Package 'synapter'

December 6, 2024

Type Package

Title Label-free data analysis pipeline for optimal identification and quantitation

Version 2.30.0

Author Laurent Gatto, Nick J. Bond, Pavel V. Shliaha and Sebastian Gibb.

Maintainer Laurent Gatto <laurent.gatto@uclouvain.be> Sebastian Gibb <mail@sebastiangibb.de>

Depends R (>= 3.1.0), methods, MSnbase (>= 2.1.2)

Imports RColorBrewer, lattice, qvalue, multtest, utils, tools, Biobase, Biostrings, cleaver (>= 1.3.3), readr (>= 0.2), rmarkdown (>= 1.0)

Suggests synapterdata (>= 1.13.2), xtable, testthat (>= 0.8), BRAIN, BiocStyle, knitr

Description The synapter package provides functionality to reanalyse label-free proteomics data acquired on a Synapt G2 mass spectrometer. One or several runs, possibly processed with additional ion mobility separation to increase identification accuracy can be combined to other quantitation files to maximise identification and quantitation accuracy.

License GPL-2

URL https://lgatto.github.io/synapter/

VignetteBuilder knitr

biocViews ImmunoOncology, MassSpectrometry, Proteomics, QualityControl

RoxygenNote 6.1.0

git_url https://git.bioconductor.org/packages/synapter

git_branch RELEASE_3_20

git_last_commit 8e5133c

git_last_commit_date 2024-10-29

Repository Bioconductor 3.20

Date/Publication 2024-12-05

Contents

`synapter-package` *Combine label-free data for optimal identification and quantitation*

Description

The synapter package provides functionality to reanalyse label-free proteomics data acquired on a Synapt G2 mass spectrometer. One or several runs, possibly processed with additional ion mobility separation to increase identification accuracy can be combined to other quantitation files to maximise identification and quantitation accuracy.

Details

Two pipelines of variying flexibility are proposed the preform data analysis: (1) the `synergise1` and `synergise2` function are single entry functions for a complete analysis and (2) low level, step-by-step data processing can be achieved as described in `?Synapter`.

A high-level overview of the package and how to operate it can be found in the vignette, accessinble with `synapterGuide()`. Detailed information about the data processing can be found in the respective function and class manual pages appropriately referenced in the vignette.

For questions, use the Biocondcutor mailing list or contact the author. The vignette has a section with details on where/how to get help.

Author(s)

Laurent Gatto, Pavel V. Shliaha, Nick J. Bond and Sebastian Gibb

Maintainer: Laurent Gatto <lg390@cam.ac.uk> and Sebastian Gibb <mail@sebastiangibb.de>

References

Improving qualitative and quantitative performance for MSE-based label free proteomics, N.J. Bond, P.V. Shliaha, K.S. Lilley and L. Gatto, J Proteome Res. 2013 Jun 7;12(6):2340-53. doi: 10.1021/pr300776t. PubMed PMID: 23510225.

The Effects of Travelling Wave Ion Mobility Separation on Data Independent Acquisition in Proteomics Studies, P.V. Shliaha, N.J. Bond, L. Gatto and K.S. Lilley, J Proteome Res. 2013 Jun 7;12(6):2323-39. doi: 10.1021/pr300775k. PMID: 23514362.

`createUniquePeptideDbRds` *Create an RDS file for the 'Unique Peptides Database'*

Description

This function creates an RDS file to store the "Unique Peptides Database" for future runs of `synergise` or `Synapter`.

Usage

```
createUniquePeptideDbRds(fastaFile, outputFile = paste0(fastaFile, ".rds"),
  missedCleavages = 0, IisL = FALSE, verbose = interactive())
```

Arguments

`fastaFile`	file path of the input fasta file
`outputFile`	file path of the target RDS file; must have the file extension ".rds"
`missedCleavages`	Number of maximal allowed missed cleavages. Default is 0.
`IisL`	If `TRUE` Isoleucin and Leucin are treated as identical. In this case sequences like "ABCI", "ABCL" are removed because they are not unqiue. If `FALSE` (default) "ABCI" and "ABCL" are reported as unique.
`verbose`	If `TRUE` a verbose output is provied.

Author(s)

Sebastian Gibb <mail@sebastiangibb.de>

See Also

`Synapter` for details about the cleavage procedure.

Examples

```
## Not run:
createUniquePeptideDbRds("uniprot.fasta", "uniprot.fasta.rds")

## End(Not run)
```

`Deprecated` *synapter Deprecated and Defunct*

Description

The function, class, or data object you have asked for have been deprecated or made defunct. Deprecated functions are provided for compatibility with older versions of the `synapter` package only, and will be defunct at the next release. Defunct functions have been removed.

Deprecated:

Defunct: `synapterGUI`; use the replacement `synergise`

`estimateMasterFdr` *Computes FDR for all possible final peptide combinations*

Description

This function takes all possible combination of `pepfiles` of length greater or equal than 2 and computes the number of estimated incorrect peptides, the number of unique peptides, the number of unique protetypic peptides and the false discovery rate after merging for each combination. The best combination has an fdr lower than `masterFdr` and the highest number of unique (proteotypic) peptides.

Usage

```
estimateMasterFdr(pepfiles, fastafile, masterFdr = 0.025, fdr = 0.01,
  proteotypic = TRUE, missedCleavages = 0, IisL = FALSE,
  maxFileComb = length(pepfiles), verbose = interactive())
```

Arguments

`pepfiles`	A `list` of `vector` of final peptide filenames.
`fastafile`	A `character` with the fasta filename.
`masterFdr`	A `numeric` indicating the maximum merged false discovery to be allowed.
`fdr`	Peptide FDR level for individual peptide files filtering.
`proteotypic`	Logical. Should number proteotypic peptides be used to choose best combination and plot results or total number of unique peptides.
`missedCleavages`	Number of maximal missed cleavage sites. Default is 0.
`IisL`	If `TRUE` Isoleucin and Leucin are treated as identical. In this case sequences like "ABCI", "ABCL" are removed because they are not unqiue. If `FALSE` (default) "ABCI" and "ABCL" are reported as unique.
`maxFileComb`	A `numeric` to limit the accepted file combinations to reduce computation time. Default is `length(pepfiles)` meaning no limit set.
`verbose`	Should progress messages be printed?

Details

The false discovery rate for the master (merged) file is calcualted by summing the number of estimated false discoveries for each individual final peptide file (number of unique peptides in that file multiplied by `fdr`) divided by the total number of unique peptides for that specific combination.

The function returns an instance of the class "`MasterFdrResults`".

Value

An instance of class "`MasterFdrResults`". See details above.

Author(s)

Laurent Gatto

References

Bond N. J., Shliaha P.V., Lilley K.S. and Gatto L., (2013) J. Prot. Research.

See Also

The `makeMaster` function to combine the peptide data as suggested by `estimateMasterFdr` into one single *master* peptide file.

The vignette, accessible with `synapterGuide()` illustrates a complete pipeline using `estimateMasterFdr` and `makeMaster`.

`inspectPeptideScores` *Inspect peptide scores.*

Description

This function takes a final peptide file and returns information about the *unique* peptide scores and their number in each peptide matchType (PepFrag1 and PepFrag2) by protein dataBaseType (Random and Regular) category. The function is a lightweight verion of `getPepNumbers` and `plotPepScores` (to be used with `Synapter` instances) for individual files.

Usage

```
inspectPeptideScores(filename)
```

Arguments

`filename`	The name of a final peptide file.

Value

A table of peptide counts in each peptide matchType * protein dataBasteType category. Also plots the distribution of respective peptide scores.

Author(s)

Laurent Gatto

`makeMaster` *Merges final peptide files*

Description

This function combines a list of peptide final peptide files into one single *master* file that is obtained by merging the unique peptides from the filtered original peptide files.
Additionally it can combine multiple final fragment files into a fragment library.

Usage

```
makeMaster(pepfiles, fragmentfiles, fdr = 0.01, method = c("BH",
  "Bonferroni", "qval"), span.rt = 0.05, span.int = 0.05,
  maxDeltaRt = Inf, removeNeutralLoss = TRUE, removePrecursor = TRUE,
  tolerance = 2.5e-05, verbose = interactive())
```

Arguments

`pepfiles`	A `character` vector of final peptide file names to be merged.
`fragmentfiles`	A `character` vector of final fragment file names to be combined into an fragment library. These files should be from the same runs as the final peptide files used in `pepfiles`.
`fdr`	A `numeric` indicating the peptide false discovery rate limit.
`method`	A `character` indicating the p-value adjustment to be used. One of `BH` (default), `Bonferroni` or `qval`.
`span.rt`	A `numeric` with the loess span parameter value to be used for retention time modelling.
`span.int`	A `numeric` with the loess span parameter value to be used for intensity modelling.
`maxDeltaRt`	A `double` value that sets a maximum limit for the retention time deviaton between master and slave run to be included
`removeNeutralLoss`	A `logical`, if `TRUE` peptides with neutral loss are removed from the fragment library.
`removePrecursor`	A `logical`, if `TRUE` precursor ions are removed from the fragment spectra.
`tolerance`	A `double` value that determines the tolerance used to look for the precursor ions.
`verbose`	A `logical` indicating if information should be printed out.

