

Package ‘synapter’

March 26, 2013

Type Package

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

Version 1.0.1

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

Maintainer Laurent Gatto <lg390@cam.ac.uk>

Depends R (>= 2.15), methods, MSnbase

Imports hwriter, tcltk, tcltk2, RColorBrewer, lattice, qvalue, multtest, utils, Biobase

Suggests synapterdata, knitr, xtable

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 <http://lgatto.github.com/synapter/>

biocViews Bioinformatics, MassSpectrometry, Proteomics, GUI

R topics documented:

synapter-package	2
estimateMasterFdr	2
inspectPeptideScores	3
makeMaster	4
MasterFdrResults-class	5
MasterPeptides-class	6
Synapter	7
synapterGUI	15
synapterGuide	16
synapterTinyData	16
synergise	17
Index	20

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

Three pipelines of varying flexibility are proposed the preform data analysis: (1) a graphical user interface is provided through the synatperGUI function, (2) the synergise function is a single entry function for a complete analysis and (3) 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, accessible 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 Bioconductor 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 and Nick J. Bond

Maintainer: Laurent Gatto <lg390@cam.ac.uk>

References

Bond N.J., Shliaha P.V., Lilley K.S. and Gatto L. *Improving qualitative and quantitative performance for label free proteomics*, in prep.

Shliaha, P.V., Bond N.J., Gatto L. and Lilley K.S. *The Effects of Ion Mobility Separation on Data Independent Acquisition in Proteomics Studies.*, in prep.

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 proteotypic 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, fastafilename, masterFdr = 0.025,  
fdr = 0.01, proteotypic = TRUE, verbose = TRUE)
```

Arguments

pepfiles	A list 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.
verbose	Should progress messages be printed?

Details

The false discovery rate for the master (merged) file is calculated 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

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

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 version 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	<i>Merges final peptide files</i>
------------	-----------------------------------

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.

Usage

```
makeMaster(pepfiles, fdr = 0.01,
           method = c("BH", "Bonferroni", "qval"), span = 0.05,
           verbose = TRUE)
```

Arguments

pepfiles A list of peptide final peptide file names to be merged.

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 A numeric with the loess span parameter value to be used for retention time modelling.

verbose A logical indicating 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 \leq fdr and (iii) proteins with a false positive rate \leq 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 individual 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.

Value

An instance of class "[MasterPeptides](#)".

Author(s)

Laurent Gatto

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 discovery 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

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

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.

Methods

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

Author(s)

Laurent Gatto

References

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

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-hoc container called a class. In the frame of the analysis of a set of 3 data files, namely as identification peptide, a quantitation peptide and a quantitation Pep3D, 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 is 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(filenamees, 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'. If missing, dialog boxes pop up to select the the four files manually. identpeptide can be a csv final peptide file (from PLGS) or a saved " MasterPeptides " data object as created by makeMaster if working with <i>master</i> peptide data. To serialise the " MasterPeptides " instance, use the saveRDS function, and file extension rds.
master	A logical that defines if the identification file is a <i>master</i> 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 opens four dialog boxes for the user to point to the input files, namely (and in that order), the identification final peptide csv file, the quantitation final peptide csv file and the quantitation Pep3D csv file (as exported from the PLGS software) and the fasta file use for peptide identification. The files are read and the data is stored in the newly created Synapter instance. The file names can also be specified as a named list with names 'identpeptide', 'quantpeptide' and 'quantpep3d' respectively.

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_H2O, 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.

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 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.

Mention `plotFeatures` here.

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` and `setRtNsd` 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 window defined by the error ppm and number of retention time standard deviations 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 ($100 * (\text{synapter} - \text{quant}) / \text{quant}$) and Venn counts.

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.

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.

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

searchGrid signature(object="Synapter", ppms="character", nsds="character", subset="numeric", n = "numeric", verbose="logical"): Performs a grid search. The grid is defined by the ppm and nsd numerical vectors, representing the sequence of values to be tested. Default are `seq(5, 20, 2)` and `seq(0.5, 5, 0.5)` respectively. `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.

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 4 with the names of the input files used as `identpeptide`, `quantpeptide`, `quantpep3d` and `fasta`.

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 percent 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, 0, 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 discovery 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.
- 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.
- 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** signature(object="Synapter", verbose = TRUE): Returns (and displays, if verbose) the performance of the synapter analysis.

Filters:

filterUniqueDbPeptides signature(object="Synapter"): This method first digests the fasta database file and keeps unique tryptic peptides. Those peptide sequences are then used as a filter against the identification and quantitation peptides, where only unique proteotypic instances (no miscleavage allowed) 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"): As filterUniqueDbPeptides for quantitation peptides only.

filterUniqueIdentDbPeptides signature(object="Synapter"): 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 = "character"): 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.

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 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 `qplot` method, one of "BH", "Bonferroni" or "qval", defines what identification statistics to use.

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

plotGrid signature(object="Synapter", what = "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.

plotFeatures signature(object="Synapter", what = "character", xlim = "numeric", ylim = "numeric"): Plots the retention time against precursor mass space. If what is "all", three 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.

Exporters:

writeMergedPeptides signature(object="Synapter", file = "character", what = "character", ...): Exportes the merged peptide data to a comma-separated file (default name is "Res-MergedPeptides.csv"). what can be "light" (default) or "full" and specifies if the full data or only selected columns are exported. ... are passed to `write.csv`.

writeMatchedEMRTs signature(object="Synapter", file = "character", what = "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", ...): As above, exporting the quantitation peptide data.

Other:

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

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

Shliaha P.V., Bond N.J., Lilley K.S. and Gatto L. Improving qualitative and quantitative performance for label free proteomics, in prep.

Shliaha, P.V., Bond N.J., Gatto L. and Lilley K.S. The Effects of Ion Mobility Separation on Data Independent Acquisition in Proteomics Studies., in prep.

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

## 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)

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

