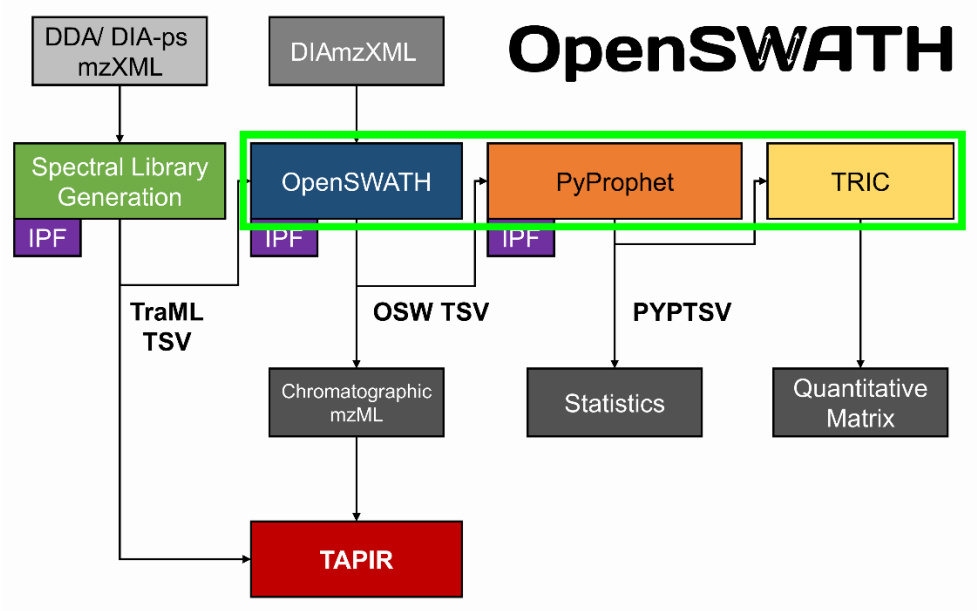
Tutorial 4: DIA data processing using OpenSWATH

1. Introduction

As you can see in the following figure (adapted from the OpenSWATH wiki at http://www.openswath.org), the OpenSWATH workflow comprises three major components namely OpenSWATH [1], PyProphet [2] and TRIC [3].



1) **OpenSWATH**
OpenSWATH as a part of OpenMS enables LC-MS/MS DIA data analysis. This proteomic software has been implemented based on a targeted peptide centric approach to assign peak groups according to its prior knowledge, spectral library, generated in tutorial 1 and 3. Please find more detailed information about OpenSWATH at
http://www.openswath.org/en/latest/docs/openswath.html

2) **PyProphet**
PyProphet is a python-based and optimized reimplementation of the mProphet algorithm, which can analyze large scale datasets generated by OpenSWATH statistically. The conventional version of PyProphet is NOT recommended for very large sample size due to the accumulation of the false targets across samples. To overcome the issue, a new version has been developed, jumbo PyProphet (JPP), which can control the error rate over thousands of samples and conditions globally. In this tutorial, we would focus on the latest version of PyProphet, which is applicable to various datasets. There are two major difference between PyProphet and Jumbo-PyProphet. First, the model generated by JPP using a semi-supervised algorithm is only made once based on all the runs and not for each run separately. Second, the generated model can be applied to the files with three different statistical modes. One of them is called “global” where the error rate can be controlled globally rather than file specific. Hence, we would analyze the data with either run-specific or experiment-wide context (depends on the sample type you want to analyze) and then the results will be filtered based on the results of the global analysis. Please find more detailed information about JPP at More detailed information is available at
http://www.openswath.org/en/latest/docs/pyprophet.html