Details

The merging process is as follows:

1. Each individual peptide final peptide file is filtered to retain (i) non-duplicated unique tryptic peptides, (ii) peptides with a false discovery rate <= `fdr` and (iii) proteins with a false positive rate <= `fpr`.
2. The filtered peptide files are ordered (1) according to their total number of peptides (for example [P1, P2, P3]) and (2) as before with the first item is positioned last ([P2, P3, P1] in the previous example). The peptide data are then combined in pairs in these respective orders. The first one is called the *master* file.

3. For each (master, slave) pair, the slave peptide file retention times are modelled according to the (original) master's retention times and slave peptides, not yet present in the master file are added to the master file.
4. The final *master* datasets, containing their own peptides and the respective slave specific retention time adjusted peptides are returned as a `MasterPeptides` instance.

The resulting `MasterPeptides` instance can be further used for a complete master vs. peptides/Pep3D analysis, as described in `Synapter`, `synergise` or using the GUI (`synapterGUI`). To do so, it must be serialised (using the `saveRDS` function) with a `.rds` file extension, to be recognised (and loaded) as a `R` object.

When several quantitation (or identification) files are combined as a master set to be mapped back against the inidividual final peptide files, the second master [P2, P3, P1] is used when analysing the peptide data that was first selected in the master generation (P1 above). This is to avoid aligning two identical sets of peptides (those of P1) and thus not being able to generate a valid retention time model. This is detected automatically for the user.

The two master peptides dataframes can be exported to disk as two `csv` files with `writeMasterPeptides`. The `MasterPeptides` object returned by `makeMaster` can be saved to disk (with `save` or `saveRDS`) and later reloaded (with `load` or `readRDS`) for further analysis.

The fragment library generation works as follows:

1. Each individual final fragment file is imported and only peptides present in the *master* dataset are used.
2. The fragments are combined based on their precursor ions.
3. The intensities of identical fragments (seen in different runs) is summed and divided by the summed precursor intensity (of the same peptide in different runs).
4. Afterwards the intensities are normalized to the average precursor intensity of the different runs.
5. Finally a `MSnExp` object is created.

The fragment library dataframe can be exported to disk as `csv` file with `writeFragmentLibrary`.

Value

An instance of class `"MasterPeptides"`.

Author(s)

Laurent Gatto, Sebastian Gibb

References

Shliaha P.V., Bond N. J., Lilley K.S. and Gatto L., in prep.

See Also

See the `Synapter` class manual page for detailed information on filtering and modelling and the general algorithm implemented in the `synapter` package.

The `estimateMasterFdr` function allows to control false dicovery rate when combining several peptide files while maximising the number of identifications and suggest which combination of peptide files to use.

The vignette, accessible with `synapterGuide()` illustrates a complete pipeline using `estimateMasterFdr` and `makeMaster`.

MasterFdrResults-class

Class "MasterFdrResults"

Description

This class stored the results of the

Objects from the Class

Objects are created with the `estimateMasterFdr` function.

Slots

`all`: Object of class "`data.frame`" with results of all possible combinations.

`files`: Object of class "`character`" with file names used in combinations.

`best`: Object of class "`data.frame`" with results of the best combination.

`masterFdr`: Object of class "`numeric`" storing the best combination.

Methods

bestComb `signature(object = "MasterFdrResults")`: ...

allComb `signature(object = "MasterFdrResults")`: ...

fileNames `signature(object = "MasterFdrResults")`: ...

masterFdr `signature(object = "MasterFdrResults")`: ...

plot `signature(x = "MasterFdrResults", y = "missing")`: ...

show `signature(object = "MasterFdrResults")`: ...

Author(s)

Laurent Gatto <lg390@cam.ac.uk>

References

Improving qualitative and quantitative performance for MSE-based label free proteomics, N.J. Bond, P.V. Shliaha, K.S. Lilley and L. Gatto, Journal of Proteome Research, 2013, in press.

The Effects of Travelling Wave Ion Mobility Separation on Data Independent Acquisition in Proteomics Studies, P.V. Shliaha, N.J. Bond, L. Gatto and K.S. Lilley, Journal of Proteome Research, 2013, in press.

Examples

```
## Not run:
library(synapterdata)
loadMaster()
class(master)
master

## End(Not run)
```

MasterPeptides-class *Class* "MasterPeptides"

Description

A class to store the results of makeMaster. This class stored the 2 versions (orders) of the *master* final peptide data.

Objects from the Class

Objects can be created by calls of the form makeMaster.

Slots

masters: Object of class "list" storing the 2 master data.frame objects.

pepfiles: Object of class "character" with the list of final peptide input files.

fdr: Object of class "numeric" with the peptide false discovery applied when creating the filter.

method: Object of class "character" with the peptide p-value adjustment method. One of BH (default), qval or Bonferroni.

orders: Object of class "list" with the numeric vectors specifying the order of pepfiles used to generate the respective masters data.frames.

fragmentfiles: Object of class "character" with the list of final fragment input files.

fragments: Object of class data.frame storing the combined final fragment data.

fragmentlibrary: Object of class "MSnExp" storing the fragment library.

Methods

show signature(object = "MasterPeptides"): to print a textual representation of the instance.

Author(s)

Laurent Gatto

References

Improving qualitative and quantitative performance for MSE-based label free proteomics, N.J. Bond, P.V. Shliaha, K.S. Lilley and L. Gatto, Journal of Proteome Research, 2013, in press.

The Effects of Travelling Wave Ion Mobility Separation on Data Independent Acquisition in Proteomics Studies, P.V. Shliaha, N.J. Bond, L. Gatto and K.S. Lilley, Journal of Proteome Research, 2013, in press.

plotFragmentMatching *Plot fragment matching results.*

Description

This method plots the results of the fragment matching procedure (fragmentMatching). A single plot contains two panels. The upper panel shows the identification fragments and the lower one the MS2 spectrum of the quantitation run. Common peaks are drawn in a slightly darker colour and with a point on the top.

Arguments

object	Object of class "Synapter" .
key	character, value to look for.
column	character, name of the column in which plotFragmentMatching looks for key.
...	Further arguments passed to internal functions.

Methods

signature(object = "Synapter", key = "character", column = "character", verbose = "logical", ...)
: Plots identification fragments against quantitation spectra. The ... arguments are passed to the internal functions. Currently legend.cex, fragment.cex, and most of the plot.default arguments (like xlim, ylim, main, xlab, ylab, ...) are supported. legend.cex and fragment.cex control the size of the legend and fragments labels (default: 0.5). Please see par for details about cex. If verbose = TRUE a progress bar is shown.

Author(s)

Sebastian Gibb <mail@sebastiangibb.de>

See Also

fragmentMatching

---

requantify,MSnSet-method
Intensity requantification

Description

This method tries to remove saturation effects from the intensity counts.

Usage

```
## S4 method for signature 'MSnSet'
requantify(object, saturationThreshold,
method=c("sum", "reference", "th.mean", "th.median",
"th.weighted.mean"), ...)
```

Arguments

object	An MSnSet object.
saturationThreshold	double, intensity of an ion (isotope of a given charge state) at which saturation is starting to occur.
method	character, requantification method, please see details section.
...	further arguments passed to internal functions. Currently onlyCommonIsotopes for method="sum" and requantifyAll for method=c("th.mean", "th.median", "th.weighted.mean") are supported.

Details

Currently requantify supports 3 (5) different requantification methods.

"sum" is the simplest requantification method. All ions of a peptide below the saturation threshold are summed to get the new intensity. This method accept an additional argument, namely onlyCommonIsotopes If onlyCommonIsotopes=TRUE (default) all ions that are not seen in all runs are removed and only the common seen ions are summed. In contrast onlyCommonIsotopes=FALSE sums all ions regardless they are present in all runs.

In "reference" the run that has the most unsaturated ions in common with all the other runs. If there are more than one run, the most intense is used as reference. The other runs are corrected as follows:

- Find common ions between current and the reference run.
- Divide the intensities of the common ions and calculate the mean of these quotients as a run specific scaling factor.
- Multiply the unsaturated ions of the current run by the scaling factor and replace the saturated ones by the product of the scaling factor and the intensities of their corresponding ions in the reference run.
- Sum the rescaled ion intensities.

The "th.*" methods are nearly identical. All of them calculate the theoretical isotopic distribution for the given sequence of the peptide. Subsequently the unsaturated ions are divided by their theoretical proportion and the mean/median/weighted.mean (proportions are used as weights) of these intensities are calculated per charge state. The sum of the charge state values is used as requantified intensity for this peptide.
If requantifyAll=FALSE (default) just peptides with at least one saturated ion are requantified (unsaturated peptides are unaffected). If requantify=TRUE all peptides even these where all ions are below saturationThreshold are requantified by their theoretical distribution.

Value

MSnSet where the assayData are requantified.

Author(s)

Sebastian Gibb <mail@sebastiangibb.de> and Pavel Shliaha

References

See discussion on github: https://github.com/lgatto/synapter/issues/39

See Also

MSnSet documentation: MSnSet

`rescaleForTop3,MSnSet,MSnSet-method`
Rescale for TOP3

Description

This method rescales the intensity values of an MSnSet object to be suiteable for TOP3 quantification.

Usage

```
## S4 method for signature 'MSnSet,MSnSet'
rescaleForTop3(before, after, saturationThreshold,
onlyForSaturatedRuns=TRUE, ...)
```

Arguments

before	An MSnSet object before requantification.
after	The same MSnSet object as before but after requantification.
saturationThreshold	double, intensity of an ion (isotope of a given charge state) at which saturation is starting to occur.
onlyForSaturatedRuns	logical, rescale just runs where at least one isotope is affected by saturation.
...	further arguments passed to internal functions. Currently ignored.

