

Contents

I. Supplementary Note 1: Overview of OpenMS	2
A. OpenMS Software	2
B. Improvements since OpenMS 1.0	2
C. OpenMS resources	3
D. OpenMS pre-built workflows	3
E. OpenMS supported data formats	6
II. Supplementary Note 2: Available tools	9
A. Graphical Tools	9
B. File Handling	9
C. Signal Processing and Preprocessing	10
D. Quantitation	11
E. Map Alignment	12
F. Protein/Peptide Identification	12
G. Protein/Peptide Processing	13
H. Targeted Experiments	14
I. Peptide Property Prediction	15
J. File Handling	15
K. Algorithm Evaluation	16
L. Quantitation	16
M. Metabolite Identification	17
N. Quality Control	17
O. Misc	17
III. Supplementary Note 3: Case studies	19
A. SWATH Analysis	19
B. Degradomics	19
C. Proteogenomics	21
References	22

I. SUPPLEMENTARY NOTE 1: OVERVIEW OF OPENMS

A. OpenMS Software

OpenMS uses modern software engineering concepts more commonly found in industry settings than in academic environments. The project places great emphasis on modularity, reusability and extensive testing (using continuous integration) leading to high code quality. The modular architecture of OpenMS tries to build upon existing standard libraries as much as possible, relying on them for sequence analysis, XML parsing, numerical computations and statistics.

The library itself contains over 1300 C++ classes representing core concepts in mass spectrometry and the corresponding ontologies defined by the Human Proteome Organization Proteomics Standard Initiative (HUPO-PSI) [1]. Modern object-oriented, template-based C++ is used exclusively throughout the code base, encapsulating raw data structures and discouraging manual heap-based memory management, thus providing robust and error-tolerant code. Coding conventions are enforced and extensive English documentation is available for several thousand C++ functions part of the public API.

All development is performed in the open, using a public source code repository and ticketing system. Stringent code reviews and continuous integration, running a multitude of functional, unit and black-box tests, ensure continued support, robustness and correctness of the code. Strict abstraction is used to hide implementation details from the user and support classes tailored to specific algorithms.

B. Improvements since OpenMS 1.0

When the first version of OpenMS was released in 2007, it was a pure C++ software library [2] and thus targeting software developers exclusively. Since then, the project has fundamentally changed in many respects and has grown far beyond a software library. In 2016, OpenMS has become a flexible collection of tools, workflows and algorithms with a strong focus on end-users and the community of MS practitioners. The recent 2.0 release of OpenMS consists of over 180 executable end-user tools fully integrated into graphical workflow managers (such as Galaxy or KNIME). The resulting workflows can be executed on computing infrastructure as diverse as laptop computers and high-performance computing

clusters. Over the last decade, the OpenMS project has also expanded in scope and functionality, adding algorithms for high-resolution shotgun proteomics, targeted proteomics, isotopic labelling, cross-linking, metaproteomics, and metabolomics analysis while providing a new Python interface to gain access to additional developers. In comparison to the initial C++ software library of limited scope, OpenMS is now a community-driven initiative which additionally provides teaching outreach, scientific collaboration and dissemination of open source software and is a major driver for the adaptation of open standards and transparent, open-source software in computational mass spectrometry. The OpenMS project has thus matured into an algorithmic cornerstone for a large number of data analysis workflows in the field and has been recognized by the community as an easy-to-use, flexible tool for all mass spectrometric data analysis needs.

C. OpenMS resources

The OpenMS project provides an extensive set of resources which can be obtained online. Table I displays an overview over these resources, starting with the OpenMS webpage, the binary downloads of the software and the user tutorials which explain the major features of OpenMS to a new user.

D. OpenMS pre-built workflows

The OpenMS project contains several pre-built workflows which can be accessed at <http://www.openms.de/workflows> and are further described in Table II.

TABLE I: Available resources from the OpenMS project

Resource	Description	URL
Web page	OpenMS home page	http://open-ms.de/
Download	Download page to obtain OpenMS binaries	http://open-ms.de/downloads/
Tutorials	User-oriented tutorials for OpenMS	https://github.com/Tutorials
Workflows	Workflows for OpenMS	http://www.openms.de/workflows
Mailing List	Public mailing list for questions relating to the use and features of OpenMS	https://lists.sourceforge.net/lists/listinfo/open-ms-general/
Source code	Complete source code of the OpenMS project	https://github.com/OpenMS/OpenMS
Documentation	Complete documentation of all public classes and functions of OpenMS, including build instructions, developer guide	http://ftp.mi.fu-berlin.de/pub/OpenMS/release-documentation/html/index.html
Bug tracker	List of currently known bugs and missing features (open to the public and preferred way of reporting issues with OpenMS)	https://github.com/OpenMS/OpenMS/issues
KNIME Forum	OpenMS section of the KNIME Forum which gives advice to issues encountered with the KNIME nodes	https://tech.knime.org/forum/openms

TABLE II: Available workflows in OpenMS

Workflow	Engine	Description	Ref.
Peptide Identification	KNIME	Basic peptide identification	
Peptide Identification	KNIME	Consensus peptide identification with multiple search engines	
Labelfree	KNIME	Label free quantification workflow suitable for large scale experiments	[3]
Labelfree	KNIME	Label free quantification and identification workflow	[3]
Protein Inference	KNIME	Label free quantification, identification and protein inference	
OpenSWATH	KNIME	SWATH MS analysis workflow	[4]
Metabolomics	KNIME	Small molecule identification and quantification	[5]
Isobaric label	TOPPAS	iTRAQ quantification and identification	

E. OpenMS supported data formats

The OpenMS project supports a large set of over 25 different data formats, which can be read by OpenMS tools and converted to more common data formats (see Table III). The supported formats range from a large number of data formats for raw spectral data to spectral library data formats to HUPO PSI standard formats for identification and quantification information. This allows OpenMS to support a large number of different workflows, integrate with different tools or act as intermediate tools to convert from a specific format to a desired data format.

In addition, OpenMS supports over 15 different external tools directly which allows the user in many cases to execute a specific external tool directly from within the OpenMS workflow environment. OpenMS is capable of writing input data and reading output format for the majority of these tools, as indicated in Table IV.

TABLE III: Supported data formats in OpenMS

Data format	Support	Tool support ^a
.mzML	Full	Tools reading and writing raw data
.featureXML	Full	Tools reading and writing feature data
.consensusXML	Full	Tools reading and writing feature data
.ini	Full	All tools
.toppas	Full	TOPPAS workflow manager
.trafoXML	Full	All relevant tools
.idXML	Full	All tools reading and writing identification data
.mzQuantML	Full	IsobaricAnalyzer, FeatureFinder tools
.TraML	Full	All OpenSWATH tools, conversion tools
.qcML	Full	QC tools
.mzTab	Write	All metabolite tools, MzTabExporter
.mzIdentML	Read and Write	IDFileConverter
.pep.xml	Read and Write	IDFileConverter and various adapters
.prot.xml	Read	IDFileConverter
.peplist	Read	FileConverter
MSPFile	Read and Write	SpecLibCreator, SpecLibSearcher
.mzXML	Read and Write	FileConverter
.mzData	Read and Write	FileConverter
.mgf	Read and Write	FileConverter
.dta	Read and Write	FileConverter
.dta2d	Read and Write	FileConverter
.edta	Read and Write	FileConverter
.fasta	Read and Write	multiple
.obo	Read	CVInspector, SemanticValidator
CV Mapping (.xml)	Read	CVInspector, SemanticValidator

^aSome file formats can only be used in OpenMS after conversion to a suitable format, e.g. .mzXML has to be converted to the current standard format .mzML before further processing.

TABLE IV: Supported software and associated formats in OpenMS

Software tool	Support	Direct Execution ^a	Tool support
Fido	Input and Output	Yes	FidoAdapter
InsPecT	Input and Output	Yes	InspectAdapter
Luciphor	Input and Output	Yes	LuciphorAdapter
Mascot	Input and Output	Yes	MascotAdapter, MascotAdatppterOnline
MSGFPlus	Input and Output	Yes	MSGFPlusAdapter
MyriMatch	Input and Output	Yes	MyriMatchAdapter
OMSSA	Input and Output	Yes	OMSSAAdapter
PepNovo	Input and Output	Yes	PepNovoAdapter
XTandem	Input and Output	Yes	XTandemAdapter
Percolator	Input and Output	Yes	TopPerc
TPP	Input and Output	No	various (pep and prot xml)
SpetraST	Read	No	ConvertTSVToTraML (MRM file format)
Hardkloer	Read	No	FileConverter (Kroenik file format)
UniMod XML	Read	N/A	multiple
SEQUEST	Input and Output	No	IDFileConverter

^aDirect execution means that these tools can be directly incorporated into an OpenMS workflow. For all other cases, OpenMS is able to read the output of the tool after separate execution.

II. SUPPLEMENTARY NOTE 2: AVAILABLE TOOLS

The following listing provides an overview of the available tools in OpenMS with a short description of their function. In total, over 180 individual tools are available to the user, please see the online listing of the TOPP tools and the UTILS for and up-to-date list of available tools.

A. Graphical Tools

- TOPPView - A viewer for mass spectrometry data.
- TOPPAS - An assistant for GUI-driven TOPP workflow design.
- INIFileEditor - An editor for OpenMS configuration files.

B. File Handling

- DTAExtractor - Extracts spectra of an MS run file to several files in DTA format.
- FileConverter - Converts between different MS file formats.
- FileFilter - Extracts or manipulates portions of data from peak, feature or consensus feature files.
- FileInfo - Shows basic information about the file, such as data ranges and file type.
- FileMerger - Merges several MS files into one file.
- IDMerger - Merges several protein/peptide identification files into one file.
- IDRipper - Splits protein/peptide identifications according their file-origin.
- IDFileConverter - Converts identification engine file formats.
- MapStatistics - Extract extended statistics on the features of a map for quality control.
- TextExporter - Exports various XML formats to a text file.
- MzTabExporter - Exports various XML formats to an mzTab file

C. Signal Processing and Preprocessing

- `BaselineFilter` - Removes the baseline from profile spectra using a top-hat filter.
- `InternalCalibration` - Applies an internal calibration.
- `MapNormalizer` - Normalizes peak intensities in an MS run.
- `MassTraceExtractor` - Annotates mass traces in centroided LC-MS maps.
- `NoiseFilterGaussian` - Removes noise from profile spectra by using different smoothing techniques.
- `NoiseFilterSGolay` - Removes noise from profile spectra by using different smoothing techniques.
- `PeakPickerHiRes` - Finds mass spectrometric peaks in profile mass spectra.
- `PeakPickerWavelet` - Finds mass spectrometric peaks in profile mass spectra.
- `PrecursorMassCorrector` - Correct the precursor entries of tandem MS scans.
- `HighResPrecursorMassCorrector` - Correct the precursor entries of tandem MS scans.
- `Resampler` - Transforms an LC-MS map into an equally-spaced (in RT and m/z) map.
- `SpectraFilterBernNorm` - Applies a filter to peak spectra.
- `SpectraFilterMarkerMower` - Applies a filter to peak spectra.
- `SpectraFilterNLargest` - Applies a filter to peak spectra.
- `SpectraFilterNormalizer` - Applies a filter to peak spectra.
- `SpectraFilterParentPeakMower` - Applies a filter to peak spectra.
- `SpectraFilterScaler` - Applies a filter to peak spectra.
- `SpectraFilterSqrtMower` - Applies a filter to peak spectra.
- `SpectraFilterThresholdMower` - Applies a filter to peak spectra.

- SpectraFilterWindowMower - Applies a filter to peak spectra.
- SpectraMerger - Merges spectra from an LC-MS map, either by precursor or by RT blocks
- TOFCalibration - Applies time of flight calibration.
- RNPxlXICFilter - Remove MS2 spectra from treatment based on the fold change between control and treatment for RNP cross linking experiments.

D. Quantitation

- AdditiveSeries - Computes an additive series to quantify a peptide in a set of samples.
- Decharger - Decharges and merges different feature charge variants of the same chemical entity.
- EICExtractor - Quantifies signals at given positions in (raw or picked) LC-MS maps.
- FeatureFinderCentroided - Detects two-dimensional features in centroided LC-MS data.
- FeatureFinderIdentification - Detects two-dimensional features in MS1 data based on peptide identifications.
- FeatureFinderIsotopeWavelet - Detects two-dimensional features in uncentroided (=raw) LC-MS data.
- FeatureFinderMetabo - Detects two-dimensional features in centroided LC-MS data of metabolites.
- FeatureFinderMRM - Quantifies features LC-MS/MS MRM data.
- FeatureFinderMultiplex - Identifies peptide multiplets (pairs, triplets etc.) and determines their relative abundance.
- IsobaricAnalyzer - Extracts and normalizes TMT and iTRAQ information from a MS experiment.

- ITRAQAnalyzer - Extracts and normalizes iTRAQ information from an MS experiment.
- ProteinQuantifier - Computes protein abundances from annotated feature/consensus maps
- ProteinResolver - A peptide-centric algorithm for protein inference.
- SeedListGenerator - Generates seed lists for feature detection.
- TMTAnalyzer - Extracts and normalizes TMT information from an MS experiment.

E. Map Alignment

- ConsensusMapNormalizer - Normalizes maps of one consensusXML file (after linking).
- MapAlignerIdentification - Corrects retention time distortions between maps based on common peptide identifications.
- MapAlignerPoseClustering - Corrects retention time distortions between maps using a pose clustering approach.
- MapAlignerSpectrum - Corrects retention time distortions between maps by spectrum alignment.
- MapRTTransformer - Applies retention time transformations to maps.
- FeatureLinkerLabeled - Groups corresponding isotope-labeled features in a feature map.
- FeatureLinkerUnlabeled - Groups corresponding features from multiple maps.
- FeatureLinkerUnlabeledQT - Groups corresponding features from multiple maps using a QT clustering approach.

F. Protein/Peptide Identification

- CompNovo - Performs a peptide/protein identification with the CompNovo engine.

- CompNovoCID - Performs a peptide/protein identification with the CompNovo engine in CID mode.
- InspectAdapter - Identifies MS/MS spectra using Inspect (external).
- MascotAdapter - Identifies MS/MS spectra using Mascot (external).
- MascotAdapterOnline - Identifies MS/MS spectra using Mascot (external).
- MSGFPlusAdapter - Identifies MS/MS spectra using MSGFPlus (external).
- MyriMatchAdapter - Identifies MS/MS spectra using MyriMatch (external).
- OMSSAAdapter - Identifies MS/MS spectra using OMSSA (external).
- PepNovoAdapter - Identifies MS/MS spectra using PepNovo (external).
- XTandemAdapter - Identifies MS/MS spectra using XTandem (external).
- SpecLibSearcher - Identifies peptide MS/MS spectra by spectral matching with a searchable spectral library.

G. Protein/Peptide Processing

- ConsensusID - Computes a consensus identification from peptide identifications of several identification engines.
- FalseDiscoveryRate - Estimates the false discovery rate on peptide and protein level using decoy searches.
- FidoAdapter - Runs the protein inference engine Fido.
- IDConflictResolver - Resolves ambiguous annotations of features with peptide identifications.
- IDFilter - Filters results from protein or peptide identification engines based on different criteria.
- IDMapper - Assigns protein/peptide identifications to feature or consensus features.

- IDPosteriorErrorProbability - Estimates posterior error probabilities using a mixture model.
- IDRTCalibration - Can be used to calibrate RTs of peptide hits linearly to standards.
- LuciphorAdapter - Scores potential phosphorylation sites in order to localize the most probable sites.
- PeptideIndexer - Refreshes the protein references for all peptide hits.
- PhosphoScoring - Scores potential phosphorylation sites in order to localize the most probable sites.
- ProteinInference - Infer proteins from a list of (high-confidence) peptides.
- DecoyDatabase - Create decoy peptide databases from normal ones.
- Digestor - Digests a protein database in-silico.
- DigestorMotif - Digests a protein database in-silico (optionally allowing only peptides with a specific motif) and produces statistical data for all peptides.
- IDExtractor - Extracts n peptides randomly or best n from idXML files.
- IDMassAccuracy - Calculates a distribution of the mass error from given mass spectra and IDs.
- IDScoreSwitcher - Switches between different scores of peptide or protein hits in identification data.
- RNPxl - Tool for RNP cross linking experiment analysis.
- SequenceCoverageCalculator - Prints information about idXML files.
- SpecLibCreator - Creates an MSP-formatted spectral library.

H. Targeted Experiments

- InclusionExclusionListCreator - Creates inclusion and/or exclusion lists for LC-MS/MS experiments.

- PrecursorIonSelector - A tool for precursor ion selection based on identification results.
- MRMMapper - MRMMapper maps measured chromatograms (mzML) and the transitions used (TraML)
- OpenSwathDecoyGenerator - Generates decoys according to different models for a specific TraML
- OpenSwathChromatogramExtractor - Extract chromatograms (XIC) from a MS2 map file.
- OpenSwathAnalyzer - Picks peaks and finds features in an SRM experiment.
- OpenSwathRTNormalizer - This tool will align an SRM / SWATH run to a normalized retention time space.
- OpenSwathFeatureXMLToTSV - Converts a featureXML to a tsv.
- OpenSwathConfidenceScoring - Computes confidence scores for OpenSwath results.

I. Peptide Property Prediction

- PTModel - Trains a model for the prediction of proteotypic peptides from a training set.
- PTPredict - Predicts the likelihood of peptides to be proteotypic using a model trained by PTModel.
- RTModel - Trains a model for the retention time prediction of peptides from a training set.
- RTPredict - Predicts retention times for peptides using a model trained by RTModel.

J. File Handling

- CVInspector - A tool for visualization and validation of PSI mapping and CV files.
- ConvertTSVToTraML - Converts a tsv file (tab separated) to TraML.

- ConvertTraMLToTSV - Converts a TraML file to TSV.
- FuzzyDiff - Compares two files, tolerating numeric differences.
- IDSplitter - Splits protein/peptide identifications off of annotated data files.
- MzMLSplitter - Splits an mzML file into multiple parts
- OpenSwathMzMLFileCacher - Caching of large mzML files
- SemanticValidator - SemanticValidator for analysisXML and mzML files.
- XMLValidator - Validates XML files against an XSD schema.

