SWATH2stats

Peter Blattmann, Moritz Heusel, and Ruedi Aebersold

Institute of Molecular Systems Biology, Department of Biology,
ETH Zurich, Switzerland

December 6, 2015

This vignette describes how the different functions from the SWATH2stats package can be applied. The functions from the SWATH2stats package are intended to be used on SWATH data that has been generated by the OpenSWATH pipeline. The SWATH2stats package provides functions to annotate such SWATH data with experimental meta-data, perform a false-discovery rate (FDR) assessment, perform a filtering to control the FDR, and to convert the SWATH data into a format readable by other statistical and quantification software tools such as MSstats, aLFQ, mapDIA or imsbInfer. The SWATH2stats package thus represents a link between the OpenSWATH pipeline and the downstream analysis packages MSstats, aLFQ, mapDIA, or imsbInfer. The SWATH2stats package was programmed and intended for use by researchers in proteomics working with SWATH data without extensive programming skills, but with basic R knowledge.

Contents

1 Introduction

1.1 SWATH-MS data analysis via open source tools

SWATH-MS as an implementation of data-independent acquisition (DIA) mass spectrometry is an emerging proteomic approach that allows systematic quantification of peptides in complex samples (Gillet et al. 2012, Venable et al. 2004). The acquired mass spectra can be queried for the presence and quantity of peptide analytes using the open-source OpenSWATH pipeline. The OpenSWATH pipeline consists of the OpenSWATH software (Roest et al. 2014) coupled to statistical validation using the mProphet algorithm (Reiter et al. 2011), or its re-implementation pyProphet (Teleman et al. 2015). OpenSWATH extracts ion chromatograms of both the peptide precursor and the fragment ions and quantifies peak groups. It then generates scores for how well a given candidate peak group corresponds to an analyte from a spectral or assay library (Roest et al. 2014). mProphet uses a machine learning algorithm to identify an optimal linear combination of these scores (d-score) to discriminate targets from decoys. In addition, it fits a function to the distribution of the d-scores for the decoy peptides, that is used as the null distribution. This null distribution is then used to calculate a q-value/m-score of each peakgroup (Storrey et al. 2003, Reiter et al. 2011). Hence, filtering the results with an m-score of 0.01 results in an FDR of 1% of the target assays within this run.

1.2 The usage of SWATH2stats in the open-source SWATH-MS data analysis workflow

This package creates a link between the OpenSWATH/mProphet .tsv output table and popular downstream tools for statistical and advanced data analysis. With very large assay libraries (Rosenberger et al. 2014) the SWATH results can become too large to analyse and process via tools such as Microsoft Excel. Therefore this package offers functionality to annotate the data with the study

design (such as condition, biological and unique MS run id) and also offers substantial filtering capabilities. The data can be filtered based on frequency of observation, number of sibling peptides of a protein entry or directly on the FDR as estimated by the mProphet model (m-score, equivalent to q-value; for details see previous section or Reiter et al. 2011). Furthermore the package features estimation of global false discovery rates according to the target-decoy rationale (Elias and Gygi 2008, Kaell et al. 2008). The last step is the conversion to formats that can be read directly by the downstream analysis packages such as MSstats (Choi et al. 2014), mapDIA (Teo et al. unpublished), aLFQ (Rosenberger et al. 2014), and imsbInfer (Wolski et al. unpublished).

1.3 Implementation of a target-decoy strategy to estimate false target discovery rates (FDR)

Mass-spectrometry-based proteomic experiments produce large amounts of data that require statistical validation. In the SWATH2stats package a target-decoy strategy was implemented to estimate the FDR (Elias and Gygi, 2007). The target-decoy strategy relies on the assumption that the decoys have the same characteristics (distribution of their scores) as the false targets. The FDR among the targets is estimated as the ratio of decoy peptides passing a certain score threshold divided by the total number of targets passing the same score threshold (Choi and Nesvizhskii, 2008). The usage of the target-decoy strategy for SWATH data and to estimate peptide and protein-level FDR has not been extensively tested yet. The target-decoy strategy has been tested to estimate protein-level FDR in DDA data and has been shown to resulted in more conservative FDR estimates compared to the model-based approach (Reiter et al. 2009). A target-decoy approach was implemented in SWATH2stats because it allows i.) estimation of an FDR over multiple runs and ii.) allows to directly assess the selectivity of a given filter for likely true (target) over false (decoy) data points.