Details

If an MSnSet object was requantified using the method="sum" requantification method (see requantify,MSnSet-method TOP3 is not valid anymore because the most abundant proteins are penalised by removing high intensity isotopes.

To overcome this rescaleForTop3 takes the proportion isotope/sum(isotopes for each requantified peptide and calculates a correction factor by comparing these proportions against the unsaturated isotopes before requantification. The new rescale intensity values of the isotopes are the mean correction factor multiplied with the corrected intensity values (see https://github.com/lgatto/synapter/issues/39#issuecomment-207987278 for the complete explanation/discussion).

Value

MSnSet where the assayData are requantified.

Author(s)

Sebastian Gibb <mail@sebastiangibb.de> and Pavel Shliaha

References

See discussion on github: https://github.com/lgatto/synapter/issues/39

See Also

MSnSet documentation: MSnSet https://github.com/lgatto/synapter/issues/39#issuecomment-207987278

Synapter *Class "Synapter"*

Description

A reference class to store, manage and process Synapt G2 data to combine identification and quantitation results.

The data, intermediate and final results are stored together in such a ad-how container called a class. In the frame of the analysis of a set of 3 or 5 data files, namely as identification peptide, a quantitation peptide and a quantitation Pep3D, and identification fragments and quantitation spectra, such a container is created and populated, updated according to the user's instructions and used to display and export results.

The functionality of the synapter package implemented in the Synapter class in described in the *Details* section below. Documentation for the individual methods is provided in the *Methods* section. Finally, a complete example of an analysis is provided in the *Examples* section, at the end of this document.

See also papers by Shliaha et al. for details about ion mobility separation and the manuscript describing the synapter methodology.

Usage

```
Synapter(filenames, master) ## creates an instance of class 'Synapter'
```

Arguments

filenames	A named list of file names to be load. The names must be 'identpeptide', 'quantpeptide', 'quantpep3d' and 'fasta' (could be an RDS file created by link{createUniquePepti and (optional 'identfragments' and 'quantspectra'). identpeptide can be a csv final peptide file (from PLGS) or a saved "MasterPeptides" data object as created by makeMaster if working with *master* peptide data. To serialise the "MasterPeptides" instance, use the saveRDS function, and file extenstion rds. This master file could contain a fragment library as well. In this case the 'identfragments' argument would be ignored.
master	A logical that defines if the identification file is a *master* file. See makeMaster for details about this strategy.

Details

A Synapter object logs every operation that is applied to it. When displayed with show or when the name of the instance is typed at the R console, the original input file names, all operations and resulting the size of the respective data are displayed. This allows the user to trace the effect of respective operations.

Loading the data: The construction of the data and analysis container, technically defined as an instance or object of class Synapter, is created with the Synapter constructor. This function requires 4 or 6 files as input, namely, the identification final peptide csv file, the quantitation final peptide csv file, the quantitation Pep3D csv file (as exported from the PLGS software), the fasta file use for peptide identification, and optional the identification fragments csv

file and the quantitation spectra xml file. The fasta file ('fasta') could be an RDS file generated by `link{createUniquePeptideDbRds}`, too. The file names need to be specified as a named list with names 'identpeptide', 'quantpeptide', 'quantpep3d', 'fasta', 'identfragments' and 'quantspectra' respectively. These files are read and the data is stored in the newly created `Synapter` instance.

The final peptide files are filtered to retain peptides with `matchType` corresponding to `PepFrag1` and `PepFrag2`, corresponding to unmodified round 1 and 2 peptide identification. Other types, like `NeutralLoss_NH3`, `NeutralLoss_H20`, `InSource`, `MissedCleavage` or `VarMod` are not considered in the rest of the analysis. The quantitation Pep3D data is filtered to retain `Function` equal to 1 and unique quantitation spectrum ids, i.e. unique entries for multiple charge states or isotopes of an EMRT (exact mass-retention time features).

Then, p-values for `Regular` peptides are computed based on the `Regular` and `Random` database types score distributions, as described in *Käll et al., 2008a*. Only unique peptide sequences are taken into account: in case of duplicated peptides, only one entry is kept. Empirical p-values are adjusted using *Bonferroni* and *Benjamini and Hochberg, 1995* (`multtest` package) and q-values are computed using the `qvalue` package (*Storey JD and Tibshirani R., 2003 and Käll et al., 2008b*). Only `Regular` entries are stored in the resulting data for subsequent analysis.

The data tables can be exported as `csv` spreadsheets with the `writeIdentPeptides` and `writeQuantPeptides` methods.

Filtering identification and quantitation peptide: The first step of the analysis aims to match reliable peptide. The final peptide datasets are filtered based on the FDR (BH is default) using the `filterQuantPepScore` and `filterIdentPepScore` methods. Several plots are provided to illustrate peptide score densities (from which p-values are estimated, `plotPepScores`; use `getPepNumbers` to see how many peptides were available) and q-values (`plotFdr`).

Peptides matching to multiple proteins in the fasta file (non-unique tryptic identification and quantitation peptides) can be discarded with the `filterUniqueDbPeptides` method. One can also filter on the peptide length using `filterPeptideLength`.

Another filtering criterion is mass accuracy. Error tolerance quantiles (in ppm, parts per million) can be visualised with the `plotPpmError` method. The values can be retrieved with `getPpmErrorQs`. Filtering is then done separately for identification and quantitation peptide data using `filterIdentPpmError` and `filterQuantPpmError` respectively. The previous plotting functions can be used again to visualise the resulting distribution.

Filtering can also be performed at the level of protein false positive rate, as computed by the PLGS application (`protein.falsePositiveRate` column), which counts the percentage of decoy proteins that have been identified prior to the regular protein of interest. This can be done with the `filterIdentProtFpr` and `filterQuantProtFpr` methods. Note that this field is erroneously called a false positive rate in the PLGS software and the associated manuscript; it is a false discovery rate.

Merging identification and quantitation peptides: Common and reliable identification and quantitation peptides are then matched based on their sequences and merged using the `mergePeptides` method.

Retention time modelling: Systematic differences between identification features and quantitation features retention times are modelled by fitting a local regression (see the `loess` function for details), using the `modelRt` method. The smoothing parameter, or number of neighbour data points used the for local fit, is controlled by the `span` parameter that can be set in the above method.

The effect of this parameter can be observed with the `plotRt` method, specifying `what = "data"` as parameters. The resulting model can then be visualised with the above method specifying `what = "model"`, specifying up to 3 number of standard deviations to plot. A histogram of retention time differences can be produced with the `plotRtDiffs` method.

To visualise the feature space `plotFeatures` could be used. It generates one or two (if ion mobility is available) plots of retention time vs mass and mass vs ion mobility for each data source, namely, Identification data, Quantitation data and Quantitation Pep3D data.

Intensity modelling: Systematic differences between intensities of identification features and quantitation features depending on retention times are modelled by fitting a local regression (see the `loess` function for details), using the `modelIntensity` method. The smoothing parameter, or number of neighbour data points used the for local fit, is controlled by the `span` parameter that can be set in the above method.

The effect of this parameter can be observed with the `plotIntensity` method, specifying `what = "data"` as parameters. The resulting model can then be visualised with the above method specifying `what = "model"`, specifying up to 3 number of standard deviations to plot.

Grid search to optimise matching tolerances: Matching of identification peptides and quantitation EMRTs is done within a mass tolerance in parts per million (ppm) and the modelled retention time +/- a certain number of standard deviations. To help in the choice of these two parameters, a grid search over a set of possible values is performed and performance metrics are recorded, to guide in the selection of a 'best' pair of parameters.

The following metrics are computed: (1) the percentage of identification peptides that matched a single quantitation EMRT (called `prcntTotal`), (2) the percentage of identification peptides used in the retention time model that matched the quantitation EMRT corresponding to the correct quantitation peptide in ident/quant pair of the model (called `prcntModel`) and (3) the detailed about the matching of the features used for modelling (accessible with `getGridDetails`) and the corresponding `details` grid that reports the percentage of correct unique assignments. The detailed grid results specify the number of non matched identification peptides (0), the number of correctly (1) or wrongly (-1) uniquely matched identification peptides, the number of identification peptides that matched 2 or more peptides including (2+) or excluding (2-) the correct quantitation equivalent are also available.

See the next section for additional details about how matching. The search is performed with the `searchGrid` method, possibly on a subset of the data (see Methods and Examples sections for further details).

The parameters used for matching can be set manually with `setPpmError`, `setRtNsd`, `setImDiff` respectively, or using `setBestGridParams` to apply best parameters as defined using the grid search. See example and method documentation for details.

Identification transfer: matching identification peptides and quantitation EMRTs: The identification peptide - quantitation EMRT matching, termed identification transfer, is performed using the best parameters, as defined above with a grid search, or using user-defined parameters.

Matching is considered successful when one and only one EMRT is found in the mass tolerance/retention time/ion mobility window defined by the error ppm, number of retention time standard deviations, and ion mobility difference parameters. The values of uniquely matched EMRTs are reported in the final *matched* dataframe that can be exported (see below). If however, none or more than one EMRTs are matched, 0 or the number of matches are reported.

As identification peptides are serially individually matched to 'close' EMRTs, it is possible for peptides to be matched the same EMRT independently. Such cases are reported as -1 in the results dataframes.

