Package 'protiq'

February 20, 2015

Type Package

Title Protein (identification and) quantification based on peptide evidence

Version 1.2

Date 2012-10-01

Author Sarah Gerster and Peter Buehlmann

Maintainer ORPHANED

Description Method for protein quantification based on identified and quantified peptides. protiq can be used for absolute and relative protein quantification. Input peptide abundance scores can come from various sources, including SRM transition areas and intensities or spectral counts derived from shotgun experiments. The package is still being extended to also include the model for protein identification, MIPGEM, presented in Gerster, S., Qeli, E., Ahrens, C.H. and Buehlmann, P. (2010). Protein and gene model inference based on statistical modeling in k-partite graphs. Proceedings of the National Academy of Sciences 107(27):12101-12106.

License GPL (>= 2)

Depends methods

Imports graph, RBGL, mvtnorm

Suggests gplots

Collate 'scampi_classes.R' 'protiq.R'

NeedsCompilation no

Repository CRAN

Date/Publication 2013-08-19 01:00:26

X-CRAN-Original-Maintainer Sarah Gerster <sarah.gerster@isb-sib.ch>

X-CRAN-Comment Orphaned on 2014-12-07 as maintainer address <sarah.gerster@isb-sib.ch> bounced.

R topics documented:

`protiq-package` *Protein (identification and) quantification based on peptide evidence*

Description

Method for protein quantification based on identified and quantified peptides. `protiq` can be used for absolute and relative protein quantification. Input peptide abundance scores can come from various sources, including SRM transition areas and intentisites or spectral counts derived from shotgun experiments. The package is being extended to also include the model for protein identification, MIPGEM, presented in Gerster et al., see references.

Details

Package:	protiq
Type:	Package
Version:	1.0
Date:	2012-09-30
License:	GPL (>= 2)
Depends:	methods
Imports:	graph, RBGL, mvtnorm

Author(s)

Sarah Gerster (maintainer) <sarah.gerster@isb-sib.ch> Peter Buehlmann

References

Gerster, S., Qeli, E., Ahrens, C.H. and Buehlmann, P. (2010). Protein and gene model inference based on statistical modeling in k-partite graphs. *Proceedings of the National Academy of Sciences* 107(27):12101-12106.

See Also

Main functions: runScampi, iterateScampi

---

checkInputData — *Generic function to check if the input data fulfills all requirements for protein inference with MIPGEM or quantification with SCAMPI.*

Description

The dataframes are checked for completeness: are all needed input variables provided? Furthermore, the numbering of the dataframes is checked and inconsistencies are reported.

Usage

```
checkInputData(scampiData, ...)
```

Arguments

scampiData	object of class scampi or mipgem
...	further arguments to be passed to the methods

Details

Currently only the functions for quantification, related to objects of the class scampi, are available.

Author(s)

Sarah Gerster <sarah.gerster@isb-sib.ch>

See Also

checkInputData.scampi for details about the checks performed on scampi objects

Examples

```
data("leptoSRM")
dataChecked <- checkInputData(scampi(peptides=leptoSRMpeptides,
                                     proteins=leptoSRMproteins,
                                     edgespp=leptoSRMedgespp),
                              rescaling=FALSE)
```

`checkInputData.scampi` *Check if input object fulfills all requirements in order to proceed with the protein quantification step.*

Description

The dataframes contained in the input object are tested for completeness and consistency. It ensures that the input data is suited to build the bipartite graph and holds all needed variables for parameter estimation, prediction of protein quantities and peptide abundance reassessment. If some optional variables are missing, the dataframes are completed with default values.

Usage

```
## S3 method for class 'scampi'
checkInputData(scampiData, rescaling = TRUE, verbose = FALSE, ...)
```

Arguments

`scampiData`	object of class `scampi-class`
`rescaling`	If TRUE, the peptide abundance scores are logarithmized (log10). If this transformation has not yet been done during preprocessing, it is strongly recommended to stick to the default: `rescaling=TRUE`.
`verbose`	if TRUE, basic information are printed to indicate the progress of the function
`...`	further arguments

Value

Object of class `scampi-class` containing the checked, and possily completed, dataframes. If the input data does not match the requirements, the function exits with an error message.