In contrast to the naive target-decoy approach counting the number of decoys, a correction factor can be supplied to many FDR estimation functions in the SWATH2stats package. For example, a correction needs to be applied to correct for the fraction of false targets (FFT), in analogy to what has been shown in Kaell et al. 2008 for DDA data. Similar correction factors have been used to adjust FDR estimation in DDA data (PIT: Kaell et al. 2008, p(-): Keller et al. 2002). In the functions, the FFT defaults to 1 to perform a naive target-decoy counting strategy without FFT correction, which will result in an overestimation of the FDR. For a less conservative estimation of the FDR, a FFT correction factor can be provided that corrects for the ratio of false targets to decoys. The number of decoys counted is multiplied with the FFT correction factor. The rationale is that for example if 50% of the samples are true targets, the number of true negative targets that are modeled by the decoy distribution is around 50% lower than the decoy distribution. Therefore 2 decoy hits passing a certain m-score threshold suggest only one false positive datapoint. The ratio of true negative targets compared to all targets (FFT) can for example be obtained from the mProphet model statistics (Injection_name]_full_stat.csv (column 1 line 2 corresponding to the maximal q-value).

Alternatively, the FFT can conservatively be approximated by the fraction of assays in the library that do not pass an m-score threshold of e.g. 0.01 (corre-

sponding to 1 % model FDR). For example, acquiring a full cell lysate and searching the data using the combined assay library (200k assays, Rosenberger et al. 2014), 50k assays are typically identified with m-score <= 0.01. Hence a FFT of 0.75 can be estimated. If a full lysate is searched by a sample-specific assay library (e.g. 70k assays) and 40k assays were identified with m-score <=0.01, a FFT of 0.57 can be estimated.
Another option is to use a correction factor so that the estimated FDR by decoy counting corresponds to the mProphet derived FDR (e.g. m-score threshold 0.01 should correspond to an assay-level FDR of 0.01).

2 Loading and annotating the data

2.1 Loading the data

To install the SWATH2stats package the following commands can be executed within R (after package has been accepted to Bioconductor).

```
> source("http://www.bioconductor.org/biocLite.R")
> biocLite("SWATH2stats")
```

Typically, the workspace is cleared and the SWATH2stats package loaded.

```
> rm(list=ls())
> library(SWATH2stats)
```

The example data, that is included in the package, consists of a reduced OpenSWATH output file generated from Hela cells. To avoid making the file of the SWATH2stats package too large, only a fraction of a typical SWATH data table is included as an example data. The example data contains data for 9 proteins, 5 decoy-proteins and a set of peptides for retention time calibration (labelled as iRT_protein). In total the data contains 284 peptides for which quantitative data has been extracted from 6 different samples measured on an ABSciex TripleTOF 5600 mass spectrometer and analyzed with the OpenSWATH + pyProphet workflow (Roest et al. 2014). These 6 samples consist of biological triplicates of Hela cells grown under control condition and Hela cells that have been perturbed by inhibiting cholesterol synthesis.
The experimental design is described in a table called Study_design that is included in the package. This file that contains the study design information needs to be a table with the following columns: Filename, Condition, BioReplicate, Run (see below). For correct assignment of identifiers into the Run, BioReplicate and Condition column for MSstats, please consult their manual. The values in the column `Filename` have to be unique for every injection file and will be matched to the OpenSWATH output in the column `align_origfilename` (caution: this matching is case sensitive).
The example SWATH data and the study design table can be loaded from the package with the function `data()`.

```
> data('OpenSWATH_data', package='SWATH2stats')
> data <- OpenSWATH_data
> data('Study_design', package='SWATH2stats')
> head(Study_design)
```

```
               Filename      Condition BioReplicate Run
1     peterb_J131223_043   Hela_Control            1   1
2     peterb_J131223_054 Hela_Treatment            1   2
3 peterb_L150425_003b_SW   Hela_Control            2   3
4  peterb_L150425_011_SW Hela_Treatment            2   4
5  peterb_L150514_001_SW   Hela_Control            3   5
6  peterb_L150514_002_SW Hela_Treatment            3   6
```