The results can be assess using the `plotEMRTtable` (or `getEMRTtable` to retrieve the values) and `performace` methods. The former shows the number of identification peptides assigned to none (0), exactly 1 (1) or more (> 2) EMRTs. The latter method reports matched identification peptides, the number of (q-value and protein FPR filtered) identification and quantitation peptides. Matched EMRT and quantitation peptide numbers are then compared calculating the synapter enrichment (100 * (synapter - quant) / quant) and Venn counts.

Remove Less Intense Peaks: As an additional step it is possible to remove less intense peaks from the spectra and fragment data. Use `plotCumulativeNumberOfFragments` to plot the number of fragments vs the intensity and to find a good threshold. The `filterFragments` method could remove peaks if the intensity is below a specified threshold via the `minIntensity` argument. Set the `maxNumber` argument to keep only the `maxNumber` highest peaks/fragments. The `what` argument controls the data on which the filter is applied. Use `what = "fragments.ident"` for the identification fragments and `what = "spectra.quant"` for the quantiation spectra data.

Fragment Matching: After importing fragment and spectra data it is possible to match peaks between the identification fragments and the quantitation spectra using the `fragmentMatching` method. Use `setFragmentMatchingPpmTolerance` to set the maximal allowed tolerance for considering a peak as identical. There are two different methods to visualise the results of the fragment matching procedure. `plotFragmentMatching` plots the fragments and spectra for each considered pair. `plotFragmentMatchingPerformance` draws two plots. On the left panel you could see the performance of different thresholds for the number of common peaks for unique matches. The right panel visualizes the performance of different differences (delta) of common peaks between the best match (highest number of common peaks) and the second best match in each non unique match group. `plotFragmentMatchingPerformance` returns the corresponding values invisible or use `fragmentMatchingPerformance` to access these data. Use `filterUniqueMatches` and `filterNonUniqueMatches` to remove unique or non unique matches below the threshold of common peaks respective the difference in common peaks from the MatchedEMRTs data.frame.

Exporting and saving data: The merged identification and quantitation peptides can be exported to csv using the `writeMergedPeptides` method. Similarly, the matched identification peptides and quantitation EMRTs are exported with `writeMatchedEMRTs`.

Complete `Synapter` instances can be serialised with `save`, as any R object, and reloaded with `load` for further analysis.

It is possible to get the fragment and spectra data from the identification and quantitation run using `getIdentificationFragments` respectively `getQuantitationSpectra`.

Methods

Analysis methods:

mergePeptides `signature(object = "Synapter")`: Merges quantitation and identification final peptide data, used to perform retention time modelling (see `modelRt` below).

modelRt `signature(object = "Synapter", span = "numeric")`: Performs local polynomial regression fitting (see `loess`) retention time alignment using `span` parameter value to control the degree of smoothing.

modelIntensity `signature(object = "Synapter", span = "numeric")`: Performs local polynomial regression fitting (see `loess`) intensity values using `span` parameter value to control the degree of smoothing.

findEMRTs `signature(object = "Synapter", ppm = "numeric", nsd = "numeric", imdiff = "numeric")`: Finds EMRTs matching identification peptides using `ppm` mass tolerance, `nsd` number of retention time standard deviations and `imdiff` difference in ion mobility. The last three parameters are optional if previously set with `setPpmError`, `setRtNsd`, `setImDiff`, or, better, `setBestGridParams` (see below).

rescueEMRTs `signature(object = "Synapter", method = c("rescue", "copy"))`: The `method` parameter defined the behaviour for those high confidence features that where identified in both identification and quantitation acquisitions and used for the retention time model (see `mergePeptides`). Prior to version 1.1.1, these features were transferred from the quantitation pep3d file if unique matches were found, as any feature (`"transfer"`). As a result, those matching 0 or > 1 EMRTs were quantified as `NA`. The default is now to `"rescue"` the

quantitation values of these by directly retrieving the data from the quantification peptide data. Alternatively, the quantitation values for these features can be directly taken from the quantitation peptide data using "copy", thus effectively bypassing identification transfer.

searchGrid signature(object="Synapter", ppms="numeric", nsds="numeric", imdiffs = "numeric", subset="numeric", n = "numeric", verbose="logical"): Performs a grid search. The grid is defined by the ppm, nsd and imdiffs numerical vectors, representing the sequence of values to be tested. Default are seq(5, 20, 2), seq(0.5, 5, 0.5), seq(0.2, 2, 0.2) respectively. To ignore ion mobility set imdiffs = Inf. subset and n allow to use a randomly chosen subset of the data for the grid search to reduce search time. subset is a numeric, between 0 and 1, describing the percentage of data to be used; n specifies the absolute number of feature to use. The default is to use all data. verbose controls whether textual output should be printed to the console. (Note, the mergedEMRTs value used in internal calls to findEMRTs is "transfer" - see findEMRTs for details).

fragmentMatching signature(object="Synapter", ppm = "numeric", verbose = "logical": Performs a fragment matching between spectra and fragment data. The ppm argument controls the tolerance that is used to consider two peaks (MZ values) as identical. If verbose is TRUE (default) a progress bar is shown.

Methods to display, access and set data:

show signature(object = "Synapter"): Display object by printing a summary to the console.

dim signature(x="Synapter"): Returns a list of dimensions for the identification peptide, quantitation peptide, merged peptides and matched features data sets.

inputFiles signature(object="Synapter"): Returns a character of length 6 with the names of the input files used as identpeptide, quantpeptide, quantpep3d, fasta, identfragments and quantspectra.

getLog signature(object="Synapter"): Returns a character of variable length with a summary of processing undergone by object.

getGrid signature(object="Synapter", digits = "numeric"): Returns a named list of length 3 with the precent of total (prcntTotal), percent of model (prcntModel) and detailed (details) grid search results. The details grid search reports the proportion of correctly assigned features (+1) to all unique assignments (+1 and -1). Values are rounded to 3 digits by default.

getGridDetails signature(object="Synapter"): Returns a list of number of ..., -2, -1, 0, +1, +2, ... results found for each of the ppm/nsd pairs tested during the grid search.

getBestGridValue signature(object="Synapter"): Returns a named numeric of length 3 with best grid values for the 3 searches. Names are prcntTotal, prcntModel and details.

getBestGridParams signature(object="Synapter"): Returns a named list of matrices (prcntTotal, prcntModel and details). Each matrix gives the ppm and nsd pairs that yielded the best grid values (see getBestGridValue above).

setBestGridParams signature(object="Synapter", what="character"): This methods set the best parameter pair, as determined by what. Possible values are auto (default), model, total and details. The 3 last ones use the (first) best parameter values as reported by getBestGridParams. auto uses the best model parameters and, if several best pairs exists, the one that maximises details is selected.

setPepScoreFdr signature(object="Synapter", fdr = "numeric"): Sets the peptide score false discovery rate (default is 0.01) threshold used by filterQuantPepScore and filterIdentPepScore.

getPepScoreFdr signature(object="Synapter"): Returns the peptide false discrovery rate threshold.

setIdentPpmError signature(object="Synapter", ppm = "numeric"): Set the identification mass tolerance to ppm (default 10).

getIdentPpmError `signature(object="Synapter")`: Returns the identification mass tolerance.

setQuantPpmError `signature(object="Synapter", ppm = "numeric")`: Set the quantitation mass tolerance to `ppm` (default 10).

getQuantPpmError `signature(object="Synapter")`: Returns the quantitation mass tolerance.

setPpmError `signature(object="Synapter", ppm = "numeric")`: Sets the identification and quantitation mass tolerance `ppm` (default is 10).

setLowessSpan `signature(object="Synapter", span = "numeric")`: Sets the `loess` span parameter; default is 0.05.

getLowessSpan `signature(object="Synapter")`: Returns the span parameter value.

setRtNsd `signature(object="Synapter", nsd = "numeric")`: Sets the retention time tolerance `nsd`, default is 2.

getRtNsd `signature(object="Synapter")`: Returns the value of the retention time tolerance `nsd`.

setImDiff `signature(object="Synapter", imdiff = "numeric")`: Sets the ion mobility tolerance `imdiff`, default is 0.5.

getImDiff `signature(object="Synapter")`: Returns the value of the ion mobility tolerance `imdiff`.

getPpmErrorQs `signature(object="Synapter", qs = "numeric", digits = "numeric")`: Returns the mass tolerance `qs` quantiles (default is `c(0.25, 0.5, 0.75, seq(0.9, 1, 0.01))`) for the identification and quantitation peptides. Default is 3 `digits`.

getRtQs `signature(object="Synapter", qs = "numeric", digits = "numeric")`: Returns the retention time tolerance `qs` quantiles (default is `c(0.25, 0.5, 0.75, seq(0.9, 1, 0.01))`) for the identification and quantitation peptides. Default is 3 `digits`.

getPepNumbers `signature(object="Synapter")`: Returns the number of regular and random quantitation and identification peptide considered for p-value calculation and used to plot the score densities (see `plotPepScores`). Especially the difference between random and regular entries are informative in respect with the confidence of the random scores distribution.

setFragmentMatchingPpmTolerance `signature(object="Synapter", ppm = "numeric")`: Sets the fragment matching mass tolerance `ppm` (default is 25).

getFragmentMatchingPpmTolerance `signature(object="Synapter")`: Returns the fragment matching mass tolerance in ppm.

showFdrStats `signature(object="Synapter", k = "numeric")`: Returns a named `list` of length 2 with the proportion of identification and quantitation peptides that are considered significant with a threshold of `k` (default is `c(0.001, 0.01, 0.5, 0.1)`) using raw and adjusted p-values/q-values.

getEMRTtable `signature(object="Synapter")`: Returns a `table` with the number of 0, 1, 2, ... assigned EMRTs.

performance `signatute(object="Synapter", verbose = TRUE)`: Returns (and displays, if `verbose`) the performance of the synapter analysis.

performance2 `signatute(object="Synapter", verbose = TRUE)`: Returns (and displays, if `verbose`) information about number of missing values and identification source of transfered EMRTs.

fragmentMatchingPerformance `signature(object="Synapter", what = c("unique", "non-unique")`: Returns the performance of the fragment matching for `unqiue` or `non-unique` matches. The return valus is a `matrix` with seven columns. The first column `ncommon/deltacommon` contains the thresholds. Column 2 to 5 are the true positives `tp`, false positives `fp`, true negatives `tn`, false negatives `fn` for the merged peptide data. The sixth column `all` shows

the corresponding number of peptides for all peptides (not just the merged ones) and the last column shows the FDR `fdr` for the current threshold (in that row) for the merged data. Please note that the FDR is overfitted/underestimated because the merged peptides are the peptides from the highest quality spectra were PLGS could easily identify the peptides. The peptides that are not present in the merged data are often of lower quality hence the FDR would be higher by trend.
See `plotFragmentMatchingPerformance` for a graphical representation.

