Package ‘pwOmics’

January 15, 2017

Type       Package
Title      Pathway-based data integration of omics data
Version    1.6.0
Date       2014-12-29
Author     Astrid Wachter <Astrid.Wachter@med.uni-goettingen.de>
Maintainer Astrid Wachter <Astrid.Wachter@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 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.

License    GPL (>= 2)
Depends    R (>= 3.2)
Imports    data.table, rBiopaxParser, igraph, STRINGdb, graphics, gplots, Biobase, BiocGenerics, AnnotationDbi, biomaRt, AnnotationHub, GenomicRanges
Suggests   ebdbNet, longitudinal, Mfuzz
biocViews  SystemsBiology, Transcription, GeneTarget
NeedsCompilation no
RoxygenNote 5.0.1

R topics documented:

  pwIntOmics-package .................................................. 3
  addFeedbackLoops ..................................................... 3
  clusterTimeProfiles .................................................. 4
  consDynamicNet ....................................................... 5
R topics documented:

- `createBiopaxnew` .................................................. 7
- `createIntIDs` ...................................................... 7
- `enrichPWs` .......................................................... 8
- `enrichTFs` ............................................................ 9
- `findXneighborsoverlap` ........................................... 10
- `findXnextneighbors` ............................................... 10
- `genfullConsensusGraph` .......................................... 11
- `genGenelists` ....................................................... 11
- `genGenelistssub` ................................................... 12
- `genIntIDs` .......................................................... 12
- `getAliasfromSTRINGIDs` .......................................... 13
- `getAlias_Ensemble` ............................................... 13
- `getBiopaxModel` ................................................... 14
- `getbipartitegraphInfo` ........................................... 14
- `getConsensusSTRINGIDs` ......................................... 15
- `getDS_PWs` .......................................................... 15
- `getDS_TFs` .......................................................... 16
- `getDS_TGs` .......................................................... 17
- `getFCsplines` ....................................................... 18
- `getGenesIntersection` ............................................ 18
- `getOmicsallGeneIDs` .............................................. 19
- `getOmicsallProteinIDs` .......................................... 20
- `getOmicsDataset` .................................................. 20
- `getOmicsTimepoints` ............................................. 21
- `getProteinIntersection` .......................................... 21
- `getSTRING_graph` .................................................. 22
- `getTFIntersection` ................................................ 23
- `getTPIntersection` ................................................ 24
- `getUS_PWs` .......................................................... 25
- `getUS_regulators` ................................................ 25
- `getUS_TFs` .......................................................... 26
- `identifyPWs` ....................................................... 27
- `identifyPWTFTGs` ................................................ 28
- `identifyRsofTFs` .................................................. 29
- `identifyTFs` ...................................................... 30
- `identPWsofTFs` .................................................... 31
- `identRegulators` .................................................. 31
- `identTFs` .......................................................... 32
- `identTFTGsinPWs` ................................................ 32
- `loadGenelists` ...................................................... 33
- `loadPWs` ............................................................ 33
- `OmicsExampleData` ................................................ 34
- `plotConsDynNet` .................................................... 34
- `plotConsensusGraph` ............................................. 35
- `plotConsensusProfiles` ........................................... 36
- `plotTimeProfileClusters` ........................................ 37
- `predictFCvals` ..................................................... 38
- `preparePWinfo` .................................................... 38
- `print.OmicsData` .................................................. 39
- `PWidentallprots` .................................................. 39
- `PWidentallprotssub` ............................................. 40
- `PWidenttps` ....................................................... 40
**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.