K. Algorithm Evaluation

- FFEval - Evaluation tool for feature detection algorithms.
- IDEvaluator - Evaluation tool, comparing peptide recovery at different q-value thresholds for multiple search engines (e.g., after ConsensusID). For interactive version use the IDEvaluatorGUI tool.
- LabeledEval - Evaluation tool for isotope-labeled quantitation experiments.
- MapAlignmentEvaluation - Evaluates alignment results against a ground truth.
- RTEvaluation - Application that evaluates TPs (true positives), TNs, FPs, and FNs for an idXML file with predicted RTs.
- TransformationEvaluation - Simple evaluation of transformations (e.g. RT transformations produced by a MapAligner tool).

L. Quantitation

- ERPairFinder - Evaluate pair ratios on enhanced resolution (zoom) scans.
- FeatureFinderSuperHirn - Find Features using the SuperHirn Algorithm (it can handle centroided or profile data, see .ini file).

- MRMPairFinder - Evaluate labeled pair ratios on MRM features.
- OpenSwathWorkflow - Complete workflow to run OpenSWATH.

M. Metabolite Identification

- AccurateMassSearch - Find potential HMDB IDs within the given mass error window.

N. Quality Control

- QC Calculator - Calculates basic quality parameters from MS experiments and compiles data for subsequent QC into a qcML file.
- QC Embedder - This application is used to embed tables or plots generated externally as attachments to existing quality parameters in qcML files.
- QC Exporter - Will extract several quality parameter from several run/sets from a qcML file into a tabular (text) format - counterpart to QC Importer.
- QC Extractor - Extracts a table attachment of a given quality parameter from a qcML file as tabular (text) format.
- QC Importer - Will import several quality parameter from a tabular (text) format into a qcML file - counterpart to QC Exporter.
- QC Merger - Merges two qcML files together.
- QC Shrinker - This application is used to remove extra verbose table attachments from a qcML file that are not needed anymore, e.g. for a final report.

O. Misc

- Generic Wrapper - Allows the generic wrapping of external tools.
- Execute Pipeline - Executes workflows created by TOPPAS.
- INI Updater - Update INI and TOPPAS files from previous versions of OpenMS as parameters and storage method might have changed

- DeMeanderize - Orders the spectra of MALDI spotting plates correctly.
- ImageCreator - Creates images from MS1 data (with MS2 data points indicated as dots).
- MSSimulator - A highly configurable simulator for mass spectrometry experiments.
- MassCalculator - Calculates masses and mass-to-charge ratios of peptide sequences.
- OpenSwathDIAPreScoring - SWATH (data-independent acquisition) pre-scoring.
- OpenSwathRewriteToFeatureXML - Rewrites results from mProphet back into featureXML.
- SvmTheoreticalSpectrumGeneratorTrainer - A trainer for SVM models as input for SvmTheoreticalSpectrumGenerator.

III. SUPPLEMENTARY NOTE 3: CASE STUDIES

A. SWATH Analysis

Large scale proteomics measurements of human blood plasma provide in-depth information about inter- and intra-person variability of protein abundances and can be used to investigate longitudinal trends during aging. In a recent study, Liu et al. [6] used SWATH-MS to acquire over 200 blood plasma samples collected from monozygotic and dizygotic twins in a longitudinal study. For maximal sensitivity, the authors chose a targeted analysis strategy for SWATH-MS data [7] which was developed using the OpenMS software library. In particular, OpenMS enabled the development of an analysis tool using existing algorithms and data structures for file parsing, RT alignment and signal processing. Second, the extensibility of OpenMS allowed the creation of a new OpenSWATH module for the targeted analysis of SWATH-MS data, built on top of the C++ library [4]. This module is now an integral analysis tool in OpenMS that executes all steps of the targeted analysis algorithm (RT alignment, targeted extraction, peak detection and peak scoring). When applied in the study of Liu et al., the high sensitivity of OpenSWATH led to the quantification of over 300 plasma proteins and allowed the authors the decomposition of the observed variance for each protein into heritable, environmental (common and individual) and longitudinal components. Interestingly, the authors found that the heritable component of plasma protein variation ranged from over 60 % to almost zero. In conclusion, the rapid development speed and the power of the OpenMS library allowed the researchers to implement a new algorithm for SWATH-MS data and successfully apply it to a complex human sample.

B. Degradomics

The human genome encodes for over 460 active proteases, which shape proteom composition and functionality [10]. Quantitative proteomics employing stable isotope labeling is widely used to investigate the (patho)-physiological role(s) of proteolytic enzymes. Often, enrichment schemes for N- or C-terminal peptides are used in order to identify proteolytically generated protein neo-termini. In combination with loss- or gain-of-function systems, such strategies are suited to unravel the substrate repertoire of a protease under investigation [10].

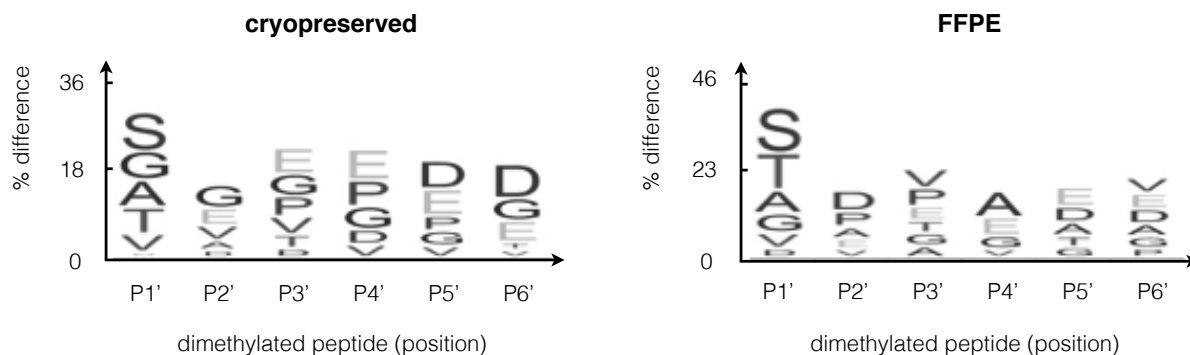


FIG. 1: **Degradomics case study.** N-terminal dimethylated peptides in cryopreserved and FFPE [8]. In both sample types, the same prominent fingerprint of serine, glycine, valine, alanine and threonine at the P1' position were observed.

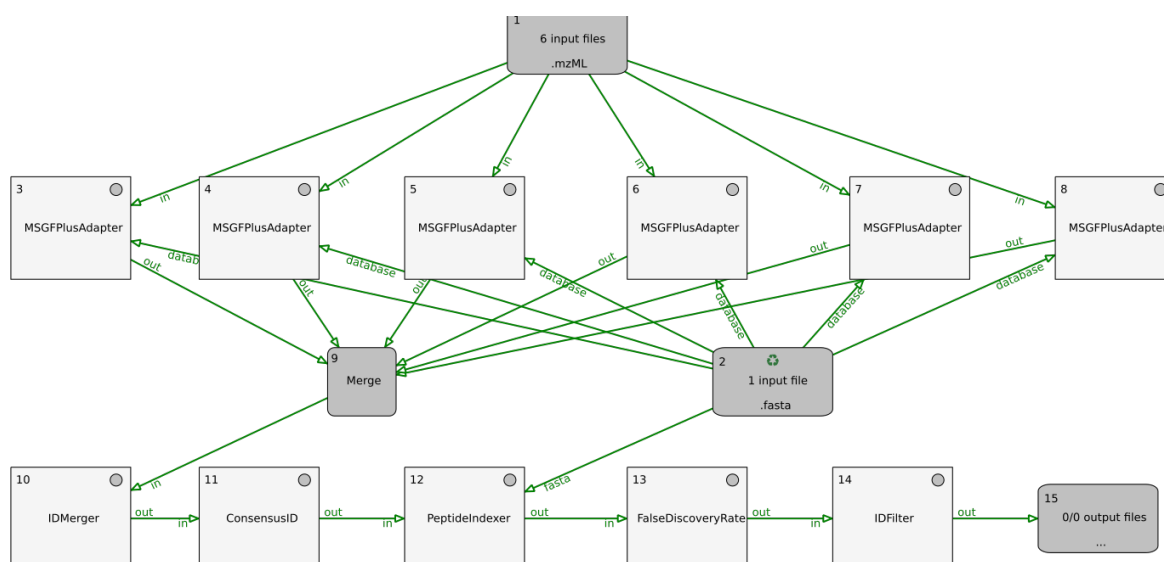


FIG. 2: **Degradomics case study workflow.** Subset of the OpenMS analysis workflow used in (Lai et al. [8]). Peptide sequence searches with six different fixed modifications (light / heavy dimethylation, monomethylation and acetylation) using the external search engine MS-GF+ [9] are subsequently combined.

In a recent study [8], this strategy has been expanded to include formalin-fixed, paraffin-embedded (FFPE) specimens, which represent the vast majority of samples that are stored in clinical archives, see Fig. 1. The aforementioned study highlighted the versatility of

OpenMS. In particular, OpenMS enabled the parallel and unbiased analysis of multiple, prevalent N-terminal modifications in different, isotopically labeled states, Fig. 2. Secondly, OpenMS enabled accurate relative quantitation of stable-isotope-labeled peptides. This included addressing so-called “knock-out” situations, i.e. peptides that occur in only one isotope state and are missing in the corresponding second isotope channel [11]. Typically, this situation is prone to distorted quantitation due to chromatographic background signals [12]; a problem that is resolved by accurate peptide feature detection in OpenMS. The workflow characterized thousands of protein N-termini in FFPE specimens and identified 23 potential substrates for the lysosomal protease cathepsin L [8].

C. Proteogenomics

Proteogenomics is an expanding field of inquiry that uses proteomic data in search of direct evidence of protein expression, in order to refine the annotation of protein-coding regions in genomes. In the case of shotgun tandem mass spectrometry data, the sought-after pieces of evidence are peptides of “unexpected” origin, i.e. peptides that do not match a known protein sequence and that could thus lead to novel annotations if supported by orthogonal evidence such as RNA expression or sequence conservation. For genomes that are already well characterized, particularly the human one, these novel peptides represent needles in a haystack of peptides matching known proteins. Notably, the difficulty in finding genuine novel peptides is exacerbated by the importance of avoiding false positive assignments that could lead to spurious annotations.

Several prerequisites are essential for a successful proteogenomics endeavor: A suitable proteomics dataset, a comprehensive database comprising both known and potential novel protein-coding sequences, and the expertise in genome annotation. Moreover, a data analysis workflow is needed that reliably and sensitively identifies peptides and filters them according to rigorous criteria. Such a workflow should operate in a reproducible fashion and allow for high throughput, to enable the analysis of large datasets. OpenMS is particularly well suited for this application, as it facilitates creating complex data analysis pipelines with access to a large variety of available functionality. Once a pipeline is established, it can be configured to account for experimental parameters and applied to large datasets. We recently reported stringent guidelines for proteogenomic data analysis and interpretation,

these analyses were based on OpenMS pipelines to run database searches of spectra followed by a rescoring step [16]. Our recent extension of the capabilities of OpenMS now enable a complete proteogenomic analysis workflow, which encompasses database searches using multiple search engines, rescoring and combining of search results, filtering according to various criteria on PSM, peptide and protein levels, and data export – within OpenMS (see Fig. 3). Processing of over 55 million tandem mass spectra from different publicly available datasets using OpenMS-based pipelines has led to over 40 new protein-coding gene annotations in the GENCODE human reference genome [17]. While this may seem like a low number, our approach for calling novel peptides is intentionally conservative and based on high quality proteomics evidence together with additional orthogonal support. Notably, only a fraction of the novel peptides identified from proteomic data gave rise to new annotations during the manual review process. On the other, the vast majority of the information generated in a proteogenomic analysis pertains to known proteins; this information may be valuable for other purposes, e.g. to elucidate tissue-specific protein expression levels [18]. Overall, the capabilities for creating flexible analysis pipelines, in connection with the wealth of functionality offered by TOPP tools and integrated third-party tools, make OpenMS especially well-suited for large-scale examination of publicly available data, be it with a proteogenomic or applications with a different focus such as differential (quantitative) or phosphoproteomic analyses.

-
- [1] L. Martens, M. Chambers, M. Sturm, D. Kessner, F. Levander, J. Shofstahl, W. H. Tang, A. Römpp, S. Neumann, A. D. Pizarro, et al., *Molecular & cellular proteomics : MCP* **10** (2011).
 - [2] M. Sturm, A. Bertsch, C. Gröpl, A. Hildebrandt, R. Hussong, E. Lange, N. Pfeifer, O. Schulz-Trieglaff, A. Zerck, K. Reinert, et al., *BMC Bioinformatics* **9**, 163 (2008).
 - [3] H. Weisser, S. Nahnsen, J. Grossmann, L. Nilse, A. Quandt, H. Brauer, M. Sturm, E. Kenar, O. Kohlbacher, R. Aebersold, et al., *Journal of Proteome Research* **12**, 1628 (2013).
 - [4] H. L. Röst, G. Rosenberger, P. Navarro, L. Gillet, S. M. Miladinović, O. T. Schubert, W. Wolfski, B. C. Collins, J. Malmström, L. Malmström, et al., *Nature Biotechnology* **32**, 219 (2014).
 - [5] E. Kenar, H. Franken, S. Forcisi, K. Wörmann, H.-U. Häring, R. Lehmann, P. Schmitt-

- Kopplin, A. Zell, and O. Kohlbacher, *Molecular & cellular proteomics: MCP* **13**, 348 (2014).
- [6] Y. Liu, A. Buil, B. C. Collins, L. C. Gillet, L. C. Blum, L.-Y. Cheng, O. Vitek, J. Mouritsen, G. Lachance, T. D. Spector, et al., *Molecular systems biology* **11**, 786 (2015).
- [7] L. C. Gillet, P. Navarro, S. Tate, H. Röst, N. Selevsek, L. Reiter, R. Bonner, and R. Aebersold, *Molecular & cellular proteomics: MCP* **11**, O111.016717 (2012).
- [8] Z. W. Lai, J. Weisser, L. Nilse, F. Costa, E. Keller, M. Tholen, J. N. Kizhakkedathu, M. Biniossek, P. Bronsert, and O. Schilling, *Molecular & Cellular Proteomics* **15**, 2203 (2016).
- [9] V. Granholm, S. Kim, J. C. Navarro, E. Sjolund, R. D. Smith, and L. Kall, *Journal of proteome research* **13**, 890 (2013).
- [10] Z. W. Lai, A. Petrera, and O. Schilling, *Current opinion in chemical biology* **24**, 71 (2015).
- [11] L. Nilse, F. C. Sigloch, M. L. Biniossek, and O. Schilling, *Proteomics Clinical Applications* **9**, 706 (2015).
- [12] S. Tholen, M. L. Biniossek, A.-L. Geßler, S. Müller, J. Weißer, J. N. Kizhakkedathu, T. Reinheckel, and O. Schilling, *Biological chemistry* **392**, 961 (2011).
- [13] D. N. Perkins, D. J. C. Pappin, D. M. Creasy, and J. S. Cottrell, *Electrophoresis* **20**, 3551 (1999).
- [14] S. Kim, N. Gupta, and P. A. Pevzner, *Journal of Proteome Research* **7**, 3354 (2008).
- [15] L. Käll, J. D. Canterbury, J. Weston, W. S. Noble, and M. J. MacCoss, *Nature Methods* **4**, 923 (2007).
- [16] J. C. Wright, J. Mudge, H. Weisser, M. P. Barzine, J. M. Gonzalez, A. Brazma, J. S. Choudhary, and J. Harrow, *Nature Communications* **7**, 11778+ (2016), ISSN 2041-1723, URL <http://dx.doi.org/10.1038/ncomms11778>.
- [17] J. Harrow, F. Denoeud, A. Frankish, A. Reymond, C.-K. Chen, J. Chrast, J. Lagarde, J. G. Gilbert, R. Storey, D. Swarbreck, et al., *Genome Biol* **7**, S4 (2006).
- [18] R. Petryszak, M. Keays, Y. A. Tang, N. A. Fonseca, E. Barrera, T. Burdett, A. Füllgrabe, A. M.-P. Fuentes, S. Jupp, S. Koskinen, et al., *Nucleic acids research* **44**, D746 (2016).