3) TRIC

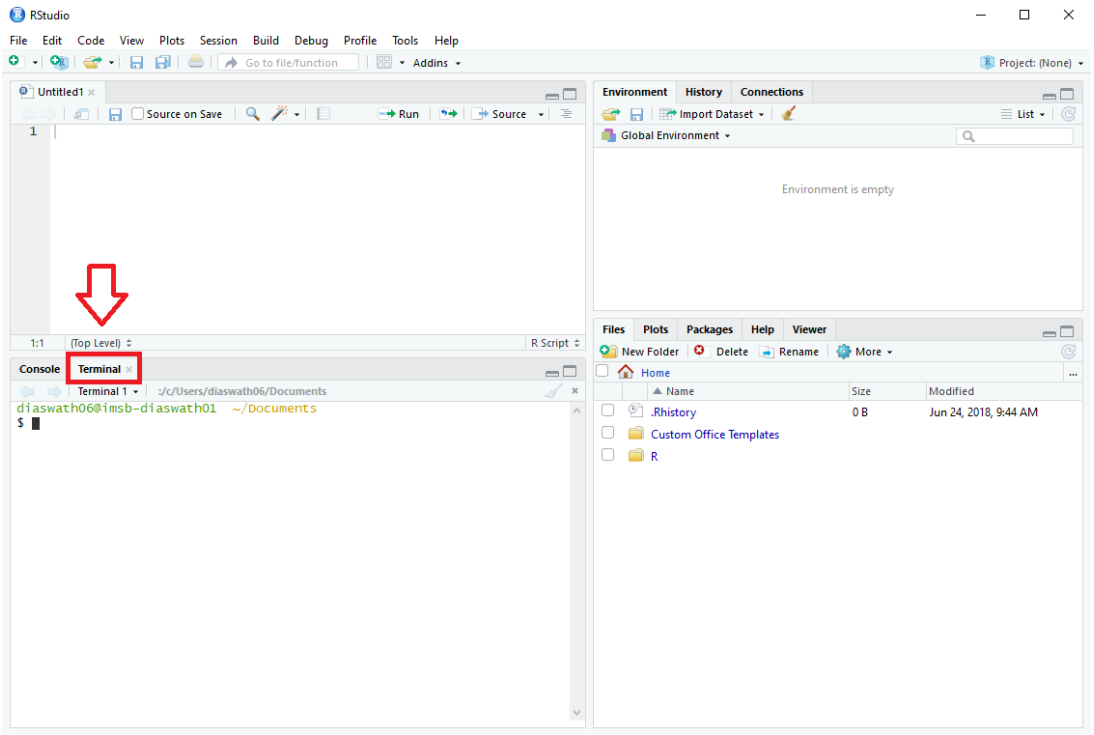
TRIC is an alignment software included in msproteomicstools, a python package, to integrate various information of each run with a graph-based method. There are two basic running modes available. The first one uses a reference-based alignment where a single run is chosen as a reference and all other runs are aligned to it. This is a useful choice for a small number of runs that are chromatographically similar. The second mode generates a guidance tree based on chromatographic similarity of the input runs and uses this tree to align the targeted proteomics runs (the nodes in the tree are runs and the edges are pairwise alignments). Generally this mode is better for a large number of runs or for chromatographically dissimilar samples. Please find more detailed information about TRIC at
http://www.openswath.org/en/latest/docs/tric.html

2. Preparing your working environment

There are two major ways that you can install all the tools required for the OpenSWATH workflow. In the conventional manner, we have to install all of them one by one. This sometimes leads those who have less computational background to issues. However, a user-friendly system through Docker (https://www.docker.com) has been implemented where you can install Docker and pull the required tools down to your PC. For the details, please look at the tutorial overview.

- **Note!** This tutorial has been prepared based on the conventional system. But you can go for the Docker option in your house with the same commands explained below.

- Start by opening your RStudio and direct yourself to its terminal (Please see the screenshot below)



- To run OpenSWATH, we need to convert the raw files (.wiff) to the mzXML format in either profiled or centroid mode. As this step has been explained in the tutorial overview, we here would use the centroid version of the files directly.
- As before, open the bash script "`run_OpenSwathWorkflow.sh`" and change your working place to the following by running line 2

```
cd /c/DIA_Course/Tutorial4_OpenSWATH/
```

3. Run OpenSWATH to assign peak groups

Since OpenSWATH extracts chromatograms and assigns peak groups for runs separately. We can run this tool on every SWATH file by a separate command line where the input file changes each time. However, for a large dataset this is not convenient though. That's why we use a "for" loop to analyze all the runs in one go.