```

synapterGUI

Launch the synapter GUI

Description

This function starts the synapter graphical user interface that allows to (1) load *n* sets of files to be analysed, (2) parameters used for data filtering and retention time modelling and (3) parameters used for grid search. The *n* file sets and the parameters recorded by the GUI are used to call [synergise](#) that performs the analysis and generates *n* html reports. See the synapter vignette for a description of the interface and [synergize](#) for a description of the parameters and data processing.

Usage

```
synapterGUI(n = 1)
```

Arguments

n Number of analysis to be run.

Value

Invisibly returns null. Used for its side effects.

Author(s)

Laurent Gatto

synapterGuide	<i>Opens the 'synapter' vignette</i>
---------------	--------------------------------------

Description

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

Usage

```
synapterGuide()
```

Author(s)

Laurent Gatto

synapterTinyData	<i>Loads a small test data for the 'synapter' package</i>
------------------	---

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 were not necessary for processing were also removed. As such, the data stored in `synapterTiny` does not reflect 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 .GlobalEnv.

Author(s)

Laurent Gatto

Source

Bond, Shliaha, Gatto and Lilley, *in prep.*

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 rda files. See details for a description of the pipeline and [Synapter](#) for manual execution of individual steps.

Usage

```
synergise(filenames, master = FALSE, object, outputdir,
  fdr = 0.01, fdrMethod = c("BH", "Bonferroni", "qval"),
  fpr = 0.01, identppm = 20, quantppm = 20,
  uniquepep = TRUE, span = 0.05, grid.ppm.from = 5,
  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"),
  css = NULL, verbose = TRUE)
```

