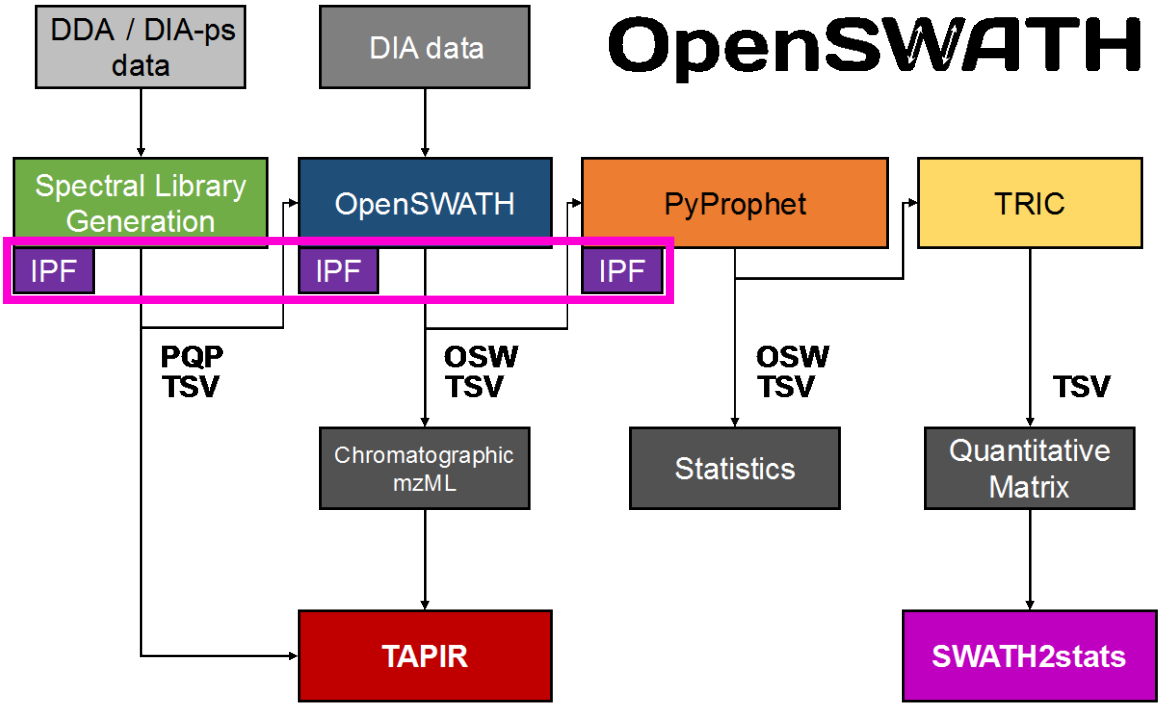
Tutorial 6: Inference of PeptidoForms using IPF

1. Introduction

Inference of PeptidoForms (IPF)[1] is an extension to the OpenSWATH workflow that improves specificity for the detection of peptidoforms, closely related peptides with similar or identical amino acid sequence and configuration of posttranslational modifications (PTMs), e.g. positional phosphopeptide isomers. Since IPF is fully integrated into OpenMS[2]/OpenSWATH[3], PyProphet[4,5] and compatible with TRIC[6] and downstream analysis tools, only few parameters need to be modified to conduct an IPF analysis. For this tutorial, we will use the same virtual environment as for the OpenSWATH part. We will focus on the following steps:



1) **Spectral Library Generation**
   Spectral libraries for IPF are generated as discussed in the previous tutorials by processing DDA data or directly from DIA data via DIA-Umpire[7,8], the TPP and SpectraST[9]. The important difference is that here we will use the OpenMS tool OpenSwathAssayGenerator in IPF mode to append *in silico* generated identification transitions to the empirical spectral library. Further information can be found at http://openswath.org/en/latest/docs/pqp.html

2) **OpenSWATH**
   OpenSWATH as a part of OpenMS enables LC-MS/MS DIA data analysis. OpenSWATH will detect candidate peak groups identically to the original implementation, however it will additionally extract and individually score all identification transitions against the candidate peak groups. Please find more detailed information about OpenSWATH at http://www.openswath.org/en/latest/docs/openswath.html

3) **PyProphet**
   PyProphet is a reimplementation and extension of the mProphet[10] algorithm, which statistical validation of the OpenSWATH results. In this tutorial, we will use the IPF extension that is fully integrated since PyProphet 2.0. Please find more detailed information about PyProphet at http://www.openswath.org/en/latest/docs/pyprophet.html