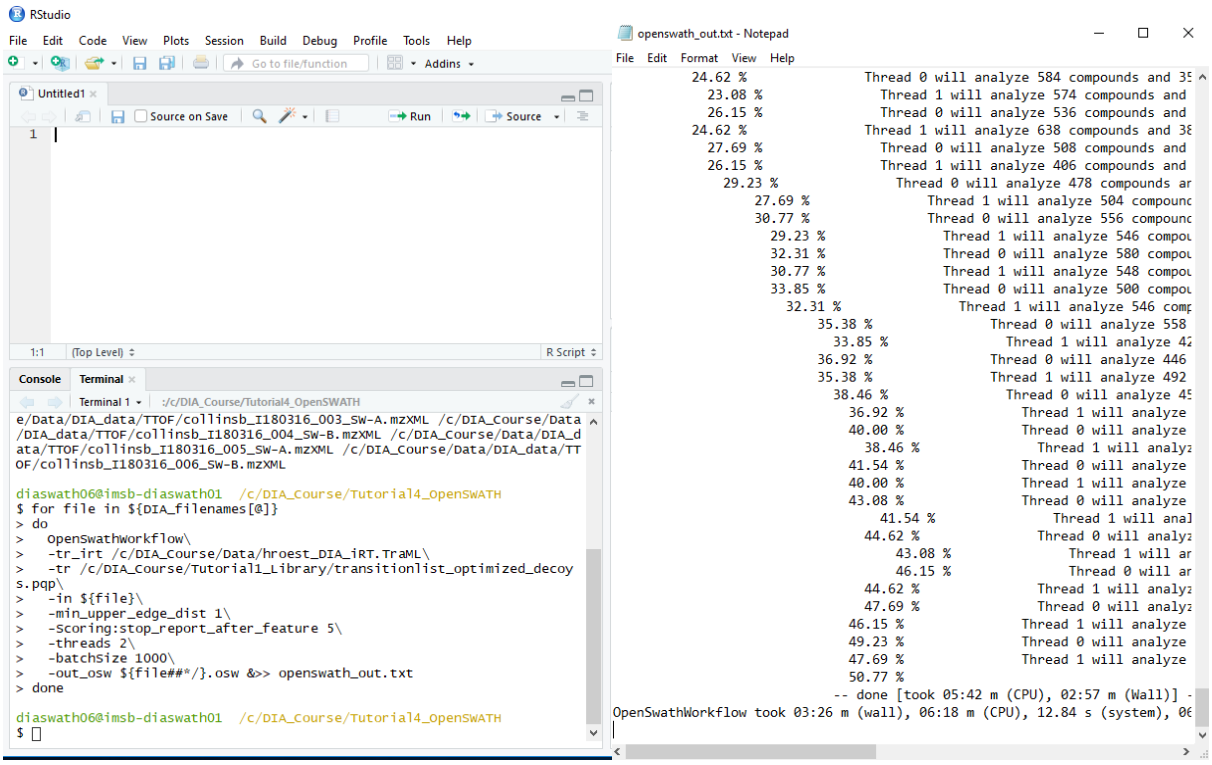
- Copy the OpenSWATH command below in your script at line 16 and run it. The parameters have been explained below. the screenshot of a successful analysis (and its log file) is provided below.
  - **Note!** **Remember to change the file path if you are using QE data.**
  - **Note!** **`min_upper_edge_dist` requires 1 for TTOF and 2 for QE data.**
  - **Note!** The asterisk is used for the variable part of the file name.

```
for file in /c/DIA_Course/Data/DIA_data/TTOF/collinsb*.mzXML
do
  OpenSwathWorkflow \
  -in ${file} \
  -tr /c/DIA_Course/Tutorial1_Library/transitionlist_optimized_decoys.pqp \
  -tr_irt /c/DIA_Course/Data/hroest_DIA_iRT.TraML \
  -batchSize 1000 \
  -min_upper_edge_dist 1 \
  -Scoring:stop_report_after_feature 5 \
  -out_osw $(basename ${file%%.*}).osw \
  -threads 2 \
  &>> Tutorial4_log.txt
done
```

  - **Attention!** Please run this command before going through and understanding the parameters as it takes quite a bit of time to be done.

- In the command above we use the spectral library that was generated from the DDA files (Tutorial 1). Alternatively, you can use the library generated by DIA-Umpire (Tutorial 3). The alternative command is below – it is exactly the same but with an alternative path to the DIA-Umpire based spectral library:

```
for file in /c/DIA_Course/Data/DIA_data/TTOF/collinsb*.mzXML
do
  OpenSwathWorkflow \
  -in ${file} \
  -tr /c/DIA_Course/Tutorial3_DIAUmpire/transitionlist_optimized_decoys.pqp
\
  -tr_irt /c/DIA_Course/Data/hroest_DIA_iRT.TraML \
  -batchSize 1000 \
  -min_upper_edge_dist 1 \
  -Scoring:stop_report_after_feature 5 \
  -out_osw $(basename ${file%%.*}).osw \
  -threads 2 \
  &>> Tutorial4_log.txt
done
```