The working directory of the analysis is defined and the file name of the OpenSWATH results file and the study design file are indicated (in this example they should be present in the same folder). Whereas the rest of the script can be kept after having optimized the filtering options, this first part of the script is changed for different data that will be analyzed.

```
> n# set working directory
> setwd('~/Documents/MyWorkingDirectory/')
> # Input data file (openSWATH output)
> file.name <- 'OpenSWATH_output_file.txt'
> # File name for annotation file
> annotation.file <- 'Study_design_file.txt'
```

The SWATH data can be loaded from the previously indicated file into R using the following command.

```
> # load data
> data <- data.frame(fread(file.name, sep='\t', header=TRUE))
```

The function `reduce_openSWATH_output` can be executed to reduce the number of columns from the OpenSWATH result table. This function reduces the number of columns to the ones necessary for MSstats, mapDIA, aLFQ. However for other packages such as imsbInfer all the columns need to be kept and this function should be omitted. In the next command the iRT peptides (peptides for retention time calibration) can be removed. Similar commands can be used to exclude other proteins.

```
> # reduce number of columns
> data <- reduce_OpenSWATH_output(data)
> # remove the iRT peptides (or other proteins)
> data <- data[grep('iRT', data$ProteinName, invert=TRUE),]
```

2.2 Annotating the data

With the first two commands the number of files in the OpenSWATH data and the names of these files can be printed. This can be helpful to generate the study design table (The script can be executed until here and then the annotation file generated with a text editor). See above for a description of the exact format and column names required for the study design table.

```
> # list number and different Files present
> nlevels(factor(data$align_origfilename))
> levels(factor(data$align_origfilename))
> # load the study design table from the indicated file
> Study_design <- read.delim2(file.path(getwd(), annotation.file),
+                                dec='.', sep='\t', header=TRUE)
```

With the function `sample_annotation` the data is annotated with the meta-data contained in the study design table. The next commands can be used to shorten the protein names and remove repetitive and non-unique parts of the Protein name as shown by the example removing some parts of the identifier keeping only the unique SwissProt accession identifier (e.g. sp_Q9GZL7_WDR12_HUMAN –> Q9GZL7).

```
> # annotate data
> data.annotated <- sample_annotation(data, Study_design)
> head(unique(data$ProteinName))

[1] 1/Protein6  1/Protein1  1/Protein7  1/Protein4  1/Protein8
[6] 10/Protein9
15 Levels: 1/Protein1 1/Protein2 1/Protein3 1/Protein4 ... DECOY_1/Protein6

> # OPTIONAL: for human, shorten Protein Name to remove non-unique information
> #(sp|Q9GZL7|WDR12_HUMAN --> Q9GZL7)
> data$ProteinName <- gsub('sp\\|([[:alnum:]]+)\\|[[:alnum:]]*_HUMAN',
+                          '\\1', data$ProteinName)
> head(unique(data$ProteinName))

[1] "1/Protein6"  "1/Protein1"  "1/Protein7"  "1/Protein4"
[5] "1/Protein8"  "10/Protein9"
```

3 FDR estimation

Mass-spectrometry-based proteomic experiments produce large amounts of data, requiring statistical validation of the obtained results. Large multi-run proteomics studies are prone to the accumulation of false positive identifications and the statistical significance scores must therefore be normalized accordingly(Benjamini and Hochberg, 1995).
This chapter describes first the functionality of SWATH2stats to estimate and visualize the global false discovery rate in OpenSWATH/mProphet result tables and second the functionality to obtain m-score thresholds (peak group level mProphet-estimated FDR quality) to control FDR on a global level.