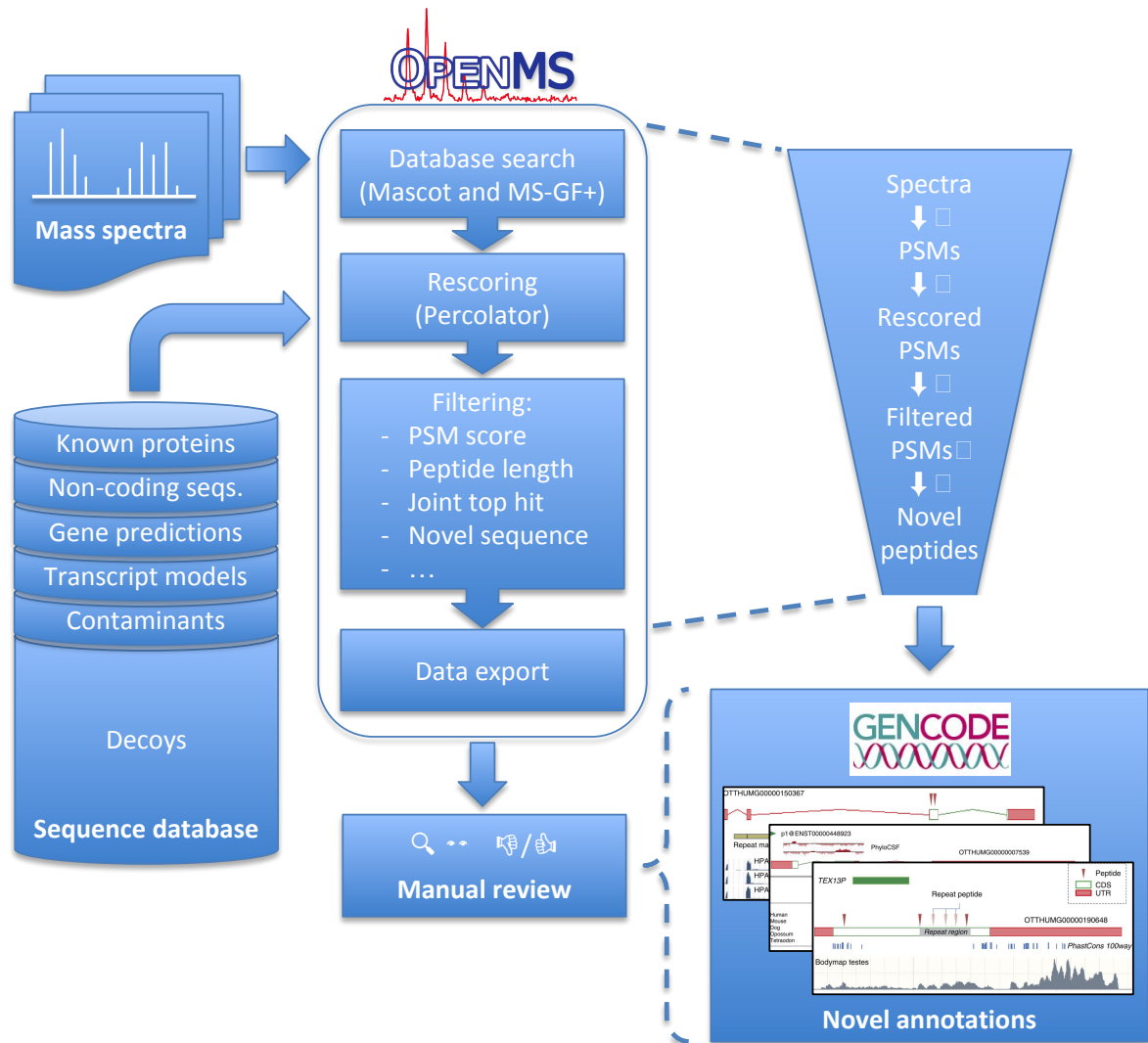
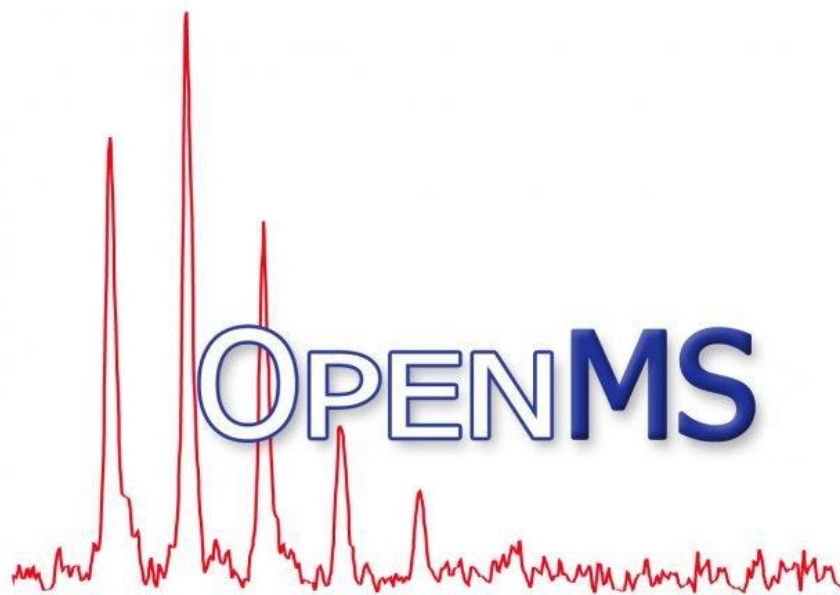


FIG. 3: Schematic overview of the OpenMS proteogenomics workflow. Based on a comprehensive sequence database, tandem mass spectra from large proteomic datasets were searched in a competitive target/decoy approach using two search engines, Mascot [13] and MS-GF+ [14]. The search results were rescored using Percolator [15], then filtered in multiple stages according to stringent quality criteria [16]. During this process, starting from a large number of spectra and initial PSMs, the set of retained PSMs was refined further and further, until in the end only high-confidence PSMs from novel peptides remained. These were exported and passed on to the annotators. In a manual review process, novel peptides and other evidence sources were integrated, which in some cases yielding novel genome annotations.

This figure uses images from a publication by Wright et al. [16], licensed under a Creative Commons Attribution 4.0 International License.



OpenMS Tutorial

The OpenMS Developers

Mathias Walzer, Timo Sachsenberg, Fabian Aicheler,
Marc Rurik, Johannes Veit,
Bludau Isabell, Patrick Pedrioli,
Julianus Pfeuffer, Xiao Liang,
Knut Reinert, and Oliver Kohlbacher

Creative Commons Attribution 4.0 International (CC BY 4.0)


Contents

1	General remarks	6
2	Getting started	7
2.1	Data conversion	7
2.2	Data visualization using TOPPView	8
2.3	Introduction to KNIME / OpenMS	11
2.3.1	Plugin and dependency installation	11
2.3.2	KNIME concepts	13
2.3.3	Overview of the graphical user interface	14
2.3.4	Creating workflows	16
2.3.5	Sharing workflows	16
2.3.6	Duplicating workflows	17
2.3.7	A minimal workflow	17
2.3.8	Advanced topic: Meta nodes	20
2.3.9	Advanced topic: R integration	21
3	Label-free quantification	23
3.1	Introduction	23
3.2	Peptide Identification	23
3.2.1	Bonus task: identification using several search engines	26
3.3	Quantification	27
3.4	Combining quantitative information across several label-free experiments	29
3.4.1	Basic data analysis in KNIME	31
4	Protein Inference	34
4.1	Extending the LFQ workflow by protein inference and quantification	34
4.2	Statistical validation of protein inference results	36
4.2.1	Data preparation	36
4.2.2	ROC curve of protein ID	36
4.2.3	Posterior probability and FDR of protein IDs	37

5	Metabolomics	39
5.1	Introduction	39
5.2	Quantifying metabolites across several experiments	39
5.3	Identifying metabolites in LC-MS/MS samples	42
5.4	Convert your data into a KNIME table	43
5.4.1	Bonus task: Visualizing data	44
5.5	Downstream data analysis and reporting	45
5.5.1	Signal processing and Data preparation ID	45
5.5.2	Data preparation Quant	45
5.5.3	Statistical analysis	46
5.5.4	Interactive visualization	47
5.5.5	Advanced visualization	48
5.5.6	Data preparation for Reporting	49
6	OpenSWATH	51
6.1	Introduction	51
6.2	Installation of OpenSWATH	51
6.3	Installation of mProphet	51
6.4	Generating the Assay Library	52
6.4.1	Generating TraML from transition lists	52
6.4.2	Appending decoys to a TraML	54
6.5	OpenSWATH KNIME	55
6.6	From the example dataset to real-life applications	56
7	An introduction to pyOpenMS	57
7.1	Introduction	57
7.2	Installation	57
7.2.1	Windows	57
7.2.2	Mac OS X 10.10	57
7.2.3	Linux	58
7.3	Build instructions	58
7.4	Your first pyOpenMS tool: pyOpenSwathFeatureXMLToTSV	58
7.4.1	Basics	59

7.4.2	Loading data structures with pyOpenMS	60
7.4.3	Converting data in the featureXML to a TSV	62
7.4.4	Putting things together	63
7.4.5	Bonus task	64
8	Quality control	65
8.1	Introduction	65
8.2	Building a qcML file per run	66
8.3	Adding brand new QC metrics	69
8.4	Set QC metrics	71
9	Troubleshooting guide	73
9.1	FAQ	73
9.1.1	General	73
9.1.2	Platform-specific problems	73
9.1.3	Nodes	74
9.2	Sources of support	74

1 General remarks

- This handout will guide you through an introductory tutorial for the OpenMS/TOPP software package [1].
- OpenMS [2] is a versatile open-source library for mass spectrometry data analysis. Based on this library, we offer a collection of command-line tools ready to be used by end users. These so-called TOPP tools (short for "The OpenMS Proteomics Pipeline") [3] can be understood as small building blocks of arbitrary complex data analysis workflows.
- In order to facilitate workflow construction, OpenMS was integrated into KNIME [4], the Konstanz Information Miner, an open-source integration platform providing a powerful and flexible workflow system combined with advanced data analytics, visualization, and report capabilities. Raw MS data as well as the results of data processing using TOPP can be visualized using TOPPView [5].
- This tutorial was designed for use in a hands-on tutorial session but can also be worked through at home using the online resources. You will become familiar with some of the basic functionalities of OpenMS/TOPP, TOPPView, and KNIME and learn how to use a selection of TOPP tools used in the tutorial workflows.
- All sample data referenced in this tutorial can be found in the  Example_Data folder on the USB stick that came with this tutorial (or online on our SourceForge).

2 Getting started

Before we get started we will install OpenMS and KNIME using the installers provided on the USB stick. Please choose the directory that matches your operating system and execute the installer. Note that these steps are not necessary if you use one of our laptops.

For example for Windows you call

- the OpenMS installer: `Windows / OpenMS-2.0_Win64_setup.exe`
- the KNIME installer: `Windows / OpenMS-2.0-prerequisites-installer.exe`
and `Windows / KNIME Full 3.1.1 Installer (64bit).exe`

on Mac you call

- the OpenMS installer: `Mac / OpenMS-2.0.0_setup.dmg`
- the KNIME installer: `Mac / knime-full_3.1.1.macosx.cocoa.x86_64.dmg`

and follow the instructions. If you are working through this tutorial at home or downloaded it from our homepage, you can get the installers under the following links:

- OpenMS
- KNIME
- OpenMS prerequisites (Windows-only): After installation, before your first use of the OpenMS plugin in KNIME you will be asked to download it automatically if certain requirements are not found in your Windows registry. Alternatively, you can get a bundled version here.

Choose the installers for the platform you are working on. We suggest to use the full installers of KNIME so that you can skip the installation of the OpenMS plugin and other dependencies for the example workflows.

2.1 Data conversion

Each MS instrument vendor has one or more formats for storing the acquired data. Converting these data into an open format (preferably mzML) is the very first step when you


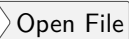


want to work with open-source mass spectrometry software. A freely available conversion tool is ProteoWizard. The OpenMS installation package for Windows automatically installs ProteoWizard, so you do not need to download and install it separately.

Please note that due to restrictions from the instrument vendors, file format conversion for most formats is only possible on Windows systems, so exporting from the acquisition PC connected to the instrument is usually the most convenient option. All files used in this tutorial have already been converted to mzML by us, so you do not need to do it yourself.

2.2 Data visualization using TOPPView

Visualizing the data is the first step in quality control, an essential tool in understanding the data, and of course an essential step in pipeline development. OpenMS provides a convenient viewer for some of the data: TOPPView.

We will guide you through some of the basic features of TOPPView. Please familiarize yourself with the key controls and visualization methods. We will make use of these later throughout the tutorial. Let's start with a first look at one of the files of our tutorial data set:

- Start TOPPView (see Start-Menu or Applications on MacOS)
- Go to  , navigate to the directory where you copied the contents of the USB stick to, and select  Example_Data ▶ Introduction ▶ datasets ▶ small ▶ velos005614.mzML . This file contains a reduced LC-MS map (only a selected RT and m/z range was extracted using the TOPP tool FileFilter) of a label-free measurement of the human platelet proteome recorded on an Orbitrap velos. The other two mzML files contain technical replicates of this experiment. First, we want to obtain a global view on the whole LC-MS map - the default option *Map view 2D* is the correct one and we can click the  button.
- Play around.
- Three basic modes allow you to interact with the displayed data: scrolling, zooming and measuring:

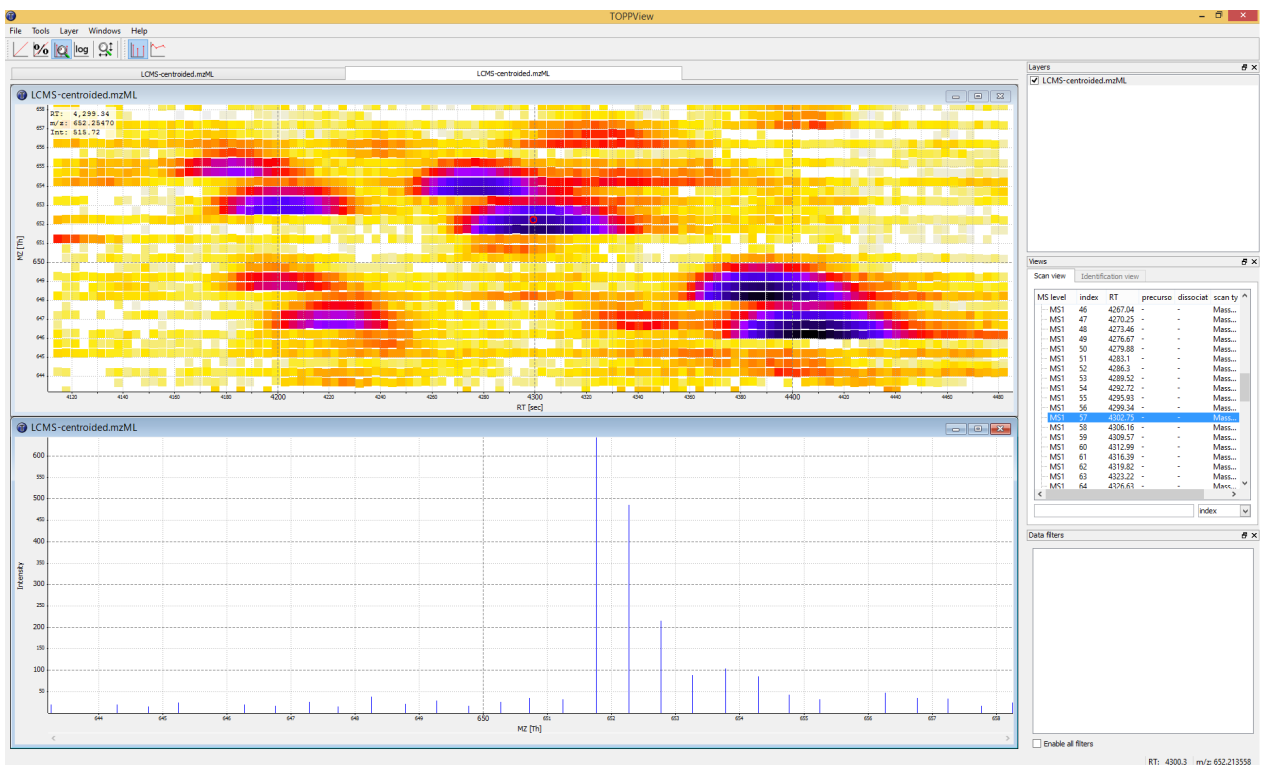


Figure 1: TOPPView, the graphical application for viewing mass spectra and analysis results. Top window shows a small region of a peak map. In this 2D representation of the measured spectra, signals of eluting peptides are colored according to the raw peak intensities. The lower window displays an extracted spectrum (=scan) from the peak map. On the right side, the list of spectra can be browsed.

- Scroll mode
 - * Is activated by default (though each loaded spectra file is displayed zoomed out first, so you do not need to scroll).
 - * Allows you to browse your data by moving around in RT and m/z range.
 - * When zoomed in, to scroll the spectra map, click-drag on the current view.
 - * Arrow keys can be used to scroll the view as well.
- Zoom mode
 - * Zooming into the data: either mark an area in the current view with your mouse while holding the left mouse button plus the `ctrl` key to zoom to this area or use your mouse wheel to zoom in and out.
 - * All previous zoom levels are stored in a zoom history. The zoom history can be traversed using `ctrl + +` or `ctrl + -` or the mouse wheel (scroll up and down).
 - * Pressing the Backspace key zooms out to show the full LC-MS map (and also resets the zoom history).
- Measure mode
 - * It is activated using the `↑` key.
 - * Press the left mouse button down while a peak is selected and drag the mouse to another peak to measure the distance between peaks.
 - * This mode is implemented in the 1D and 2D mode only.
- Right click on your 2D map and select `Switch to 3D view` and examine your data in 3D mode
- Go back to the 2D view. In 2D mode, visualize your data in different normalization modes, use linear, percentage and log-view (icons on the upper left tool bar).

Note: On *Apple OS X*, due to a bug in one of the external libraries used by OpenMS, you will see a small window of the 3D mode when switching to 2D. Close the 3D tab in order to get rid of it.
- In TOPPView you can also execute TOPP tools. Go to `Tools >> Apply tool (whole layer)` and choose a TOPP tool (e.g., FileInfo) and inspect the results.

2.3 Introduction to KNIME / OpenMS

Using OpenMS in combination with KNIME you can create, edit, open, save, and run workflows combining TOPP tools with the powerful data analysis capabilities of KNIME. Workflows can be created conveniently in a graphical user interface. The parameters of all involved tools can be edited within the application and are also saved as part of the workflow. Furthermore, KNIME interactively performs validity checks during the workflow editing process, in order to make it more difficult to create an invalid workflow.


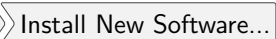


Throughout most of the parts of this tutorial you will use KNIME to create and execute workflows. This first step is to make yourself familiar with KNIME. Additional information on basic usage of KNIME can be found on the KNIME Getting Started page. However, the most important concepts will also be reviewed in this tutorial.

2.3.1 Plugin and dependency installation

Note: If you installed the binaries from our USB Stick or downloaded and installed the full release of KNIME with all contributing plugins, you can skip this section.

Before we can start with the tutorial we need to install all the required extensions for KNIME.

First, we install some additional extensions that are required by our OpenMS nodes or used in the Tutorials e.g. for visualization and file handling.