Arguments

- | | |
|-----------|--|
| filenames | A named list of file names to be load. The names must be identpeptide, quantpeptide, quantpep3d and fasta. If missing, a dialog box opens to select files interactively. identpeptide can be a csv final peptide file (from PLGS) or a saved " MasterPeptides " data object as created by makeMaster if working with <i>master</i> peptide data. To serialise the " MasterPeptides " instance, use the saveRDS function, and file extension 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.
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.
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 5.
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.
css	An optional path to a custom css file. If NULL (default), uses synapter.css.
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. If none of object and filenames are provided, file section menus are open to select input files. The html report and result files will be created in the outputdir folder. If not provided, the destination can be selected through a selection menu. 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 \leq fdR (default is 0.01) using the "BH" method (see fdR and fdRMethod parameters for details).
3. Peptide with a mass tolerance $>$ 20 ppm (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 5 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

References

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

Examples

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

## End(Not run)
```

Index

*Topic **classes**

MasterFdrResults-class, 5

MasterPeptides-class, 6

Synapter, 7

*Topic **datasets**

synapterTinyData, 16

*Topic **package**

synapter-package, 2

allComb (MasterFdrResults-class), 5

allComb,MasterFdrResults-method
(MasterFdrResults-class), 5

as.MSnSet.Synapter (Synapter), 7

bestComb (MasterFdrResults-class), 5

bestComb,MasterFdrResults-method
(MasterFdrResults-class), 5

class:MasterFdrResults
(MasterFdrResults-class), 5

class:MasterPeptides
(MasterPeptides-class), 6

class:Synapter (Synapter), 7

dim,Synapter-method (Synapter), 7

estimateMasterFdr, 2, 5

fileNames,MasterFdrResults-method
(MasterFdrResults-class), 5

filterIdentPepScore (Synapter), 7

filterIdentPepScore,Synapter-method
(Synapter), 7

filterIdentPpmError (Synapter), 7

filterIdentPpmError,Synapter-method
(Synapter), 7

filterIdentProtFpr (Synapter), 7

filterIdentProtFpr,Synapter-method
(Synapter), 7

filterQuantPepScore (Synapter), 7

filterQuantPepScore,Synapter-method
(Synapter), 7

filterQuantPpmError (Synapter), 7

filterQuantPpmError,Synapter-method
(Synapter), 7

filterQuantProtFpr (Synapter), 7

filterQuantProtFpr,Synapter-method
(Synapter), 7

filterUniqueDbPeptides (Synapter), 7

filterUniqueDbPeptides,Synapter-method
(Synapter), 7

filterUniqueIdentDbPeptides (Synapter), 7

filterUniqueIdentDbPeptides,Synapter-method
(Synapter), 7

filterUniqueQuantDbPeptides (Synapter), 7

filterUniqueQuantDbPeptides,Synapter-method
(Synapter), 7

findEMRTs (Synapter), 7

findEMRTs,Synapter-method (Synapter), 7

getBestGridParams (Synapter), 7

getBestGridParams,Synapter-method
(Synapter), 7

getBestGridValue (Synapter), 7

getBestGridValue,Synapter-method
(Synapter), 7

getEMRTtable (Synapter), 7

getEMRTtable,Synapter-method
(Synapter), 7

getGrid (Synapter), 7

getGrid,Synapter-method (Synapter), 7

getGridDetails (Synapter), 7

getGridDetails,Synapter-method
(Synapter), 7

getIdentPpmError (Synapter), 7

getIdentPpmError,Synapter-method
(Synapter), 7