Author(s)

Sarah Gerster <sarah.gerster@isb-sib.ch>

See Also

`scampi-class`

Examples

```
data("leptoSRM")
dataChecked <- checkInputData(scampi(peptides=leptoSRMpeptides,
                                     proteins=leptoSRMproteins,
                                     edgespp=leptoSRMedgespp),
                              rescaling=FALSE)
```

`estimateModelParameters` *Estimate the SCAMPI model paramteres alpha, beta, mu and tau from the data*

Description

Estimate the parameter values with maximum likelihood (`MLE`) or a method of moments approach (`LSE`), or both (`all`)

Usage

```
estimateModelParameters(method = "all", ccList, peptides = NULL,
                        numIter = 10, verbose = FALSE)
```

Arguments

`method`	method to be used for the parameter estimation; can be `all` (MLE and LSE, default), `LSE` or `MLE`
`ccList`	list of pre-processed connected components
`peptides`	data frame with pre-processed peptide info (only used for LSE)
`numIter`	number of successful munerical optimizations to perform (only used for MLE, see details)
`verbose`	If `TRUE`, detailed output is provided.

Details

To use `method="MLE"` the inverses of the covariance matrices (of the connected components) are needed. Depending on the chosen parameters, this can lead to stability issues. To avoid the function from crashing, a `try(...)` bolck is used: the parameter estimation is performed until it was successful `numIter` times. Among these `numIter` sets, the one with the lowest negative log-likelihood value is returned.

Value

Named list of vectors with at least four named elements: `alphaH`, `betaH`, `muH` and `tauH` (estimates for the four model parameters). The name of the list elements corresponds to the parameter estimation method, namely `LSE` or `MLE`.

Note

This function is called by one of the main functions of the package, runScampi. Calling this function directly is seldomly necessary.

Author(s)

Sarah Gerster <sarah.gerster@isb-sib.ch>

See Also

runScampi, iterateScampi

Examples

```
## Not run:
data("leptoSRM")

dataChecked <- checkInputData(scampi(peptides=leptoSRMpeptides,
                                     proteins=leptoSRMproteins,
                                     edgespp=leptoSRMedgespp),
                              rescaling=FALSE)
tmpPrepro <- preprocessInputData(dataChecked)
dataPrepro <- tmpPrepro[["dataPrepro"]]
ppGraph <- tmpPrepro[["ppGraph"]]
myCCList <- tmpPrepro[["ccList"]]
rm("tmpPrepro")
scampiParam <- estimateModelParameters(method="all", ccList=myCCList,
                                       peptides=dataPrepro@peptides,
                                       numIter=10)

## End(Not run)
```

---

getCovU *Compute covariance matrix of peptide abundances*

Description

Compute the covariance matrix of the peptide abundances in the same connected component for given parameter values (beta and tau).

Usage

```
getCovU(cc, beta, tau)
```

Arguments

cc	pre-processed connected component (list)
beta	model parameter ("weight" of protein abundance contribution)
tau	model parameter (variance of error term)

Value

Same list as cc with an additional element named covU

Note

This function is called by higher level functions (quantifyProteins). Calling it directly is seldomly necessary.

Author(s)

Sarah Gerster <sarah.gerster@isb-sib.ch>

See Also

quantifyProteins

Examples

```
## get data
data("leptoSRM")
## check input data
#dataChecked <-
dataChecked <-
  checkInputData(scampiData=scampi(peptides=leptoSRMpeptides,
                                   proteins=leptoSRMproteins,
                                   edgespp=leptoSRMedgespp),
                 rescaling=FALSE)
## preprocess input data
tmpPrepro <- preprocessInputData(scampiData=dataChecked)
dataPrepro <- tmpPrepro[["dataPrepro"]]
myCCList <- tmpPrepro[["ccList"]]
rm(tmpPrepro)

## compute covariance matrices
myCCList <- lapply(myCCList, getCovU,
                   beta=0.2,
                   tau=0.5)
```

---

iterateScampi *SCAMPI protein quantification function with iterative outlier removal*

Description

Estimate a protein abundance score for each protein in the dataset, based on the input peptide abundance scores and the connectivity information between peptides and proteins. The expected values for the peptide abundances are computed as well. Comparing these values with the initial measurements allows to detect outliers in the input data. Several iterations of abundance estimation and outlier removal can then be performed.