1. Click on  
2. From the  drop down list select 
3. Now select the following plugins from the *KNIME & Extensions* category
 - KNIME Base Chemistry Types & Nodes
 - KNIME Chemistry Add-Ons
 - KNIME File Handling Nodes (required for OpenMS nodes in general)
 - KNIME Interactive R Statistics Integration
 - KNIME R Statistics Integration (Windows Binaries)

- KNIME Report Designer
- KNIME SVG Support
- KNIME XLS Support
- KNIME XML-Processing
- (Older versions of KNIME do not have KNIME Math Expression (JEP) installed which is used in one of the workflows)

4. From the **Work with:** drop down list select

<http://tech.knime.org/update/community-contributions/trusted/3.1>

5. Now select the following plugin from the "KNIME Community Contributions - Cheminformatics" category

- RDKit KNIME integration

6. Follow the instructions and after a restart of KNIME the dependencies will be installed.

You are now ready to install the OpenMS nodes.

1. Open KNIME.

2. Click on **Help** >> **Install New Software...**

3. From the **Work with:** drop down list select the

<http://tech.knime.org/update/community-contributions/trusted/3.1>

4. Select the **OpenMS** nodes in the category:

"KNIME Community Contributions - Bioinformatics & NGS" and click **Next**.

5. Follow the instructions and after a restart of KNIME the OpenMS nodes will be available under "Community Nodes".

2.3.2 KNIME concepts

A **workflow** is a sequence of computational steps applied to a single or multiple input data sets to process and analyze the data. In KNIME such workflows are implemented graphically by connecting so-called **nodes**. A node represents a single analysis step in a workflow. Nodes have input and output **ports** where the data enters the node or the results are provided for other nodes after processing, respectively. KNIME distinguishes between different port types, representing different types of data. The most common representation of data in KNIME are tables (similar to an excel sheet). Ports that accept tables are marked with a small triangle. For OpenMS we use a different port type, so called **file ports**, representing complete files. Those ports are marked by a small blue box. Filled blue boxes represent mandatory inputs and empty blue boxes optional inputs. A typical OpenMS workflow in KNIME can be divided in two conceptually different parts:

- Nodes for signal and data processing, filtering and data reduction. Here, files are passed between nodes. Execution times of the individual steps are longer as the main computational steps are performed.
- Downstream statistical analysis and visualization. Here, tables are passed between nodes.

Between file-based processing and table-based analysis a conversion node typically performs the conversion from OpenMS results into KNIME tables. Nodes can have three different states, indicated by the small traffic light below the node.

- Inactive, failed, and not yet fully configured nodes are marked red.
- Configured but not yet executed nodes are marked yellow.
- Successfully executed nodes are marked green.

If the node execution failed the node will switch to the red state. Other anomalies and warnings like missing information or empty results will be presented with a yellow exclamation mark sign above the traffic light. Most nodes will be configured as soon as all input ports are connected. For some nodes additional parameters have to be provided that cannot be either guessed from the data or filled with sensible defaults. In this case,

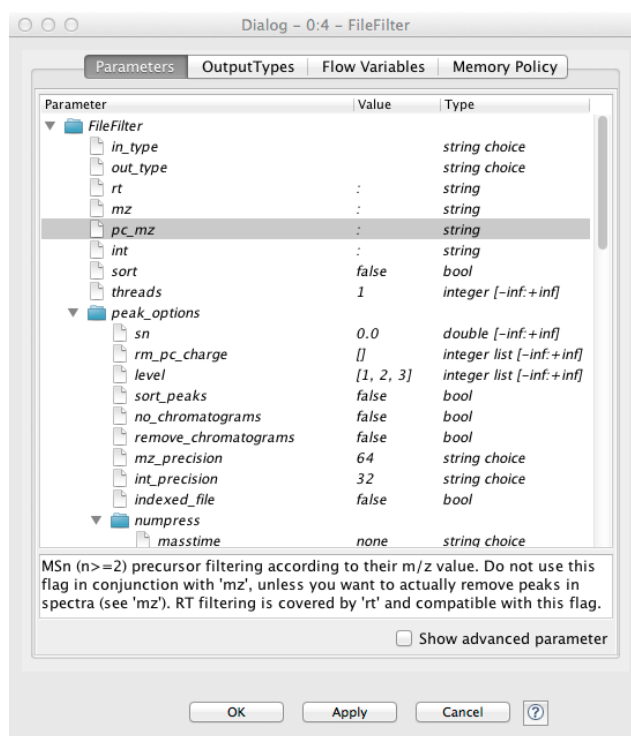


Figure 2: Node configuration dialog of an OpenMS node.

of if you want to customize the default configuration, you can open the configuration dialog of a node with a double-click on the node. For all OpenMS nodes you will see a configuration dialog like the one shown in Figure 2.

Note: OpenMS distinguishes between normal parameters and advanced parameters. Advanced parameters are by default hidden from the users since they should only rarely be customized. In case you want to have a look at the parameters or need to customize them in one of the tutorials you can show them by clicking on the checkbox Show advanced parameter in the lower part of the dialog.

The dialog shows the individual parameters, their current value and type, and, in the lower part of the dialog, the documentation for the currently selected parameter.

2.3.3 Overview of the graphical user interface

The graphical user interface (GUI) of KNIME consists of different components or so called panels that are shown in Figure 3. We will shortly introduce the individual panels and their

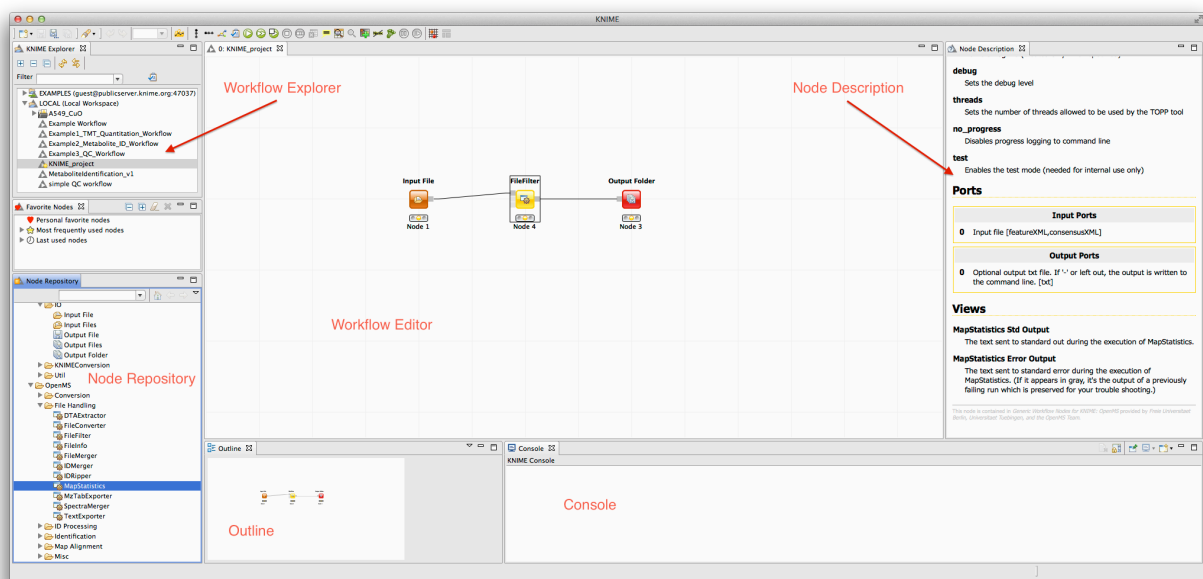


Figure 3: The KNIME workbench.

purposes below.

Workflow Editor: The workflow editor is the central part of the KNIME GUI. Here you assemble the workflow by adding nodes from the Node Repository via "drag & drop". For quick creation note that double-clicking on a node in the repository automatically connects it with the selected node in the workbench. Nodes can be connected by clicking on the output port of one node and dragging the edge until releasing the mouse at the desired input port of the next node.

Workflow Explorer: Shows a list of available workflows (also called workflow projects). You can open a workflow by double clicking it. A new workflow can be created with a right-click in the Workflow Explorer followed by selecting `New KNIME Workflow...`. Remember to save your workflow often with the Ctrl+S shortcut.

Node Repository: Shows all nodes that are available in your KNIME installation. Every plugin you install will provide new nodes that can be found here. The OpenMS nodes can be found in `Community Nodes >> OpenMS`. Nodes for managing files (e.g., Input Files or Output Folders) can be found in `Community Nodes >> GenericKnodeNodes`. You


can search the node repository by typing the node name into the small text box in the upper part of the node repository.

Outline: The Outline panel contains a small overview of the complete workflow. While of limited use when working on a small workflow, this feature is very helpful as soon as the workflows get bigger.



Console: In the console panel warning and error messages are shown. This panel will provide helpful information if one of the nodes failed or shows a warning sign.


Node Description: As soon as a node is selected, the Node Description window will show the documentation of the node including documentation for all its parameters. For OpenMS nodes you will also find a link to the tool page in the online documentation.

2.3.4 Creating workflows

Workflows can easily be created by a right click in the Workflow Explorer followed by clicking on .

2.3.5 Sharing workflows

To be able to share a workflow with others, KNIME supports the import and export of complete workflows. To export a workflow, select it in the Workflow Explorer and select . KNIME will export workflows as a zip file containing all the information on nodes, their connections, and their configuration. Those zip files can again be imported by selecting .

Note: For your convenience we added all workflows discussed in this tutorial to the  workflows folder on the USB Stick. Additionally the zip files can be found on our GitHub repository. If you want to check your own workflow by comparing it to the solution or got stuck, simply import the full workflow from the corresponding zip file.

2.3.6 Duplicating workflows

During the tutorial a lot of the workflows will be created based on the workflow from a previous task. To keep the intermediate workflows we suggest you create copies of your workflows so you can see the progress. To create a copy of your workflow follow the next steps.

- Right click on the workflow you want to create a copy of in the Workflow Explorer and select `Copy`.
- Right click again somewhere on the workflow explorer and select `Paste`.
- This will create a workflow with same name as the one you copied with a (2) appended.
- To distinguish them later on you can easily rename the workflows in the Workflow Explorer by right clicking on the workflow and selecting `Rename`.

Note: To rename a workflow it has to be closed.

2.3.7 A minimal workflow

Let us now start with the creation of our very first, very simple workflow. As a first step, we will gather some basic information about the data set before starting the actual development of a data analysis workflow.

- Create a new workflow.
- Add an Input File node and an Output Folder node (to be found in `Community Nodes` `GenericKnomeNodes` `IO`) and a FileInfo node (to be found in the category `Community Nodes` `OpenMS` `File Handling`) to the workflow.
- Connect the Input File node to the FileInfo node, and the first output port of the FileInfo node to the Output Folder node.

Note: In case you are unsure about which node port to use, hovering the cursor over the port in question will display the port name and what kind of input it expects.

The complete workflow is shown in Figure 4. FileInfo can produce two different kinds of output files.

- All nodes are still marked red, since we are missing an actual input file. Double-click the Input File node and select **Browse**. In the file system browser select **Example_Data** ▶ **Introduction** ▶ **datasets** ▶ **tiny** ▶ **velos005614.mzML** and click **Open**. Afterwards close the dialog by clicking **Ok**.

Note: Make sure to use the "tiny" version this time, not "small", for the sake of faster workflow execution.

- The Input File node and the FileInfo node should now have switched to yellow, but the Output Folder node is still red. Double-click on the Output Folder node and click on **Browse** to select an output directory for the generated data.
- Great! Your first workflow is now ready to be run. Press **↑** + **F7** to execute the complete workflow. You can also right click on any node of your workflow and select **Execute** from the context menu.
- The traffic lights tell you about the current status of all nodes in your workflow. Currently running tools show either a progress in percent or a moving blue bar, nodes waiting for data show the small word "queued", and successfully executed ones become green. If something goes wrong (e.g., a tool crashes), the light will become red.
- In order to inspect the results, you can just right-click the Output Folder node and select **View: Open the output folder**. You can then open the text file and inspect its contents. You will find some basic information of the data contained in the mzML file, e.g., the total number of spectra and peaks, the RT and m/z range, and how many MS1 and MS2 spectra the file contains.



Figure 4: A minimal workflow calling FileInfo on a single file.

Workflows are typically constructed to process a large number of files automatically. As a simple example, consider you would like to gather this information for more than one file. We will now modify the workflow to compute the same information on three different files and then write the output files to a folder.

- We start from the previous workflow.
- First we need to replace our single input file with multiple files. Therefore we add the Input Files node from the category `Community Nodes > GenericKnlmeNodes > IO`.
- To select the files we double-click on the Input Files node and click on `Add`. In the filesystem browser we select all three files from the directory `Example_Data > Introduction > datasets > tiny`. And close the dialog with `Ok`.
- We now add two more nodes: the ZipLoopStart and the ZipLoopEnd node from the category `Community Nodes > GenericKnlmeNodes > Flow`.
- Afterwards we connect the Input Files node to the first port of the ZipLoopStart node, the first port of the ZipLoopStart node to the FileInfo node, the first output port of the FileInfo node to the first input port of the ZipLoopEnd node, and the first output port of the ZipLoopEnd node to the Output Folder node (NOT to the Output File). The complete workflow is shown in Figure 5
- The workflow is already complete. Simply execute the workflow and inspect the output as before.

In case you had trouble to understand what ZipLoopStart and ZipLoopEnd do - here is a brief explanation:

- The Input Files node passes a list of files to the ZipLoopStart node.



Figure 5: A minimal workflow calling FileInfo on multiple files in a loop.

- The ZipLoopStart node takes the files as input, but passes the single files sequentially (that is: one after the other) to the next node.
- The ZipLoopEnd collects the single files that arrive at its input port. After all files have been processed, the collected files are passed again as file list to the next node that follows.

2.3.8 Advanced topic: Meta nodes

Workflows can get rather complex and may contain dozens or even hundreds of nodes. KNIME provides a simple way to improve handling and clarity of large workflows:

Meta Nodes allow to bundle several nodes into a single Meta Node.

Task



Select multiple nodes (e.g. all nodes of the ZipLoop including the start and end node). To select a set of nodes, draw a rectangle around them with the left mouse button or hold **Ctrl** to add/remove single nodes from the selection. Open the context menu (right-click on a node in the selection) and select **Collapse into Meta Node**. Enter a caption for the Meta Node. The previously selected nodes are now contained in the Meta Node. Double clicking on the Meta Node will display the contained nodes in a new tab window.

Task



Undo the packaging. First select the Meta Node, open the context menu (right-click) and select **Expand Meta Node**.

2.3.9 Advanced topic: R integration

KNIME provides a large number of nodes for a wide range of statistical analysis, machine learning, data processing and visualization. Still, more recent statistical analysis methods, specialized visualizations or cutting edge algorithms may not be covered in KNIME. In order to expand its capabilities beyond the readily available nodes, external scripting languages can be integrated. In this tutorial, we primarily use scripts of the powerful statistical computing language R. Note that this part is considered advanced and might be difficult to follow if you are not familiar with R. In this case you might skip this part.

R View (Table) allows to seamlessly include R scripts into KNIME. We will demonstrate on a minimal example how such a script is integrated.

Task



First we need some example data in KNIME, which we will generate using the Data Generator node. You can keep the default settings and execute the node. The table contains 4 columns, each containing random coordinates and one column containing a cluster number (Cluster_0 to Cluster_3). Now place a R View (Table) node into the workflow and connect the upper output port of the Data Generator node to the input of the R View (Table) node. Right-click and configure the node.

If you get an error message like "Execute failed: R_HOME does not contain a folder with name 'bin'.": please change the R settings in the preferences. To do so open **File > Preferences > KNIME > R** and enter the path to your R installation (the folder that contains the bin directory).

If R is correctly recognized we can start writing an R script. Consider that we are interested in plotting the first and second coordinates and color them according to their cluster number. In R this can be done in a single line.

In the R View (Table) text editor, enter the following code:

```
plot(x=knime.in$Universe_0_0, y=knime.in$Universe_0_1, main="Plotting column ←  
Universe_0_0 vs. Universe_0_1", col=knime.in$"Cluster Membership")
```

Explanation: The table provided as input to the R View (Table) node

is available as R `data.frame` with name `knime.in`. Columns (also listed on the left side of the R View window) can be accessed in the usual R way by first specifying the `data.frame` name and then the column name (e.g. `knime.in$Universe_0_0`). `plot` is the plotting function we use to generate the image. We tell it to use the data in column `Universe_0_0` of the dataframe object `knime.in` (denoted as `knime.in$Universe_0_1`) as x-coordinate and the other column `knime.in$Universe_0_1` as y-coordinate in the plot. `main` is simply the main title of the plot and `col` the column that is used to determine the color (in this case it is the `Cluster Membership` column).

Now press the `Eval script` and `Show plot` buttons.

Note: Note that we needed to put some extra quotes around `Cluster Membership`. If we omit those, R would interpret the column name only up to the first space (`knime.in$Cluster`) which is not present in the table and leads to an error. Quotes are regularly needed if column names contain spaces, tabs or other special characters like `$` itself.

3 Label-free quantification

3.1 Introduction

In this chapter, we will build a workflow with OpenMS / KNIME to quantify a label-free experiment. Label-free quantification is a method aiming to compare the relative amounts of proteins or peptides in two or more samples. We will start from the minimal workflow of the last chapter and, step-by-step, build a label-free quantitation workflow.

3.2 Peptide Identification

As a start, we will extend the minimal workflow so that it performs a peptide identification using the OMSSA [6] search engine. Since OpenMS version 1.10, OMSSA is included in the OpenMS installation, so you do not need to download and install it yourself.

- Let's start by replacing the input files in our `Input Files` node by the three mzML files in `Example_Data > Labelfree > datasets > lfq_spikein_dilution_1-3.mzML`. This is a reduced toy dataset where each of the three runs contains a constant background of *S. pyogenes* peptides as well as human spike-in peptides in different concentrations. [7]
- Instead of `FileInfo`, we want to perform OMSSA identification, so we simply replace the `FileInfo` node with the OMSSAadapter node `Community Nodes >> OpenMS >> Identification`, and we are almost done. Just make sure you have connected the `ZipLoopStart` node with the `in` port of the OMSSAadapter node.
- OMSSA, like most mass spectrometry identification engines, relies on searching the input spectra against sequence databases. Thus, we need to introduce a search database input. As we want to use the same search database for all of our input files, we can just add a single `Input File` node to the workflow and connect it directly with the OMSSAadapter database port. KNIME will automatically reuse this `Input` node each time a new `ZipLoop` iteration is started. In order to specify the database, select `Example_Data > Labelfree > databases > s_pyo_sf370_potato_human_target_decoy_with_contaminants.fasta`, and we have a very basic peptide identification workflow.