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

Examples

# please run with whole database files (prepared according to vignette)
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("chea", "pazar"))
## Not run:
data_omics = readTFdata(data_omics)
data_omics_plus = readPWdata(data_omics,
                              loadgenelists = FALSE)
data_omics = identifyPWS(data_omics_plus)
data_omics = identifyTFs(data_omics)
data_omics = enrichPWS(data_omics)
data_omics = identifyRsofTFs(data_omics, only_enriched = FALSE,
                           noTFs_inPW = 1, order_neighbors = 10)
data_omics = identifyPWFTGs(data_omics, only_enriched = FALSE)
statConsNet = staticConsensusNet(data_omics)
consDynNet = consDynamicNet(data_omics, statConsNet)
clusterTimeProfiles(consDynNet)
## End(Not run)

---

consDynamicNet

**Dynamic analysis.**

Description

Generates continuous 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

```r
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


Examples

#please run with whole database files (prepared according to vignette)
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("chea", "pazar"))
# Not run:
data_omics = readTFdata(data_omics)
data_omics_plus = readPWdata(data_omics,
                              loadgenelists = FALSE)
data_omics = identifyPWS(data_omics_plus)
data_omics = identifyTFs(data_omics)
data_omics = enrichPWS(data_omics)
data_omics = identifyRosoftFs(data_omics, only_enriched = FALSE,
                              noTFs_inPW = 1, order_neighbors = 10)
data_omics = identifyPWTFTs(data_omics, only_enriched = FALSE)
statConsNet = staticConsensusNet(data_omics)
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 biopaxSdf:

**Usage**

createIntIDs(data_omics, PWinfo)

**Arguments**

- data_omics: OmicsData object.
- PWinfo: pathway database information from chosen pathway databases as from load-PWs.

**Value**

list of internal IDs from specified pathway databases.
Pathway enrichment - downstream analysis.

Description

This function does pathway enrichment for pathways determined via identifyPWs function.

Usage

enrichPWs(data_omics, method = "BH", alpha = 0.05, ...)

Arguments

data_omics OmicsData object.
method correction method for multiple testing correction as specified in p.adjust documentation; default is Benjamini & Hochberg correction.
alpha significance level for pathway enrichment; default is alpha = 0.05.
... further input parameters for multiple comparison adjustment.

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

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"),
                     TFFtargetdatabase = c("chea"))
data_omics = readTFdata(data_omics)
data_omics_plus = readPWdata(data_omics,
                           loadgenelists = "Genelists")

## Not run:
data_omics = identifyPWs(data_omics_plus)
data_omics = identifyTFs(data_omics)
data_omics = enrichPWs(data_omics)

## End(Not run)
enrichTFs

Transcription factor enrichment - upstream analysis.

Description
This function does transcription factor enrichment for transcription factor identified to be upstream of the diff. expressed genes/transcripts in the omics data set imported with readOmics function. In order to use this function first read in the transcription factor target gene information via readTFdata and identify the upstream TFs of the differentially expressed genes/transcripts with the identifyTFs function.

Usage
enrichTFs(data_omics, method = "BH", alpha = 0.05, ...)

Arguments
- **data_omics**: OmicsData object.
- **method**: correction method for multiple testing correction as specified in p.adjust documentation; default is Benjamini & Hochberg correction.
- **alpha**: significance level for transcription factor enrichment; default is alpha = 0.05.
- **...**: further input parameters for multiple comparison adjustment.

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
```r
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"),
TFtargetdatabase = c("chea"))
setwd(system.file("extdata", package = "pwOmics"))
data_omics = readTFdata(data_omics)
data_omics_plus = readPWdata(data_omics,
loadgenelists = "Genelists")
## Not run:
data_omics = identifyPWs(data_omics_plus)
data_omics = identifyTFs(data_omics)
data_omics = enrichTFs(data_omics)
## End(Not run)
```
findxneighboroverlap 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
findxneighboroverlap(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 specifying 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.
**genfullConsensusGraph**

Combine SteinerNet with bipartite graph to get full consensus network.

**Description**

Combine SteinerNet with bipartite graph to get full consensus network.

**Usage**

```r
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 det-Pathways function.