Usage

```
iterateScampi(peptides, proteins, edgespp, rescaling = TRUE,
              method = "LSE", numIter = 2, numMLEIter = 10,
              thresh = 2, verbose = FALSE)
```

Arguments

peptides	Data frame with peptide information. The following columns are required: pepId (unique identification number for each distinct peptide sequence, numbering from 1:n where n=number of distinct peptide sequences), pepSeq (peptide sequence, optionally including modifications and charge states), and pepQty (peptide abundance score). An additional column pepObs (peptide observability or identification score) is used if provided. Each row in the data frame describes one observed distinct peptide sequence.
proteins	Data frame with the protein information. The following columns are required: protId (unique identification number for each distinct protein sequence, numbering from (n+1):(n+m) where m=number of distinct protein sequences), protName (protein identifier or protein sequence). Each row describes a distinct protein sequence to which at least one of the observed peptides is matching.
edgespp	Data frame with two mandatory columns: pepId and protId. Each row defines an edge of the bipartite graph.
rescaling	If TRUE, the peptide abundance scores are logarithmized (log10). If this transformation has not yet been done during preprocessing, it is strongly recommended to stick to the default: rescaling=TRUE.
method	Describes which method should be used for the parameter estimation. Available: method="LSE" (default), method="MLE" and method="all".
numIter	Number of estimation/outlier-removal iterations to be performed.
numMLEIter	Only used with method="MLE", see details. Default: numIter=10.
thresh	Constant to tune the outlier selection process. See details.
verbose	If TRUE, detailed output is provided.

Details

To use method="MLE" the inverses of the covariance matrices (of the connected components) are needed. Depending on the chosen parameters, this can lead to stability issues. To avoid the function from crashing, a try(...) bolck is used: the parameter estimation is performed until it was successful numIter times. Among these numIter sets, the one with the lowest negative log-likelihood value is returned.

Peptide outlier detection is based on an interquartile range criterion on the peptide abundance residuals. The larger the chosen thresh, the less peptides get discarded.

Value

Named list. Each element corresponds to one iteration step, and is a list itself with

scampiRes	object of class scampiVal

peptideOutliers
: dataframe with the peptides selected as outliers and not used (removed from the graph) for this iteration step

Author(s)

Sarah Gerster <sarah.gerster@isb-sib.ch>

See Also

runScampi to perform a single iteration

Examples

```
data("leptoSRM")
scampiIterRes <- iterateScampi(peptides=leptoSRMpeptides,
                               proteins=leptoSRMproteins,
                               edgespp=leptoSRMedgespp, rescaling=FALSE,
                               method="LSE", numIter=3, thresh=1.37)
```

---

leptoSRM *SRM quantification example data*

Description

This dataset holds peptide SRM measurements matching to 39 proteins. The data is summarized in three data frames.

Usage

```
data(leptoSRM)
```

Format

Three data frames (leptoSRMpeptides, leptoSRMproteins and leptoSRMedgespp) containing the required input data for SCAMPI.

leptoSRMpeptides: data frame with 151 distinct peptides (=151 rows)

leptoSRMproteins: data frame with 39 distinct protein sequences (=39 rows)

leptoSRMedgespp: data frame with 151 edges (=151 rows) between peptides and proteins

Source

Ludwig et al., see references

References

Ludwig et al. Estimation of absolute protein quantities of unlabeled samples by selected reaction monitoring mass spectrometry (2011) *Molecular & Cellular Proteomics*

leptoSRMedgespp *SRM quantification: edges informationu.*

Description

This dataframe holds the edge information linking leptoSRMpeptides to leptoSRMproteins

Usage

```
data(leptoSRM)
```

Format

data frame with 151 edge definitions (=151 rows)

Source

Ludwig et al., see references

References

Ludwig et al. Estimation of absolute protein quantities of unlabeled samples by selected reaction monitoring mass spectrometry (2011) *Molecular & Cellular Proteomics*

leptoSRMpeptides *SRM quantification: peptide example data*

Description

This dataframe holds the peptide SRM measurements.

