

Package: bppg (via r-universe)

October 11, 2026

Title Bipartite peptide-protein graphs

Version 0.99.1

Date 2026-08-12

Description This package contains functionality to create and characterize bipartite graphs that model the relationship between peptides and proteins in bottom-up proteomics. With these graphs, protein ratios (fold change between two sample groups) are calculated from the respective measured peptide ratios. The main aim is to make use of quantitative information contained in shared peptides and making it possible to quantify proteins without shared peptides. Further, the package offers general analysis of peptide and protein relationships using bipartite graphs and visualization for these graphs.

License BSD_3_clause + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

biocViews Proteomics, GraphAndNetwork, MassSpectrometry, MultipleComparison, Visualization

Imports checkmate, ggplot2, ggraph, limma, openxlsx, pbapply, robustbase, Rsolnp, stringr, SummarizedExperiment, vsn

Suggests covr, rmarkdown, testthat (>= 3.0.0), tidySummarizedExperiment, seqinr, BiocStyle, knitr, pak, RefManageR, sessioninfo, batchtools

Config/testthat/edition 3

VignetteBuilder knitr

Depends igraph (>= 2.0.0)

Config/roxygen2/version 8.1.0

URL <https://github.com/mpc-bioinformatics/bppg>

BugReports <https://github.com/mpc-bioinformatics/bppg/issues>

Config/pak/sysreqs

libfontconfig1-dev libfreetype6-dev libglpk-dev libicu-dev libxml2-dev zlib1g-dev

Repository <https://biocstaging.r-universe.dev>

Date/Publication 2026-08-12 12:14:48 UTC

RemoteUrl <https://github.com/BiocStaging/bppg>

RemoteRef HEAD

RemoteSha e3f64d0f75108a1114b1ab00d2e1b34c0b657f81

Contents

.addUniquenessAttributes	3
.analyseResultSingleProt	4
.calcConstraints	4
.calcObjectiveFunction	5
.calcResultGridpoint	6
.cleave	7
.contractGraph	8
.digest2	9
.directBipartiteGraph	10
.errorEquation	10
.extractIntensities	11
.foldChange	12
.geomMean	12
.getContractMapping	13
.initializeCi	14
.initializeRi	15
.isomorphicBipartite	15
.minimizeSquaredError	16
.setNodeLabels	17
.trackingDataFrame	18
aggregateReplicates	19
automatedAnalysisIteratedCi	20
calculatePeptideRatios	21
combineComparisons	22
digestFASTA	22
exportSE	24
generateGraphsFromEdgelist	25
generateGraphsFromFASTA	26
generateGraphsFromQuantData	27
generateQuantGraphs	29
iterateOverCi	30
normalizePeptideIntensities	32
plotBipartiteGraph	33
proteinElimination	34
readFpPeptideTable	36
readMqPeptideTable	37

Index

39

`.addUniquenessAttributes`

Adds vertex attributes with uniqueness of peptides and number of unique peptides for proteins.

Description

Adds vertex attributes with uniqueness of peptides and number of unique peptides for proteins.

Usage

```
.addUniquenessAttributes(G)
```

Arguments

<code>G</code>	igraph graph Bipartite peptide protein graph.
----------------	---

Value

A graph with 2 additional vertex attributes, uniqueness and nr_unique_peptides

See Also

[generateGraphsFromFASTA\(\)](#), [generateQuantGraphs\(\)](#)

Examples

```
file <- system.file("extdata", "quantGraphs.rds", package = "bppg")
graphs <- readRDS(file)

res <- bppg:::.addUniquenessAttributes(graphs[[1]][[2]])

# get uniqueness of all nodes (NA means it is a protein node, not a peptide)
igraph::V(res)$uniqueness

# get number of unique and shared peptides per protein node:
igraph::V(res)$nr_unique_peptides
igraph::V(res)$nr_shared_peptides
```

```
.analyseResultSingleProt
```

Analyse results for a single protein

Description

Analyse results for a single protein

Usage

```
.analyseResultSingleProt(
  protNr,
  resProt,
  error_tol = 1e-10,
  ratioLog_tol = 1e-06
)
```

Arguments

protNr	integer(1) Index of the protein to be analysed.
resProt	data.frame The data.frame resulting from the iterateOverCi() function, filtered for a specific protein.
error_tol	numeric(1) tolerance for the error term.
ratioLog_tol	numeric(1) tolerance for the log protein ratios.

Value

Vector with minimal error, estimate for Ri (single value or min/max), estimate for Ci (single value or min/max).

```
.calcConstraints
```

Calculate equality and inequality constraints for the optimization step

Description

Calculate equality and inequality constraints for the optimization step

Usage

```
.calcConstraints(fixedCi, m)
```

Arguments

<code>fixedCi</code>	numeric vector The fixed protein weights, variable weights set as NA. Sum of fixed weights must not exceed 1. If NULL, all Ci will be considered as variable. This argument is needed to fix Ci on a grid point in the <code>iterated_Ci</code> function.
<code>m</code>	integer(1) number of proteins in the respective graph

Details

`x` is a vector containing first the RiLog values and then the Ci values (length $2*m$ if no Ci are fixed).

For the equality constraint (`eqfun` with corresponding bound `eqB`), the difference of the sum of the Ci values and 1 is calculated. The bound is set to 0, i.e. forcing the sum of the Ci to be 1. In case of fixed Ci values, the sum of the fixed and the free Ci values is considered.

The lower bound (LB) for the RiLog values is set to -Inf (no constraint) and to 0 for the Ci.

Value

list containing the following elements (see also [solnp](#)):

<code>eqfun</code>	function for calculating equality constraints
<code>eqB</code>	vector of equality bounds for the variables
<code>LB</code>	lower bound for the variables

Examples

```
# without fixed Ci
bppg:::.calcConstraints(fixedCi = NULL, m = 3)

# with fixed Ci
bppg:::.calcConstraints(fixedCi = c(NA, 0.8, NA), m = 3)
```

`.calcObjectiveFunction`*Calculate objective function for the optimization step*

Description

Calculate objective function for the optimization step

Usage

```
.calcObjectiveFunction(fixedCi, M, rjLog)
```

Arguments

<code>fixedCi</code>	numeric vector The fixed protein weights, variable weights set as NA. Sum of fixed weights must not exceed 1. If NULL, all C _i s will be considered as variable. This argument is needed to fix C _i on a grid point in the <code>iterated_Ci</code> function.
<code>M</code>	matrix Biadjacency matrix of the graph.
<code>rjLog</code>	numeric vector Contains the measured peptide ratios (log2-scale).