**Usage**

```r
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 det-
Pathways 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 load-
PWs.
- **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, consSTRINGIDs, tps, string_db)

Arguments
- data_omics: OmicsData object.
- ST_net: steiner tree graph generated by SteinerTree_cons function.
- 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 col-names '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**

```r
getBiopaxModel(data_omics)
```

**Arguments**

- `data_omics` OmicsData object.

**Value**

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

**Examples**

```r
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"),
  TFtargetdatabase = c("chea"))
## Not run:
data_omics = readTFdata(data_omics)
data_omics_plus = readPWdata(data_omics, loadgenelists = FALSE)
data_omics = identifyPWs(data_omics_plus)
data_omics = identifyTFs(data_omics)
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**

```r
getbipartitegraphInfo(data_omics, tps)
```

**Arguments**

- `data_omics` OmicsData object.
- `tps` integer specifying current timepoint under consideration.
**getConsensusSTRINGIDs**

*Get consensus graph in STRING IDs.*

**Description**

Get consensus graph in STRING IDs.

**Usage**

```
getConsensusSTRINGIDs(data_omics, tps, string_db)
```

**Arguments**

- `data_omics`: OmicsData object.
- `tps`: integer specifying current timepoint under consideration.
- `string_db`: STRING_db object.

**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, corresponding pathway names and flags if those pathways are enriched (1 = enriched, NA = not enriched).
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

#please run with whole database files (prepared according to vignette)
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("chea", "pazar"))
## Not run:
data_omics = readTFdata(data_omics)
data_omics_plus = readPWdata(data_omics, loadgenelists = FALSE)
data_omics = identifyPWs(data_omics_plus)
data_omics = identifyTFs(data_omics)
data_omics = enrichPWs(data_omics)
setwd(system.file("extdata/Genelists", package = "pwOmics"))
data_omics = identifyPWTFTGs(data_omics, only_enriched = FALSE)
getDS_PWts(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

```r
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

# please run with whole database files (prepared according to vignette)
```r
data_omics = identifyPWs(data_omics_plus)
data_omics = identifyTFs(data_omics)
data_omics = enrichPWs(data_omics)
setwd(system.file("extdata/Genelists", package = "pwOmics"))
data_omics = identifyPWTFTGs(data_omics, only_enriched = FALSE)
getDS_TFs(data_omics)
## End(Not run)
```
getFCsplines

*Get fold change splines.*

**Description**

Calculate the splines used for the dynamic analysis.

**Usage**

```r
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**

```r
getGenesIntersection(data_omics, tp_prot, tp_genes)
```

**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.

**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.
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
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("chea", "pazar"))
## Not run:
data_omics = readTFdata(data_omics)
data_omics_plus = readPWdata(data_omics,
loadgenelists = FALSE)
data_omics = identifyPWS(data_omics_plus)
data_omics = identifyTFs(data_omics)
data_omics = enrichPWS(data_omics)
data_omics = identifyRsoftTFs(data_omics, only_enriched = FALSE,
notFs_inPW = 1, order_neighbors = 10)
data_omics = identifyPWTFTGs(data_omics, only_enriched = FALSE)
getGenesIntersection(data_omics, tp_prot = 4, tp_genes = 4)
## 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**

```r
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"),
                      TFtargetdatabase = c("chea"))
getOmicsAllProteinIDs(data_omics)
```

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.
getOmicsTimepoints

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"),
  TFtargetdatabase = c("chea"))
getOmicsDataset(data_omics)

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

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"),
  TFtargetdatabase = c("chea"))
getOmicsDataset(data_omics)

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.
**getProteinIntersection**

**Usage**

```r
getProteinIntersection(data_omics, tp_prot, tp_genes)
```

**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.

**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**

```r
# please run with whole database files (prepared according to vignette)
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("chea", "pazar"))
## Not run:
data_omics = readTFdata(data_omics)
data_omics_plus = readPWdata(data_omics,
loadgenelists = FALSE)
data_omics = identifyPWs(data_omics_plus)
data_omics = identifyTFs(data_omics)
data_omics = enrichPWs(data_omics)
data_omics = identifyRsofTFs(data_omics, only_enriched = FALSE,
noTFs_inPW = 1, order_neighbors = 10)
data_omics = identifyPWTFTGs(data_omics, only_enriched = FALSE)
getProteinIntersection(data_omics, tp_prot = 4, tp_genes = 4)
## End(Not run)
```

---

**getSTRING_graph**

Generate STRING PPI graph.