Usage

```
data(leptoSRM)
```

Format

data frame with 151 distinct peptides (=151 rows)

Source

Ludwig et al., see references

References

Ludwig et al. Estimation of absolute protein quantities of unlabeled samples by selected reaction monitoring mass spectrometry (2011) *Molecular & Cellular Proteomics*

leptoSRMproteins *SRM quantification: protein example data*

Description

This dataframe holds the proteins matching the the peptides in leptoSRMpeptides.

Usage

```
data(leptoSRM)
```

Format

data frame with 39 distinct proteins (=39 rows)

Source

Ludwig et al., see references

References

Ludwig et al. Estimation of absolute protein quantities of unlabeled samples by selected reaction monitoring mass spectrometry (2011) *Molecular & Cellular Proteomics*

preprocessInputData *Generic function to preprocess the input data for protein inference with MIPGEM or quantification with SCAMPI to speed up further computations.*

Description

Quantities which do not depend on the parameter values are precomputed and stored to be accessed quickly in further computations.

Usage

```
preprocessInputData(scampiData, ...)
```

Arguments

scampiData	object of class scampi or mipgem
...	further arguments to be passed to the methods

Details

Currently only the functions for quantification, related to objects of the class scampi, are available.

Author(s)

Sarah Gerster <sarah.gerster@isb-sib.ch>

See Also

preprocessInputData.scampi for details about the preprocessing performed on scampi objects

Examples

```
data("leptoSRM")
dataChecked <- checkInputData(scampi(peptides=leptoSRMpeptides,
                                     proteins=leptoSRMproteins,
                                     edgespp=leptoSRMedgespp),
                              rescaling=FALSE)
dataPrepro <- preprocessInputData(dataChecked)
```

---

preprocessInputData.scampi

Precompute all quantities which will be used (repetitously) for further computations and do not depend on the parameter values.

Description

Build the bipartite graph, generate the list of connected components, store information about the graph structure for each nodes (e.g. number of neighbors or index of the connected components), optionally transform peptide abundance score.

Usage

```
## S3 method for class 'scampi'
preprocessInputData(scampiData, verbose = FALSE, ...)
```

Arguments

scampiData	Object of class scampi
verbose	If TRUE, detailed output is provided.
...	further arguments

Value

List

dataPrepro	object of class scampi
ppGraph	object of class graphNEL
ccList	list of connected components of ppGraph

Note

This function is called by one of the main functions of the package, runScampi. Calling this function directly is seldomly necessary.

Author(s)

Sarah Gerster <sarah.gerster@isb-sib.ch>

See Also

runScampi, iterateScampi

Examples

```
data("leptoSRM")
dataChecked <- checkInputData(scampi(peptides=leptoSRMpeptides,
                                     proteins=leptoSRMproteins,
                                     edgespp=leptoSRMedgespp),
                              rescaling=FALSE)
dataPrepro <- preprocessInputData(dataChecked)
```

---

quantifyPeptide *Compute expected value for the peptide abundance.*

Description

Use provided model parameter values to compute the expected value of the abundance for a given petpitde. This value can the be compared to the originally measure value to identify outliers in the input data.

Usage

```
quantifyPeptide(pepInfo, ccList, param, verbose = FALSE)
```

Arguments

pepInfo	Vector with two elements: pepId and index of connected component (in list ccList) which contains pepId.
ccList	List of pre-processed connected components.
param	Vector with at least four named elements: alphaH, betaH, muH and tauH (estimates for the four model parameters).
verbose	If TRUE, detailed output is provided.

Details

In order to avoid overfitting, the value computed by this function E[U_k|U_i-k], hence the kth measurement is not used to predict the expected value of peptide k.