Details

The objective function is the sum of squared error terms calculated in [.errorEquation](#).

`x` is a vector containing first the `RiLog` values and then the `Ci` values (length $2*m$ if no `Ci` are fixed). If there are fixed `Ci` and `m2` non-fixed ones, `x` has the length $m + m2$.

Value

objective function that will be minimized

Examples

```
M <- matrix(c(1, 1, 1, 0, 1, 1), ncol = 2, byrow = FALSE)
rjLog <- log2(c(0.6, 1.2, 1.5))

fun <- bppg:::.calcObjectiveFunction(fixedCi = NULL, M = M, rjLog = rjLog)
```

`.calcResultGridpoint` *Calculate optimal solution for a single gridpoint*

Description

Calculate optimal solution for a single gridpoint

Usage

```
.calcResultGridpoint(j, gridpoint, cnames, G, n, verbose, ...)
```

Arguments

<code>j</code>	integer(1) index of protein
<code>gridpoint</code>	numeric(1) C _i value to fix for this protein

<code>cnames</code>	character column names for result dataframe
<code>G</code>	igraph object bipartite peptide-protein graph
<code>n</code>	integer(1) number of proteins in the graph
<code>verbose</code>	logical If TRUE, print additional information (see solnp function).
<code>...</code>	additional arguments for .minimizeSquaredError() , e.g. <code>verbose</code> , <code>control</code>

Value

Vector with the following elements:

<code>protein</code>	index of protein (j)
<code>grid</code>	gridpoint
<code>res_Ri_Ci</code>	multiple values: estimated RiLog and Ci values
<code>error</code>	error term

<code>.cleave</code>	<i>Cleavage of a single protein sequence.</i>
----------------------	---

Description

Cleavage of a single protein sequence.

Usage

```
.cleave(sequence, start, stop, miss)
```

Arguments

<code>sequence</code>	character The protein sequence.
<code>start</code>	integer vector Index of where peptides start
<code>stop</code>	integer vector Index of where peptides stop
<code>miss</code>	integer which order of missed cleavages, e.g. 0 = no missed cleavage

Value

A dataframe with information (inkl. peptide sequence and start value) of the peptides.

See Also

[.digest2\(\)](#), [digestFASTA\(\)](#)

<code>.contractGraph</code>	<i>Contracting of peptide and protein nodes.</i>
-----------------------------	--

Description

Contracting of peptide and protein nodes.

Usage

```
.contractGraph(G, vMapping, collProtNodes, collPeptNodes)
```

Arguments

<code>G</code>	igraph graph Bipartite peptide protein graph.
<code>vMapping</code>	list A list with two dataframes from .getContractMapping() .
<code>collProtNodes</code>	logical If TRUE, the protein nodes will be collapsed.
<code>collPeptNodes</code>	logical If TRUE, the peptide nodes will be collapsed.

Value

An edgelist with collapsed protein and/or peptide nodes.

See Also

For graph collapsing: [.getContractMapping\(\)](#)
[generateGraphsFromEdgelist\(\)](#)

Examples

```
library(seqinr)
file <- system.file("extdata", "uniprot_proteome_Scerevisiae_filtered.fasta",
  package = "bppg")
fasta <- seqinr::read.fasta(file = file, seqtype = "AA", as.string = TRUE)
edgelist <- digestFASTA(fasta)
vMapping <- bppg::getContractMapping(edgelist)
G <- igraph::graph_from_edgelist(as.matrix(edgelist), directed = FALSE)
igraph::V(G)[igraph::V(G)$name %in% edgelist[, 1]]$type <- TRUE
igraph::V(G)[igraph::V(G)$name %in% edgelist[, 2]]$type <- FALSE
res <- bppg::contractGraph(G, vMapping)
```

`.digest2`*Digestion of a single protein sequence.*

Description

Digestion of a single protein sequence.

Usage

```
.digest2(  
  sequence,  
  enzyme = "trypsin",  
  missed = 0,  
  warn = TRUE,  
  remove_initial_M = TRUE  
)
```

Arguments

<code>sequence</code>	character The protein sequence.
<code>enzyme</code>	character The enzyme used in digestion e.g. "trypsin" (does not cut before proline).
<code>missed</code>	character The maximal number of missed cleavages.
<code>warn</code>	logical If TRUE, warnings will be printed e.g. if a protein has no cleavage site.
<code>remove_initial_M</code>	logical If TRUE, the initial methionine will be removed from the peptides.

Value

A vector of peptides.

See Also

[digestFASTA\(\)](#)

Examples

```
library(seqinr)  
file <- system.file("extdata", "uniprot_proteome_Scerevisiae_filtered.fasta",  
  package = "bppg")  
fasta <- seqinr::read.fasta(file = file, seqtype = "AA", as.string = TRUE)  
  
# digest first protein sequence in the fasta file:  
digested_proteins <- bppg::digest2(fasta[[1]])
```

`.directBipartiteGraph` *Transform a bipartite graph into a directed graph.*

Description

Transform a bipartite graph into a directed graph.

Usage

```
.directBipartiteGraph(bip_graph, from_type = FALSE)
```

Arguments

<code>bip_graph</code>	graph (igraph) A bipartite graph.
<code>from_type</code>	logical If TRUE, the edges will go out from the vertices with the type TRUE from the bipartite graph.

Value

A bipartite graph that is know directed.

`.errorEquation` *Function to set up the error equations for the optimization problem*

Description

Function to set up the error equations for the optimization problem

Usage

```
.errorEquation(RiLog, Ci, M, rjLog)
```

Arguments

<code>RiLog</code>	numeric vector Contains the (estimated) protein ratios (log2-scale).
<code>Ci</code>	numeric vector Contains the protein weights (estimated, sum up to 1)
<code>M</code>	matrix The biadjacency matrix of the corresponding graphs.
<code>rjLog</code>	numeric vector Contains the measured peptide ratios (log2-scale).