Note: You might also want to save your new identification workflow under a different name. Have a look at Section 2.3.6 for information on how to create copies of workflows.

- The result of a single OMSSA run is basically a number of peptide-spectrum-matches (PSM) with a score each, and these will be stored in an idXML file. Now we can run the pipeline and after execution is finished, we can have a first look at the results: just open the input files folder with a file browser and from there open an mzML file in TOPPView.
- Here, you can annotate this spectrum data file with the peptide identification results. Choose **Tools** > **Annotate with identification** from the menu and select the idXML file that OMSSAAdapter generated (it is located within the output directory that you specified when starting the pipeline).
- On the right, select the tab **Identification view**. Using this view, you can see all identified peptides and browse the corresponding MS2 spectra.

Note: Opening the output file of OMSSAAdapter (the idXML file) directly is also possible, but the direct visualization of an idXML file is less useful.

- The search results stored in the idXML file can also be read back into a KNIME table for inspection and subsequent analyses: Add a TextExporter **Community Nodes** > **OpenMS** > **File Handling** node to your workflow and connect the output port of your OMSSAAdapter (the same port your ZipLoopEnd is connected to) to its input port. This tool will convert the idXML file to a more human-readable text file which can also be read into a KNIME table using the IDTextReader node. Add an IDTextReader node **Community Nodes** > **OpenMS** > **Conversion** after TextExporter and execute it. Now you can right-click IDTextReader and select **ID Table** to browse your peptide identifications.
- From here, you can use all the tools KNIME offers for analyzing the data in this table. As a simple example, you could add a Histogram **Data Views** node after IDTextReader, double-click it, select *peptide_charge* as binning column, hit **OK**, and execute it. Right-clicking and selecting **View: Histogram view** will open a plot showing the charge state distribution of your identifications.

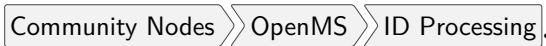
In the next step, we will tweak the parameters of OMSSA to better reflect the instrument's accuracy. Also, we will extend our pipeline with a false discovery rate (FDR) filter to retain only those identifications that will yield an FDR of $< 1\%$.

- Open the configuration dialog of OMSSAadapter. The dataset was recorded using an LTQ Orbitrap XL mass spectrometer, so we can set the precursor mass tolerance to a smaller value, say 10 ppm. Set *precursor_mass_tolerance* to 10 and *precursor_mass_tolerance_unit_ppm* to *true*.

Note: Whenever you change the configuration of a node, the node as well as all its successors will be reset to the Configured state.

- Set *max_precursor_charge* to 5, in order to also search for peptides with charges up to 5.
- Add Carbamidomethyl (C) as fixed modification and Oxidation (M) as variable modification.

Note: To add a modification click on the empty value field in the configuration dialog to open the list editor dialog. In the new dialog click . Then select the newly added modification to open the drop down list where you can select the correct modification.

- A common step in analysis is to search not only against a regular protein database, but to also search against a decoy database for FDR estimation. The fasta file we used before already contains such a decoy database. For OpenMS to know which OMSSA PSM came from which part of the file (i.e. target versus decoy), we have to index the results. Therefore extend the workflow with a PeptideIndexer node . This node needs the idXML as input as well as the database file.

Note: You can direct the files of an Input File node to more than just one destination port.

- The decoys in the database are prefixed with ``REV_``, so we have to set *decoy_string* to *REV_* and *prefix* to *true* in the configuration dialog of PeptideIndexer.

- Now we can go for the FDR estimation, which the `FalseDiscoveryRate` node will calculate for us `Community Nodes` `OpenMS` `ID Processing`. As we have a combined search database and thus only one `idXML` per `mzML` we will only use the `in` port of the `FalseDiscoveryRate` node.
- In order to set the FDR level to 1%, we need an `IDFilter` node from `Community Nodes` `OpenMS` `ID Processing`. Configuring its parameter `score` \rightarrow `pep` to 0.01 will do the trick. The FDR calculations (embedded in the `idXML`) from the `FalseDiscoveryRate` node will go into the `in` port of the `IDFilter` node.
- Execute your workflow and inspect the results using `IDTextReader` like you did before. How many peptides did you identify at this FDR threshold?

Note: The finished identification workflow is now sufficiently complex that we might want to encapsulate it in a Meta node. For this, select all nodes inside the `ZipLoop` (including the `Input File` node) and right-click to select `Collapse into Meta node` and name it `ID`. Meta nodes are useful when you construct even larger workflows and want to keep an overview.

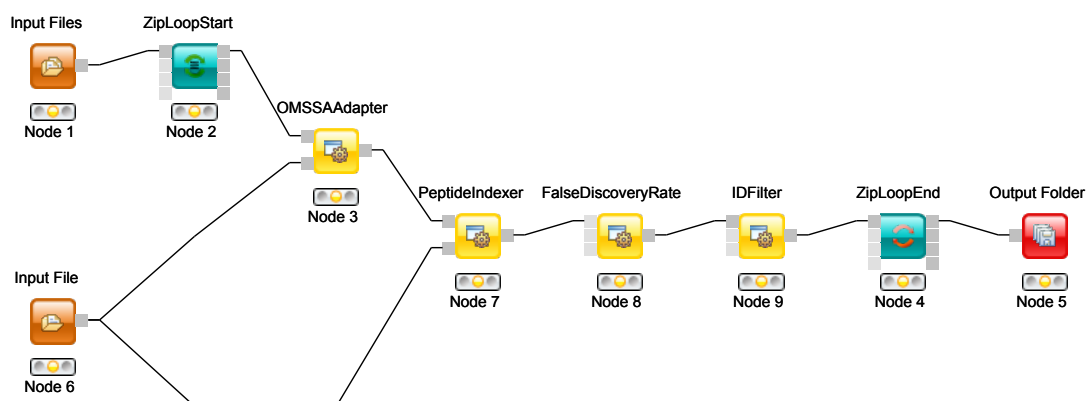
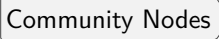
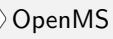
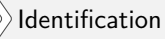

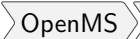








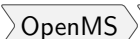



Figure 6: OMSSA ID pipeline including FDR filtering.

3.2.1 Bonus task: identification using several search engines

Note: If you are ahead of the tutorial or later on, you can further improve your FDR identification workflow by a so-called consensus identification using several search engines. Otherwise, just continue with section 3.3.

It has become widely accepted that the parallel usage of different search engines can increase peptide identification rates in shotgun proteomics experiments. The ConsensusID algorithm is based on the calculation of posterior error probabilities (PEP) and a combination of the normalized scores by considering missing peptide sequences.

- Next to the OMSSAAdapter add a XTandemAdapter    node and set its parameters and ports analogously to the OMSSAAdapter.
- To calculate the PEP, introduce each a IDPosteriorErrorProbability    node to the output of each ID engine adapter node. This will calculate the PEP to each hit and output an updated idXML.
- To create a consensus, we must first merge these two files with a FileMerger node    so we can then merge the corresponding IDs with a IDMerger   .
- Now we can create a consensus identification with the ConsensusID    node. We can connect this to the PeptideIndexer and go along with our existing FDR filtering.

Note: By default, X!Tandem takes additional enzyme cutting rules into consideration (besides the specified tryptic digest). Thus for the tutorial files, you have to set PeptideIndexer's *enzyme* → *specificity* parameter to none to accept X!Tandem's non-tryptic identifications as well.

3.3 Quantification

Now that we have successfully constructed a peptide identification pipeline, we can add quantification capabilities to our workflow.

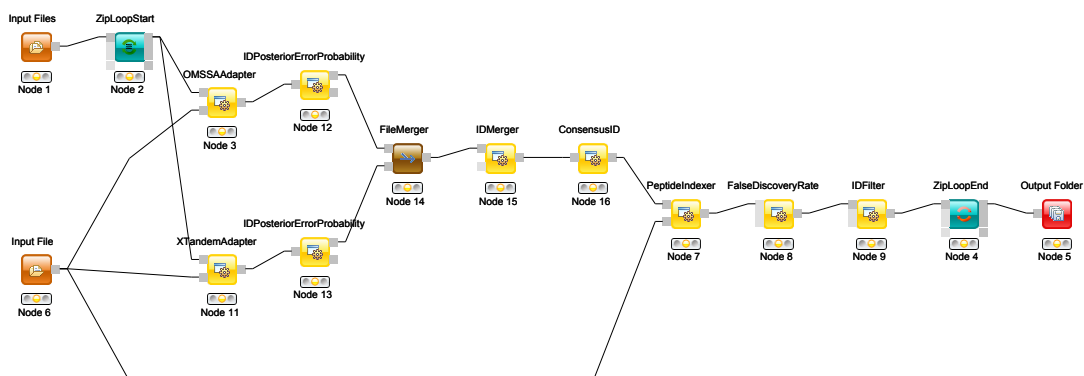


Figure 7: Complete consensus identification workflow.

- Add a FeatureFinderCentroided node Community Nodes OpenMS Quantitation which gets input from the first output port of the ZipLoopStart node. Also, add an IDMapper node Community Nodes OpenMS ID Processing which gets input from the FeatureFinderCentroided node and the ID Meta node (or IDFilter node if you haven't used the Meta node). The output of the IDMapper is then connected to an *in* port of the ZipLoopEnd node.
- FeatureFinderCentroided finds and quantifies peptide ion signals contained in the MS1 data. It reduces the entire signal, i.e., all peaks explained by one and the same peptide ion signal, to a single peak at the maximum of the chromatographic elution profile of the monoisotopic mass trace of this peptide ion and assigns an overall intensity.
- FeatureFinderCentroided produces a featureXML file as output, containing only quantitative information of so-far unidentified peptide signals. In order to annotate these with the corresponding ID information, we need the IDMapper node.
- Run your pipeline and inspect the results of the IDMapper node in TOPPView.
- In order to assess how well the feature finding worked, you can project the features contained in the featureXML file on the raw data contained in the mzML file. In TOPPView choose File Open file and select the mzML file corresponding to your featureXML file in Example_Data Labelfree datasets. In the dialog that pops up, select Open in New layer. Zoom in until you see boxes (found features) around the peptide signals in the raw data.

Note: The RT range is very narrow. Thus, select the full RT range and zoom only into the m/z dimension by holding down CTRL (CMD on Mac) and repeatedly dragging a narrow box from the very left to the very right.

- You can see which features were annotated with a peptide identification by first selecting the featureXML file in the Layers window on the upper right side and then clicking on the icon with the letters A, B and C on the upper icon bar. Now, click on the small triangle next to that icon and select Peptide identification.

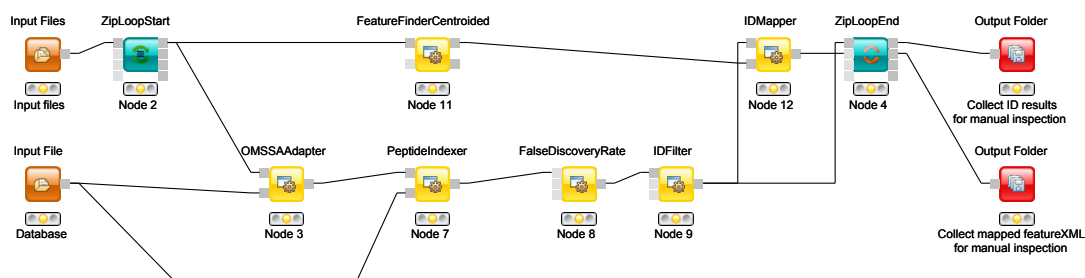


Figure 8: Extended workflow featuring peptide identification and quantification.

3.4 Combining quantitative information across several label-free experiments

So far, we successfully performed peptide identification as well as quantification on individual LC-MS runs. For differential label-free analyses, however, we need to identify and quantify corresponding signals in different experiments and link them together to compare their intensities. Thus, we will now run our pipeline on all three available input files and extend it a bit further, so that it is able to find and link features across several runs.

- To find features across several maps, we first have to align them to correct for retention time shifts between the different label-free measurements. With the `MapAlignerPoseClustering` `Community Nodes` `OpenMS` `Map Alignment`, we can align corresponding peptide signals to each other as closely as possible by applying a transformation in the RT dimension.

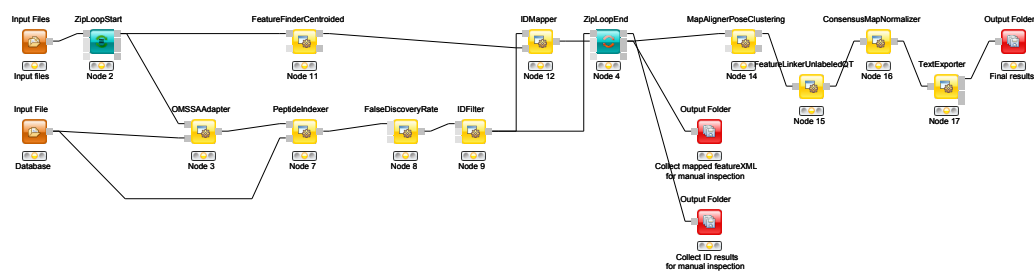


Figure 9: Complete identification and label-free quantification workflow.

Note: MapAlignerPoseClustering consumes several featureXML files and its output should still be several featureXML files containing the same features, but with the transformed RT values. In its configuration dialog, make sure that *OutputTypes* is set to featureXML.

- With the FeatureLinkerUnlabeledQT node Community Nodes OpenMS Map Alignment, we can then perform the actual linking of corresponding features. Its output is a consensusXML file containing linked groups of corresponding features across the different experiments.
- Since the overall intensities can vary a lot between different measurements (for example, because the amount of injected analytes was different), we apply the ConsensusMapNormalizer Community Nodes OpenMS Map Alignment as a last processing step. Configure its parameters with setting *algorithm_type* to median. It will then normalize the maps in such a way that the median intensity of all input maps is equal.
- Finally, we export the resulting normalized consensusXML file to a csv format using TextExporter. Connect its out port to a new Output Folder node.

Note: You can specify the desired column separation character in the parameter settings (by default, it is set to ` ` (a space)). The output file of TextExporter can also be opened with external tools, e.g., Microsoft Excel, for downstream statistical analyses.

3.4.1 Basic data analysis in KNIME

For downstream analysis of the quantification results within the KNIME environment, you can use the `ConsensusTextReader` node (Community Nodes >> OpenMS >> Conversion) instead of the `Output Folder` node to convert the output into a KNIME table (indicated by a triangle as output port). After running the node you can view the KNIME table by right clicking on the `ConsensusTextReader` and selecting `Consensus Table`. Every row in this table corresponds to a so-called consensus feature, i.e., a peptide signal quantified across several runs. The first couple of columns describe the consensus feature as a whole (average RT and m/z across the maps, charge, etc.). The remaining columns describe the exact positions and intensities of the quantified features separately for all input samples (e.g., `intensity_0` is the intensity of the feature in the first input file). The last 11 columns contain information on peptide identification.

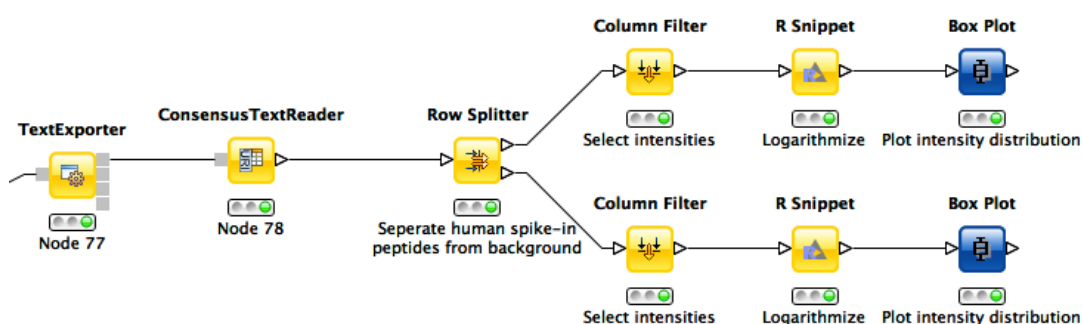





Figure 10: Simple KNIME data analysis example for LFQ.

- Now, let's say we want to plot the log intensity distributions of the human spike-in peptides for all input files. In addition, we will plot the intensity distributions of the background peptides.
- As shown in Fig. 10, add a `Row Splitter` node (Data Manipulation >> Row >> Filter) after `ConsensusTextReader`. Double-click it to configure. The human spike-in peptides have accessions starting with ```hum```. Thus, set the column to test to `accessions`, select pattern matching as matching criterion, enter `hum*` into the corresponding text field, and check the `contains wild cards` box. Press OK and execute the node.




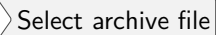
- Row Splitter produces two output tables: the first one contains all rows from the input table matching the filter criterion, and the second table contains all other rows. You can inspect the tables by right-clicking and selecting *Filtered* and *Filtered Out*. The former table should now contain only peptides with a human accession, whereas the latter should contain all remaining peptides (including unidentified ones).
- Now, since we only want to plot intensities, we can add a Column Filter node , connect its input port to the *Filtered* output port of the Row Filter, and open its configuration dialog. We could either manually select the columns we want to keep, or, more elegantly, select *Wildcard/Regex Selection* and enter *intensity_?* as the pattern. KNIME will interactively show you which columns your pattern applies to while you're typing.
- Since we want to plot log intensities, we will now compute the log of all intensity values in our table. The easiest way to do this in KNIME is a small piece of R code. Add an R Snippet node  after Column Filter and double-click to configure. In the *R Script* text editor, enter the following code:

```
x <- knime.in           # store copy of input table in x
x[x == 0] <- NA         # replace all zeros by NA (= missing value)
x <- log10(x)           # compute log of all values
knime.out <- x          # write result to output table
```

- Now we are ready to plot! Add a Box Plot node  after the R Snippet node, execute it, and open its view. If everything went well, you should see a significant fold change of your human peptide intensities across the three runs.
- In order to verify that the concentration of background peptides is constant in all three runs, you can just copy and paste the three nodes after Row Splitter and connect the duplicated Column Filter to the second output port (*Filtered Out*) of Row Splitter, as shown in Fig. 10. Execute and open the view of your second Box Plot.
- That's it! You have constructed an entire identification and label-free quantification workflow including a simple data analysis using KNIME!