**Description**

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

**Usage**

```r
getSTRING_graph(string_db)
```

**Arguments**

- `string_db`: STRING_db object generated by getConsensusSTRINGIDs function.
getTFIntersection

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, only_enriched = FALSE)

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.

only_enriched  Boolean value defining if transcription factors should be identified only for enriched pathways (TRUE); or for all identified pathways (FALSE); default is FALSE.

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

# please run with whole database files (prepared according to vignette)
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("chea", "pazar"))
## Not run:
data_omics = readTFdata(data_omics)
data_omics_plus = readPWdata(data_omics,
                              loadgenelists = FALSE)
data_omics = identifyPWs(data_omics_plus)
data_omics = identifyTFs(data_omics)
data_omics = enrichPWs(data_omics)
data_omics = identifyRsofTFs(data_omics, only_enriched = FALSE,
                             noTFs_inPW = 1, order_neighbors = 10)
data_omics = identifyPWTFTGs(data_omics, only_enriched = FALSE)
getTFIntersection(data_omics, 4,4,only_enriched = 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, only_enriched = FALSE)

Arguments

data_omics  OmicsData object.

only_enriched  Boolean value defining if transcription factors should be identified only for enriched pathways (TRUE); or for all identified pathways (FALSE); default is FALSE.

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

#please run with whole database files (prepared according to vignette)
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("chea", "pazar"))
## Not run:
data_omics = readTFdata(data_omics)
data_omics_plus = readPWdata(data_omics,
                            loadgenelists = FALSE)
data_omics = identifyPWs(data_omics_plus)
data_omics = identifyTFs(data_omics)
data_omics = enrichPWs(data_omics)
data_omics = identifyRsofTFs(data_omics, only_enriched = FALSE,
                            noTFs_inPW = 1, order_neighbors = 10)
data_omics = identifyPWFTGs(data_omics, only_enriched = FALSE)
gettpIntersection(data_omics)

## End(Not run)
getUS_PWs

*Get upstream pathways of identified transcription factors.*

**Description**

Get upstream pathways of identified transcription factors.

**Usage**

```r
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**

```r
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"),
TFtargetdatabase = c("chea"))
## Not run:
data_omics = readTFdata(data_omics)
data_omics_plus = readPWdata(data_omics,
loadgenelists = FALSE)
data_omics = identifyPWs(data_omics_plus)
data_omics = identifyTFs(data_omics)
data_omics = enrichPWs(data_omics)
setwd(system.file("extdata/Genelists", package = "pwOmics"))
data_omics = identifyRsofTFs(data_omics, only_enriched = FALSE,
  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**

```r
getUS_regulators(data_omics)
```
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 and flag if these transcription factors are enriched (1 = enriched, NA = not enriched).
identifyPWs

*Examples*

```r
data(OmicsExampleData)
data_omics = readOmics(tp_prots = c(0.25, 1, 4, 8, 13, 18, 24),
  tp_gens = c(1, 4, 8, 13, 18, 24), OmicsExampleData,
  PWdatabase = c("biocarta"),
  TFtargatdatabase = c("chea"))
## Not run:
data_omics = readTFdata(data_omics)
data_omics_plus = readPWdata(data_omics,
  loadgenelists = FALSE)
data_omics = identifyPWs(data_omics_plus)
data_omics = identifyTFs(data_omics)
data_omics = enrichPWs(data_omics)
setwd(system.file("extdata/Genelists", package = "pwOmics"))
data_omics = identifyRsofTFs(data_omics, only_enriched = FALSE,
  noTFs_inPW = 1, order_neighbors = 10)
getUS_TFs(data_omics)
## End(Not run)
```