4) **TRIC**
   TRIC is an alignment software included in the Python package msproteomicstools, which transfers the detection confidence across multiple runs with a graph-based method. While we will not use TRIC for this tutorial, it is fully compatible with IPF. Please find more detailed information about TRIC at http://www.openswath.org/en/latest/docs/tric.html whereas IPF-specific parameters can be found at http://openswath.org/en/latest/docs/ipf.html

We will analyze a subset (3 replicates each) of the enriched U2OS phosphoproteomic data set[1]. U2OS cell lines were either treated with nocodazole or left untreated (control). Nocodazole arrests cells at the mitotic stage and has a substantial effect on signaling pathways involving phosphorylation[1]. To probe various signaling pathways, we will use a sample-specific reduced library focusing on the proteins identified in the P100[11,12] assay panel that provides peptide query parameter for ~100 phosphoproteins covering those pathway as "sentinel"[13] assays.

2. Prepare the working environment

- Please open "`run_IPF.sh`" bash script from the Tutorial-6_IPF folder as we have done it for the previous command line Tutorials.
- First change the working directory by running the first command in your script (line 2):

```
cd /c/DIA_Course/Tutorial6_IPF/
```

  - **Note!** Please make sure you are back in the Terminal window of R Studio.

3. Generate the spectral library for IPF

As many peptide-centric scoring algorithms, IPF requires a spectral library to derive peptide query parameters for targeted data extraction of DIA data sets. Here, we will use the tool "OpenSwathAssayGenerator" to do this. Two main input files are required: The spectral library "pqp_p100.tsv" is the spectral library that was generated as part of the original study, but it was prefiltered to contain only the proteins covered by the P100 assays. In general, any spectral library generated by DDA (e.g. TPP or MaxQuant, etc.) or DIA pseudo-spectra (e.g. DIA-Umpire) will work. The second file is a modified "unimod.xml" file that contains the information about residue modifiability. Unimod is a database that stores this information, but also contains non-canonical annotations, for example phosphorylation is a valid PTM for

S,T,Y,D,H,C,R,K instead of the canonical S,T,Y. This increases the search space substantially and thus it is highly recommended to restrict the options. For this tutorial, we provide a modified XML file, but if other PTM types are investigated, the file should be edited.

- Copy the following command on line 5 in your bash script and execute it:

```
OpenSwathAssayGenerator \
-in pqp_p100.tsv \
-out pqp_p100_ipf.TraML \
-enable_ipf \
-unimod_file unimod.xml \
&>> Tutorial6_log.txt
```

- This will generate the file “pqp_p100_ipf.TraML”. Now append decoys and then convert the file to the “PQP” format:

```
OpenSwathDecoyGenerator \
-in pqp_p100_ipf.TraML \
-out pqp_p100_ipf_decoys.TraML \
&>> Tutorial6_log.txt

TargetedFileConverter \
-in pqp_p100_ipf_decoys.TraML \
-out pqp_p100.pqp \
&>> Tutorial6_log.txt
```

- The file “pqp_p100.pqp” will be the input for “OpenSwathWorkflow” in the next step. Most often, the default parameters for “OpenSwathAssayGenerator” and “OpenSwathDecoyGenerator” should be used, however there are a few options worth exploring if you use IPF for different scenarios.

Parameters and Options	Description
-in	Input file (valid formats: 'tsv', 'mrm', 'pqp', 'TraML')
-out	Output file (valid formats: 'tsv', 'pqp', 'TraML')
-min_transitions	Minimal number of transitions (default: '6')
-max_transitions	Maximal number of transitions (default: '6')
-allowed_fragment_types	Allowed fragment types (default: 'b,y')
-allowed_fragment_charges	Allowed fragment charge states (default: '1,2,3,4')
-enable_detection_specific_losses	Set this flag if specific neutral losses for detection fragment ions should be allowed
-enable_detection_unspecific_losses	Set this flag if unspecific neutral losses (H2O1, H3N1, C1H2N2, C1H2N1O1) for detection fragment ions should be allowed
-precursor_mz_threshold	MZ threshold in Thomson for precursor ion selection (default: '0.025')
-precursor_lower_mz_limit	Lower MZ limit for precursor ions (default: '400')

