# Package 'synapter'

October 8, 2014

**Type** Package

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

**Version** 1.6.0

**Author** Laurent Gatto, Nick J. Bond and Pavel V. Shliaha with contributions from Sebastian Gibb.

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

**Depends** R (>= 2.15), methods, MSnbase

**Imports** hwriter, tcltk, tcltk2, RColorBrewer, lattice, qvalue,multtest, utils, Biobase, knitr, Biostrings, cleaver,BiocParallel

**Suggests** synapterdata, 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/

**VignetteBuilder** knitr

**biocViews** MassSpectrometry, Proteomics, GUI

## R topics documented:

---

`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 variying 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, 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 and Nick J. Bond

Maintainer: 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, 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.

`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 petides, 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, verbose = TRUE)
```

## 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 missed cleavage sites. Default is 0. |
| `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.

## 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 preptide 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 <= 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 loded) 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.

## 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 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` obejcts.

`pepfiles`**:** Object of class `"character"` with the list of final peptide iinput 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

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.

---

`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 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 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'. If missing, dialog boxes pop up to select 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 *master* peptide data. To serialise the `"MasterPeptides"` instance, use the `saveRDS` function, and file extenstion `rds`. |
| `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 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_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.

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 * ( synapter - quant ) / 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", mergedEMRTs = c("rescue"`
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). The `mergedEMRTs` 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="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. (Note, the `mergedEMRTs` value used in internal calls to `findEMRTs` is `"transfer"` - see `findEMRTs` for details).

### 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 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`.

**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** `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.

**Filters:**

**filterUniqueDbPeptides** `signature(object="Synapter")`: 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 missed cleavages can be set as `missedCleavages` (default is 0). Those 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, verbose = TRUE)`: As `filterUniqueDbPeptides` for quantitation peptides only.

**filterUniqueIdentDbPeptides** `signature(object="Synapter", missedCleavages = 0, 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.

**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.

**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 or 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", ...)`: Exports 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", ...)`: A 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

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

```
library(synapter) ## always needed

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

## End(Not run)

## lets 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, 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-filted 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 analyses, (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. |

## Details

The GUI code requires the Tcl package 'BWidget' to display the tree drill-down file selection menu. The package is available on Windows (in case of issues, please read http://www.stat.berkeley.edu/users/spector/s133/Bwidget.html). On GNU/Linux, the package needs to be installed separately; on Ubuntu, the package is called 'bwidget'.

## Value

Invisibly returns `null`. Used for its side effects.

## Author(s)

Laurent Gatto

---

# 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

`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 `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, peplen = 7,
  missedCleavages = 0, 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"),
  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 *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. |
| `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. |
| `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. |
| 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. |
| 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. |
| fdr | Peptide false discovery rate. Default is 0.01. |
| 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. |

## 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 <= 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

## 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, 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