Parameters and Options	Description
-in	Input file (valid format: mzXML, mzML)
-out_osw	Output file (valid format: osw)
-tr	An assay library containing mass spectrometric and chromatographic coordinates for peptides. This file has been generated during tutorial 1 and 3 (valid format: TraML, tsv, csv)
-threads	Set the number of threads allowed to be used by the TOPP tool
-min_upper_edge_dist	Minimal distance to the edge to still consider a precursor, in Thomson (default: '0')
-tr_irt	Transition file for the iRT peptides (valid format: TraML)
-Scoring:stop_report_after_feature	Stop reporting after feature (ordered by quality; -1 means do not stop). (default: '-1')

- **Note!** You can find all the parameters by running the following commands.

`OpenSwathWorkflow –help` or
`OpenSwathWorkflow --helphelp` for the full list

- You can't see much when OpenSWATH is running on your terminal since all the info is saved into a log file namely Tutorial4_log.txt". You don't necessarily need to know what those reports mean. However, they would be very helpful for any potential troubleshooting. Here you can see a screenshot of a successful OpenSWATH DIA data extraction on mzXML files.

4. PyProphet to analyze the assigned peaks statistically

- **Merging:** once OpenSWATH is finished you can run PyProphet on the SQLight based output files. In most scenarios, there are more than a single file (here, we have six files for instance). Therefore the following commands merges and subsamples osw files respectively.

```
pyprophet merge \
--out=training.osw \
--subsample_ratio=0.33 \
collinsb*.osw \
&>> Tutorial4_log.txt
```

```
pyprophet merge \
--out=merged.osw \
--subsample_ratio=1 \
collinsb*.osw \
&>> Tutorial4_log.txt
```

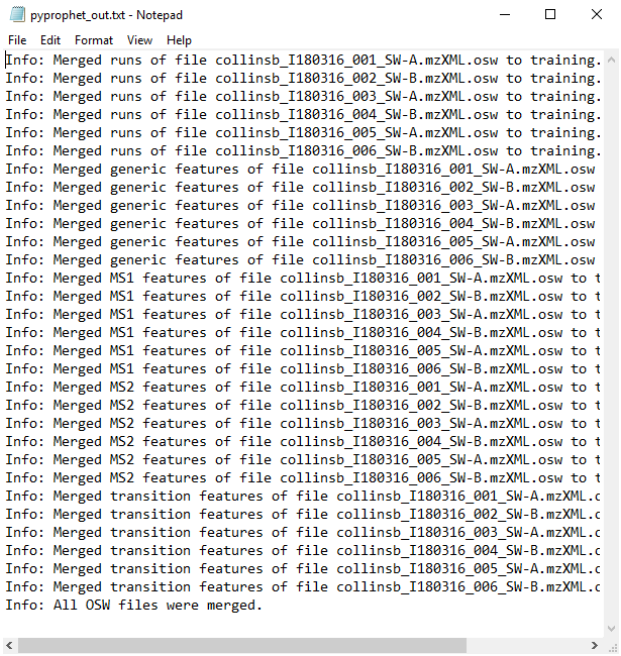
  - **Note!** If semi-supervised learning is not fast enough, you can merge osw files with a smaller subsample_ratio (< 0.33) and apply the created model to the full dataset.

  - **Note!** For more information, please run the commands below.

```
pyprophet --help
pyprophet merge --help
```

Parameters and Options	Description
--out	Merged osw output file
--subsample_ratio	Subsample ratio used per run (default:1)

Here, there is a screenshot of a successful PyProphet merging on the OpenSWATH output files. Check this by opening the “Tutorial4_log.txt” file in Notepad