**identifyPWs**

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

**Description**

This function identifies the pathways of the differentially abundant proteins 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**

```r
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.
identifyPWTFTGs

Identify TFs in enriched pathways and their target genes - downstream analysis.

Description

This function identifies the transcription factors being part of the enriched 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, only_enriched = TRUE)

Arguments

data_omics OmicsData object.
only_enriched boolean value defining if transcription factors should be identified only for enriched pathways (TRUE); or for all identified pathways (FALSE); default is TRUE.

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

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"),
TFtargetdatabase = c("chea"))

data_omics = readTFdata(data_omics)
data_omics_plus = readPWdata(data_omics,
loadgenelists = "Genelists")

## Not run:
data_omics = identifyPWS(data_omics_plus)

## End(Not run)
identifyRsofTFs

Identify regulators of enriched transcription factors - upstream analysis.

Description

This function identifies the regulators upstream of the enriched/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, only_enriched = TRUE, noTFs_inPW = 2, order_neighbors = 6)

Arguments

data_omics	OmicsData object.

only_enriched	boolean value defining if transcription factors should be identifies only for enriched pathways (TRUE); or for all identified pathways (FALSE); default is TRUE.

noTFs_inPW	integer; only regulators in upstream pathways with more than this number of TFs are identified.

order_neighbors	integer specifying 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

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"),
TFtargetdatabase = c("chea"))
data_omics = readTFdata(data_omics)
Identify transcription factors (TFs) using the `identifyTFs` function. This function identifies the upstream TFs of the provided gene IDs.

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

### Usage
```r
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
```r
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"),
                       TFtargetdatabase = c("chea"))
data_omics = readTFdata(data_omics)
data_omics = identifyTFs(data_omics)
data_omics = identifyPWs(data_omics)
data_omics = enrichPWs(data_omics)
setwd(system.file("extdata/Genelists", package = "pwOomics"))
identifyRsofTFs(data_omics, only_enriched = FALSE, noTFs_inPW = 1,
                 order_neighbors = 10)
```

---

**identifyTFs**

Transcription factor identification.

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

### Usage
```r
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
```r
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"),
                       TFtargetdatabase = c("chea"))
data_omics = readTFdata(data_omics)
data_omics = identifyTFs(data_omics)
data_omics = identifyPWs(data_omics)
data_omics = enrichPWs(data_omics)
setwd(system.file("extdata/Genelists", package = "pwOomics"))
identifyRsofTFs(data_omics, only_enriched = FALSE, noTFs_inPW = 1,
                 order_neighbors = 10)
```

---

**identifyTFs**

Transcription factor identification.

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

### Usage
```r
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
```r
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"),
                       TFtargetdatabase = c("chea"))
data_omics = readTFdata(data_omics)
data_omics = identifyTFs(data_omics)
data_omics = identifyPWs(data_omics)
data_omics = enrichPWs(data_omics)
setwd(system.file("extdata/Genelists", package = "pwOomics"))
identifyRsofTFs(data_omics, only_enriched = FALSE, noTFs_inPW = 1,
```
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 and the flag for enrichment 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 specifying 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 and the flag for enrichment.

**Description**

This function provides a data.table of upstream transcription factors and the flag for enrichment.

**Usage**

`identTFs(data_omics, glen)`

**Arguments**

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

**Value**

data.table of upstream TFs and an enrichment flag.