Filters:

filterUniqueDbPeptides `signature(object="Synapter", missedCleavages = 0, IisL = TRUE, verbose = TRUE)`: This method first digests the fasta database file and keeps unique tryptic peptides. (NOTE: since version 1.5.3, the tryptic digestion uses the `cleaver` package, replacing the more simplistic inbuild function. The effect of this change is documented in https://github.com/lgatto/synapter/pull/47). The number of maximal missed cleavages can be set as `missedCleavages` (default is 0). If `IisL = TRUE` Isoleucin and Leucin are treated as the same aminoacid. In this case sequences like "ABCI", "ABCL" are removed because they are not unqiue anymore. If `IisL = FALSE` (default) "ABCI" and "ABCL" are reported as unique. The peptide sequences are then used as a filter against the identification and quantitation peptides, where only unique proteotyptic instances (no miscleavage allowed by default) are eventually kept in the `object` instance. This method also removes any additional duplicated peptides, that would not match any peptides identified in the fasta database.

filterUniqueQuantDbPeptides `signature(object="Synapter", missedCleavages = 0, IisL = TRUE, verbose = TRUE)`: As `filterUniqueDbPeptides` for quantitation peptides only.

filterUniqueIdentDbPeptides `signature(object="Synapter", missedCleavages = 0, IisL = TRUE, verbose = TRUE)`: As `filterUniqueDbPeptides` for identification peptides only.

filterQuantPepScore `signature(object="Synapter", fdr = "numeric", method = "character")`: Filters the quantitation peptides using `fdr` false discovery rate. `fdr` is missing by default and is retrieved with `getPepScoreFdr` automatically. If not set, default value of 0.01 is used. `method` defines how to performe p-value adjustment; one of `BH`, `Bonferrone` or `qval`. See details section for more information.

filterIdentPepScore `signature(object="Synapter", fdr = "numeric", method = "charactet")`: As `filterQuantPepScore`, but for identification peptides.

filterQuantProtFpr `signature(object="Synapter", fpr = "numeric")`: Filters quantitation peptides using the protein false positive rate (erroneously defined as a FPR, should be FDR), as reported by PLGS, using threshold set by `fpr` (missing by default) or retrieved by `getProtFpr`.

filterIdentProtFpr `signature(object="Synapter", fpr = "numeric")`: as `filterQuantProtFpr`, but for identification peptides.

filterQuantPpmError `signature(object="Synapter", ppm = "numeric")`: Filters the quantitation peptides based on the mass tolerance `ppm` (default missing) provided or retrieved automatically using `getPpmError`.

filterIdentPpmError `signature(object="Synapter")`: as `filterQuantPpmError`, but for identification peptides.

filterFragments `signature(object = "Synapter", what = c("fragments.ident", "spectra.quant"), minIntensity = "numeric", maxNumber = "numeric", verbose = "logical")`: Filters the spectra/fragment data using a minimal intensity threshold (`minIntensity`) or a maximal number of peaks/fragments threshold (`maxNumber`). Please note that the maximal number is transfered to an intensity threshold and the result could contain less peaks than specified by `maxNumber`. If both arguments are given, the more aggressive one is chosen. Use the `what` argument to specify the data that should be filtered. Set `what = "fragments.ident"` for

the identification fragment data or what = "spectra.quant" for the quantiation spectra. If verbose is TRUE (default) a progress bar is shown.

filterUniqueMatches signature(object="Synapter", minNumber = "numeric"): Removes all unique matches that have less than minNumber of peaks/fragments in common. Use fragmentMatchingPerformance(..., what="unique")/plotFragmentMatchingPerformance (left panel) to find an ideal threshold.

filterNonUniqueMatches signature(object="Synapter", minDelta = "numeric"): Removes all non unique matches that have a difference between the best match (highest number of common peaks/fragments, treated as true match) and the second best match (second highest number of common peaks/fragments) less than minDelta. For the matches above the threshold only the one with the highest number of common peaks/fragments in each match group is kept. Use fragmentMatchingPerformance(..., what="non-unique")/plotFragmentMatchingPerformance (right panel) to find an ideal threshold.

filterNonUniqueIdentMatches signature(object="Synapter"): Removes all non unique identification matches. In rare circumstances (if the grid search parameters are to wide/relaxed or a fragment library is used) it could happen that the searchGrid methods matches a single quantification EMRT to multiple identification EMRTs. This methods removes all these non unique matches.

Plotting:

plotPpmError signature(object="Synapter", what = "character"): Plots the proportion of data against the mass error tolerance in ppms. Depending on what, the data for identification (what = "Ident"), quantitation (what = "Quant") or "both" is plotted.

plotRtDiffs signature(object="Synapter", ...): Plots a histogram of retention time differences after alignments. ... is passed to hist.

plotRt signature(object="Synapter", what = "character", f = "numeric", nsd = "numeric"): Plots the Identification - Quantitation retention time difference as a function of the Identification retention time. If what is "data", two plots are generated: one ranging the full range of retention time differences and one focusing on the highest data point density and showing models with various span parameter values, as defined by f (default is 2/3, 1/2, 1/4, 1/10, 1/16, 1/25, 1/50, passed as a numed numeric). If what is "model", a focused plot with the applied span parameter is plotted and areas of nsd (default is x(1, 3, 5) number of standard deviations are shaded around the model.

plotIntensity signature(object="Synapter", what = "character", f = "numeric", nsd = "numeric"): Plots the (log2) ratio of Identification and Quantitation intensities as a function of the Identification retention time. If what is "data", two plots are generated: one ranging the full range of ratios and one focusing on the highest data point density and showing models with various span parameter values, as defined by f (default is 2/3, 1/2, 1/4, 1/10, 1/16, 1/25, 1/50, passed as a numed numeric). If what is "model", a focused plot with the applied span parameter is plotted and areas of nsd (default is x(1, 3, 5) number of standard deviations are shaded around the model.

plotPepScores signature(object="Synapter"): Plots the distribution of random and regular peptide scores for identification and quantitation features. This reflects how peptide p-values are computed. See also getPepNumbers.

plotFdr signature(object="Synapter", method = "character"): Displays 2 plots per identification and quantitation peptides, showing the number of significant peptides as a function of the FDR cut-off and the expected false number of false positive as a number of significant tests. PepFrag 1 and 2 peptides are illustrated on the same figures. These figures are adapted from plot.qvalue. method, one of "BH", "Bonferroni" or "qval", defines what identification statistics to use.

plotEMRTtable signature(object="Synapter"): Plots the barchart of number or 0, 1, 2, ... assigned EMRTs (see getEMRTtable) .

plotGrid `signature(object="Synapter", what = "character"), maindim = "character"`: Plots a heatmap of the respective grid search results. This grid to be plotted is controlled by `what`: `"total"`, `"model"` or `"details"` are available. If ion mobility was used in the grid search you can use `maindim` to decided which dimensions should be shown. `maindim` could be one of `"im"` (default), `"rt"` and `"mz"`. If `maindim = "im"` a heatmap for each ion mobility threshold is drawn. For `maindim = "rt"` and `maindim` you get a heatmap for each retention time respective mass threshold.

plotFeatures `signature(object="Synapter", what = "character", xlim = "numeric", ylim = "numeric", ionmobiltiy = "logical")`: Plots the retention time against precursor mass space. If `what` is `"all"`, three (six if ion mobility is available and `ionmobility = TRUE` (default is `FALSE`); three additional plots with precursor mass against ion mobility) such plots are created side by side: for the identification peptides, the quantitation peptides and the quantitation Pep3D data. If `what` is `"some"`, a subset of the rt/mass space can be defined with `xlim` (default is `c(40, 60)`) and `ylim` (default is `c(1160, 1165)`) and identification peptide, quantitation peptides and EMRTs are presented on the same graph as grey dots, blue dots and red crosses respectively. In addition, rectangles based on the ppm and nsd defined tolerances (see `setPpmError` and `setNsdError`) are drawn and centered at the expected modelled retention time. This last figure allows to visualise the EMRT matching.

plotFragmentMatching `signature(object = "Synapter", key = "character", column = "character", verbose = "logical", ...)`: Plots two spectra and fragments against each other. Please see `plotFragmentMatching` for details.

plotFragmentMatchingPerformance `signature(object = "Synapter", showAllPeptides = FALSE)`: Creates two plots. The left panel shows the performance of filtering the unique matches of the merged peptides using a different number of common peaks. The right panel shows the performance of filtering the non unique matches of the merged peptides using different differences (delta) in common peaks/fragments. These differences (delta) are calculated between the match with the highest number of common peaks/fragments and the second highest one. Use `filterUniqueMatches` and `filterNonUniqueMatches` to filter the `MatchedEMRT` `data.frame` using one of these thresholds. This function returns a `list` with two named elements (`unqiue` and `nonunqiue` invisibly. These are the same data as return by `fragmentMatchingPerformance`.
Use `showAllPeptides=TRUE` to add a line for all peptides (not just the merged onces) to both plots.

plotCumulativeNumberOfFragments `signature(object = "Synapter", what = c("fragments.ident", "spectra.quant"))`: Plots the cumulative number of the fragments/peaks vs their intensity (log10 scaled). Use the `what` argument to create this plot for the identification fragments (`what = "fragments.quant"`) or the the quantitation spectra (`what = "spectra.quant"`).

