find minimal graph, default is 3.
updown	boolean value; TRUE in case up- and downregulation should be checked individually for intersection. Type of checking is defined with parameter 'phospho'.

tp_prot	integer specifying the time point that should be included into the static consensus net for the phosphoprotein data
tp_gene	integer specifying the time point that should be included into the static consensus net for the transcriptome data
phospho	boolean value; TRUE in case up- and downregulation should be checked based on provided downstream phosphoprotein influence from identifyPR function; FALSE in case up- and downregulation should be checked for without phosphoprotein database knowledge. Default is TRUE.

Value

list of igraph objects; length corresponds to number of overlapping time points from upstream and downstream analysis.

References

1. Path heuristic and Original path heuristic, Section 4.1.3 of the book "The Steiner tree Problem", Peter L. Hammer

2. "An approximate solution for the Steiner problem in graphs", H Takahashi, A Matsuyama

Examples

```
## Not run:
data(OmicsExampleData)
data_omics = readOmics(tp_prots = c(0.25, 1, 4, 8, 13, 18, 24),
tp_genes = c(1, 4, 8, 13, 18, 24), OmicsExampleData,
PWdatabase = c("biocarta", "kegg", "nci", "reactome"),
TFtargetdatabase = c("userspec"))
data_omics = readPhosphodata(data_omics,
phosphoreg = system.file("extdata", "phospho_reg_table.txt",
package = "pwOmics.newupdown"))
data_omics = readTFdata(data_omics,
TF_target_path = system.file("extdata", "TF_targets.txt",
package = "pwOmics.newupdown"))
data_omics_plus = readPWdata(data_omics,
loadgenelists = system.file("extdata/Genelists", package = "pwOmics.newupdown"))

## End(Not run)
## Not run:
data_omics_plus = identifyPR(data_omics_plus)
setwd(system.file("extdata/Genelists", package = "pwOmics.newupdown"))
data_omics = identifyPWs(data_omics_plus)
data_omics = identifyTFs(data_omics)
data_omics = identifyRsofTFs(data_omics,
noTFs_inPW = 1, order_neighbors = 10)
data_omics = identifyPWTFTGs(data_omics)
statConsNet = staticConsensusNet(data_omics)

## End(Not run)
```

SteinerTree_cons *Steiner tree algorithm.*

Description

Use this function to get the Steiner tree based on the STRING protein-protein interaction database.

Usage

```
SteinerTree_cons(terminal_nodes, PPI_graph, run_times)
```

Arguments

terminal_nodes	character vector of final nodes used for generation of Steiner tree.
PPI_graph	igraph object; graph should be connected and have undirected edges.
run_times	integer specifying number of times to run SP Steiner tree algorithm to find minimal graph.

Value

igraph object including Steiner tree.

temp_correlations *Plot temporal correlations of phosphoprotein expression levels and affected downstream transcripts.*

Description

This function plots an overview of consensus phosphoprotein expression level correlations with those transcripts that are affected downstream in a newly generated folder in the working directory. The following additional information needs to be provided: Phosphorylation information amino acid, position and multiplicity with the expression levels for each of the time points the correlations should be plotted for. Furthermore data tables for fold changes/ratios of phosphoproteins and transcripts that are part of the data_omics object.

Usage

```
temp_correlations(ConsensusGraph, timepointsprot, timepointstrans,
  foldername = "ProtCons_", trans_sign = "signif_single.csv",
  trans_sign_names, phospho_sign = "mat_phospho.csv", phospho_sign_names)
```

Arguments

ConsensusGraph	result from static analysis: consensus graph generated by staticConsensusNet function.
timepointsprot	numeric vector with measurement time points in phosphoproteome data set, which should be used for correlation plots

timepointstrans
: numeric vector with measurement time points in transcriptome data set, which should be used for correlation plots

foldername
: character vector specifying the name of the folder that will be generated for the temporal correlations

trans_sign
: character vector specifying a tab-delimited file with the transcriptome expression levels for all time points in timepointstrans

trans_sign_names
: character vector specifying column names in the transcriptome file corresponding to the expression levels at different time points.

phospho_sign
: character vector specifying a tab-delimited file with the phosphoproteome information (columns 'Gene.names', 'Amino.acid', 'Position', 'Multiplicity') and expression levels for all time points in timepointstrans

phospho_sign_names
: character vector specifying column names in the phosphoproteome file corresponding to the expression levels at different time points.

...
: further plotting/legend parameters.

Value

pdf file in current working directory.

Examples

```
data(OmicsExampleData)
data_omics = readOmics(tp_prots = c(0.25, 1, 4, 8, 13, 18, 24),
tp_genes = c(1, 4, 8, 13, 18, 24), OmicsExampleData,
PWdatabase = c("biocarta", "kegg", "nci", "reactome"),
TFtargetdatabase = c("userspec"))
data_omics = readPhosphodata(data_omics,
phosphoreg = system.file("extdata", "phospho_reg_table.txt",
package = "pwOmics"))
data_omics = readTFdata(data_omics,
TF_target_path = system.file("extdata", "TF_targets.txt",
package = "pwOmics"))
data_omics_plus = readPWdata(data_omics,
loadgenelists = system.file("extdata/Genelists", package = "pwOmics"))
## Not run:
data_omics_plus = identifyPR(data_omics_plus)
setwd(system.file("extdata/Genelists", package = "pwOmics"))
data_omics = identifyPWs(data_omics_plus)
data_omics = identifyTFs(data_omics)
data_omics = identifyRsofTFs(data_omics,
noTFs_inPW = 1, order_neighbors = 10)
data_omics = identifyPWTFTGs(data_omics)
statConsNet = staticConsensusNet(data_omics)
temp_correlations(statConsNet, timepointsprot = c(1,4,8,13,18,24),
timepointstrans = c(1,4,8,13,18,24), foldername = "ProtCons_",
trans_sign = system.file("extdata", "signif_single.csv", package = "pwOmics")
trans_sign_names = c("FC_1",  "FC_2", "FC_3", "FC_4"),
phospho_sign = system.file("extdata", "mat_phospho.csv", package = "pwOmics")
phospho_sign_names = c("Rat1", "Rat2", "Rat3", "Rat4")) )

## End(Not run)
```

TFidentallgenes *Identification of upstream transcription factors for all genes.*

Description

Identification of upstream transcription factors for all genes.

Usage

```
TFidentallgenes(data_omics)
```

Arguments

data_omics	OmicsData object.

Value

data_omics OmicsData object with upstream TFs identified for all genes.

TFidenttps *Identification of upstream transcription factors.*

Description

Identification of upstream transcription factors for the differentially expressed genes of the different timepoints.

Usage

```
TFidenttps(data_omics)
```

Arguments

data_omics	OmicsData object.

Value

data_omics OmicsData object with upstream TFs of differentially expressed genes for separate timepoints identified.

Index