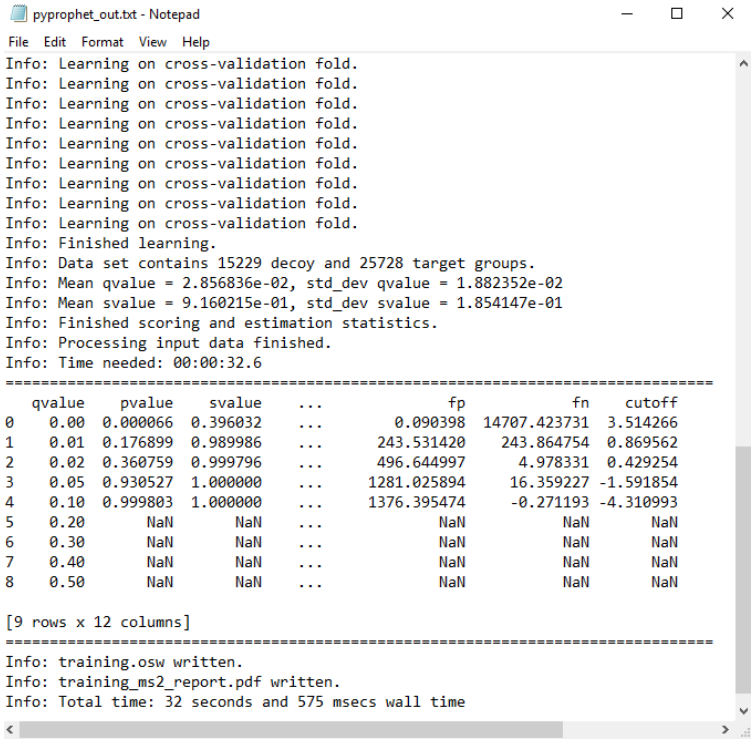


- **Scoring:** The main command will conduct a semi-supervised learning and error-rate estimation in a fully automated fashion. The default parameters are recommended for SCIEX TripleTOF 5600/6600 instrument data, but can be adjusted in other scenarios.
- To compute MS2 scores, please run the following command (please see the screenshot of the log file below)

```
pyprophet score \
--in=training.osw \
--level=ms2 \
&>> Tutorial4_log.txt
```

  - **Note!** For more information on the available options:

```
pyprophet score --help
```



- Now, compute MS2 scores for the merged file including all runs by the following command:

```
pyprophet score \
--in=merged.osw \
--level=ms2 \
--apply_weights=training.osw \
&>> Tutorial4_log.txt
```

- To conduct peptide inference in run-specific, experiment-wide and global contexts, the following command can be applied

```
pyprophet \
peptide --in=merged.osw --context=run-specific \
peptide --in=merged.osw --context=experiment-wide \
peptide --in=merged.osw --context=global \
&>> Tutorial4_log.txt
```

- Similarly, the same analysis can be applied on protein level by the command below:

```
pyprophet \
protein --in=merged.osw --context=run-specific \
protein --in=merged.osw --context=experiment-wide \
protein --in=merged.osw --context=global \
&>> Tutorial4_log.txt
```

  - **Note!** More information can be found through the following commands.

```
pyprophet peptide --help
pyprophet protein --help
```

- Finally, filter the dataset globally on both peptide and protein level and convert the osw format to tsv and also create pdf reports

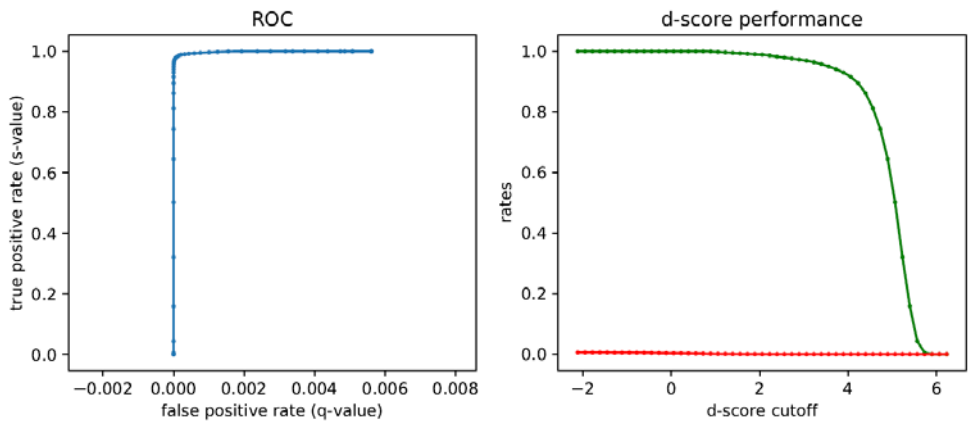
```
pyprophet export \
--in=merged.osw \
--out=merged_export.tsv \
--format=legacy_merged \
--no-ipf \
--max_rs_peakgroup_qvalue 0.1 \
--max_global_peptide_qvalue 0.05 \
--max_global_protein_qvalue 0.01 \
&>> Tutorial4_log.txt

pyprophet export \
--in=merged.osw \
--format=score_plots \
&>> Tutorial4_log.txt
```