Exporters:

writeMergedPeptides `signature(object="Synapter", file = "character", what = "character", ...)`: Exports the merged peptide data to a comma-separated `file` (default name is `"Res-MergedPeptides.csv"`

writeMatchedEMRTs `signature(object="Synapter", file = "character", ...)`: As above, saving the matched EMRT table.

writeIdentPeptides `signature(object="Synapter", file = "character", ...)`: As above, exporting the identification peptide data.

writeQuantPeptides `signature(object="Synapter", file = "character", ...)`: A above, exporting the quantitation peptide data.

getIdentificationFragments `signature(object="Synapter")`: returns the identification fragments as `MSnExp`.

getQuantitationSpectra `signature(object="Synapter")`: returns the quantitation spectra as `MSnExp`.

Other:

as(, "MSnSet") `signature(x = "Synapter")`: Coerce object from `Synapter` to `MSnSet` class.

validObject `signature(object = "Synapter")`: Test whether a given `Synapter` object is valid.

updateObject `signature(object = "Synapter")`: Updates an old `Synapter` object.

Author(s)

Laurent Gatto <lg390@cam.ac.uk>

References

Käll L, Storey JD, MacCoss MJ, Noble WS Posterior error probabilities and false discovery rates: two sides of the same coin. J Proteome Res. 2008a Jan; 7:(1)40-4

Bonferroni single-step adjusted p-values for strong control of the FWER.

Benjamini Y. and Hochberg Y. Controlling the false discovery rate: a practical and powerful approach to multiple testing. J. R. Statist. Soc. B., 1995, Vol. 57: 289-300.

Storey JD and Tibshirani R. Statistical significance for genome-wide experiments. Proceedings of the National Academy of Sciences, 2003, 100: 9440-9445.

Käll, Storey JD, MacCoss MJ, Noble WS Assigning significance to peptides identified by tandem mass spectrometry using decoy databases. J Proteome Res. 2008b Jan; 7:(1)29-34

Improving qualitative and quantitative performance for MSE-based label free proteomics, N.J. Bond, P.V. Shliaha, K.S. Lilley and L. Gatto, Journal of Proteome Research, 2013, in press.

The Effects of Travelling Wave Ion Mobility Separation on Data Independent Acquisition in Proteomics Studies, P.V. Shliaha, N.J. Bond, L. Gatto and K.S. Lilley, Journal of Proteome Research, 2013, in press.

Trypsin cleavage:

Glatter, Timo, et al. Large-scale quantitative assessment of different in-solution protein digestion protocols reveals superior cleavage efficiency of tandem Lys-C/trypsin proteolysis over trypsin digestion. Journal of proteome research 11.11 (2012): 5145-5156. http://dx.doi.org/10.1021/pr300273g

Rodriguez, Jesse, et al. Does trypsin cut before proline?. Journal of proteome research 7.01 (2007): 300-305. http://dx.doi.org/10.1021/pr0705035

Brownridge, Philip, and Robert J. Beynon. The importance of the digest: proteolysis and absolute quantification in proteomics. Methods 54.4 (2011): 351-360. http://dx.doi.org/10.1016/j.ymeth.2011.05.005

cleaver's rules are taken from: http://web.expasy.org/peptide_cutter/peptidecutter_enzymes.html#Tryps

Examples

```
library(synapter) ## always needed

## Not run:
## (1) Construction - to create your own data objects
synapterTiny <- Synapter()

## End(Not run)

## let's use synapterTiny, shipped with the package
synapterTinyData() ## loads/prepares the data
```

```
synapterTiny ## show object

## (2) Filtering
## (2.1) Peptide scores and FDR

## visualise/explore peptide id scores
plotPepScores(synapterTiny)
getPepNumbers(synapterTiny)

## filter data
filterUniqueDbPeptides(synapterTiny) ## keeps unique proteotypic peptides
filterPeptideLength(synapterTiny, l = 7) ## default length is 7

## visualise before FDR filtering
plotFdr(synapterTiny)

setPepScoreFdr(synapterTiny, fdr = 0.01) ## optional
filterQuantPepScore(synapterTiny, fdr = 0.01) ## specifying FDR
filterIdentPepScore(synapterTiny) ## FDR not specified, using previously set value

## (2.2) Mass tolerance
getPpmErrorQs(synapterTiny)
plotPpmError(synapterTiny, what="Ident")
plotPpmError(synapterTiny, what="Quant")

setIdentPpmError(synapterTiny, ppm = 20) ## optional
filterQuantPpmError(synapterTiny, ppm = 20)
## setQuantPpmError(synapterTiny, ppm = 20) ## set quant ppm threshold below
filterIdentPpmError(synapterTiny, ppm=20)

filterIdentProtFpr(synapterTiny, fpr = 0.01)
filterQuantProtFpr(synapterTiny, fpr = 0.01)

getPpmErrorQs(synapterTiny) ## to be compared with previous output

## (3) Merge peptide sequences
mergePeptides(synapterTiny)

## (4) Retention time modelling
plotRt(synapterTiny, what="data")
setLowessSpan(synapterTiny, 0.05)
modelRt(synapterTiny) ## the actual modelling
getRtQs(synapterTiny)
plotRtDiffs(synapterTiny)
## plotRtDiffs(synapterTiny, xlim=c(-1, 1), breaks=500) ## pass parameters to hist()
plotRt(synapterTiny, what="model") ## using default nsd 1, 3, 5
plotRt(synapterTiny, what="model", nsd=0.5) ## better focus on model

plotFeatures(synapterTiny, what="all")
setRtNsd(synapterTiny, 3)     ## RtNsd and PpmError are used for detailed plot
setPpmError(synapterTiny, 10) ## if not set manually, default values are set automatically
plotFeatures(synapterTiny, what="some", xlim=c(36,44), ylim=c(1161.4, 1161.7))
## best plotting to svg for zooming

set.seed(1) ## only for reproducibility of this example

## (5) Grid search to optimise EMRT matching parameters
```

```
searchGrid(synapterTiny,
           ppms = 7:10,  ## default values are 5, 7, ..., 20
           nsds = 1:3,   ## default values are 0.5, 1,  ..., 5
           subset = 0.2) ## default is 1
## alternatively, use 'n = 1000' to use exactly
## 1000 randomly selected features for the grid search
getGrid(synapterTiny)  ## print the grid
getGridDetails(synapterTiny)  ## grid details
plotGrid(synapterTiny, what = "total")   ## plot the grid for total matching
plotGrid(synapterTiny, what = "model")   ## plot the grid for matched modelled feature
plotGrid(synapterTiny, what = "details") ## plot the detail grid
getBestGridValue(synapterTiny)  ## return best grid values
getBestGridParams(synapterTiny) ## return parameters corresponding to best values
setBestGridParams(synapterTiny, what = "auto") ## sets RtNsd and PpmError according the grid results
## 'what' could also be "model", "total" or "details"
## setPpmError(synapterTiny, 12) ## to manually set values
## setRtNsd(synapterTiny, 2.5)

## (6) Matching ident peptides and quant EMRTs
findEMRTs(synapterTiny)
plotEMRTtable(synapterTiny)
getEMRTtable(synapterTiny)
performance(synapterTiny)
performance2(synapterTiny)

## (7) Exporting data to csv spreadsheets
writeMergedPeptides(synapterTiny)
writeMergedPeptides(synapterTiny, file = "myresults.csv")
writeMatchedEMRTs(synapterTiny)
writeMatchedEMRTs(synapterTiny, file = "myresults2.csv")
## These will export the filter peptide data
writeIdentPeptides(synapterTiny, file = "myIdentPeptides.csv")
writeQuantPeptides(synapterTiny, file = "myQuantPeptides.csv")
## If used right after loading, the non-filted data will be exported
```

`synapterGuide` *Opens the 'synapter' vignette*

Description

Opens the relevant vignette; a shortcut to using `vignette`. `synapterGuide()` gives access to the main overview vignette.

Usage

```
synapterGuide()
```

Author(s)

Laurent Gatto

`synapterPlgsAgreement,MSnSet-method`
Synapter/PLGS Agreement

Description

This method checks the agreement between synapter analysis and PLGS results.