Value

list containing the following elements:

<code>res_Mat</code>	matrix containing the estimated peptide ratios using the given <code>Ri</code> and <code>Ci</code>
<code>res_equ</code>	vector of error terms for each peptide
<code>res_squ_err</code>	sum of squared error terms
<code>W</code>	internal weight matrix

Examples

```
Ri <- c(0.5, 1.3)
Ci <- c(0.3, 0.7)
M <- matrix(c(1, 0, 1, 1), nrow = 2, byrow = TRUE)
rj <- c(0.6, 1.2)
bppg:::errorEquation(Ri, Ci, M, rj)
```

<code>.extractIntensities</code>	<i>Helper function that extracts the intensity columns and columns of interest from a given dataframe.</i>
----------------------------------	--

Description

Helper function that extracts the intensity columns and columns of interest from a given dataframe.

Usage

```
.extractIntensities(D, col_pattern, rename_columns)
```

Arguments

<code>D</code>	data.frame Data frame of peptides.txt from MaxQuant
<code>col_pattern</code>	character Pattern to recognize intensity columns by. Should be "Intensity." or "LFQ.intensity."
<code>rename_columns</code>	logical If TRUE, "Intensity." or "LFQ.intensity." are removed

Value

returns intensity dataframe

<code>.foldChange</code>	<i>Calculate peptide ratios for pairwise comparisons of groups (Y/X).</i>
--------------------------	---

Description

Calculate peptide ratios for pairwise comparisons of groups (Y/X).

Usage

```
.foldChange(D, X, Y, useNA = FALSE)
```

Arguments

D	data.frame The data set.
X	character The column name of group1.
Y	character The column name of group2.
useNA	logical If TRUE, results 0 and Inf are possible, otherwise ratio is NA, if value for X or Y is NA

Value

The fold changes (Y/X).

Examples

```
D <- data.frame(s1 = c(1,4), s2 = c(2,5), s3 = c(3,6))
X <- "s1"
Y <- "s2"
bppg:::foldChange(D,X,Y)
```

<code>.geomMean</code>	<i>Calculate the geometric mean.</i>
------------------------	--------------------------------------

Description

Calculate the geometric mean.

Usage

```
.geomMean(x, useprod = FALSE, na.rm = FALSE)
```

Arguments

<code>x</code>	numeric vector Input data.
<code>useprod</code>	logical If TRUE, $\text{prod}(x)^{(1/n)}$ will be calculated, otherwise $\exp(\text{mean}(\log(x)))$.
<code>na.rm</code>	logical If TRUE, missing values are removed before the calculation. FALSE is default.

Value

The geometric mean of the provided data points.

Examples

```
data <- c(1,6,3.5)
result <- bppg:::geomMean(data, useprod = FALSE)
```

`.getContractMapping` *Create Mapping signature for `igraph::contract` function.*

Description

Create Mapping signature for `igraph::contract` function.

Usage

```
.getContractMapping(edgelist, collProtNodes = TRUE, collPeptNodes = FALSE)
```

Arguments

<code>edgelist</code>	data.frame An edgelist eg. created with digestFASTA() .
<code>collProtNodes</code>	logical If TRUE, the protein nodes will be collapsed.
<code>collPeptNodes</code>	logical If TRUE, the peptide nodes will be collapsed.

Value

A list with two dataframes, one for peptide and one for protein signatures.

See Also

For collapsing graphs:
[generateGraphsFromFASTA\(\)](#), [generateQuantGraphs\(\)](#)

Examples

```
library(seqinr)
file <- system.file("extdata", "uniprot_proteome_Scerevisiae_filtered.fasta",
  package = "bppg")
fasta <- seqinr::read.fasta(file = file, seqtype = "AA", as.string = TRUE)
edgelist <- digestFASTA(fasta)
res <- bppg:::.getContractMapping(edgelist)
```

.initializeCi

Calculate initial values for Ci (protein weights) for optimization

Description

Calculate initial values for Ci (protein weights) for optimization

Usage

```
.initializeCi(fixedCi, m)
```

Arguments

fixedCi	numeric vector The fixed protein weights, variable weights set as NA. Sum of fixed weights must not exceed 1. If NULL, all Cis will be considered as variable. This argument is needed to fix Ci on a grid point in the iterated_Ci function.
m	integer(1) number of proteins in the respective graph

Details

If no Ci are fixed, all Ci are initialized with equal weights (1/m). If at least one Ci is fixed, the remaining weight (1 - sum of fixed Ci) is distributed equally among the non-fixed Ci.

Value

Ci_start: vector with initial start values for Ci for the optimization step

Examples

```
# without fixed Ci -> equal starting weights
bppg:::.initializeCi(fixedCi = NULL, m = 3)

# with fixed Ci -> fixes these and equal weight for the others
bppg:::.initializeCi(fixedCi = c(NA, 0.8, NA), m = 3)
```

<code>.initializeRi</code>	<i>Calculate initial values for Ri (protein ratios) for optimization</i>
----------------------------	--

Description

Calculate initial values for Ri (protein ratios) for optimization

Usage

```
.initializeRi(M, rjLog)
```

Arguments

<code>M</code>	matrix Biadjacency matrix of the graph.
<code>rjLog</code>	numeric Vector of measured peptide ratios (log2-scale).

Details

For each protein, the mean of the peptide ratios (log2-scale) of the peptides belonging to this protein is calculated. If there are unique peptides for this protein, only those are used.