Value

Expected value of the peptide abundance (see details).

Note

This function is intended to assess a single peptide when the parameter values have been computed previously. To run a whole analysis (check input data, preprocess connected components, estimate model parameters and compute abundance sore estimates) you should use the function runScampi or iterateScampi.

Author(s)

Sarah Gerster <sarah.gerster@isb-sib.ch>

See Also

runScampi, iterateScampi

Examples

```
## get data
data("leptoSRM")
## check input data
dataChecked <- checkInputData(scampiData=scampi(peptides=leptoSRMpeptides,
                                                proteins=leptoSRMproteins,
                                                edgespp=leptoSRMedgespp),
                              rescaling=FALSE)
## preprocess input data
tmpPrepro <- preprocessInputData(scampiData=dataChecked)
dataPrepro <- tmpPrepro[["dataPrepro"]]
myCCList <- tmpPrepro[["ccList"]]
rm(tmpPrepro)

## compute covariance matrices
myCCList <- lapply(myCCList, getCovU,
                   beta=0.2,
                   tau=0.5)

## compute expected value of abundance for peptide 13
pepAbundanceScore <-
  quantifyPeptide(pepInfo=dataPrepro@peptides[13, c("pepId", "ccInd")],
                  ccList=myCCList, param=c(alphaH=0, betaH=0.2, muH=0.3,
                  tauH=0.2))
```

quantifyProtein *Compute expected value and variance of a protein abundance.*

Description

Use provided model parameter values to compute the expected value and variance of the abundance for a given protein.

Usage

```
quantifyProtein(protInfo, ccList, param, verbose = FALSE)
```

Arguments

protInfo	Vector with two elements: protId and index of connected component (in list ccList) which contains protId.
ccList	List of pre-processed connected components.
param	Vector with at least four named elements: alphaH, betaH, muH and tauH (estimates for the four model parameters).
verbose	If TRUE, detailed output is provided.

Value

Vector with two values: expected value and variance of the protein abundance.

Note

This function is intended to quantify a single protein when the parameter values have been computed previously. To run a whole analysis (check input data, preprocess connected components, estimate model parameters and compute abundance sore estimates) you should use the function runScampi or iterateScampi.

Author(s)

Sarah Gerster <sarah.gerster@isb-sib.ch>

See Also

runScampi, iterateScampi

Examples

```
## get data
data("leptoSRM")
## check input data
dataChecked <- checkInputData(scampiData=scampi(peptides=leptoSRMpeptides,
                                                proteins=leptoSRMproteins,
                                                edgespp=leptoSRMedgespp),
```

```
                              rescaling=FALSE)
## preprocess input data
tmpPrepro <- preprocessInputData(scampiData=dataChecked)
dataPrepro <- tmpPrepro[["dataPrepro"]]
myCCList <- tmpPrepro[["ccList"]]
rm(tmpPrepro)

## compute covariance matrices
myCCList <- lapply(myCCList, getCovU,
                   beta=0.2,
                   tau=0.5)

## compute expected value and variance of abundance for protein 7
protAbundanceScore <-
  quantifyProtein(protInfo=dataPrepro@proteins[7, c("protId", "ccInd")],
                  ccList=myCCList, param=c(alphaH=0, betaH=0.2, muH=0.3,
                  tauH=0.2))[1]
protAbundanceVariance <-
  quantifyProtein(protInfo=dataPrepro@proteins[7, c("protId", "ccInd")],
                  ccList=myCCList, param=c(alphaH=0, betaH=0.2, muH=0.3,
                  tauH=0.2))[2]
```

`quantifyProteins` *Compute the protein (and optionally the peptide) abundance scores.*

Description

Compute the protein (and optionally the peptide) abundance scores for all proteins given the specified parameter values

Usage

```
quantifyProteins(scampiData, ccList, paramList, quantifyPeptides = FALSE,
                 verbose = FALSE)
```

Arguments

`scampiData`	Object of class `scampi`
`ccList`	List of pre-processed connected components
`paramList`	Named list of vectors with at least four named elements: `alphaH`, `betaH`, `muH` and `tauH` (estimates for the four model parameters). The name of list element corresponds to the parameter estimation method, namely `LSE` or `MLE`.
`quantifyPeptides`	If TRUE, also compute peptide abundance scores/residuals.
`verbose`	If `TRUE`, detailed output is provided.