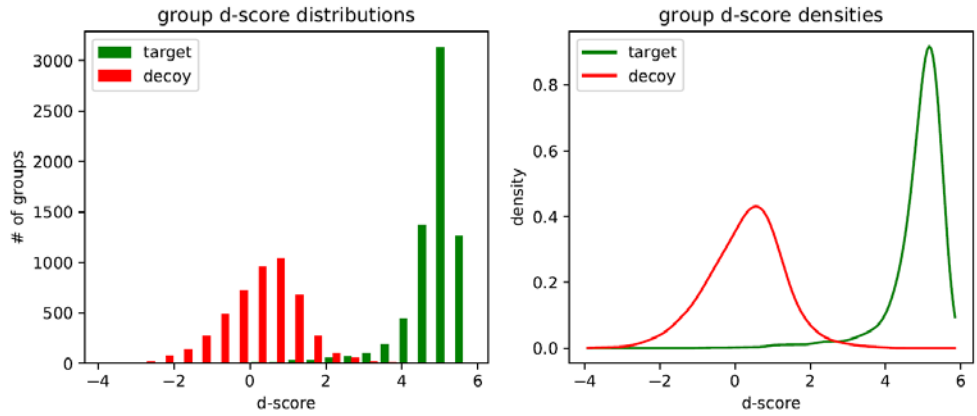
  - **Note!** The data is filtered with a less strict cutoff on peptide level. This gives more flexibility to TRIC to conduct the features alignment optimally.
  - **Note!** Please open the report files (PDFs) and make sure you understand all the plots perfectly. Similar plots are generated at each step of scoring.

- **Question:** How many proteptypic peptides and proteins have been identified? To count the number of protein and peptide, please open your output file (merged_export.tsv) in excel. If you have an extra '\r' at the end of each line, please remove them quickly by notepad++ before importing to Excel.

- **Question:** What does the following ROC curve mean? What do you expect to see if the data quality was not so high? If the size of your spectral library (sample specific vs pan-spectral library) can affect the following plot? Why?



- **Question:** What is the underlying reason that decoys and targets are separated perfectly in the analysis? Please see the plot below. In which scenario, you expect a higher overlap?



5. TRIC to integrate runs based on various types of data

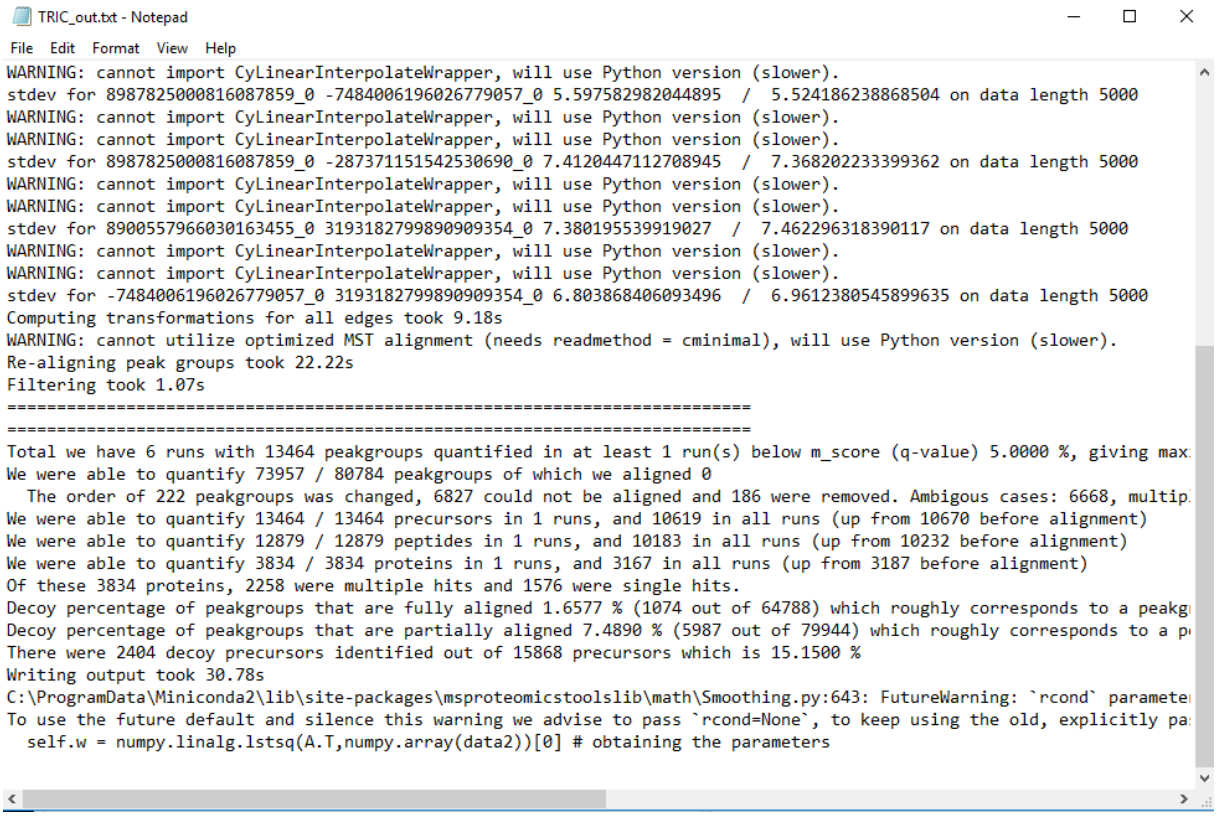
TRIC is run on "merged_export.tsv" file by the following command. This command conduct a tree based alignment (LocalMST) where the pairwise alignment is performed by a non-linear method (lowess). This mode of the alignment is recommended for a large number of runs or for chromatographically dissimilar samples. Please see the screenshot of a successful run below:

- Copy and run the following command:

```
feature_alignment.py \
--in merged_export.tsv \
--out aligned.tsv \
--method LocalMST \
--realign_method lowess \
--max_rt_diff 60 \
--mst:useRTCorrection True \
--mst:Stdev_multiplier 3.0 \
--target_fdr -1 \
--fdr_cutoff 0.05 \
--max_fdr_quality -1 \
--alignment_score 0.05  \
&>> Tutorial4_log.txt
```

Parameters and Options	Description
--in	Input files (valid format: tsv)
--out	Output file
--file_format	Input file format (openswath, mprophet, peakview)
--method	Method to use for the clustering (best_overall, best_cluster_score, global_best_cluster_score, global_best_overall, LocalMST, LocalMSTALLCluster)

--max_rt_diff	Maximum difference in RT for two aligned features
--target_fdr	If parameter estimation is used, which target FDR should be optimized for. If set to lower than 0, parameter estimation is turned off.
--max_fdr_quality	Extension m-score score cutoff, peakgroup of this quality will still be considered for alignment during extension
--mst:useRTCorrection	Use aligned peakgroup RT to continue threading
--mst:Stdev_multiplier	How many standard deviations the peakgroup can deviate in RT during the alignment.
--alignment_score	Minimal score needed for a feature to be considered for alignment between runs
--realign_method	RT alignment method (diRT, linear, splineR, splineR_external, splinePy, lowess, nonCVSpline, CVSpline, Earth)
--matrix_output_method	Which columns are written besides Intensity (none, RT, score, source, full)
--dscore_cutoff	Discard all peakgroups below this d-score
--frac_selected	Do not write peakgroup if selected in less than this fraction of runs (range 0 to 1)
--disable_isotopic_grouping	Disable grouping of isotopic variants by peptide group label
--out_matrix	Matrix containing one peakgroup per row (.csv, .tsv, .xlsx)
--out_meta	Outfile containing meta information



- **Note!** You can find all the parameters by running the following command.

```
feature_alignment.py --help
```

- **Question:** Once the TRIC alignment is done, please open the output file (aligned.tsv) via Excel and count the number of protein IDs for each organism.

References:

1. Röst, H. L. *et al.* OpenSWATH enables automated, targeted analysis of data-independent acquisition MS data. *Nat. Biotechnol.* **32,** 219–223 (2014).
2. Rosenberger, G. *et al.* Statistical control of peptide and protein error rates in large-scale targeted data-independent acquisition analyses. *Nat. Methods* **14,** 921–927 (2017).
3. Röst, H. L. *et al.* TRIC: an automated alignment strategy for reproducible protein quantification in targeted proteomics. *Nat. Methods* **13,** 777–783 (2016).

We would like to thank SystemsX for supporting the Zurich DIA / SWATH Course 2018.

SystemsX.ch
The Swiss Initiative in Systems Biology