Value

RiLog_start: vector with initial start values for Ri for the optimization step (log2-scale)

Examples

```
M <- matrix(c(1, 1, 1, 0, 1, 1), ncol = 2, byrow = FALSE)
rjLog <- log2(c(0.6, 1.2, 1.5))
bppg:::initializeRi(M, rjLog)
```

<code>.isomorphicBipartite</code>	<i>Enhanced version of the igraph::isomorphic function that also considers the node type in bipartite graphs, e.g. that W- and M-shaped graphs are NOT isomorphic</i>
-----------------------------------	---

Description

Enhanced version of the igraph::isomorphic function that also considers the node type in bipartite graphs, e.g. that W- and M-shaped graphs are NOT isomorphic

Usage

```
.isomorphicBipartite(graph1, graph2, ...)
```

Arguments

graph1	graph (igraph) First graph.
graph2	graph (igraph) Second graph.
...	currently unused

Value

TRUE if graphs are isomorphic, FALSE if not.

See Also

[generateGraphsFromEdgelist\(\)](#)

Examples

```
M1 <- matrix(c(1, 0, 1, 1), nrow = 2, byrow = TRUE)
G1 <- igraph::graph_from_biadjacency_matrix(M1)

M2 <- matrix(c(1, 1, 0, 1), nrow = 2, byrow = TRUE)
G2 <- igraph::graph_from_biadjacency_matrix(M2)

bpgg:::isomorphicBipartite(G1, G2)
```

`.minimizeSquaredError` *Function to set up the optimization problem and minimize the sum of squared error terms*

Description

Function to set up the optimization problem and minimize the sum of squared error terms

Usage

```
.minimizeSquaredError(G, fixedCi = NULL, verbose = FALSE, control = list())
```

Arguments

G	igraph object bipartite peptide-protein graph
fixedCi	numeric vector The fixed protein weights, variable weights set as NA. Sum of fixed weights must not exceed 1. If NULL, all Cis will be considered as variable. This argument is needed to fix Ci on a grid point in the iterated_Ci function.

<code>verbose</code>	logical If TRUE, additional information on each iteration of the optimization is printed (see solnp).
<code>control</code>	list The control parameters for <code>solnp</code> (see solnp).

Value

list containing the following elements:

<code>RiLog</code>	estimated protein ratios (Log2-scale)
<code>Ci</code>	estimated protein weights
<code>RES</code>	final result of .errorEquation , which also contains the final, minimal error term
<code>Tracking</code>	Tracking of <code>Ri</code> , <code>Ci</code> and error term for the different iterations (-1: no optimization)
<code>outer.iter</code>	Number of outer iterations needed for the optimization algorithm to converge or stop (see also solnp)
<code>convergence</code>	Indicates whether the solver has converged (0) or not (1 or 2) (see also solnp)

Examples

```
file <- system.file("extdata", "quantGraphs.rds", package = "bppg")
graphs <- readRDS(file)
G <- graphs$"1_2"[[2]]
bppg:::minimizeSquaredError(G)
```

<code>.setNodeLabels</code>	<i>Set names for plotting with <code>plotBipartiteGraph</code>.</i>
-----------------------------	---

Description

Set names for plotting with `plotBipartiteGraph`.

Usage

```
.setNodeLabels(G, node_labels_peptides, node_labels_proteins, round_digits)
```

Arguments

<code>G</code>	igraph graph object A bipartite peptide-protein graph.
<code>node_labels_peptides</code>	character The type of labels for the peptides. Options are "numbers" or "pep_ratios" or "pep_ratio_aggr".

node_labels_proteins **character**
The type of labels for the proteins. Options are "letters" or "acessions".

round_digits **integer**
The number of digits to round the peptide ratios to.

Value

Graph with updated names

.trackingDataFrame	<i>Funktion to format Tracking information into a dataframe with columnnames.</i>
--------------------	---

Description

Funktion to format Tracking information into a dataframe with columnnames.

Usage

.trackingDataFrame(i, RES, RiLog, Ci)

Arguments

i **numeric**
iterrator, for when this tracking was performed

RES **list**
The result from [.errorEquation](#)

RiLog **numeric vector**
Contains the (estimated) protein ratios (log2-scale).

Ci **numeric vector**
Contains the protein weights (estimated, sum up to 1)

Value

\strong{data.frame} \cr
combined information in one data.frame.

aggregateReplicates *Aggregate replicates of the same experimental group.*

Description

Aggregate replicates of the same experimental group.

Usage

```
aggregateReplicates(
  D,
  group = NULL,
  missing.limit = 0,
  method = "mean",
  seq_col = "Sequence"
)
```

Arguments

D	SummarizedExperiment SummarizedExperiment Dataset containing peptide intensities, e.g. the result of normalizePeptideIntensities .
group	character factor The groups per sample for aggregation, if not already in SummarizedExperiment::colData(D)\$group.
missing.limit	numeric The proportion of missing values that is allowed (e.g. 0 means no missings allowed).
method	character The method of aggregation. Options are "mean", "sum" or "median"
seq_col	character The column name containaining the peptide sequences in the rowData of the SummarizedExperiment. Default is "Sequence"

Value

SummarizedExperiment

A SummarizedExperiment with aggregated intensities (\$intensities).

Examples

```
file <- system.file("extdata", "peptides_filtered.txt", package = "bppg")
group <- factor(rep(1:9, each = 3))
D <- readMqPeptideTable(path = file, group = group, LFQ = TRUE,
  remove_contaminants = FALSE)
D_norm <- bppg::normalizePeptideIntensities(D)
group <- factor(rep(1:9, each = 3))
```

```
aggregateReplicates(D_norm, group = group)
```

```
automatedAnalysisIteratedCi
```

Extract protein ratio solutions from the result of iterateOverCi()

Description

Extract protein ratio solutions from the result of iterateOverCi()

Usage

```
automatedAnalysisIteratedCi(
  G,
  res,
  use_results_from_other_proteins = TRUE,
  verbose = FALSE,
  job = NULL,
  error_tol = 1e-10,
  ratioLog_tol = 1e-06
)
```

Arguments

G	igraph object An igraph graph of the bipartite peptide-protein graph with peptide ratios
res	data.frame The data.frame resulting from the iterateOverCi() function.
use_results_from_other_proteins	logical If TRUE, the results from other proteins within the same graph will be used to calculate the optimal solution for each protein node. Default is TRUE
verbose	logical If TRUE, additional information will be printed.
job	BatchExperiment job object Is used to print the job id and parameters in the output
error_tol	numeric(1) Tolerance for a constant error term. Default is 1e-10.
ratioLog_tol	numeric(1) Tolerance for a constant log2 protein ratios. The default is 1e-6.