-precursor_upper_mz_limit	Upper MZ limit for precursor ions (default: '1200')
-product_mz_threshold	MZ threshold in Thomson for fragment ion annotation (default: '0.025')
-product_lower_mz_limit	Lower MZ limit for fragment ions (default: '350')
-product_upper_mz_limit	Upper MZ limit for fragment ions (default: '2000')
-swath_windows_file	Tab separated file containing the SWATH windows for exclusion of fragment ions falling into the precursor isolation window
-unimod_file	(Modified) Unimod XML file (http://www.unimod.org/xml/unimod.xml) describing residue modifiability (valid formats: 'xml')
-enable_ipf	IPF: set this flag if identification transitions should be generated for IPF. Note: Requires setting 'unimod_file'.
-max_num_alternative_localizations	IPF: maximum number of site-localization permutations (default: '10000')
-disable_identification_ms2_precursors	IPF: set this flag if MS2-level precursor ions for identification should not be allowed for extraction of the precursor signal from the fragment ion data (MS2-level).
-disable_identification_specific_losses	IPF: set this flag if specific neutral losses for identification fragment ions should not be allowed
-enable_identification_unspecific_losses	IPF: set this flag if unspecific neutral losses (H2O1, H3N1, C1H2N2, C1H2N1O1) for identification fragment ions should be allowed
-enable_swath_specifity	IPF: set this flag if identification transitions without precursor specificity (i.e. across whole precursor isolation window instead of precursor MZ) should be generated.

- **Note!** You can find all the parameters by running the following commands.
  `OpenSwathAssayGenerator –help` or
  `OpenSwathAssayGenerator --helphelp` for the full list

- **Attention!** If you change the fragment ion annotation parameters in OpenSwathAssayGenerator, they should also be changed in OpenSwathDecoyGenerator.

4. Run OpenSWATH on each LC-MS/MS run

- Using the spectral library generated in the step above, we will run "OpenSwathWorkflow" similarly as described in Tutorial 4, however we will add a few parameters to append IPF scores:
  - **Attention!** OpenSWATH will need crucially longer to run with IPF parameters. Therefore please make yourself aware with the new parameters. You can also start running the command, but then kill it by pressing CTRL and C.
- In the Backup folder DIA_Course_completed you can find the completed osw files. Please copy them into your own Tutorial-6_IPF folder.

```
for run in *.mzML.gz
   do
     OpenSwathWorkflow \
     -in $run \
     -tr pqp_p100.pqp \
     -tr_irt iRTkit.TraML \
     -batchSize 10 \
     -readOptions workingInMemory \
     -Scoring:stop_report_after_feature 5 \
     -min_upper_edge_dist 1 \
     -use_ms1_traces \
     -enable_uis_scoring \
     -Scoring:Scores:use_uis_scores \
     -Scoring:Scores:use_ms1_mi \
     -Scoring:Scores:use_mi_score \
     -Scoring:Scores:use_total_mi_score \
     -out_osw ${run%%.*}.osw \
     -threads 1 \
     &>> Tutorial6_log.txt
   done
```

Parameters and Options	Description
-batchSize	We set this parameter to a lower value, because OpenSwathWorkflow now needs to extract many more transitions per peptide.
-use_ms1_traces	We extract precursor ion chromatograms from the MS1 maps.
-enable_uis_scoring	We enable scoring on transition-level for IPF.
-Scoring:Scores:use_ms1_mi	Enable MS1-level mutual information-based score
-Scoring:Scores:use_mi_score	Enable MS2 peak group and transition-level mutual information-based score
-Scoring:Scores:use_total_mi_score	Enable MS2 peak group and transition-level total mutual information-based score

  - **Note!** You can find all the parameters by running the following commands. `OpenSwathWorkflow –help` or `OpenSwathWorkflow --helphelp` for the full list

5. PyProphet to analyze the assigned peaks statistically

- Start here again with running the rest of the Tutorial with the copied OpenSWATH output and merge all files by running:

```
pyprophet merge \
--out=merged.osw \
*.osw \
&>> Tutorial6_log.txt
```

- Then score all runs on MS1, MS2 and transition-level.
    - **Note!** We use “feature_id” as group_id, since we are not sure about the specificity of the peptide query parameters, e.g. one set of peptide query parameters might detect several positional isomers of the targeted phosphopeptide.

```
pyprophet score \
--in=merged.osw \
--level=ms1 \
--group_id=feature_id \
&>> Tutorial6_log.txt

pyprophet score \
--in=merged.osw \
--level=ms2 \
--group_id=feature_id \
&>> Tutorial6_log.txt

pyprophet score \
--in=merged.osw \
--level=transition \
&>> Tutorial6_log.txt
```

- Export also the reports:

```
pyprophet export \
--format=score_plots \
--in=merged.osw \
&>> Tutorial6_log.txt
```

Have a look at the PDF reports and compare the target and decoy distributions with the OpenSWATH results of Tutorial 4.