Assays are identified by unique identifiers in the column `transition_group_id` of the SWATH data table, peptides by unique identifiers in the column `FullPeptideName` and protein(group)s by unique identifiers in the column `ProteinName`. Different MS injections (also termed runs) are identified based on a unique entry in the column run_id.

3.1 FDR: Overview and visualization

SWATH2stats supplies three functions to assess and visualize the false discovery rate in multi-run SWATH data. These functions are useful to get an overview on the relationship between false discovery rate and m-score thresholds. A suitable m-score threshold can subsequently be used to filter the data with the filtering functions described in the next chapter.
The FDR within the results passing a given score cutoff is evaluated as explained in the introduction:

FDR = (number of decoys * FFT)/(number of targets)

Application of the decoy-counting-based FDR assessment functions in interplay with the meta-data filters can help the researcher in selecting an efficient strategy to establish highest possible data quality for downstream analyses. By counting the decoys before and after application of a filter, the selectivity of a given filter for likely true (target) over false (decoy) data can be estimated. With a first basic function `assess_decoy_rate` the overall number of decoy peptides can be counted in the data:

```
> data('OpenSWATH_data_FDR', package='SWATH2stats')
> data.FDR<-sample_annotation(OpenSWATH_data_FDR, Study_design)
> assess_decoy_rate(data.FDR)
```

The function `assess_fdr_overall` creates a global assessment of decoy rates (and estimated FDR) on assay, peptide and protein level. Results are reported by default as .csv table and visualized in a .pdf report. Setting the output option to "Rconsole" reports back the results to R. Included in the pdf report are plots showing the estimated global FDR in relation to the m-score threshold. Because false-positive hits accumulate over different runs, the false discovery rate estimated by this function will be higher than if assessed within each run individually.

```
> # count decoys and targets on assay, peptide and protein level
> # and report FDR at a range of m_score cutoffs
> assess_fdr_overall(data.FDR, FFT = 0.7, output = "pdf_csv", plot = TRUE,
+                    filename='assess_fdr_overall_testrun')
> # The results can be reported back to R for further calculations
> overall_fdr_table <- assess_fdr_overall(data.FDR, FFT = 0.7,
+                                         output = "Rconsole")
```

The function `plot.fdr_table` allows to create the report plots from this overall fdr table.

```
> # create plots from fdr_table
> plot(overall_fdr_table, output = "Rconsole",
+                filename = "FDR_report_overall")
```

The function `assess_fdr_byrun` investigates the decoy rate or FDR in individual runs and by default reports the results in a .csv table and .pdf file. Setting the output option to "Rconsole" reports back the results to R. This function is used if the FDR for different injections should be estimated separately.

```
> # count decoys and targets on assay, peptide and protein level per run
> # and report FDR at a range of m_score cutoffs
> assess_fdr_byrun(data.FDR, FFT = 0.7, output = "pdf_csv", plot = TRUE,
+                  filename='assess_fdr_byrun_testrun')
> # The results can be reported back to R for further calculations
> byrun_fdr_cube <- assess_fdr_byrun(data.FDR, FFT = 0.7,
+                                    output = "Rconsole")
```

The function `plot._fdr_cube` allows to create the report plots from this by-run fdr cube.

```
> # create plots from fdr_table
> plot(byrun_fdr_cube, output = "Rconsole",
+               filename = "FDR_report_overall")
```

3.2 Identification of useful m-score cutoffs to satisfy desired FDR criteria

SWATH2stats supplies three functions for the identification of useful m-score cutoffs to satisfy FDR criteria on assay, peptide and protein level over many different runs. These functions return an m-score value, which can be used to filter the data of these different runs in order to obtain a desired overall FDR. The following functions report an m-score cutoff to achieve a strict global FDR target.
The function `mscore4assayfdr` reports an m-score cutoff to achieve a desired overall (global) assay FDR:

```
> # select and return a useful m_score cutoff in order
> # to achieve the desired FDR quality for the entire table
> mscore4assayfdr(data.FDR, FFT = 0.7, fdr_target=0.02)

[1] 8.912509e-05
```