Value

A SummarizedExperiment object with one row for each protein in the assay.

See Also

[iterateOverCi\(\)](#), [.minimizeSquaredError\(\)](#)

Examples

```
file <- system.file("extdata", "quantGraphs.rds", package = "bppg")
graphs <- readRDS(file)
G <- graphs$"1_2"[[2]]
# small example with a small grid size
res <- iterateOverCi(G, gridSize = 100)
automatedAnalysisIteratedCi(G, res)
```

calculatePeptideRatios

Calculation of peptide ratios from aggregated intensities.

Description

Calculation of peptide ratios from aggregated intensities.

Usage

```
calculatePeptideRatios(D, group_levels = NULL)
```

Arguments

D	SummarizedExperiment The result from function aggregateReplicates() .
group_levels	character factor The group information in the sample order. They are extracted from D by default

Value

A SummarizedExperiment with log2 peptide ratios (logRatios).

Examples

```
file <- system.file("extdata", "peptides_filtered.txt", package = "bppg")
group <- factor(rep(1:9, each = 3))
D <- readMqPeptideTable(path = file, group = group, LFQ = TRUE,
  remove_contaminants = FALSE)
D_norm <- bppg::normalizePeptideIntensities(D)
dAgg <- aggregateReplicates(D_norm)
calculatePeptideRatios(dAgg)
```

combineComparisons	<i>Combine relative quantification results over multiple pairwise comparisons into a single SummarizedExperiments object.</i>
--------------------	---

Description

Combine relative quantification results over multiple pairwise comparisons into a single SummarizedExperiments object.

Usage

```
combineComparisons(compResultList)
```

Arguments

compResultList **list**
List of results to be concatenated. Each list element is the result of the optimization step for one pairwise comparison, i.e. a SummarizedExperiment object. Ideally, the element names should correspond to the comparison names.

Value

A SummarizedExperiment object with one assay per comparison. The row names of the assay matrices are the union of all row names across the input SummarizedExperiment objects. For rows that are not present in a given comparison, NA values are introduced. Row names are sorted alphabetically.

Examples

```
# prepared result list. Check "inst/scripts/filter_fasta_and_quantData.R" for
# details on how it was generated.
file <- system.file("extdata", "resultsList.rds", package = "bppg")
resultList <- readRDS(file)
SE <- combineComparisons(resultList)
```

digestFASTA	<i>In silico tryptic digestion of whole FASTA file.</i>
-------------	---

Description

In silico tryptic digestion of whole FASTA file.

Usage

```
digestFASTA(
  fasta,
  missed_cleavages = 2,
  min_aa = 6,
  max_aa = 50,
  protOrigin = NULL,
  verbose = FALSE,
  ...
)
```

Arguments

<code>fasta</code>	list of character sequence A fasta file, already read into R by <code>seqinr::read.fasta()</code> . If several protein origins are used they should be combined into one flatten list and a list with the corresponding origins should be provided for protOrigin .
<code>missed_cleavages</code>	integer The maximal number of missed cleavages.
<code>min_aa</code>	integer The minimal number of amino acids (set to 0 for no filtering).
<code>max_aa</code>	integer The maximal number of amino acids (set to Inf for no filtering).
<code>protOrigin</code>	list A list with the protein origin corresponding to <code>fasta</code> , proteins are used as index.
<code>verbose</code>	logical If TRUE, additional information on each iteration of the digestion is printed.
<code>...</code>	Arguments passed on to <code>.digest2</code>
<code>sequence</code>	character The protein sequence.
<code>enzyme</code>	character The enzyme used in digestion e.g. "trypsin" (does not cut before proline).
<code>missed</code>	character The maximal number of missed cleavages.
<code>warn</code>	logical If TRUE, warnings will be printed e.g. if a protein has no cleavage site.
<code>remove_initial_M</code>	logical If TRUE, the initial methionine will be removed from the peptides.

Value

data.frame with proteins and their peptide sequences, filtered for minimal and maximal number of amino acids.

See Also

[.digest2\(\)](#)

Examples

```
library(seqinr)
file <- system.file("extdata", "uniprot_proteome_Scerevisiae_filtered.fasta",
  package = "bppg")
fasta <- seqinr::read.fasta(file = file, seqtype = "AA", as.string = TRUE)
res <- digestFASTA(fasta)
```

exportSE	<i>Export SummarizedExperiment object to an Excel file, with one assay per sheet.</i>
----------	---

Description

Export SummarizedExperiment object to an Excel file, with one assay per sheet.

Usage

```
exportSE(SE, file)
```

Arguments

SE	SummarizedExperiment object Object to be exported. E.g. the result of combineComparisons .
file	character file path to the output Excel file. If the file already exists, it will be overwritten.

Value

nothing, but an Excel file is written containing the assay matrices. Each assay is written to a separate sheet, and the sheet name corresponds to the assay name. Row names of the assay matrices are included in the Excel file.

Examples

```
# prepared result list. Check "inst/scripts/filter_fasta_and_quantData.R" for
# details on how it was generated.
file <- system.file("extdata", "resultsList.rds", package = "bppg")
resultList <- readRDS(file)
SE <- combineComparisons(resultList)
TMP <- tempdir()
exportSE(SE, file.path(TMP,"results.xlsx"))
```

`generateGraphsFromEdgelist`*Generate bipartite peptide-protein graphs from a list of digested proteins via an edgelist. Peptide and protein nodes can be contracted.*

Description

Generate bipartite peptide-protein graphs from a list of digested proteins via an edgelist. Peptide and protein nodes can be contracted.

Usage

```
generateGraphsFromEdgelist(  
  edgelist,  
  collProtNodes = FALSE,  
  collPeptNodes = FALSE  
)
```

Arguments

<code>edgelist</code>	data.frame An edgelist, output from <code>digestFASTA()</code> . Quant data needs to be in the column <code>\$pep_logRatio</code> .
<code>collProtNodes</code>	logical If TRUE, the protein nodes will be collapsed.
<code>collPeptNodes</code>	logical If TRUE, the peptide nodes will be collapsed.