---

**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 enriched/not enriched PWs.

**Value**

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

```r
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**

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

Arguments
- `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

# please run with whole database files (prepared according to vignette)
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("chea", "pazar"))
## Not run:
data_omics = readTFdata(data_omics)
data_omics_plus = readPWdata(data_omics,
loadgenelists = FALSE)
data_omics = identifyPWs(data_omics_plus)
data_omics = identifyTFs(data_omics)
data_omics = enrichPWs(data_omics)
data_omics = identifyRsofTFs(data_omics, only_enriched = FALSE,
notFs_inPW = 1, order_neighbors = 10)
data_omics = identifyPWTFGs(data_omics, only_enriched = FALSE)
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

# please run with whole database files (prepared according to vignette)
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"),
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

...
进一步的绘图/图例参数。

Value

df file in current working directory.

Examples

# please run with whole database files (prepared according to vignette)
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("chea", "pazar"))

## Not run:
data_omics = readTFdata(data_omics)
data_omics_plus = readPWdata(data_omics,
  loadgenelists = FALSE)
data_omics = identifyPWs(data_omics_plus)
data_omics = identifyTFs(data_omics)
data_omics = enrichPWs(data_omics)
data_omics = identifyRsofTFs(data_omics, only_enriched = FALSE,
  noTFs_inPW = 1, order_neighbors = 10)
data_omics = identifyPWTFTGs(data_omics, only_enriched = FALSE)
statConsNet = staticConsensusNet(data_omics)
plot(statConsNet)

## End(Not run)
loadgenelists = FALSE)
data_omics = identifyPWs(data_omics_plus)
data_omics = identifyTFs(data_omics)
data_omics = enrichPWs(data_omics)
data_omics = identifyRsofTFs(data_omics, only_enriched = FALSE, noTFs_inPW = 1, order_neighbors = 10)
data_omics = identifyPWMFTGs(data_omics, only_enriched = FALSE)
statConsNet = staticConsensusNet(data_omics)
plotConsensusProfiles(statConsNet)

## 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

#please run with whole database files (prepared according to vignette)
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("chea", "pazar"))

## Not run:
data_omics = readTFdata(data_omics)
data_omics_plus = readPWdata(data_omics,
loadgenelists = "Genelists")
data_omics = identifyPWs(data_omics_plus)
data_omics = identifyTFs(data_omics)
data_omics = enrichPWs(data_omics)
data_omics = identifyRsofTFs(data_omics, only_enriched = FALSE, noTFs_inPW = 1, order_neighbors = 10)
data_omics = identifyPWMFTGs(data_omics, only_enriched = FALSE)
statConsNet = staticConsensusNet(data_omics)
consDynNet = consDynamicNet(data_omics, statConsNet)
timeprof = clusterTimeProfiles(consDynNet)
plotTimeProfileClusters(timeprof)
## End(Not run)

### predictFCvals

**Prediction of continuous data points via smoothing splines.**

**Description**

Prediction of continuous data points via smoothing splines.

**Usage**

predictFCvals(data_omics, nopredpoints, splineslist, title)

**Arguments**

- **data_omics**: OmicsData object.
- **nopredpoints**: integer number; how many timepoints 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

```r
## 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

```r
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**

```r
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**

```r
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 fold changes 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

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"),
                  TFtargetdatabase = c("chea"))
## Not run:
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("chea", "pazar", "userspec"))
## End(Not run)
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**

```r
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**


**Examples**

```r
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),
                        PWdatabase = c("biocarta"),
                        TFtargetdatabase = c("chea"))

data_omics = readTFdata(data_omics)
data_omics_plus = readPWdata(data_omics, loadgenelists = "Genelists")
data_omics_plus[[2]][[1]]
```
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

```r
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.
- `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

```r
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"),
                     TFtargetdatabase = c("pazar"))
data_omics = readTFdata(data_omics)
data_omics[[3]]
```

---

**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**

```r
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**

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

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.

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.

Usage

staticConsensusNet(data_omics, run_times = 3)

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.

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
Examples

# please run with whole database files (prepared according to vignette)
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("chea", "pazar"))
## Not run:
data_omics = readTFdata(data_omics)
data_omics_plus = readPWdata(data_omics,
loadgenelists = FALSE)
data_omics = identifyPWs(data_omics_plus)
data_omics = identifyTFs(data_omics)
data_omics = enrichPWs(data_omics)
data_omics = identifyRsofTFs(data_omics, only_enriched = FALSE,
noTFs_inPW = 1, order_neighbors = 10)
data_omics = identifyPWTFTGs(data_omics, only_enriched = FALSE)
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.
**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