getLog (Synapter), 7

getLog,Synapter-method (Synapter), 7

getLowessSpan (Synapter), 7

getLowessSpan,Synapter-method
(Synapter), 7

getPepNumbers, 3

getPepNumbers (Synapter), 7

getPepNumbers,Synapter-method
(Synapter), 7

getPepScoreFdr (Synapter), 7

getPepScoreFdr,Synapter-method
(Synapter), 7

- getPpmErrorQs (Synapter), 7
- getPpmErrorQs,Synapter-method (Synapter), 7
- getProtFpr (Synapter), 7
- getProtFpr,Synapter-method (Synapter), 7
- getQuantPpmError (Synapter), 7
- getQuantPpmError,Synapter-method (Synapter), 7
- getRtNsd (Synapter), 7
- getRtNsd,Synapter-method (Synapter), 7
- getRtQs (Synapter), 7
- getRtQs,Synapter-method (Synapter), 7
- inputFiles (Synapter), 7
- inputFiles,Synapter-method (Synapter), 7
- inspectPeptideScores, 3
- loess, 9, 10, 18
- makeMaster, 3, 4, 6, 8, 17, 19
- masterFdr (MasterFdrResults-class), 5
- masterFdr,MasterFdrResults-method (MasterFdrResults-class), 5
- MasterFdrResults, 3
- MasterFdrResults (MasterFdrResults-class), 5
- MasterFdrResults-class, 5
- MasterPeptides, 5, 8, 17
- MasterPeptides-class, 6
- mergePeptides (Synapter), 7
- mergePeptides,Synapter-method (Synapter), 7
- modelRt (Synapter), 7
- modelRt,Synapter-method (Synapter), 7
- performance (Synapter), 7
- performance,Synapter-method (Synapter), 7
- plot,MasterFdrResults,missing-method (MasterFdrResults-class), 5
- plotEMRTtable (Synapter), 7
- plotEMRTtable,Synapter-method (Synapter), 7
- plotFdr (Synapter), 7
- plotFdr,Synapter-method (Synapter), 7
- plotFeatures (Synapter), 7
- plotFeatures,Synapter-method (Synapter), 7
- plotGrid (Synapter), 7
- plotGrid,Synapter-method (Synapter), 7
- plotPepScores, 3
- plotPepScores (Synapter), 7
- plotPepScores,Synapter-method (Synapter), 7
- plotPpmError (Synapter), 7
- plotPpmError,Synapter-method (Synapter), 7
- plotRt (Synapter), 7
- plotRt,Synapter-method (Synapter), 7
- plotRtDiffs (Synapter), 7
- plotRtDiffs,Synapter-method (Synapter), 7
- qplot, 13
- searchGrid (Synapter), 7
- searchGrid,Synapter-method (Synapter), 7
- setBestGridParams (Synapter), 7
- setBestGridParams,Synapter-method (Synapter), 7
- setIdentPpmError (Synapter), 7
- setIdentPpmError,Synapter-method (Synapter), 7
- setLowessSpan (Synapter), 7
- setLowessSpan,Synapter-method (Synapter), 7
- setPepScoreFdr (Synapter), 7
- setPepScoreFdr,Synapter-method (Synapter), 7
- setPpmError (Synapter), 7
- setPpmError,Synapter-method (Synapter), 7
- setProtFpr (Synapter), 7
- setProtFpr,Synapter-method (Synapter), 7
- setQuantPpmError (Synapter), 7
- setQuantPpmError,Synapter-method (Synapter), 7
- setRtNsd (Synapter), 7
- setRtNsd,Synapter-method (Synapter), 7
- show,MasterFdrResults-method (MasterFdrResults-class), 5
- show,MasterPeptides-method (MasterPeptides-class), 6
- show,Synapter-method (Synapter), 7
- showFdrStats (Synapter), 7
- showFdrStats,Synapter-method (Synapter), 7
- Synapter, 5, 7, 16–18
- synapter (synapter-package), 2
- synapter-package, 2
- synapterGUI, 5, 15
- synapterGuide, 16
- synapterTiny (synapterTinyData), 16
- synapterTinyData, 16
- synergise, 5, 15, 17
- synergize, 15
- synergize (synergise), 17

write.csv, [13](#)
writeIdentPeptides (Synapter), [7](#)
writeIdentPeptides,Synapter-method
(Synapter), [7](#)
writeMasterPeptides (makeMaster), [4](#)
writeMatchedEMRTs (Synapter), [7](#)
writeMatchedEMRTs,Synapter-method
(Synapter), [7](#)
writeMergedPeptides (Synapter), [7](#)
writeMergedPeptides,Synapter-method
(Synapter), [7](#)
writeQuantPeptides (Synapter), [7](#)
writeQuantPeptides,Synapter-method
(Synapter), [7](#)