Value

A list of subgraphs as igraph objects.

See Also

[digestFASTA\(\)](#)

Examples

```
library(seqinr)  
file <- system.file("extdata", "uniprot_proteome_Scerevisiae_filtered.fasta",  
  package = "bppg")  
fasta <- seqinr::read.fasta(file = file, seqtype = "AA", as.string = TRUE)  
edgelist <- digestFASTA(fasta)  
res <- bppg::generateGraphsFromEdgelist(edgelist, collProtNodes = TRUE,  
  collPeptNodes = TRUE)
```

generateGraphsFromFASTA

Generate graphs from a FASTA file

Description

Generate graphs from a FASTA file

Usage

```
generateGraphsFromFASTA(
  fasta,
  collProtNodes = TRUE,
  collPeptNodes = TRUE,
  outpath = NULL,
  suffix = NULL,
  protOrigin = NULL,
  verbose = FALSE,
  ...
)
```

Arguments

fasta	list of vector of chars A fasta file, already read into R by <code>seqinr::read.fasta()</code> .
collProtNodes	logical If TRUE, the protein nodes will be collapsed.
collPeptNodes	logical If TRUE, the peptide nodes will be collapsed.
outpath	character The path where intermediate results are saved. If NULL, results are not saved.
suffix	character The suffix for saving results. will also be saved.
protOrigin	list or data.frame A list with the protein origin corresponding to fasta , proteins are used as row-names/index.
verbose	logical If TRUE, additional information on each iteration of the optimization is printed
...	Additional arguments to <code>bppg::digestFASTA()</code>

Value

Subgraphs (i.e. connected components) from the graph generated from the FASTA file.

Examples

```
library(seqinr)
file <- system.file("extdata", "uniprot_proteome_Scerevisiae_filtered.fasta",
  package = "bppg")
fasta <- seqinr::read.fasta(file = file, seqtype = "AA", as.string = TRUE)
graphs <- bppg::generateGraphsFromFASTA(fasta)
```

generateGraphsFromQuantData

Generate graphs from quantitative peptide-level data

Description

Generate graphs from quantitative peptide-level data

Usage

```
generateGraphsFromQuantData(
  D,
  fasta,
  outpath = NULL,
  missed_cleavages = 2,
  min_aa = 6,
  max_aa = 50,
  seq_column = "Sequence",
  norm_method = "loess",
  collProtNodes = TRUE,
  collPeptNodes = FALSE,
  suffix = "",
  protOrigin = NULL,
  verbose = FALSE,
  ...
)
```

Arguments

D	data.frame A data set with peptide sequence as first column and peptide intensities in subsequent columns, e.g. created with readMqPeptideTable() .
fasta	list of vector of characters A fasta file used for identification of peptides in already read into R by seqinr::read.fasta() .
outpath	character The output path for the results.
missed_cleavages	integer The number of allowed missed cleavages in a peptide.

<code>min_aa</code>	integer The minimum number of amino acids in a peptide.
<code>max_aa</code>	integer The maximum number of amino acids in a peptide.
<code>seq_column</code>	character The column name of the column of D with the peptide sequences.
<code>norm_method</code>	character The method of normalization. Options are "nonorm" (no normalization), "median", "loess", "quantile" or "lts" normalization. Default is "loess"
<code>collProtNodes</code>	logical If TRUE, the protein nodes will be collapsed.
<code>collPeptNodes</code>	logical If TRUE, the peptide nodes will be collapsed.
<code>suffix</code>	character The suffix for output files.
<code>protOrigin</code>	list or data.frame A list with the protein origin corresponding to fasta , proteins are used as row-names/index.
<code>verbose</code>	logical If TRUE, additional information on each iteration of the optimization is printed
<code>...</code>	Additional arguments for <code>.digest2()</code> .

Value

A list of list of graphs. The outer list is for the different possible comparisons, the inner layer is for the independent graphs in that comparison.

See Also

`readMqPeptideTable()`, `seqinr::read.fasta()`, `generateQuantGraphs()`, `generateGraphsFromFASTA()`

Examples

```
library(seqinr)
file <- system.file("extdata", "uniprot_proteome_Scerevisiae_filtered.fasta",
  package = "bppg")
fasta <- seqinr::read.fasta(file = file, seqtype = "AA", as.string = TRUE)

file <- system.file("extdata", "peptides_filtered.txt", package = "bppg")
group <- factor(rep(1:9, each = 3))
D <- readMqPeptideTable(path = file, group = group, LFQ = TRUE,
  remove_contaminants = FALSE)

graphs <- bppg::generateGraphsFromQuantData(D, fasta)
```

generateQuantGraphs	<i>Generate graphs from peptide ratio table, using an edgelist calculated on the fasta file.</i>
---------------------	--

Description

Generate graphs from peptide ratio table, using an edgelist calculated on the fasta file.

Usage

```
generateQuantGraphs(  
  exp_peptide_ratios,  
  fasta_edgelist,  
  seq_column = "Sequence",  
  outpath = NULL,  
  collProtNodes = TRUE,  
  collPeptNodes = FALSE,  
  suffix = ""  
)
```

Arguments

exp_peptide_ratios	SummarizedExperiment A SummarizedExperiment from calculatePeptideRatios with peptide ratios.
fasta_edgelist	data.frame An edgelist created from the corresponding FASTA file, eg. created with digestFASTA() .
seq_column	character The column name of the peptide sequence.
outpath	character The output path for the results.
collProtNodes	logical If TRUE, the protein nodes will be collapsed.
collPeptNodes	logical If TRUE, the peptide nodes will be collapsed.
suffix	character The suffix for saving results.