Note: For further inspiration you might want to take a look at the more advanced KNIME data analysis examples in the metabolomics tutorial.

4 Protein Inference

In the last chapter, we have successfully quantified peptides in a label-free experiment. As a next step, we will further extend this label-free quantification workflow by protein inference and protein quantification capabilities. This workflow uses some of the more advanced concepts of KNIME, as well as a few more nodes containing R code. For these reasons, you will not have to build it yourself. Instead, we have already prepared and copied this workflow to the USB sticks. Just import  Workflows > Protein Inference > protein_inference.zip into KNIME via  File >>  Import KNIME workflow >>  Select archive file and double-click the imported workflow in order to open it.

Before you can execute the workflow, you again have to correct the locations of the files in the Input Files nodes (don't forget the one for the FASTA database inside the ``ID'' meta node). Try and run your workflow.

4.1 Extending the LFQ workflow by protein inference and quantification

We have made the following changes compared to the original label-free quantification workflow from the last chapter:

- First, we have added a ProteinQuantifier node and connected its input port to the output port of ConsensusMapNormalizer.
- This already enables protein quantification. ProteinQuantifier quantifies peptides by summarizing over all observed charge states and proteins by summarizing over their quantified peptides. It stores two output files, one for the quantified peptides and one for the proteins.
- In this example, we consider only the protein quantification output file, which is written to the first output port of ProteinQuantifier
- Because there is no dedicated node in KNIME to read back the ProteinQuantifier output file format into a KNIME table, we have to use a workaround. Here, we have added an additional URI Port to Variable node which converts the name of the output file to a so-called ``flow variable'' in KNIME. This variable is passed on to the

next node CSV Reader, where it is used to specify the name of the input file to be read. If you double-click on CSV Reader, you will see that the text field, where you usually enter the location of the CSV file to be read, is greyed out. Instead, the flow variable is used to specify the location, as indicated by the small green button with the ``v=?" label on the right.

- The table containing the ProteinQuantifier results is filtered one more time in order to remove decoy proteins. You can have a look at the final list of quantified protein groups by right-clicking the Row Filter and selecting Filtered.
- By default, i.e., when the second input port *protein_groups* is not used, ProteinQuantifier quantifies proteins using only the unique peptides, which usually results in rather low numbers of quantified proteins.
- In this example, however, we have performed protein inference using Fido and used the resulting protein grouping information to also quantify indistinguishable proteins
- As a prerequisite for using FidoAdapter, we have added an IDPosteriorErrorProbability node within the ID meta node, between OMSSAAdapter and PeptideIndexer. We have set its parameter *prob_correct* to *true*, so it computes posterior probabilities instead of posterior error probabilities (1 - PEP). These are stored in the resulting idXML file and later on used by the Fido algorithm.
- Next, we have added a third outgoing connection to our ID meta node and connected it to the second input port of ZipLoopEnd. Thus, KNIME will wait until all input files have been processed by the loop and then pass on the resulting list of idXML files to the subsequent IDMerger node, which merges all identifications from all idXML files into a single idXML file.
- Instead of the meta node Protein inference with FidoAdapter, we could have just used a FidoAdapter node Community Nodes OpenMS ID Processing. However, the meta node contains an additional subworkflow which, besides calling FidoAdapter, performs a statistical validation of the protein inference results using some of the more advanced KNIME nodes.

4.2 Statistical validation of protein inference results

In the following, we will explain the subworkflow contained in the Protein inference with FidoAdapter meta node.

4.2.1 Data preparation

For downstream analysis on the protein ID level in KNIME, it is again necessary to convert the idXML-file-format result generated from FidoAdapter into a KNIME table.

- By setting *proteins_only* to true in TextExporter, only the protein IDs are exported.
- As the built-in table file reader IDTextReader only reads peptide hits, we have to use URI Port to Variable which collects the URIs from a URI port object and puts them into variables.
- However, doing this will cause missing column information in the converted KNIME table. Open the File Viewer and check what each column stands for, filter away unused columns, such as sequence, coverage and rank in the Column Filter node. Add the column names manually in Column Rename.

4.2.2 ROC curve of protein ID

ROC Curves (Receiver Operating Characteristic curves) are graphical plots that visualize sensitivity (true-positive rate) against fall-out (false positive rate). They are often used to judge the quality of a discrimination method like e.g., peptide or protein identification engines. ROC Curve already provides the functionality of drawing ROC curves for binary classification problems. Before applying this node, an extra column with the class values (target and decoy proteins) has to be appended in the Rule engine node.

In protein or peptide identification, the ground-truth (i.e., which target identifications are true, which are false) is usually not known. Instead, so called pseudo-ROC Curves are regularly used to plot the number of target proteins against the false discovery rate (FDR). The FDR is approximated by using the target-decoy estimate in order to distinguish true IDs from false IDs by separating target IDs from decoy IDs.

4.2.3 Posterior probability and FDR of protein IDs

ROC curves illustrate the discriminative capability of the scores of IDs. In the case of protein identifications, Fido produces the posterior probability of each protein as the output score. However, a perfect score should not only be highly discriminative (distinguishing true from false IDs), it should also be "calibrated" (for probability indicating that all IDs with reported posterior probability scores of 95% should roughly of 5% probability be false. This implies that the estimated number of false positives can be computed as the sum of posterior error probability ($= 1 - \text{posterior probability}$), further an posterior probability estimated FDR is also possible to be computed. Therefore, we can plot calibration curves to help us visualize the quality of the score (when the score is interpreted as a probability as Fido does), by comparing how similar the target-decoy estimated FDR and the posterior probability estimated FDR are. Good results should show a close correspondence between these two measurements.

The calculation is done by using a simple R script in R snippet. First, the target decoy protein FDR is computed as the proportion of decoy proteins among all significant protein IDs. Then posterior probabilistic-driven FDR is estimated by the average of the posterior error probability of all significant protein IDs. Since FDR is the property for a group of protein IDs, we can also calculate a local property: the q -value of a certain protein ID by the minimum value of FDRs of any groups of protein IDs that contain this protein ID. We plot the protein ID results versus two different kinds of FDR estimates in R View(Table) (see Figure 12).

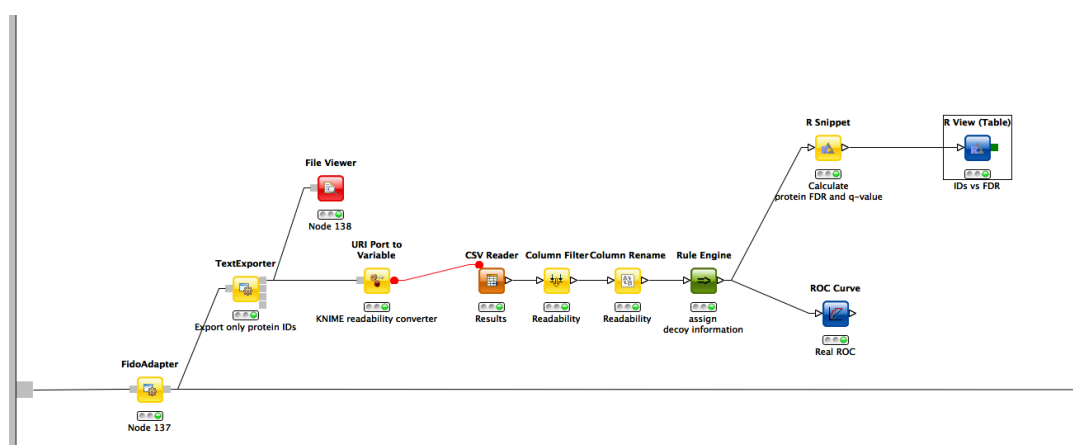


Figure 11: The workflow of statistical analysis of protein inference results

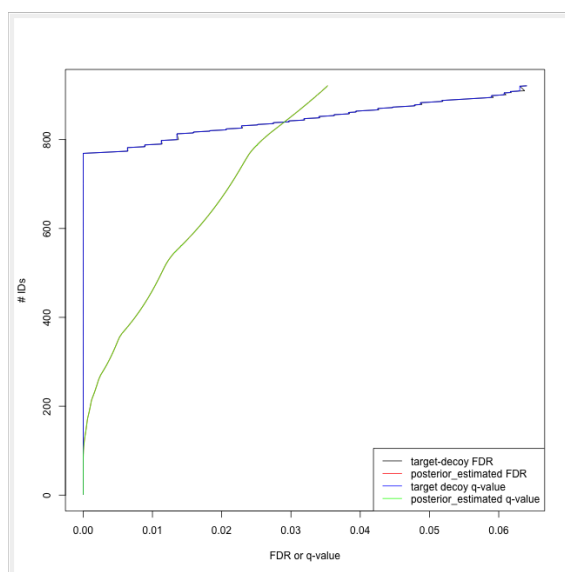


Figure 12: the pseudo-ROC Curve of protein IDs. The accumulated number of protein IDs is plotted on two kinds of scales: target-decoy protein FDR and Fido posterior probability estimated FDR. The largest value of posterior probability estimated FDR is already smaller than 0.04, this is because the posterior probability output from Fido is generally very high.


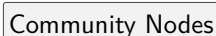
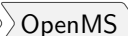
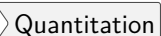
5 Metabolomics

5.1 Introduction

Quantitation and identification of chemical compounds are basic tasks in metabolomic studies. In this tutorial session we construct a UPLC-MS based, label-free quantitation and identification workflow. Following quantitation and identification we then perform statistical downstream analysis to detect quantitation values that differ significantly between two conditions. This approach can, for example, be used to detect biomarkers. Here, we use two spike-in conditions of a dilution series (0.5 mg/l and 10.0 mg/l, male blood background, measured in triplicates) comprising seven isotopically labeled compounds. The goal of this tutorial is to detect and quantify these differential spike-in compounds against the complex background.

5.2 Quantifying metabolites across several experiments

For the metabolite quantification we choose an approach similar to the one used for peptides, but this time based on the OpenMS FeatureFinderMetabo method. This feature finder again collects peak picked data into individual mass traces. The reason why we need a different feature finder for metabolites lies in the step after trace detection: the aggregation of isotopic traces belonging to the same compound ion into the same feature. Compared to peptides with their averagine model, small molecules have very different isotopic distributions. To group small molecule mass traces correctly, an aggregation model tailored to small molecules is thus needed.

- Create a new workflow called for instance "Metabolomics".
- Add an Input Files node and configure it with all mzML files from  Example_Data ▶ Metabolomics ▶ datasets.
- Add a ZipLoopStart node and connect the Input Files node to the first port of the ZipLoopStart node.
- Add a FeatureFinderMetabo node (from  Community Nodes >>  OpenMS >>  Quantitation) and connect the first output port of the ZipLoopStart to the FeatureFinderMetabo.

- For an optimal result adjust the following settings. Please note that some of these are advanced parameters.

parameter	value
<i>algorithm</i> → <i>common</i> → <i>chrom_fwhm</i>	8.0
<i>algorithm</i> → <i>mtd</i> → <i>trace_termination_criterion</i>	sample_rate
<i>algorithm</i> → <i>mtd</i> → <i>min_trace_length</i>	3.0
<i>algorithm</i> → <i>mtd</i> → <i>max_trace_length</i>	600.0
<i>algorithm</i> → <i>epd</i> → <i>width_filtering</i>	off

- Add a ZipLoopEnd node and connect the output of the FeatureFinderMetabo to the first port of the ZipLoopEnd node.

To facilitate the collection of features corresponding to the same compound ion across different samples, an alignment of the samples' feature maps along retention time is often helpful. In addition to local, small-scale elution differences, one can often see constant retention time shifts across large sections between samples. We can use linear transformations to correct for these large scale retention differences. This brings the majority of corresponding compound ions close to each other. Finding the correct corresponding ions is then faster and easier, as we don't have to search as far around individual features.

- After the ZipLoopEnd node add a MapAlignerPoseClustering node (Community Nodes > OpenMS > Map Alignment), set its Output Type to featureXML, and adjust the following settings

parameter	value
<i>algorithm</i> → <i>max_num_peaks_considered</i>	-1
<i>algorithm</i> → <i>superimposer</i> → <i>mz_pair_max_distance</i>	0.005
<i>algorithm</i> → <i>superimposer</i> → <i>num_used_points</i>	10000
<i>algorithm</i> → <i>pairfinder</i> → <i>distance_RT</i> → <i>max_difference</i>	20.0
<i>algorithm</i> → <i>pairfinder</i> → <i>distance_MZ</i> → <i>max_difference</i>	20.0
<i>algorithm</i> → <i>pairfinder</i> → <i>distance_MZ</i> → <i>unit</i>	ppm

The next step after retention time correction is the grouping of corresponding features in multiple samples. In contrast to the previous alignment, we assume no linear relations of features across samples. The used method is tolerant against local swaps in elution order.

- After the `MapAlignerPoseClustering` add a `FeatureLinkerUnlabeledQT` (Community Nodes >> OpenMS >> Map Alignment) and adjust the following settings

parameter	value
<code>algorithm → distance_RT → max_difference</code>	40.0
<code>algorithm → distance_MZ → max_difference</code>	20.0
<code>algorithm → distance_MZ → unit</code>	ppm

- After the `FeatureLinkerUnlabeledQT` add a `TextExporter` node (Community Nodes >> OpenMS >> File Handling).
- Add an `Output Folder` node and configure it with an output directory where you want to store the resulting files.
- Run the pipeline and inspect the output.

You should find a single, tab-separated file containing the information on where metabolites were found and with which intensities. You can also add `Output Folder` nodes at different stages of the workflow and inspect the intermediate results (e.g., identified metabolite features for each input map). The complete workflow can be seen in Figure 13. In the following section we will try to identify those metabolites.

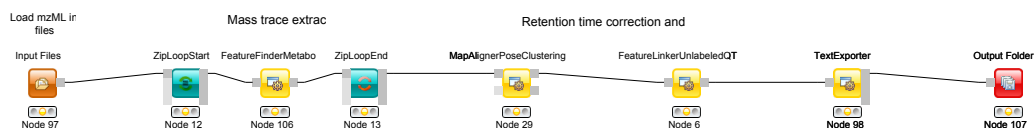


Figure 13: Label-free quantification workflow for metabolites

5.3 Identifying metabolites in LC-MS/MS samples

At the current state we found several metabolites in the individual maps but so far don't know what they are. To identify metabolites OpenMS provides multiple tools, including search by mass: the `AccurateMassSearch` node searches observed masses against the Human Metabolome Database (HMDB)[8, 9, 10]. We start with the workflow from the previous section (see Figure 13).

- Add a `FileConverter` node (Community Nodes >> OpenMS >> File Handling) and connect the output of the `FeatureLinkerUnlabeledQT` to the incoming port.
- Open the Configure dialog of the `FileConverter` and select the tab "OutputTypes". In the drop down list for `FileConverter.1.out` select "featureXML".
- Add an `AccurateMassSearch` node (Community Nodes >> OpenMS >> Utilities) and connect the output of the `FileConverter` to the first port of the `AccurateMassSearch`.
- Add four Input File nodes and configure them with the following files
 - `Example_Data > Metabolomics > databases > PositiveAdducts.tsv`
This file specifies the list of adducts that are considered in the positive mode. Each line contains the formula and charge of an adduct separated by a semi-colon (e.g. `M+H;1+`). The mass of the adduct is calculated automatically.
 - `Example_Data > Metabolomics > databases > NegativeAdducts.tsv`
This file specifies the list of adducts that are considered in the negative mode analogous to the positive mode.
 - `Example_Data > Metabolomics > databases > HMDBMappingFile.tsv`
This file contains information from a metabolite database in this case from HMDB. It has three (or more) tab-separated columns: mass, formula, and identifier(s). This allows for an efficient search by mass.
 - `Example_Data > Metabolomics > databases > HMDB2StructMapping.tsv`
This file contains additional information about the identifiers in the mapping file. It has four tab-separated columns that contain the identifier, name, SMILES, and INCHI. These will be included in the result file. The identifiers in this file must match the identifiers in the `HMDBMappingFile.tsv`.

- In the same order as they are given above connect them to the remaining input ports of the AccurateMassSearch node.
- Add an Output Folder node and connect the first output port of the AccurateMassSearch node to the Output Folder.

The result of the AccurateMassSearch node is in the mzTab format [11] so you can easily open it in a text editor or import it into Excel or KNIME, which we will do in the next section. The complete workflow from this section is shown in Figure 14.