Usage

```
## S4 method for signature 'MSnSet'
synapterPlgsAgreement(object, ...)
```

Arguments

`object`	An `MSnSet` object.
`...`	further arguments, not used yet.

Details

Each synapter object has `synapterPlgsAgreement` column in its `MatchedEMRTs` `data.frame` (see `writeMatchedEMRTs`). After converting the synapter object into an `MSnSet` instance via `as(synapterobject, "MSnSet")` this column could be find in the feature data (`fData(msnset)$synapterPlgsAgreement`). In the `synapterPlgsAgreement` each peptide is classified as:

- `"agree"`: EMRT identified in identification and quantitation run by PLGS and same EMRT matched in synapter's grid search.
- `"disagree"`: EMRT identified in identification and quantitation run by PLGS and a different EMRT was matched in synapter's grid search.
- `"no_plgs_id"`: EMRT was *not* identified in the quantitation run by PLGS but matched in synapter's grid search.
- `"no_synapter_transfer"`: EMRT was identified in the identification and quantitation run by PLGS but not matched in synapter's grid search.
- `"no_id_or_transfer"`: EMRT was *not* identified in the quantitation run by PLGS and not matched in synapter's grid search.
- `"multiple_ident_matches"`: a single quantitation EMRT was matched by synapter to multiple identification EMRTs found by PLGS (could happen if the grid search parameters are too relaxed).

After combining multiple `MSnSet` the method `synapterPlgsAgreement` adds additional columns to the feature data:

- `nIdentified`: how often a peptide was identified across multiple runs?
- `nAgree`: how often a peptide was identified by PLGS and synapter across multiple runs (counts `"agree"` entries)?
- `nDisagree`: how often a peptide was differently identified by PLGS and synapter across multiple runs (counts `"disagree"` entries)?
- `synapterPlgsAgreementRatio`: `nAgree/(nAgree + nDisagree)`.

Value

MSnSet where the columns nIdentified, nAgree, nDisagree and synapterPlgsAgreementRatio were added to the feature data.

Author(s)

Sebastian Gibb <mail@sebastiangibb.de>

References

See discussion on github: https://github.com/lgatto/synapter/issues/73

See Also

MSnSet documentation: MSnSet

---

synapterTinyData *Loads a small test data for the 'synapter' package*

Description

Instead of using data to load the synapterTiny data set, synapterTinyData will load it and initialise it for proper downstream analysis, during which the 04_test_database.fasta file, provided with the package and references inside the object needs, to be accessed. However, as the exact location can not be known in advance, the reference is updated with the file's correct local path.

This data set has been generated with the Synapter constructor. Note that the input data file sizes have been reduced by depleting many rows (peptides and EMRTs) from the original csv files.

In addition, several columns that where not necessary for processing were also removed. As such, the data stored in synapterTiny does not relect the data obtained when following the section 'Preparing the input data' in the section, without however affecting the processing and final results.

Usage

```
synapterTinyData()
```

Value

A character vector with the data set name, "synapterTiny". Used for its side effect of loading synapterTiny, an instance of class Synapter, in .GlovalEnv.

Author(s)

Laurent Gatto

Source

Improving qualitative and quantitative performance for MSE-based label free proteomics, N.J. Bond, P.V. Shliaha, K.S. Lilley and L. Gatto, Journal of Proteome Research, 2013, in press.

The Effects of Travelling Wave Ion Mobility Separation on Data Independent Acquisition in Proteomics Studies, P.V. Shliaha, N.J. Bond, L. Gatto and K.S. Lilley, Journal of Proteome Research, 2013, in press.

Examples

```
synapterTinyData()
synapterTiny
```

`synergise` *Synergise identification and quantitation results*

Description

Performs a complete default analysis on the files defined in `filenames`, creates a complete html report and saves/exports all results as `csv` and `rds` files. See details for a description of the pipeline and `Synapter` for manual execution of individual steps.

Usage

```
synergise(filenames, master = FALSE, object, outputdir,
  outputfile = paste0("synapter_report_", strftime(Sys.time(),
  "%Y%m%d-%H%M%S"), ".html"), fdr = 0.01, fdrMethod = c("BH",
  "Bonferroni", "qval"), fpr = 0.01, peplen = 7, missedCleavages = 0,
  IisL = FALSE, identppm = 20, quantppm = 20, uniquepep = TRUE,
  span = 0.05, grid.ppm.from = 2, grid.ppm.to = 20,
  grid.ppm.by = 2, grid.nsd.from = 0.5, grid.nsd.to = 5,
  grid.nsd.by = 0.5, grid.subset = 1, grid.n = 0,
  grid.param.sel = c("auto", "model", "total", "details"),
  mergedEMRTs = c("rescue", "copy", "transfer"),
  template = system.file("reports", "synergise1.Rmd", package =
  "synapter"), verbose = interactive())
```

Arguments

`filenames`	A named `list` of file names to be load. The names must be `identpeptide`, `quantpeptide`, `quantpep3d` and `fasta` (could be an RDS file created by `link{createUniquePeptic` `identpeptide` can be a `csv` final peptide file (from PLGS) or a saved `"MasterPeptides"` data object as created by `makeMaster` if working with *master* peptide data. To serialise the `"MasterPeptides"` instance, use the `saveRDS` function, and file extenstion `rds`.
`master`	A `logical` indicating if the identification final peptide files are master (see `makeMaster`) or regular files. Default is `FALSE`.
`object`	An instance of class `Synapter` that will be copied, processed and returned. If `filenames` are also provided, the latter and `object`'s `inputFiles` will be checked for equality.
`outputdir`	A `character` with the full path to an existing directory.
`outputfile`	A `character` with the file name for the report.
`fdr`	Peptide false discovery rate. Default is 0.01.
`fdrMethod`	P-value adjustment method. One of `"BH"` (default) for Benjamini and HochBerg (1995), `"Bonferroni"` for Bonferroni's single-step adjusted p-values for strong control of the FWER and `"qval"` from the `qvalue` package. See `Synapter` for references.
`fpr`	Protein false positive rate. Default is 0.01.

peplen	Minimum peptide length. Default is 7.
missedCleavages	Number of allowed missed cleavages. Default is 0.
IisL	If TRUE Isoleucin and Leucin are treated as equal. In this case sequences like "ABCI", "ABCL" are removed because they are not unqiue. If FALSE (default) "ABCI" and "ABCL" are reported as unique.
identppm	Identification mass tolerance (in ppm). Default is 20.
quantppm	Quantitation mass tolerance (in ppm). Default is 20.
uniquepep	A logical is length 1 indicating if only unique peptides in the identification and quantitation peptides as well as unique tryptic peptides as defined in the fasta file. Default is TRUE.
span	The loess span parameter. Default is 0.05.
grid.ppm.from	Mass tolerance (ppm) grid starting value. Default is 2.
grid.ppm.to	Mass tolerance (ppm) grid ending value. Default is 20.
grid.ppm.by	Mass tolerance (ppm) grid step value. Default is 2.
grid.nsd.from	Number of retention time stdev grid starting value. Default is 0.5.
grid.nsd.to	Number of retention time stdev grid ending value. Default is 5.
grid.nsd.by	Number of retention time stdev grid step value. Default is 0.5.
grid.subset	Percentage of features to be used for the grid search. Default is 1.
grid.n	Absolute number of features to be used for the grid search. Default is 0, i.e ignored.
grid.param.sel	Grid parameter selection method. One of auto (default), details, model or total. See Synapter for details on these selection methods.
mergedEMRTs	One of "rescue" (default), "copy" or "transfer". See the documentation for the findEMRTs function in Synapter for details.
template	A character full path to Rmd template.
verbose	A logical indicating if progress output should be printed to the console. Default is TRUE.

Details

Data can be input as a Synapter object if available or as a list of files (see filenames) that will be used to read the data in. The html report and result files will be created in the outputdir folder. All other input parameters have default values.

The data processing and analysis pipeline is as follows:

1. If uniquepep is set to TRUE (default), only unique proteotypic identification and quantitation peptides are retained.
2. Peptides are filtered for a FDR <= fdr (default is 0.01) using the "BH" method (see fdr and fdrMethod parameters for details).
3. Peptide with a mass tolerance > 20 ppms (see quantppm and identppm) are filtered out.
4. Peptides with a protein false positive rate (as reported by the PLGS software) > fpr are filtered out.
5. Common identification and quantitation peptides are merged and a retention time model is created using the Local Polynomial Regression Fitting (loess function for the stats package) using a default span value of 0.05.

6. A grid search to optimise the width in retention time and mass tolerance for EMRTs matching is performed. The default grid search space is from 0.5 to 5 by 0.5 retention time model standard deviations (see `grid.nsd.from`, `grid.nsd.to` and `grid.nsd.by` parameters) and from 2 to 20 by 2 parts per million (ppm) for mass tolerance (see `grid.ppm.from`, `grid.ppm.to` and `grid.ppm.by` parameters). The data can be subset using using an absolute number of features (see `grid.n`) or a fixed percentage (see `grid.subset`). The pair of optimal `nsd` and `ppm` is chosen (see `grid.param.sel` parameter).
7. The quantitation EMRTs are matched using the optimised parameters.

If a master identification file is used (`master` is set to `TRUE`, default is `FALSE`), the relevant actions that have already been executed when the file was created with `makeMaster` are not repeated here.

Value

Invisibly returns an object of class `Synapter`. Used for its side effect of creating an html report of the run in `outputdir`.