*Topic** datasets
OmicsExampleData, 34

*Topic** manip
addFeedbackLoops, 3
clusterTimeProfiles, 4
consDynamicNet, 5
createBiopaxnew, 7
createIntIDs, 7
enrichPWs, 8
enrichTFs, 9
findxneighborsoverlap, 10
findxnextneighbors, 10
genFullConsensusGraph, 11
genGenelists, 11
genGenelistssub, 12
genIntIDs, 12
getAlias_Ensemble, 13
getAliasFromSTRINGIDs, 13
getBiopaxModel, 14
getBipartitegraphInfo, 14
genConsensusSTRINGIDs, 15
genPw_Pws, 15
genPw_Tfs, 16
genPw_TGs, 17
genFCsplines, 18
getGenesIntersection, 18
getOmicsallGeneIDs, 19
getOmicsallProteinIDs, 20
getOmicsDataset, 20
getOmicsTimepoints, 21
getProteinIntersection, 21
getSTRING_graph, 22
getTFlntersection, 23
getTPlntersection, 24
getUS_Pws, 25
getUS_regulators, 25
getUS_Tfs, 26
identifyPws, 27
identifyPWFTFTGs, 28
identifyRsoftTFs, 29
identifyTFs, 30
identifyPwsofTFs, 31
identRegulators, 31
identTFs, 32
identTFTGsinPws, 32
loadGenelists, 33
loadPws, 33
plotConsDynNet, 34
plotConsensusGraph, 35
plotConsensusProfiles, 36
plotTimeProfileClusters, 37
predictFCvals, 38
preparePWinfo, 38
print.OmicsData, 39
PWidentallprots, 39
PWidentallprotssub, 40
PWidenttpts, 40
readOmics, 41
readPwdata, 42
readTFdata, 43
readTFtargets, 44
selectPwsofTFs, 44
staticConsensusNet, 45
SteinerTree_cons, 46
TFidentallgenes, 47
TFidenttpts, 47
*Topic** package
pwIntOmics-package, 3

addFeedbackLoops, 3
clusterTimeProfiles, 4
consDynamicNet, 5
createBiopaxnew, 7
createIntIDs, 7
enrichPWs, 8
enrichTFs, 9
findxneighborsoverlap, 10
findxnextneighbors, 10
genFullConsensusGraph, 11
genGenelists, 11
genGenelistssub, 12
genIntIDs, 12
getAlias_Ensemble, 13
getAliasFromSTRINGIDs, 13

48
INDEX

getBiopaxModel, 14
getBipartiteGraphInfo, 14
getConsensusSTRINGIDs, 15
getAddress, 15
getDS_trees, 16
getDS_TGs, 17
getFCsplines, 18
getGenesIntersection, 18
getGenesAllGeneIDs, 19
getGenesAllProteinIDs, 20
getGeneDataset, 20
getGeneTimepoints, 21
getProteinIntersection, 21
getSTRING_graph, 22
getTBIintersection, 23
getTBIintersection, 23
getUS_trees, 25
getUS_regulators, 25
getUS_TFs, 26
identifyPWs, 27
identifyPWCTFs, 28
identifyRsofTFs, 29
identifyTFs, 30
identifyPWsofTFs, 31
identifyRegulators, 31
identifyTFs, 32
identifyTFTGsinPWs, 32
loadGenerelists, 33
loadPWs, 33
OomicsExampleData, 34
plotConsDynNet, 34
plotConsensusGraph, 35
plotConsensusProfiles, 36
plotTimeProfileClusters, 37
predictFCvals, 38
preparePWinfo, 38
print.OmicsData, 39
PWidentallprots, 39
PWidentallprotssub, 40
PWidenttps, 40
pwIntOomics (pwIntOomics-package), 3
pwIntOomics-package, 3
readOomics, 41
readPWdata, 42
readTFdata, 43
readTFTargets, 44
selectPWsofTFs, 44
staticConsensusNet, 45
SteinerTree.cons, 46
TFidentifygenes, 47
TFidentifytps, 47