Value

Object of class `scampi`.

Note

This function is intended to quantify (a subset of) proteins and reassess peptide scores when the parameter values have already been computed previously. To run a whole analysis (check input data, preprocess connected components, estimate model parameters and compute abundance sore estimates) you should use the function runScampi or iterateScampi.

Author(s)

Sarah Gerster <sarah.gerster@isb-sib.ch>

See Also

runScampi, iterateScampi

Examples

```
## get data
data("leptoSRM")
## check input data
dataChecked <- checkInputData(scampiData=scampi(peptides=leptoSRMpeptides,
                                                proteins=leptoSRMproteins,
                                                edgespp=leptoSRMedgespp),
                              rescaling=FALSE)
## preprocess input data
tmpPrepro <- preprocessInputData(scampiData=dataChecked)
dataPrepro <- tmpPrepro[["dataPrepro"]]
myCCList <- tmpPrepro[["ccList"]]
rm(tmpPrepro)

## compute covariance matrices
myCCList <- lapply(myCCList, getCovU,
                   beta=0.2,
                   tau=0.5)

## compute protein and peptide abundance scores
scampiRes <-
  quantifyProteins(scampiData=dataPrepro, ccList=myCCList,
                   paramList=list(LSE=c(alphaH=0, betaH=0.2, muH=0.3,
                                        tauH=0.2)),
                   quantifyPeptides=FALSE)
```

---

runScampi *SCAMPI protein quantification function*

Description

Estimate a protein abundance score for each protein in the dataset, based on the input peptide abundance scores and the connectivity information between peptides and proteins. Optionally, the peptide abundances can be estimated as well to compare the predicted values with the input measurements.

Usage

```
runScampi(peptides, proteins, edgespp, rescaling = TRUE, method = "all",
          quantifyPeptides = TRUE, numIter = 10, verbose = FALSE)
```

Arguments

peptides	Data frame with peptide information. The following columns are required: pepId (unique identification number for each distinct peptide sequence, numbering from 1:n where n=number of distinct peptide sequences), pepSeq (peptide sequence, optionally including modifications and charge states), and pepQty (peptide abundance score). An additional column pepObs (peptide observability or identification score) is used if provided. Each row in the data frame describes one observed distinct peptide sequence.
proteins	Data frame with the protein information. The following columns are required: protId (unique identification number for each distinct protein sequence, numbering from (n+1):(n+m) where m=number of distinct protein sequences), protName (protein identifier or protein sequence). Each row describes a distinct protein sequence to which at least one of the observed peptides is matching.
edgespp	Data frame with two mandatory columns: pepId and protId. Each row defines an edge of the bipartite graph.
rescaling	If TRUE, the peptide abundance scores are logarithmized (log10). If this transformation has not yet been done during preprocessing, it is strongly recommended to stick to the default: rescaling=TRUE.
method	Describes which method should be used for the parameter estimation. Available: method="LSE", method="MLE" and method="all" (default).
quantifyPeptides	If TRUE (default) do also re-quantify the peptides and assess the peptide abundance scores.
numIter	Only used with method="MLE", see details. Default: numIter=10.
verbose	If TRUE, detailed output is provided.

Details

To use method="MLE" the inverses of the covariance matrices (of the connected components) are needed. Depending on the chosen parameters, this can lead to stability issues. To avoid the function from crashing, a try(...) bolck is used: the parameter estimation is performed until it was successful numIter times. Among these numIter sets, the one with the lowest negative log-likelihood value is returned.

Value

An object of class scampiVal containing estimates for the model parameters, protein abundances and, optionally, for the peptide abundances/residuals.