The function `mscore4pepfdr` reports an m-score cutoff to achieve a desired overall (global) peptide FDR:

```
> # select and return a useful m_score cutoff
> # in order to achieve the desired FDR quality for the entire table
> mscore4pepfdr(data.FDR, FFT = 0.7, fdr_target=0.02)

[1] 8.912509e-05
```

The function `mscore4protfdr` reports an m-score cutoff to achieve a desired overall (global) protein FDR. Protein FDR control on peak group quality level is a very strict filter and should be handled with caution. Alternatively, a function `filter_mscore_fdr` is described below applying a two-tiered filtering approach.

```
> # select and return a useful m_score cutoff in order
> # to achieve the desired FDR quality for the entire table
> mscore4protfdr(data.FDR, FFT = 0.7, fdr_target=0.02)

[1] 8.912509e-08
```

4 Filtering the data

In this chapter the SWATH data is filtered based on the study design or desired global FDR criteria to be achieved. By setting the option rm.decoy=FALSE, the decoy peptides can be kept in the data in order to evaluate the selectivity of a given filter for likely true (target) over false (decoy) data by decoy counting with the functions described in the previous chapter.

Before converting the data for statistical analysis the rm.decoy option is set to 'TRUE' in order to remove any decoy peptides and proteins from the data.

4.1 Filter on m-score

The function `filter_mscore` removes all measured precursor peptides that are above a certain m-score value. The number of rows removed by the function is indicated.

```
> data.filtered.mscore <- filter_mscore(data.annotated, 0.01)
```

The function `filter_mscore_requant` takes into account how many times in the different injection runs a peak group has been confidently (as defined by the m-score threshold) identified. This is useful in large data of many different replicates. For example the data for a certain precursor that has been confidently identified in most of the replicates but does not pass the threshold in one replicate still should be kept for statistical analysis. In order to keep such data, but discard data of precursors that have not been confidently identified in the other replicates, the function `filter_mscore_requant` can be used. Here, precursors passing a m-score threshold of 0.01 in 80 % of the replicates are selected. The option rm.decoy is set to FALSE to keep the decoys for subsequent FDR assessment.

```
> data.filtered.mscore <- filter_mscore_requant(data.annotated, 0.01, 0.8,
+                                               rm.decoy=FALSE)
```

The function `filter_mscore_condition` selects only precursors that have passed a certain m-score threshold in at least 3 replicates for the same condition (as defined by the study design table). In contrast to the previous function, this selects precursors that are confidently identified a certain number of times in the same condition.

```
> data.filtered.mscore <- filter_mscore_condition(data.annotated, 0.01, 3)
```

In order to reach a compromise between a very stringent m-score filter controlling the global protein FDR, and keeping valid peptide quantifications in the data, we introduce here a two-tiered filtering approach with the function `filter_mscore_fdr`. This uses a similar approach as implemented for extracting quantitative data from multi-run DDA data sets (Fermin et al. 2011). In the first step, a m-score cutoff is applied to reach a desired protein-level FDR. All proteins passing this m-score cutoff criterion are collected in a protein master list. The original data is then filtered i.) for the proteins present in the master list and ii.) and all peptide quantifications passing an m-score cutoff to achieve a desired global peptide-level FDR. Note that the m-score cutoff to filter the protein list will typically be more stringent than the second m-score cutoff to filter the peptides. The function `filter_mscore_fdr` also automatically runs `assess_fdr_byrun` to estimate the FDR in individual runs after application of the second m-score cutoff used to control the peptide-level FDR over all runs (this estimation is performed without filtering for the protein master list). The rationale of this second FDR estimation is to obtain a feeling for the FDR quality of the quantitative values when analyzing the runs individually.

```
> data.filtered.fdr <- filter_mscore_fdr(data.FDR, FFT=0.7,
+                                        overall_protein_fdr_target = 0.03,
+                                        upper_overall_peptide_fdr_limit = 0.05)
```