- Next, we will integrate all evidence with the following command:

```
pyprophet ipf \
--in=merged.osw \
&>> Tutorial6_log.txt
```

    - **Note!** You can find all the parameters by running the following command. `pyprophet score –-help` or `pyprophet ipf –-help`

6. Export data to matrix format and run mapDIA

- In the last step, we will export the IPF results from the SQLite-based OSW format to a TSV table in matrix representation with the following command:

```
pyprophet export \
   --format=matrix \
   --in=merged.osw \
   --max_rs_peakgroup_qvalue=0.05 \
   &>> Tutorial6_log.txt
```

  - **Note!** In the standard OpenSWATH tutorial we have exported the a tsv file. Here we now use the option to directly export a quantification matrix. If you want to see the difference you open both files in excel to compare the different formats.

Parameters and Options	Description
--format	Export format, either matrix, legacy (mProphet/PyProphet) or score_plots format. [default: legacy]
--max_rs_peakgroup_qvalue	[format: matrix] Filter results to maximum run-specific peak group-level q-value. [default: 0.01]

  - **Note!** You can find all the parameters by running the following command. `pyprophet export –-help`

Now you are done with the IPF analysis. Now reformat the matrix and apply mapDIA to the data set to identify the phosphorylated proteins that change most between the two conditions.

- First apply some data curation to only extract information that mapDIA requires:

```
cat merged.tsv | awk -F'\t' '{OFS="\t"; print $4, $1, $5, $7, $9,
$6, $8, $10}' > mapDIA_input.tsv
```

- Now run mapDIA:

```
./mapDIA_win64.exe mapDIA.params
```

- This will generate a table called analysis_output.txt that can be opened in Excel. Sort the list according to descending FDR. You can look-up the top protein identifiers at http://www.uniprot.org. Remember the experimental design of the introduction. Does this make sense?

References:

1. Rosenberger, G. *et al.* Inference and quantification of peptidoforms in large sample cohorts by SWATH-MS. *Nat. Biotechnol.* **35,** 781–788 (2017).
2. Röst, H. L. *et al.* OpenMS: a flexible open-source software platform for mass spectrometry data analysis. *Nat. Methods* **13,** 741–748 (2016).
3. Röst, H. L. *et al.* OpenSWATH enables automated, targeted analysis of data-independent acquisition MS data. *Nat. Biotechnol.* **32,** 219–223 (2014).
4. Teleman, J. *et al.* DIANA-algorithmic improvements for analysis of data-independent acquisition MS data. *Bioinformatics* **31,** 555–562 (2015).
5. 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).
6. Röst, H. L. *et al.* TRIC: an automated alignment strategy for reproducible protein quantification in targeted proteomics. *Nat. Methods* **13,** 777–783 (2016).
7. Tsou, C.-C. *et al.* DIA-Umpire: comprehensive computational framework for data-independent acquisition proteomics. *Nat. Methods* **12,** 258–264 (2015).
8. Tsou, C. C., Tsai, C. F., Teo, G. C., Chen, Y. J. & Nesvizhskii, A. I. Untargeted, spectral library-free analysis of data-independent acquisition proteomics data generated using Orbitrap mass spectrometers. *Proteomics* **16,** 2257–2271 (2016).
9. Schubert, O. T. *et al.* Building high-quality assay libraries for targeted analysis of SWATH MS data. *Nat. Protoc.* **10,** 426–441 (2015).
10. Reiter, L. *et al.* mProphet: automated data processing and statistical validation for large-scale SRM experiments. *Nat. Methods* **8,** 430–435 (2011).
11. Keenan, A. B. *et al.* The Library of Integrated Network-Based Cellular Signatures NIH Program: System-Level Cataloging of Human Cells Response to Perturbations. *Cell Syst.* **0,** (2017).
12. Litichevskiy, L. *et al.* A Library of Phosphoproteomic and Chromatin Signatures for Characterizing Cellular Responses to Drug Perturbations. *Cell Systems* **0,** (2018).
13. Soste, M. *et al.* A sentinel protein assay for simultaneously quantifying cellular processes. *Nat. Methods* **11,** 1045–1048 (2014).

Congratulation! You completed the whole DIA/SWATH Course 2018. We hope you will now be able to apply such a workflow similarly on your own data. ☺

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

SystemsX.ch
The Swiss Initiative in Systems Biology