Value

A list of list of subgraphs

See Also

[digestFASTA\(\)](#)

Examples

```
library(seqinr)
file <- system.file("extdata", "uniprot_proteome_Scerevisiae_filtered.fasta",
  package = "bppg")
fasta <- seqinr::read.fasta(file = file, seqtype = "AA", as.string = TRUE)
edgelist <- digestFASTA(fasta)

file <- system.file("extdata", "peptides_filtered.txt", package = "bppg")
group <- factor(rep(1:9, each = 3))
D <- readMqPeptideTable(path = file, group = group, LFQ = TRUE,
  remove_contaminants = FALSE)
D_norm <- bppg::normalizePeptideIntensities(D)
dAgg <- aggregateReplicates(D_norm)
exp_peptide_ratios <- calculatePeptideRatios(dAgg)

res <- generateQuantGraphs(exp_peptide_ratios, edgelist)
```

iterateOverCi	<i>Iterate over possible Ci values</i>
---------------	--

Description

Iterate over possible Ci values

Usage

```
iterateOverCi(
  G,
  gridStart = 0,
  gridStop = 1,
  gridSize = 1000,
  omit_grid_borders = TRUE,
  extend_grid_at_borders = FALSE,
  verbose = FALSE,
  control = list()
)
```

Arguments

G	igraph object Bipartite peptide-protein graph
gridStart	integer The start of the grid (default is 0).
gridStop	integer The end of the grid (default is 1).
gridSize	integer The number of grid points for the Cis.

omit_grid_borders	logical If TRUE (default), omit exact value of 1 and 0 from the grid (recommended, as they may cause numerical issues).
extend_grid_at_borders	logical If TRUE, the grid will be extend close to the borders (0 and 1). While exact values of 0 and 1 may cause numerical problems, values close to those may be valuable to get a better estimate of the protein ratios. Default is FALSE.
verbose	logical If TRUE, print additional information (see solnp function).
control	list The control object to be passed to the .minimizeSquaredError() function.

Details

With [.minimizeSquaredError\(\)](#) each protein node receives one estimate for the protein ratio. However, in some cases, there are multiple possible values for the protein ratios that lead to the same, minimal error term. To get a better coverage of the optimal solutions, this function iterates over a grid of possible weights C_i for each protein node. For each protein node, the C_i value is fixed on a point on the grid, while the other C_i values and all R_i values are optimized using [.minimizeSquaredError\(\)](#). The resulting table can be used to assess a range of possible solutions for the protein ratios. The table can be further processed with [automatedAnalysisIteratedCi\(\)](#).

Value

A data.frame containing the optimal C_i and R_i values together with the reached minimal error term for each grid point.

See Also

[.minimizeSquaredError\(\)](#), [automatedAnalysisIteratedCi\(\)](#)

Examples

```
file <- system.file("extdata", "quantGraphs.rds", package = "bppg")
graphs <- readRDS(file)
G <- graphs$"1_2"[[2]]
# small example with a small grid size
iterateOverCi(G, gridSize = 100)
```

`normalizePeptideIntensities`*Normalization of peptide intensities*

Description

Normalization of peptide intensities

Usage

```
normalizePeptideIntensities(  
  D,  
  method = "loess",  
  lts.quantile = 0.8,  
  log_base = 2  
)
```

Arguments

<code>D</code>	SummarizedExperiment SummarizedExperiment dataset containing peptide intensities, e.g. readMqPeptideTable .
<code>method</code>	character The method of normalization. Options are "nonorm" (no normalization), "median", "loess", "quantile" or "lts" normalization. Default is "loess"
<code>lts.quantile</code>	numeric The quantile for the lts normalization. Default is 0.8.
<code>log_base</code>	numeric The base for log-transformation. Default is 2.

Value

SummarizedExperiment
Dataset containing normalized peptide intensities.

Examples

```
file <- system.file("extdata", "peptides_filtered.txt", package = "bppg")  
D <- readMqPeptideTable(path = file, LFQ = TRUE, remove_contaminants = FALSE)  
D_norm <- normalizePeptideIntensities(D, method = "loess")
```

plotBipartiteGraph *Plotting of bipartite peptide-protein graphs.*

Description

Plotting of bipartite peptide-protein graphs.

Usage

```
plotBipartiteGraph(
  G,
  legend = TRUE,
  vertex.color = c("mediumseagreen", "cadetblue2", "coral1"),
  vertex.size = 15,
  edge.width = 0.5,
  node_labels_proteins = "letters",
  node_labels_peptides = "numbers",
  round_digits = 2,
  output_path = NULL,
  ...
)
```

Arguments

G	igraph graph object A bipartite peptide-protein graph.
legend	logical If TRUE, a legend will be added.
vertex.color	character vector The colours for the different vertex types.
vertex.size	numeric The size of vertices.
edge.width	numeric The width of the edges.
node_labels_proteins	character The type of labels for the proteins. Options are "letters" or "accessions".
node_labels_peptides	character The type of labels for the peptides. Options are "numbers" or "pep_logRatios" or "pep_logRatios_mean".
round_digits	integer The number of digits to round the peptide ratios to.
output_path	character file path for optional save of figure.
...	Additional arguments for ggsave.

Value

Plot of one bipartite graph.

Examples

```
file <- system.file("extdata", "quantGraphs_collpept.rds", package = "bppg")
graphs <- readRDS(file)
G <- graphs$"1_2"[[2]]

plotBipartiteGraph(G)
```

proteinElimination	<i>Tries to recursively remove protein nodes from graph while keeping the error term in the optimization step low</i>
--------------------	---

Description

Tries to recursively remove protein nodes from graph while keeping the error term in the optimization step low

Usage

```
proteinElimination(
  G,
  threshold = 1.05,
  control = list(),
  min_error_ref = NULL,
  resDF = NULL,
  protsOriginIDs = NULL,
  res_best = NULL
)
```

Arguments

G	igraph graph object The graph to eliminate the proteins in.
threshold	numeric The threshold for increase of error term. The default 1.05 refers to 5% increase.
control	list A list of control parameters for the optimization step. See solnp for details.
min_error_ref	numeric The minimal error term on the whole graph using all available protein nodes.
resDF	data.frame Dataframe for collecting results from each iteration. It contains information on the tested combinations of protein nodes, the error terms and the currently best combination.

protosOriginIDs	igraph node list This will be initialized during the first iteration, please keep the default NULL value. It is the current list of protein nodes relative to the original graph.
res_best	list A list containing the currently best solution including the protein node combination, the error term and the corresponding graph object.