4.2 Filter on proteotypic peptides

With the functions `filter_proteotypic_peptides` and `filter_all_peptides` the number of proteins is assessed. The function `filter_proteotypic_peptides` selects only data that is based on proteotypic peptides (peptides only contained in one protein and marked by "1/" in the beginning of the protein identifier). These functions also remove the '1' in front of the protein identifier from proteotypic peptides.

```
> data <- filter_proteotypic_peptides(data.filtered.mscore)
> data.all <- filter_all_peptides(data.filtered.mscore)
```

4.3 Filter on sibling peptides

With the function `filter_on_max_peptides` the peptides showing the strongest signal over the entire table can be selected. Removing the lower intense peptides for a protein can make the statistical analysis faster or result in more accurate quantification of proteins under the assumption that quantification of more intense peptides is more robust.

```
> data.filtered.max <- filter_on_max_peptides(data.filtered.mscore, 5)
```

Conversely maybe only data for proteins with a minimum number of supporting peptides should be selected. With the function `filter_on_min_peptides` only the proteins for which at least a certain number of peptides have been measured are selected. This filter can also be powerful to remove false positive hits from the data as these are enriched in the fraction of single hits. FDR assessment based on decoy counting may still be valid after such filtering (Reiter et al. 2009).

```
> data.filtered.max.min <- filter_on_min_peptides(data.filtered.max, 2)
```

5 Conversion of data for other tools

In order to use the filtered and annotated data in other programs and packages the data has to be converted into the required format. This chapter describes the different functions within SWATH2stats that can be used to convert the filtered SWATH data into the desired format.

5.1 Results on protein level

SWATH2stats can write a protein-level summary matrix showing the summed signals of Proteins (unique `ProteinName` identifiers) over the MS runs (unique `run_id`) using the function `write_matrix_proteins`. It calculates the sum of all transition intensities per assay, all charge states per peptide, and all peptides for the different protein groups. Note that this function does not select consistently quantified peptides, or a certain number of highest intense peptides, and therefore the summed signal should be used with caution as a direct measure of protein abundance or to compare protein abundance between runs. For other quantitative protein inference strategies, the R package aLFQ can be used

(Rosenberger et al. 2014, see below).
A more detailed overview can be generated using the function `write_matrix_peptides`.

Writing the overview matrix of summed intensities per protein entry per MS run:

```
> write_matrix_proteins(data, filename = "SWATH2stats_overview_matrix_proteinlevel",
+                       rm.decoy = FALSE)
```

5.2 Results on peptide level

With these commands a table is generated that shows the SWATH data on peptide level.

```
> data.peptide <- data
> data.peptide$aggr_Fragment_Annotation <- NULL
> data.peptide$aggr_Peak_Area <- NULL
```

```
> write.csv(data.peptide, file='peptide_level_output.csv',
+           row.names=FALSE, quote=FALSE)
```

Alternatively, an overview matrix can be written which lists the peptide intensity (sum of all transitions per assay and all precursors per peptide) over the MS runs (unique `run_id`), using the function `write_matrix_peptides`:

```
> write_matrix_peptides(data,
+                       filename = "SWATH2stats_overview_matrix_peptidelevel",
+                       rm.decoy = FALSE)
```

5.3 Results on transition level

With the function `disaggregate` the SWATH data is changed from a table where one row corresponds to one peptide to a table where one row corresponds to one measured transition.

```
> data.transition <- disaggregate(data)
```

```
> write.csv(data.transition, file='transition_level_output.csv',
+           row.names=FALSE, quote=FALSE)
```

5.3.1 Conversion using a python script

For very large SWATH data it is faster to use a custom-made python script to transform the data from a peptide-level format to a transition-level format. With the function `convert4pythonscript` the necessary columns are selected and the nomenclature for modified peptides is changed. Subsequently the data is written to disk.

```
> data <- convert4pythonscript(data)
> write.table(data, file="input.tsv", sep="\t", row.names=FALSE, quote=FALSE)
> head(data)
```

The .tsv table can be transformed into a transition level table using a python script (as example featurealigner2msstats_withRT.py from msproteomicstools which is available in the scripts folder of the package).