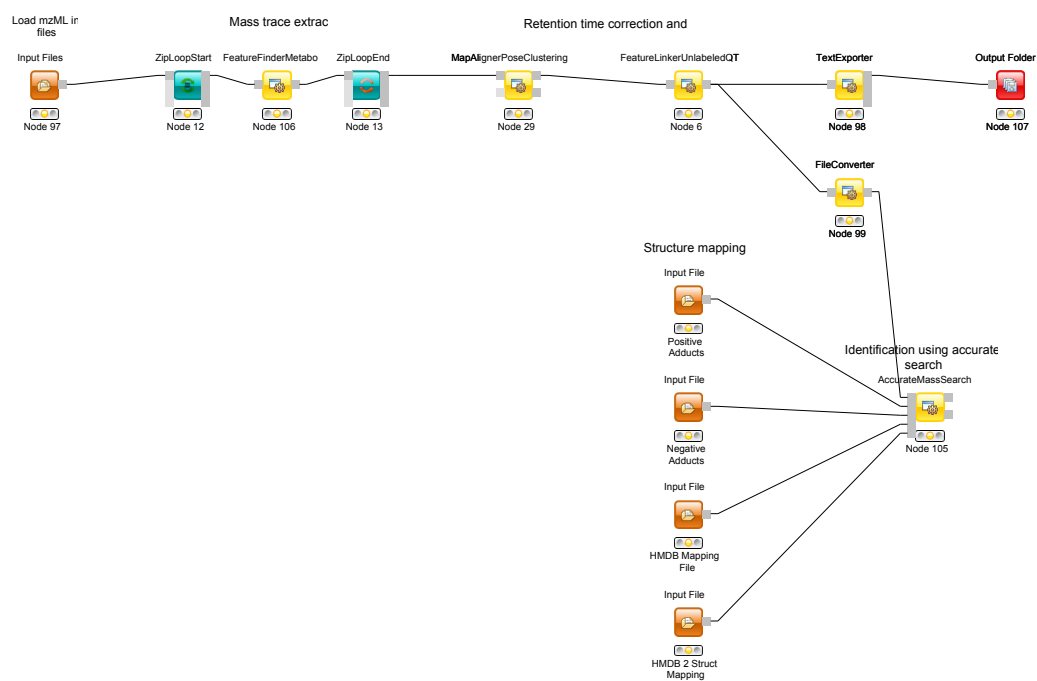


Figure 14: Label-free quantification and identification workflow for metabolites

5.4 Convert your data into a KNIME table

The result from the TextExporter node as well as the result from the AccurateMassSearch node are files while standard KNIME nodes display and process only KNIME tables. To convert these files into KNIME tables we need two different nodes. For the AccurateMassSearch results we use the MzTabReader node (Community Nodes >> OpenMS >> Conversion >> mzTab) and its

Small Molecule Section port. For the result of the TextExporter we use the ConsensusTextReader (Community Nodes >> OpenMS >> Conversion).

When executed, both nodes will import the OpenMS files and provide access to the data as KNIME tables. You can now easily combine both tables using the Joiner node (Manipulation >> Column >> Split & Combine) and configure it to match the m/z and retention time values of the respective tables. The full workflow is shown in Figure 15.

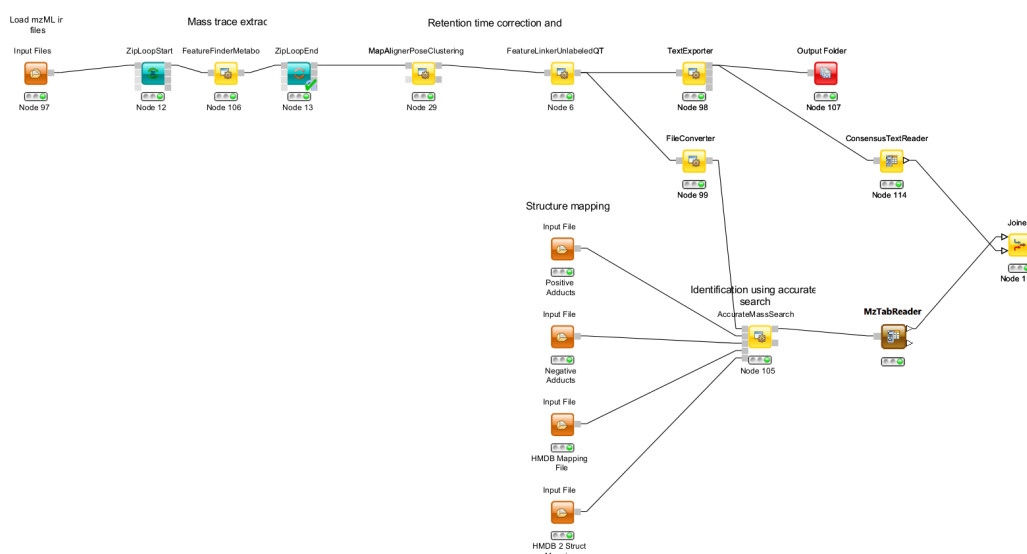


Figure 15: Label-free quantification and identification workflow for metabolites that loads the results into KNIME and joins the tables.

5.4.1 Bonus task: Visualizing data

Now that you have your data in KNIME you should try to get a feeling for the capabilities of KNIME.

Task



Check out the Molecule Type Cast node (Chemistry >> Translators) together with subsequent cheminformatics nodes (e.g. RDKit From Molecule (Community Nodes >> RDKit >> Converters)) to render the structural formula contained in the result table.

Task



Have a look at the Column Filter node to reduce the table to the interesting columns, e.g., only the Ids, chemical formula, and intensities.

Task





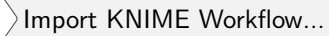
Try to compute and visualize the m/z and retention time error of the different elements of the consensus features.

5.5 Downstream data analysis and reporting

In this part of the metabolomics session we take a look at more advanced downstream analysis and the use of the statistical programming language R. As laid out in the introduction we try to detect a set of spike-in compounds against a complex blood background. As there are many ways to perform this type of analysis we provide a complete workflow.

Task



Import the workflow from  Workflows ▶ metabolite_ID.zip in KNIME:  

The section below will guide you in your understanding of the different parts of the workflow. Once you understood the workflow you should play around and be creative. Maybe create a novel visualization in KNIME or R? Do some more elaborate statistical analysis? Note that some basic R knowledge is required to fully understand the processing in R Snippet nodes.

5.5.1 Signal processing and Data preparation ID

This part is analogous to what you did for the simple metabolomics pipeline.

5.5.2 Data preparation Quant

The first part is identical to what you did for the simple metabolomics pipeline. Additionally, we convert zero intensities into NA values and remove all rows that contain at least

one NA value from the analysis. We do this using a very simple R Snippet and subsequent Missing Value filter node.

Task



Inspect the R Snippet by double-clicking on it. The KNIME table that is passed to an R Snippet node is available in R as a data.frame named `knime.in`. The result of this node will be read from the data.frame `knime.out` after the script finishes. Try to understand and evaluate parts of the script (Eval Selection). In this dialog you can also print intermediary results using for example the R command `head()` or `cat()` to the Console pane.

5.5.3 Statistical analysis

After we linked features across all maps, we want to identify features that are significantly deregulated between the two conditions. We will first scale and normalize the data, then perform a t-test, and finally correct the obtained p-values for multiple testing using Benjamini-Hochberg. All of these steps will be carried out in individual R Snippet nodes.

- Double-click on the first R Snippet node labeled "log scaling" to open the R Snippet dialog. In the middle you will see a short R script that performs the log scaling. To perform the log scaling we use a so-called regular expression (`grep`) to select all columns containing the intensities in the six maps and take the \log_2 logarithm.
- The output of the log scaling node is also used to draw a boxplot that can be used to examine the structure of the data. Since we only want to plot the intensities in the different maps (and not `m/z` or `rt`) we first use a Column Filter node to keep only the columns that contain the intensities. We connect the resulting table to a Box Plot node which draws one box for every column in the input table. Right-click and select `View: Box Plot`.
- The median normalization is performed in a similar way to the log scaling. First we calculate the median intensity for each intensity column, then we subtract the median from every intensity.

- Open the `Box Plot` connected to the normalization node and compare it to the box plot connected to the log scaling node to examine the effect of the median normalization.
- To perform the t-test we defined the two groups we want to compare. Then we call the t-test for every consensus feature unless it has missing values. Finally we save the p-values and fold-changes in two new columns named p-value and FC.
- The `Numeric Row Splitter` is used to filter less interesting parts of the data. In this case we only keep columns where the fold-change is ≥ 2 .
- We adjust the p-values for multiple testing using Benjamini-Hochberg and keep all consensus features with a q-value ≤ 0.01 (i.e. we target a false-discovery rate of 1%).

5.5.4 Interactive visualization

KNIME supports multiple nodes for interactive visualization with interrelated output. The nodes used in this part of the workflow exemplify this concept. They further demonstrate how figures with data dependent customization can be easily realized using basic KNIME nodes. Several simple operations are concatenated in order to enable an interactive volcano plot.

- We first log-transform fold changes and p-values in the `R Snippet` node. We then append columns noting interesting features (concerning fold change and p-value).
- With this information, we can use various `Manager` nodes (`Views` `Property`) to emphasize interesting data points. The configuration dialogs allow us to select columns to change color, shape or size of data points dependent on the column values.
- The `Scatter Plot` node (`Views`) enables interactive visualization of the logarithmized values as a volcano plot: the log-transformed values can be chosen in the 'Column Selection' tab of the plot view. Data points can be selected in the plot and HiLited via the menu option. HiLiteing transfers to all other interactive nodes connected to the same data table. In our case, selection and HiLiteing will also occur in the `Interactive Table` node (`Views`).

- Output of the interactive table can then be filtered via the HiLite menu tab. For example, we could restrict shown rows to points HiLited in the volcano plot.

Task



Inspect the nodes of this section. Customize your visualization and possibly try to visualize other aspects of your data.

5.5.5 Advanced visualization

R Dependencies: This section requires that the R packages `ggplot2` and `ggbiplot` are both installed. `ggplot2` is part of the *KNIME R Statistics Integration (Windows Binaries)* which should already be installed via the full KNIME installer, `ggbiplot` however is not. In case that you use an R installation where one or both of them are not yet installed, add an R Snippet node and double-click to configure. In the *R Script* text editor, enter the following code:

```
#Include the next line if you also have to install ggplot2:  
install.packages("ggplot2")  
#Include the following lines to install ggbiplot:  
install.packages("devtools")  
library(devtools)  
install_github("vqv/ggbiplot")
```

Press to execute the script.

Even though the basic capabilities for (interactive) plots in KNIME are valuable for initial data exploration, professional looking depiction of analysis results often relies on dedicated plotting libraries. The statistics language R supports the addition of a large variety of packages, including packages providing extensive plotting capabilities. This part of the workflow shows how to use R nodes in KNIME to visualize more advanced figures. Specifically, we make use of different plotting packages to realize heatmaps.

- The used RView (Table) nodes combine the possibility to write R snippet code with visualization capabilities inside KNIME. Resulting images can be looked at in the output RView, or saved via the Image Port Writer node.

- The heatmap nodes make use of the `gplots` library, which is by default part of the R Windows binaries for the KNIME 3.1.1 full installation. We again use regular expressions to extract all measured intensity columns for plotting. For clarity, feature names are only shown in the heatmap after filtering by fold changes.

5.5.6 Data preparation for Reporting

Following the identification, quantification and statistical analysis our data is merged and formatted for reporting. First we want to discard our normalized and logarithmized intensity values in favor of the original ones. To this end we first remove the intensity columns (Column Filter) and add the original intensities back (Joiner). Note that we use an *Inner Join*¹. Combining ID and Quantification table into a single table is again achieved using a Joiner node.

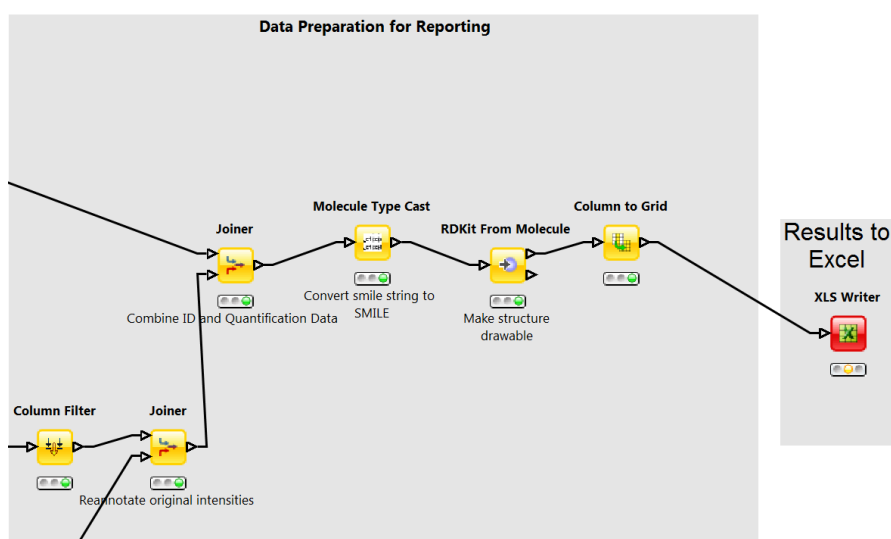


Figure 16: Data preparation for reporting

Question



What happens if we use an *Left Outer Join*, *Right Outer Join* or *Full Outer Join* instead of the *Inner Join*?

¹*Inner Join* is a technical term that describes how database tables are merged.

Task



- Inspect the output of the join operation after the Molecule Type Cast and RDKit molecular structure generation.

While all relevant information is now contained in our table the presentation could be improved. Currently, we have several rows corresponding to a single consensus feature (=linked feature) but with different, alternative identifications. It would be more convenient to have only one row for each consensus feature with all accurate mass identifications added as additional columns. To this end, we use the `Column to Grid` node that flattens several rows with the same consensus number into a single one. Note that we have to specify the maximum number of columns in the grid so we set this to a large value (e.g. 100). We finally export the data to an Excel file (`XLS Writer`).

6 OpenSWATH

6.1 Introduction

OpenSWATH [12] is a module of OpenMS that allows analysis of LC-MS/MS DIA (data independent acquisition) data using the approach described by Gillet *et al.* [13]. The DIA approach described there uses 32 cycles to iterate through precursor ion windows from 400-426 Da to 1175-1201 Da and at each step acquires a complete, multiplexed fragment ion spectrum of all precursors present in that window. After 32 fragmentations (or 3.2 seconds), the cycle is restarted and the first window (400-426 Da) is fragmented again, thus delivering complete "snapshots" of all fragments of a specific window every 3.2 seconds.

The analysis approach described by Gillet *et al.* extracts ion traces of specific fragment ions from all MS2 spectra that have the same precursor isolation window, thus generating data that is very similar to SRM traces.

6.2 Installation of OpenSWATH

OpenSWATH has been fully integrated since OpenMS 1.10 [3, 2, 14]).

6.3 Installation of mProphet

mProphet (<http://www.mprophet.org/>) [15] is available as standalone script in `External_Tools` ▶ `mProphet`. R (<http://www.r-project.org/>) and the package MASS (<http://cran.r-project.org/web/packages/MASS/>) are further required to execute mProphet. Please obtain a version for either Windows, Mac or Linux directly from CRAN.

pyprophet, a much faster reimplementaion of the mProphet algorithm is available from PyPI (<https://pypi.python.org/pypi/pyprophet/>). The usage of pyprophet instead of mProphet is suggested for large-scale applications, but the installation requires more dependencies and therefore, for this tutorial the application of mProphet is described.

6.4 Generating the Assay Library

6.4.1 Generating TraML from transition lists

OpenSWATH requires the assay libraries to be supplied in the TraML format [16]. To enable manual editing of transition lists, the TOPP tool `ConvertTSVToTraML` is available that uses tab separated files as input. Example datasets are provided in `OpenSWATH\assay`. Please note that the transition lists need to be named `.csv` or `.tsv`.

The header of the transition list contains the following variables (with example values in brackets):

PrecursorMz

The mass-to-charge (m/z) of the precursor ion. (728.88)

ProductMz

The mass-to-charge (m/z) of the product or fragment ion. (924.539)

Tr_recalibrated

The **normalized** retention time (or iRT) [17] of the peptide. (26.5)

transition_name

A **unique** identifier for the transition.

(AQUA4SWATH_HMLangeA_ADSTGLVITDPTR(UniMod:267)/2_y8)

CE

The collision energy that should be used for the acquisition. (27)

Optional (not used by OpenSWATH)

LibraryIntensity

The relative intensity of the transition. (3305.3)

transition_group_id

A **unique** identifier for the transition group.

(AQUA4SWATH_HMLangeA_ADSTGLVITDPTR(UniMod:267)/2)

decoy

A binary value whether the transition is target or decoy (target:0, decoy:1). (0)

PeptideSequence

The unmodified peptide sequence. (ADSTGTLVITDPTR)

ProteinName

A unique identifier for the protein. (AQUA4SWATH_HMLangeA)

Annotation

The fragment ion annotation. (y8)
Optional (not used by OpenSWATH)

FullUniModPeptideName

The peptide sequence with UniMod modifications. (ADSTGTLVITDPTR(UniMod:267))

MissedCleavages

The number of missed cleavages during enzymatic digestion. (0)
Optional (not used by OpenSWATH)

Replicates

The number of replicates. (0)
Optional (not used by OpenSWATH)

NrModifications

The number of modifications. (0)
Optional (not used by OpenSWATH)

PrecursorCharge

The precursor ion charge. (2)

GroupLabel

The stable isotope label. (light)
Optional (not used by OpenSWATH)

UniprotID

The Uniprot ID of the protein. ()
Optional (not used by OpenSWATH)

To convert transitions lists to TraML, use `ConvertTSVToTraML`:

Linux or Mac

On the Terminal:

```
ConvertTSVToTraML -in OpenSWATH_SGS_AssayLibrary.csv -out OpenSWATH_SGS_AssayLibrary.↔  
TraML
```

Windows

On the TOPP command line:

```
ConvertTSVToTraML.exe -in OpenSWATH_SGS_AssayLibrary.csv -out OpenSWATH_SGS_AssayLibrary↔  
.TraML
```

6.4.2 Appending decoys to a TraML

In addition to the target assays, OpenSWATH further requires decoy assays in the library which are later used for classification and error rate estimation. For the decoy generation it is crucial that the decoys represent the targets in a realistic but unnatural manner without interfering with the targets. The methods for decoy generation implemented in OpenSWATH include 'shuffle', 'pseudo-reverse', 'reverse' and 'shift'. To append decoys to a TraML, the TOPP tool `OpenSwathDecoyGenerator` can be used:

Linux or Mac

On the Terminal:

```
OpenSwathDecoyGenerator -in OpenSWATH_SGS_AssayLibrary.TraML -out ↔  
OpenSWATH_SGS_AssayLibrary_with_Decoy.TraML -method shuffle -append ↔  
exclude_similar -remove_unannotated
```


Windows



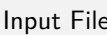

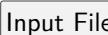

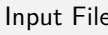

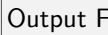
On the TOPP command line:


```
OpenSwathDecoyGenerator.exe -in OpenSWATH_SGS_AssayLibrary.TraML -out ↔  
OpenSWATH_SGS_AssayLibrary_with_Decoy.TraML -method shuffle -append ↔  
exclude_similar -remove_unannotated
```

The flag `-append` generates an output TraML with the complete set of decoy and target assays. The flag `-exclude_similar` is used to exclude decoys which are very similar to the target assays.