Details

This function works in a recursive way. In the first iteration, the error term on the whole graph is assessed. Then, one of the protein nodes is deleted and the function is recursively applied. The results of all iterations are collected in the resDF dataframe.

For starting the first iteration, only G, threshold and if necessary control have to be defined, everything else will be calculated during the first iteration for all future iterations.

Value

list containing the following elements:

min_error_ref	reference error term of the whole graph
protosOriginIDs	list of all available protein nodes in the beginning
resDF	dataframe with results of all iterations
res_best	List of the overall best solution. The graph object(s) in this list can directly be used by iterateOverCi .

See Also

[.minimizeSquaredError\(\)](#)

Examples

```
## TODO
file <- system.file("extdata", "quantGraphs.rds", package = "bppg")
graphs <- readRDS(file)
G <- graphs[[1]][[4]]
# This graph has 4 protein nodes.

proteinElimination(G)
# Only 3 protein nodes are kept after protein elimination.
# The error term only increases slightly.
```

readFpPeptideTable *Import of FragPipe's combined_peptide.tsv-table.*

Description

Import of FragPipe's combined_peptide.tsv-table.

Usage

```
readFpPeptideTable(
  path,
  group = NULL,
  rename_columns = TRUE,
  zeroToNA = TRUE,
  remove_empty_rows = TRUE,
  further_columns_to_keep = NULL,
  verbose = FALSE
)
```

Arguments

path	character The path to the peptides.txt table.
group	character List or vector of group names corresponding to the order of samples.
rename_columns	logical If TRUE, "Intensity." or "LFQ.intensity." are removed
zeroToNA	logical If TRUE, zeros are converted to NAs.
remove_empty_rows	logical If TRUE, rows with only NAs are removed.
further_columns_to_keep	integer vector Indices of additional columns to keep, except peptide sequence and intensities
verbose	logical If TRUE, additional information on the data extraction is printed

Value

A SummarizedExperiment with intensities, sequences, and optional data for the rowData dataframe.

Examples

```
file <- system.file("extdata", "peptides_FP_filtered.tsv", package = "bppg")
D1 <- bppg::readFpPeptideTable(file)
```

readMqPeptideTable	<i>Import of MaxQuant's peptide.txt-table.</i>
--------------------	--

Description

Import of MaxQuant's peptide.txt-table.

Usage

```
readMqPeptideTable(  
  path,  
  group = NULL,  
  LFQ = FALSE,  
  remove_contaminants = FALSE,  
  rename_columns = TRUE,  
  zeroToNA = TRUE,  
  remove_empty_rows = TRUE,  
  further_columns_to_keep = NULL,  
  verbose = FALSE  
)
```

Arguments

path	character The path to the peptides.txt table.
group	character List or vector of group names corresponding to the order of samples.
LFQ	logical If TRUE, LFQ intensities are used, if FALSE, raw (unnormalized) intensities
remove_contaminants	logical If TRUE, peptide sequences from potential contaminants are removed
rename_columns	logical If TRUE, "Intensity." or "LFQ.intensity." are removed
zeroToNA	logical If TRUE, zeros are converted to NAs.
remove_empty_rows	logical If TRUE, rows with only NAs are removed.
further_columns_to_keep	integer vector Indices of additional columns to keep, except peptide sequence and intensities
verbose	logical If TRUE, additional information on the data extraction is printed

Value

A SummarizedExperiment with intensities, sequences, and optional data for the rowData dataframe.

Examples

```
file <- system.file("extdata", "peptides_filtered.txt", package = "bppg")  
D <- readMqPeptideTable(path = file, LFQ = TRUE, remove_contaminants = FALSE)
```

Index

.addUniquenessAttributes, [3](#)
.analyseResultSingleProt, [4](#)
.calcConstraints, [4](#)
.calcObjectiveFunction, [5](#)
.calcResultGridpoint, [6](#)
.cleave, [7](#)
.contractGraph, [8](#)
.digest2, [9](#), [23](#)
.digest2(), [7](#), [24](#), [28](#)
.directBipartiteGraph, [10](#)
.errorEquation, [6](#), [10](#), [17](#), [18](#)
.extractIntensities, [11](#)
.foldChange, [12](#)
.geomMean, [12](#)
.getContractMapping, [13](#)
.getContractMapping(), [8](#)
.initializeCi, [14](#)
.initializeRi, [15](#)
.isomorphicBipartite, [15](#)
.minimizeSquaredError, [16](#)
.minimizeSquaredError(), [7](#), [21](#), [31](#), [35](#)
.setNodeLabels, [17](#)
.trackingDataFrame, [18](#)

aggregateReplicates, [19](#)
aggregateReplicates(), [21](#)
automatedAnalysisIteratedCi, [20](#)
automatedAnalysisIteratedCi(), [31](#)

calculatePeptideRatios, [21](#), [29](#)
combineComparisons, [22](#), [24](#)

digestFASTA, [22](#)
digestFASTA(), [7](#), [9](#), [13](#), [25](#), [29](#)

exportSE, [24](#)

fasta, [23](#), [26](#), [28](#)

generateGraphsFromEdgelist, [25](#)
generateGraphsFromEdgelist(), [8](#), [16](#)

generateGraphsFromFASTA, [26](#)
generateGraphsFromFASTA(), [3](#), [13](#), [28](#)
generateGraphsFromQuantData, [27](#)
generateQuantGraphs, [29](#)
generateQuantGraphs(), [3](#), [13](#), [28](#)

iterateOverCi, [30](#), [35](#)
iterateOverCi(), [4](#), [20](#), [21](#)

normalizePeptideIntensities, [19](#), [32](#)

plotBipartiteGraph, [33](#)
proteinElimination, [34](#)

readFpPeptideTable, [36](#)
readMqPeptideTable, [32](#), [37](#)
readMqPeptideTable(), [27](#), [28](#)

seqinr::read.fasta(), [23](#), [27](#), [28](#)
solnp, [5](#), [7](#), [17](#), [31](#), [34](#)