`python ./featurealigner2msstats.py input.csv output.csv`

Afterwards the generated .csv table is loaded again into R.

```
> data.transition <- data.frame(fread('output.csv',
+                                     sep=',', header=TRUE))
```

5.4 MSstats

In order to use the data in the R Bioconductor package `MSstats`, the transition-level data needs to be converted using the function `convert4MSstats`. Afterwards the data can directly be processed using the `MSstats` package as shown here by application of the function `dataProcess` from the MSstats package.

```
> MSstats.input <- convert4MSstats(data.transition)
> library(MSstats)
> quantData <- dataProcess(MSstats.input)

  Summary of Features :
                         count
# of Protein                 8
# of Peptides/Protein    1-103
# of Transitions/Peptide   6-6

  Summary of Samples :
                           Hela_Control Hela_Treatment
# of MS runs                          3              3
# of Biological Replicates            3              3
# of Technical Replicates             1              1
```

5.5 aLFQ

The package `aLFQ` can read the original OpenSWATH output. Alternatively the `aLFQ` package can also be applied to the filtered and annotated data from the `SWATH2stats` package. To convert the data after filtering to the format for aLFQ, the function `convert4aLFQ` is applied to the transition-level data.

```
> aLFQ.input <- convert4aLFQ(data.transition)
> library(aLFQ)
> prots <- ProteinInference(aLFQ.input, peptide_method = 'top',
+                           peptide_topx = 3,
+                           peptide_strictness = 'loose',
+                           peptide_summary = 'mean',
+                           transition_topx = 3,
+                           transition_strictness = 'loose',
+                           transition_summary = 'sum',
+                           fasta = NA, model = NA,
+                           combine_precursors = FALSE)
```

5.6 mapDIA

In order to convert the data into the format for the mapDIA program, the function `convert4mapDIA` is used. Technical replicates included in the data are not taken into account by mapDIA. Therefore the function `convert4mapDIA` averages the SWATH data from technical replicates if contained.

```
> mapDIA.input <- convert4mapDIA(data.transition, RT =TRUE)
> head(mapDIA.input)
```

```
  ProteinName PeptideSequence         FragmentIon Hela_Control_1
1    Protein2       TVVVAFLGR 1000686_TVVVAFLGR_2           1930
2    Protein2       TVVVAFLGR 1000688_TVVVAFLGR_2            924
3    Protein2       TVVVAFLGR 1000689_TVVVAFLGR_2           3264
4    Protein2       TVVVAFLGR 1000693_TVVVAFLGR_2           5681
5    Protein2       TVVVAFLGR 1000695_TVVVAFLGR_2           1386
6    Protein2       TVVVAFLGR 1000697_TVVVAFLGR_2            924
  Hela_Control_2 Hela_Control_3 Hela_Treatment_1 Hela_Treatment_2
1           5973           2792              378            14198
2           3192           1785               84             6535
3          10092           4130              483            28929
4          16316           6745             1025            45055
5           4573           1995              294            12174
6           3856           1512              147            11405
  Hela_Treatment_3       RT
1             6494 68.68395
2             3401 68.68395
3            12443 68.68395
4            20683 68.68395
5             4945 68.68395
6             5165 68.68395

> write.table(mapDIA.input, file='mapDIA.txt', quote=FALSE,
+           row.names=FALSE, sep='\t')
```