Author(s)

Laurent Gatto, Sebastian Gibb

References

Bond N. J., Shliaha P.V., Lilley K.S. and Gatto L. (2013) J. Prot. Research.

Examples

```
output <- tempdir() ## a temporary directory
synapterTinyData()
synergise(object = synapterTiny, outputdir = output, outputfile = "synapter.html",
          grid.subset = 0.2)
htmlReport <- paste0("file:///", file.path(output, "synapter.html")) ## the result report
## Not run:
browseURL(htmlReport) ## open the report with default browser

## End(Not run)
```

`synergise2` *Synergise identification and quantitation results*

Description

Performs a complete default analysis on the files defined in `filenames`, creates a complete html report and saves/exports all results as `csv` and `rds` files. See details for a description of the pipeline and `Synapter` for manual execution of individual steps.

Usage

```
synergise2(filenames, master = FALSE, object, outputdir,
  outputfile = paste0("synapter_report_", strftime(Sys.time(),
  "%Y%m%d-%H%M%S"), ".html"), fdr = 0.01, fdrMethod = c("BH",
  "Bonferroni", "qval"), fpr = 0.01, peplen = 7, missedCleavages = 2,
  IisL = FALSE, identppm = 20, quantppm = 20, uniquepep = TRUE,
  span.rt = 0.05, span.int = 0.05, grid.ppm.from = 2,
  grid.ppm.to = 20, grid.ppm.by = 2, grid.nsd.from = 0.5,
  grid.nsd.to = 5, grid.nsd.by = 0.5, grid.imdiffs.from = 0.6,
  grid.imdiffs.to = 1.6, grid.imdiffs.by = 0.2, grid.subset = 1,
  grid.n = 0, grid.param.sel = c("auto", "model", "total", "details"),
  fm.ppm = 25, fm.ident.minIntensity = 70,
  fm.quant.minIntensity = 70, fm.minCommon = 1, fm.minDelta = 1,
  fm.fdr.unique = 0.05, fm.fdr.nonunique = 0.05,
  mergedEMRTs = c("rescue", "copy", "transfer"),
  template = system.file("reports", "synergise2.Rmd", package =
  "synapter"), verbose = interactive())
```

Arguments

filenames	A named list of file names to be load. The names must be identpeptide, quantpeptide, quantpep3d and fasta (could be an RDS file created by link{createUniquePeptic If fragmentmatching should be used identfragments (could be skipped if a master RDS files is used for identpeptide) and quantspectra have to be given as well. identpeptide can be a csv final peptide file (from PLGS) or a saved "MasterPeptides" data object as created by makeMaster if working with *master* peptide data. To serialise the "MasterPeptides" instance, use the saveRDS function, and file extenstion rds.
master	A logical indicating if the identification final peptide files are master (see makeMaster) or regular files. Default is FALSE.
object	An instance of class Synapter that will be copied, processed and returned. If filenames are also provided, the latter and object's inputFiles will be checked for equality.
outputdir	A character with the full path to an existing directory.
outputfile	A character with the file name for the report.
fdr	Peptide false discovery rate. Default is 0.01.
fdrMethod	P-value adjustment method. One of "BH" (default) for Benjamini and HochBerg (1995), "Bonferroni" for Bonferroni's single-step adjusted p-values for strong control of the FWER and "qval" from the qvalue package. See Synapter for references.
fpr	Protein false positive rate. Default is 0.01.
peplen	Minimum peptide length. Default is 7.
missedCleavages	Number of allowed missed cleavages. Default is 2.
IisL	If TRUE Isoleucin and Leucin are treated as equal. In this case sequences like "ABCI", "ABCL" are removed because they are not unqiue. If FALSE (default) "ABCI" and "ABCL" are reported as unique.
identppm	Identification mass tolerance (in ppm). Default is 20.
quantppm	Quantitation mass tolerance (in ppm). Default is 20.

`uniquepep` A `logical` is length 1 indicating if only unique peptides in the identification and quantitation peptides as well as unique tryptic peptides as defined in the fasta file. Default is `TRUE`.

`span.rt` The loess span parameter for retention time correction. Default is 0.05.

`span.int` The loess span parameter for intensity correction. Default is 0.05.

`grid.ppm.from` Mass tolerance (ppm) grid starting value. Default is 2.

`grid.ppm.to` Mass tolerance (ppm) grid ending value. Default is 20.

`grid.ppm.by` Mass tolerance (ppm) grid step value. Default is 2.

`grid.nsd.from` Number of retention time stdev grid starting value. Default is 0.5.

`grid.nsd.to` Number of retention time stdev grid ending value. Default is 5.

`grid.nsd.by` Number of retention time stdev grid step value. Default is 0.5.

`grid.imdiffs.from`
Ion mobility difference grid starting value. value. Default is 0.6.

`grid.imdiffs.to`
Ion mobility difference grid ending value. Default is 1.6.

`grid.imdiffs.by`
Ion mobility difference grid step value. Default is 0.2.

`grid.subset` Percentage of features to be used for the grid search. Default is 1.

`grid.n` Absolute number of features to be used for the grid search. Default is 0, i.e ignored.

`grid.param.sel` Grid parameter selection method. One of `auto` (default), `details`, `model` or `total`. See `Synapter` for details on these selection methods.

`fm.ppm` Fragment Matching tolerance: Peaks in a range of `fm.ppm` are considered as identical. Default is 25.

`fm.ident.minIntensity`
Minimal intensity of a Identification fragment to be not filtered prior to Fragment Matching.

`fm.quant.minIntensity`
Minimal intensity of a peaks in a Quantitation spectra to be not filtered prior to Fragment Matching.

`fm.minCommon` Minimal number of peaks that unique matches need to have in common. Default 1.

`fm.minDelta` Minimal difference in number of peaks that non-unique matches need to have to be considered as true match. Default 1.

`fm.fdr.unique` Minimal FDR to select `fm.minCommon` automatically (if both values are given the more restrictive one (that filters more fragments) is used). Default 0.05.

`fm.fdr.nonunique`
Minimal FDR to select `fm.minDelta` automatically (if both values are given the more restrictive one (that filters more fragments) is used). Default 0.05.

`mergedEMRTs` One of `"rescue"` (default), `"copy"` or `"transfer"`. See the documentation for the `findEMRTs` function in `Synapter` for details.

`template` A `character` full path to Rmd template.

`verbose` A `logical` indicating if progress output should be printed to the console. Default is `TRUE`.

Details

In contrast to synergise1 synergise2 extends the default analysis and offers the follwing unique features:

- Performing 3D grid search (M/Z, Retention Time, Ion Mobility) for HDMSE data.
- Applying intensity correction.
- Filtering results by fragment matching.

Data can be input as a Synapter object if available or as a list of files (see filenames) that will be used to read the data in. The html report and result files will be created in the outputdir folder. All other input parameters have default values.

The data processing and analysis pipeline is as follows:

1. If uniquepep is set to TRUE (default), only unique proteotypic identification and quantitation peptides are retained.
2. Peptides are filtered for a FDR <= fdr (default is 0.01) using the "BH" method (see fdr and fdrMethod parameters for details).
3. Peptide with a mass tolerance > 20 ppms (see quantppm and identppm) are filtered out.
4. Peptides with a protein false positive rate (as reported by the PLGS software) > fpr are filtered out.
5. Common identification and quantitation peptides are merged and a retention time model is created using the Local Polynomial Regression Fitting (loess function for the stats package) using a default span.rt value of 0.05.
6. A grid search to optimise the width in retention time and mass tolerance (and ion mobility for HDMSE) for EMRTs matching is performed. The default grid search space is from 0.5 to 5 by 0.5 retention time model standard deviations (see grid.nsd.from, grid.nsd.to and grid.nsd.by parameters) and from 2 to 20 by 2 parts per million (ppm) for mass tolerance (see grid.ppm.from, grid.ppm.to and grid.ppm.by parameters). If HDMSE data are used the search space is extended from ion mobility difference 0.6 to 1.6 by 0.2 (see grid.imdiffs.from, grid.imdiffs.to and grid.imdiffs.by). The data can be subset using using an absolute number of features (see grid.n) or a fixed percentage (see grid.subset). The pair of optimal nsd and ppm is chosen (see grid.param.sel parameter).
7. Fragment Matching is used to filter false-positive matches from the grid search using a default of 1 common peak for unique matches and at least a difference of 1 common peaks to choose the correct match out of non-unique matches (see fm.minCommon and fm.minDelta).
8. The intensity is corrected by a Local Polynomial Regression Fitting (loess function for the stats package) using a default span.int value of 0.05.
9. The quantitation EMRTs are matched using the optimised parameters.

If a master identification file is used (master is set to TRUE, default is FALSE), the relevant actions that have already been executed when the file was created with makeMaster are not repeated here.

Value

Invisibly returns an object of class Synapter. Used for its side effect of creating an html report of the run in outputdir.

Author(s)

Laurent Gatto, Sebastian Gibb

References

Bond N. J., Shliaha P.V., Lilley K.S. and Gatto L. (2013) J. Prot. Research.

Examples

```
## Not run:
library(synapterdata)
data(synobj2)
output <- tempdir() ## a temporary directory
synergise2(object = synobj2, outputdir = output, outputfile = "synapter.html")
htmlReport <- paste0("file:///", file.path(output, "synapter.html")) ## the result report
browseURL(htmlReport) ## open the report with default browser

## End(Not run)
```

Index