Author(s)

Sarah Gerster <sarah.gerster@isb-sib.ch>

See Also

Function `iterateScampi` tuns the model iteratively, by removing outlying peptides in each step.

Examples

```
data("leptoSRM")
scampiOut <- runScampi(peptides=leptoSRMpeptides,
                       proteins=leptoSRMproteins,
                       edgespp=leptoSRMedgespp,
                       rescaling=FALSE, method="LSE")
```

---

scampi-class *Class* "scampi"

Description

This class of objects is used to group the input data for various functions in the `protiq` package. It is not needed to run the main functions such as `runScampi` and `iterateScampi` (which return objects of the class `scampiVal`). However, it is needed to run "under-functions" such as `quantifyProteins`. Objects of this class have methods for the functions summary, show and plot.

Creation of objects

Objects are created e.g. by `new("scampi", peptides, proteins, edgespp)`.

Slots

`peptides`: Object of class `"data.frame"`: dataframe summarizing information about the peptides (input data as well as (optionally) reassessed abundances).

`proteins`: Object of class `"data.frame"`: dataframe summarizing information about the proteins (input data as well as estimated protein abundances).

`edgespp`: Object of class `"data.frame"`: dataframe summarizing the information about the edges of `ppGraph`, connecting the `peptides` to the `proteins`.

Methods

plot `signature(x = "scampi", y = "missing")`: Plot the distribution of the measured peptide abundances (histogram).

show `signature(object = "scampi")`: Display basic properties of the fitted object.

summary `signature(object = "scampi")`: Display details of the fitted object.

Author(s)

Sarah Gerster <sarah.gerster@isb-sib.ch>

Examples

```
showClass("scampi")

## generate a scampi object
data("leptoSRM")
scampiData <- scampi(peptides=leptoSRMpeptides,
                     proteins=leptoSRMproteins,
                     edgespp=leptoSRMedgespp)

## use methods of class scampiVal
show(scampiData)
summary(scampiData)
plot(scampiData)
```

---

scampiVal-class *Class* "scampiVal"

Description

This class of objects is returned by the functions runScampi and iterateScampi to contain the estimated protein abundances (and optionally the peptie abundance reassessment). Objects of this class have methods for the functions summary, show and plot.

Creation of objects

Objects are typically generated by calls to the functions runScampi and iterateScampi (in latter the scampiVal class object is in a nested list).

Slots

call: Object of the class "call": the original function call.

parameters: Object of class "list": list of estimated model parameter values, each list element corresponds to a different estimation method.

ppGraph: Object of class "graphNEL": the bipartite undirected graph with peptides and proteins underlying the comuptations.

ccList: Object of class "list": list of connected components (with some pre-processed properties) of ppGraph underlying the computations.

peptides: Object of class "data.frame": dataframe summarizing information about the peptides (input data as well as (optionally) reassessed abundances).

proteins: Object of class "data.frame": dataframe summarizing information about the proteins (input data as well as estimated protein abundances).

edgespp: Object of class "data.frame": dataframe summarizing the information about the edges of ppGraph, connecting the peptides to the proteins.

Extends

Class scampi, directly.

Methods

plot `signature(x = "scampiVal", y = "missing")`: For each parameter estimation method, plot the distribution of the computed protein abundance scores (histogram). If the peptide score reassessment was performed, also display a Tukey-Anscombe and Normal Q-Q plot for the peptide residuals.

show `signature(object = "scampiVal")`: Display basic properties of the fitted object

summary `signature(object = "scampiVal")`: Display details of the fitted object

Author(s)

Sarah Gerster <sarah.gerster@isb-sib.ch>

See Also

runScampi, iterateScampi

Examples

```
showClass("scampiVal")

## generate a scampiVal object
data("leptoSRM")
scampiRes <- runScampi(peptides=leptoSRMpeptides,
                       proteins=leptoSRMproteins,
                       edgespp=leptoSRMedgespp,
                       rescaling=FALSE, method="LSE",
                       quantifyPeptides=FALSE)

## use methods of class scampiVal
show(scampiRes)
summary(scampiRes)
plot(scampiRes)
```

Index