5.7 imsbInfer

The package imsbInfer needs all the columns from the OpenSWATH output, therefore the function `reduce_OpenSWATH_output` needs to be omitted in the workflow (see above). If the package imsbInfer should be used after SWATH2stats, a decoy column needs to be added as exemplified below if it has been removed by the filtering functions.

```
> data.annotated.full <- sample_annotation(OpenSWATH_data, Study_design)
> data.annotated.full <- filter_mscore(data.annotated.full,
+                                      mscore4assayfdr(data.annotated.full, 0.01))
> data.annotated.full$decoy <- 0 ### imsbInfer needs the decoy column
> library(imsbInfer)
> specLib <- loadTransitonsMSExperiment(data.annotated.full)
```

6 Acknowledgments

We want to acknowledge Koh Ching Chiek, and Dr. Olga Schubert for help in testing the usability of this package and helpful comments to implementation and description. We want to acknowledge George Rosenberger for discussion and advice to bioinformatic questions and the OpenSWATH + pyProphet workflow.

7 Software and tools

`OpenSWATH`: http://www.openswath.org
`MSstats`: MSstats is available on Bioconductor (www.bioconductor.org) or on

www.mssstats.org
`aLFQ`: aLFQ is available on CRAN (https://cran.r-project.org)
`mapDIA`: mapDIA is available on Sourceforge (http://sourceforge.net/projects/mapdia/)
`imsbInfer`: imsbInfer is available on Github (https://github.com/wolski/imsbInfer)
`msproteomicstools`: https://github.com/msproteomicstools

8 References

Benjamini, Y., and Hochberg, Y. (1995). Controlling the False Discovery Rate: a Practical and Powerful Approach to Multiple Testing. J. R. Statist. Soc. B, 57(1), 289-300.

Choi, H., and Nesvizhskii, A. I. (2008). False discovery rates and related statistical concepts in mass spectrometry-based proteomics. Journal of Proteome Research, 7(1), 47-50.

Choi, M., et al. (2014). MSstats: an R package for statistical analysis of quantitative mass spectrometry-based proteomic experiments. Bioinformatics 30(17): 2524-2526.

Elias, J. E., and Gygi, S. P. (2007). Target-decoy search strategy for increased confidence in large-scale protein identifications by mass spectrometry. Nature Methods, 4(3), 207-214.

Fermin, D. et al. (2011). Abacus: A computational tool for extracting and pre-processing spectral count data for label-free quantitative proteomic analysis. Proteomics, 11(7), 1340-1345.

Gillet, L., et al. (2012). Targeted data extraction of the MS/MS spectra generated by data-independent acquisition: a new concept for consistent and accurate proteome analysis. Mol Cell Proteomics 11(6).

Kaell, L. et al. (2008). Assigning significance to peptides identified by tandem mass spectrometry using decoy databases. Journal of Proteome Research, 7(1), 29-34.

Nesvizhskii, A. I. (2010). A survey of computational methods and error rate estimation procedures for peptide and protein identification in shotgun proteomics. Journal of Proteomics, 73(11), 2092-123.

Reiter, L.et al. (2009). Protein identification false discovery rates for very large proteomics data sets generated by tandem mass spectrometry. Molecular and Cellular Proteomics : MCP, 8(11), 2405-17.

Reiter, L.et al. (2011). mProphet: automated data processing and statistical validation for large-scale SRM experiments. Nature Methods, 8(5), 430-5.

Rosenberger, G., et al. (2014). A repository of assays to quantify 10,000 human

proteins by SWATH-MS.' Sci Data 1: 140031.

Rosenberger, G., et al. (2014). aLFQ: an R-package for estimating absolute protein quantities from label-free LC-MS/MS proteomics data. Bioinformatics 30(17): 2511-2513.

Rost, H. L., et al. (2014). OpenSWATH enables automated, targeted analysis of data-independent acquisition MS data. Nat Biotechnol 32(3): 219-223.

Teleman, J., et al. (2015). DIANA–algorithmic improvements for analysis of data-independent acquisition MS data. Bioinformatics 31(4): 555-562.

Venable, J. D., et al. (2004). Automated approach for quantitative analysis of complex peptide mixtures from tandem mass spectra. Nat Methods 1(1): 39-45.