6.5 OpenSWATH KNIME

An example KNIME workflow for OpenSWATH is supplied in  Workflows (Figure 17). The example dataset can be used for this workflow (filenames in brackets):

1. Open  Workflows \rightarrow OpenSWATH \rightarrow OpenSWATH.zip in KNIME:  Import KNIME Workflow...
2. Select the normalized retention time (iRT) assay library in TraML format by double-clicking on node  iRT Assay Library.
( OpenSWATH \rightarrow assay \rightarrow OpenSWATH_iRT_AssayLibrary.TraML)
3. Select the SWATH MS data in mzML format as input by double-clicking on node  SWATH-MS files.
( OpenSWATH \rightarrow data \rightarrow split_napedro_L120420_010_SW-*.nf.pp.mzML)
4. Select the target peptide assay library in TraML format as input by double-clicking on node  Assay Library.
( OpenSWATH \rightarrow assay \rightarrow OpenSWATH_SGS_AssayLibrary.TraML)
5. Set the output destination by double-clicking on node .
6. Run the workflow.

The resulting output can be found at your selected path, which will be used as input for mProphet. Execute the script on the Terminal (Linux or Mac) or `cmd.exe` (Windows) in  OpenSWATH \rightarrow result:

```
R --slave --args bin_dir=../../External_Tools/mProphet/ mquest=OpenSWATH_output.csv workflow= $\leftrightarrow$ 
  LABEL_FREE num_xval=5 run_log=FALSE write_classifier=1 write_all_pg=1 < ../../ $\leftrightarrow$ 
  External_Tools/mProphet/mProphet.R
```

The main output will be called

OpenSWATH\result\mProphet_all_peakgroups.xls

with statistical information available in

OpenSWATH\result\mProphet.pdf.

Please note that due to the semi-supervised machine learning approach of mProphet the results differ slightly when mProphet is executed several times.

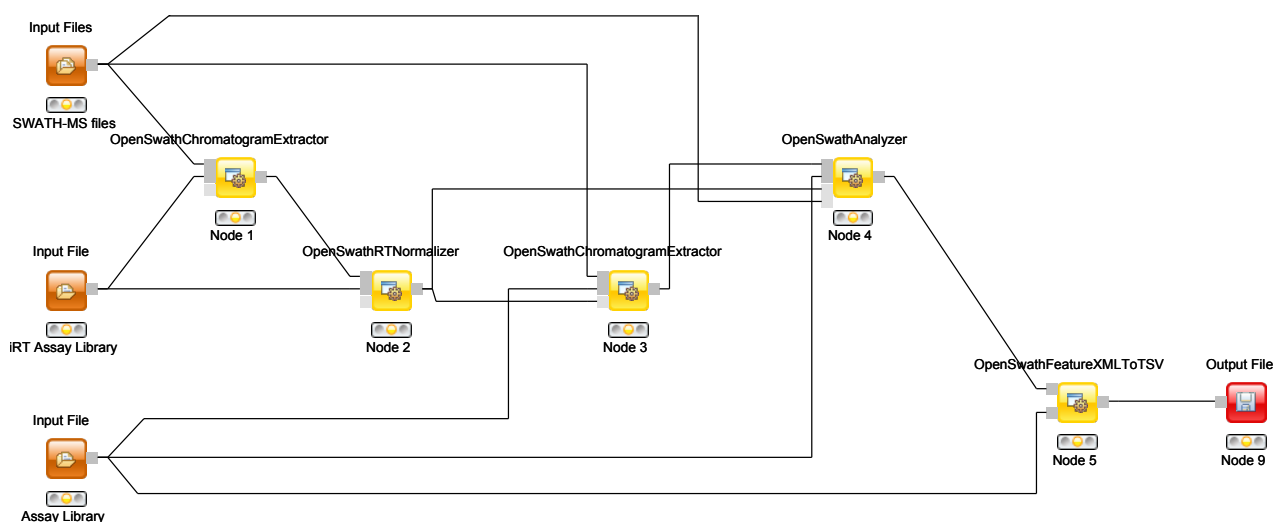


Figure 17: OpenSWATH KNIME Workflow.

6.6 From the example dataset to real-life applications

The sample dataset used in this tutorial is part of the larger SWATH MS Gold Standard (SGS) dataset which is described in the publication of Roest et al. [12]. It contains one of 90 SWATH-MS runs with significant data reduction (peak picking of the raw, profile data) to make file transfer and working with it easier. Usually SWATH-MS datasets are huge with several gigabyte per run. Especially when complex samples in combination with large assay libraries are analyzed, the TOPP tool based workflow requires a lot of computational resources. For this reason, an integrated tool (OpenSwathWorkflow) has been developed, combining all the steps shown in the KNIME-Workflow into a single executable. It is shipped with OpenMS 2.0.0. Instructions on how to use this tool can be found on <http://www.openswath.org>.

7 An introduction to pyOpenMS

7.1 Introduction

pyOpenMS provides Python bindings for a large part of the OpenMS library for mass spectrometry based proteomics. It thus provides access to a feature-rich, open-source algorithm library for mass-spectrometry based proteomics analysis. These Python bindings allow raw access to the data-structures and algorithms implemented in OpenMS, specifically those for file access (mzXML, mzML, TraML, mzIdentML among others), basic signal processing (smoothing, filtering, de-isotoping and peak-picking) and complex data analysis (including label-free, SILAC, iTRAQ and SWATH analysis tools).

pyOpenMS is integrated into OpenMS starting from version 1.11. This tutorial is addressed to people already familiar with Python. If you are new to Python, we suggest to start with a Python tutorial (http://en.wikibooks.org/wiki/Non-Programmer%27s_Tutorial_for_Python_2.6).

7.2 Installation

7.2.1 Windows

1. Install Python 2.7 (<http://www.python.org/download/>)
2. Install NumPy (<http://www.lfd.uci.edu/~gohlke/pythonlibs/#numpy>)
3. Install setuptools (<https://pypi.python.org/pypi/setuptools>)
4. On the command line:

```
easy_install pyopenms
```

7.2.2 Mac OS X 10.10

1. On the Terminal:

```
sudo easy_install pyopenms
```

7.2.3 Linux

1. Install Python 2.6 or 2.7 (Debian: python-dev, RedHat: python-devel)
2. Install NumPy (Debian / RedHat: python-numpy)
3. Install setuptools (Debian / RedHat: python-setuptools)
4. On the Terminal:

```
easy_install pyopenms
```

7.3 Build instructions

Instructions on how to build pyOpenMS can be found online (<http://ftp.mi.fu-berlin.de/OpenMS/release-documentation/html/pyOpenMS.html>).

7.4 Your first pyOpenMS tool: pyOpenSwathFeatureXMLToTSV

The first tool that you are going to re-implement is a TOPP tool called OpenSwathFeatureXMLToTSV. Take a look at the help of the tool:

```
OpenSwathFeatureXMLToTSV —help

OpenSwathFeatureXMLToTSV — Converts a featureXML to a mProphet tsv.
Version: 2.0.0 Apr 11 2015, 02:02:58, Revision: 66a7739

Usage:
  OpenSwathFeatureXMLToTSV <options>

Options (mandatory options marked with '*'):
  -in <files>*           Input files separated by blank (valid formats: 'featureXML')
  -tr <file>*            TraML transition file (valid formats: 'traML')
  -out <file>*           Tsv output file (mProphet compatible) (valid formats: 'csv')
  -short_format           Whether to write short (one peptide per line) or long format (←
                        one transition per line).
```

```
-best_scoring_peptide <varname> If only the best scoring feature per peptide should be printed  
    , give the variable name
```

Common TOPP options:

```
-ini <file> Use the given TOPP INI file  
-threads <n> Sets the number of threads allowed to be used by the TOPP tool  
    (default: '1')  
-write_ini <file> Writes the default configuration file  
-help Shows options  
-helphelp Shows all options (including advanced)
```

OpenSwathFeatureXMLToTSV converts a featureXML file to a tab-separated value text file. This example will teach you how to use pyOpenMS in combination with Python to implement such a tool very quickly.

7.4.1 Basics

The first task that your tool needs to be able to do is to read the parameters from the command line and provide a main routine. This is all standard Python and no pyOpenMS is needed so far:

```
#!/usr/bin/env python  
import sys  
  
def main(options):  
  
    # test parameter handling  
    print options.infile, options.traml_in, options.outfile  
  
def handle_args():  
    import argparse  
  
    usage = ""  
    usage += "\nOpenSwathFeatureXMLToTSV -- Converts a featureXML to a mProphet tsv."  
  
    parser = argparse.ArgumentParser(description = usage )  
    parser.add_argument('-in', dest='infile', help = 'An input file containing []features ←  
        featureXMLJ')  
    parser.add_argument('-tr', dest='traml_in', help='An input file containing the transitions ←  
        TraMLJ')  
    parser.add_argument('-out', dest='outfile', help='Output mProphet TSV file [tsv]')  
  
    args = parser.parse_args(sys.argv[1:])  
    return args
```

```
if __name__ == '__main__':
    options = handle_args()
    main(options)
```

Execute this code in the example script

```
./pyOpenMS/OpenSwathFeatureXMLToTSV_basics.py
```

```
python OpenSwathFeatureXMLToTSV_basics.py --help
usage: OpenSwathFeatureXMLToTSV_basics.py [-h] [--in INFILE] [--tr TRAML_IN]
      [-out OUTFILE]
```

OpenSwathFeatureXMLToTSV -- Converts a featureXML to a mProphet tsv.

optional arguments:

```
-h, --help    show this help message and exit
--in INFILE   An input file containing features [featureXML]
--tr TRAML_IN An input file containing the transitions [TraML]
--out OUTFILE Output mProphet TSV file [tsv]
```

```
python OpenSwathFeatureXMLToTSV_basics.py --in data/example.featureXML --tr assay/↔
OpenSWATH_SGS_AssayLibrary.TraML --out example.tsv
data/example.featureXML assay/OpenSWATH_SGS_AssayLibrary.TraML example.tsv
```

The parameters are being read from the command line by the function `handle_args()` and given to the `main()` function of the script, which prints the different variables.

7.4.2 Loading data structures with pyOpenMS

Now we're going to import the `pyOpenMS` module with `import pyopenms` in the header of the script and load the `featureXML`:

```
#!/usr/bin/env python
import pyopenms
import sys

def main(options):
    # load featureXML
    features = pyopenms.FeatureMap()
    fh = pyopenms.FileHandler()
```

```

fh.loadFeatures(options.infile, features)
keys = []
features[0].getKeys(keys)
print keys

def handle_args():
    import argparse

    usage = ""
    usage += "\nOpenSwathFeatureXMLToTSV -- Converts a featureXML to a mProphet tsv."

    parser = argparse.ArgumentParser(description = usage )
    parser.add_argument('-in', dest='infile', help = 'An input file containing features [↵
        featureXML]')
    parser.add_argument('-tr', dest='traml_in', help='An input file containing the transitions [↵
        TraML]')
    parser.add_argument('-out', dest='outfile', help='Output mProphet TSV file [tsv]')

    args = parser.parse_args(sys.argv[1:])
    return args

if __name__ == '__main__':
    options = handle_args()
    main(options)

```

The function `pyopenms.FeatureMap()` initializes an OpenMS FeatureMap data structure. The function `pyopenms.FileHandler()` prepares a filehandler with the variable name `fh` and `fh.loadFeatures(options.infile, features)` takes the filename and imports the featureXML into the FeatureMap data structure.

In the next step, we're accessing the keys using the function `getKeys()` and printing them to stdout:

```

python OpenSwathFeatureXMLToTSV_datastructures1.py -in data/example.featureXML -tr assay/↵
OpenSWATH_SGS_AssayLibrary.TraML -out example.tsv
['PeptideRef', 'leftWidth', 'rightWidth', 'total_xic', 'peak_apices_sum', 'var_xcorr_coelution', ↵
'var_xcorr_coelution_weighted ', 'var_xcorr_shape', 'var_xcorr_shape_weighted', '↵
var_library_corr', 'var_library_rmsd', 'var_library_manhattan', 'var_library_dotprod', '↵
delta_rt', 'assay_rt', 'norm_RT', 'rt_score', 'var_norm_rt_score', 'var_intensity_score', '↵
nr_peaks', 'sn_ratio', 'var_log_sn_score', 'var_elution_model_fit_score', '↵
xx_lda_prelim_score', 'var_isotope_correlation_score', 'var_isotope_overlap_score', '↵
var_massdev_score', 'var_massdev_score_weighted', 'var_bseries_score', 'var_yseries_score', ↵
'var_dotprod_score', 'var_manhatt_score', 'main_var_xx_swath_prelim_score', 'PrecursorMZ', '↵
xx_swath_prelim_score']

```

In the next task, please load the TraML into an OpenMS TargetedExperiment data structure, analogously to the featureXML. You might want to consult the pyOpenMS manual (http://proteomics.ethz.ch/pyOpenMS_Manual.pdf), which provides an overview of all functionality. If you have trouble reading the TraML, search for TraMLFile(). If you can't solve the task, take a look at OpenSwathFeatureXMLToTSV_datastructures2.py

7.4.3 Converting data in the featureXML to a TSV

Now that all data structures are populated, we need to access the data using the provided API and store it in something that is directly accessible from Python. We prepared two functions for you: `get_header()` & `convert_to_row()`:

```
def get_header(features):
    keys = []
    features[0].getKeys(keys)
    header = [
        "transition_group_id",
        "run_id",
        "filename",
        "RT",
        "id",
        "Sequence",
        "FullPeptideName",
        "Charge",
        "m/z",
        "Intensity",
        "ProteinName",
        "decoy"]
    header.extend(keys)
    return header
```

`get_header()` takes as input a FeatureMap and uses the `getKeys()` function that you have seen before to extend a predefined header list based on the contents of the FeatureMap. The return variable is a native Python list.

```
def convert_to_row(first, targ, run_id, keys, filename):
    peptide_ref = first.getMetaValue("PeptideRef")
    pep = targ.getPeptideByRef(peptide_ref)
    full_peptide_name = "NA"
    if (pep.metaValueExists("full_peptide_name")):
        full_peptide_name = pep.getMetaValue("full_peptide_name")
```

```

decoy = "0"
peptidetransitions = [t for t in targ.getTransitions() if t.getPeptideRef() == peptide_ref]
if len(peptidetransitions) > 0:
    if peptidetransitions[0].getDecoyTransitionType() == pyopenms.DecoyTransitionType().DECOY↔
        :
        decoy = "1"
    elif peptidetransitions[0].getDecoyTransitionType() == pyopenms.DecoyTransitionType().↔
        TARGET:
        decoy = "0"

protein_name = "NA"
if len(pep.protein_refs) > 0:
    protein_name = pep.protein_refs[0]

row = [
    first.getMetaValue("PeptideRef"),
    run_id,
    filename,
    first.getRT(),
    first.getUniqueId(),
    pep.sequence,
    full_peptide_name,
    pep.getChargeState(),
    first.getMetaValue("PrecursorMZ"),
    first.getIntensity(),
    protein_name,
    decoy
]

for k in keys:
    row.append(first.getMetaValue(k))

return row

```

convert_to_row() is a bit more complicated and takes as first input a Feature OpenMS class. From this, we access stored values using the provided functions (getRT(), getUniqueId(), etc). It further takes a TargetedExperiment to access information from the TraML with the provided routines. This data is then stored in a standard Python list with the variable name row and returned.

7.4.4 Putting things together

Now put these two functions into the header of

OpenSwathFeatureXMLToTSV_datastructures2.py.

Your final goal is to implement the conversion functionality into the main function using `get_header()` & `convert_to_row()` and to write a TSV using the standard `csv` module from Python <http://docs.python.org/2/library/csv.html>. Compare the results with `./result/example.tsv`. Are the results identical? Congratulations to your first `pyOpenMS` tool!

Hint: If you struggle at any point, take a look at `OpenSwathFeatureXMLToTSV_solution.py`.

7.4.5 Bonus task

Task



Implement all other 184 TOPP tools using `pyOpenMS`.

8 Quality control

8.1 Introduction

In this chapter, we will build on an existing workflow with OpenMS / KNIME to add some quality control (QC). We will utilize the qcML tools in OpenMS to create a file with which we can collect different measures of quality to the mass spectrometry runs themselves and the applied analysis. The file also serves the means of visually reporting on the collected quality measures and later storage along the other analysis result files. We will, step-by-step, extend the label-free quantitation workflow from section 3 with QC functions and thereby enrich each time the report given by the qcML file. But first, to make sure you get the most of this tutorial section, a little primer on how we handle QC on the technical level.

QC metrics and qcML

To assert the quality of a measurement or analysis we use quality metrics. Metrics are describing a certain aspect of the measurement or analysis and can be anything from a single value, over a range of values to an image plot or other summary. Thus, qcML metric representation is divided into QC parameters (QP) and QC attachments (QA) to be able to represent all sorts of metrics on a technical level.

A QP may (or may not) have a value which would equal a metric describable with a single value. If the metric is more complex and needs more than just a single value, the QP does not require the single value but rather depends on an attachment of values (QA) for full meaning. Such a QA holds the plot or the range of values in a table-like form. Like this, we can describe any metric by a QP and an optional QA.

To assure a consensual meaning of the quality parameters and attachments, we created a controlled vocabulary (CV). Each entry in the CV describes a metric or part/extension thereof. We embed each parameter or attachment with one of these and by doing so, connect a meaning to the QP/QA. Like this, we later know exactly what we collected and the programs can find and connect the right dots for rendering the report or calculating new metrics automatically. You can find the constantly growing controlled vocabulary here:

<https://github.com/qcML/qcML-development/blob/master/cv/qc-cv.obo>.

Finally, in a qcml file, we split the metrics on a per mass-spectrometry-run base or a set of mass-spectrometry-runs respectively. Each run or set will contain its QP/QA we calculate for it, describing their quality.

8.2 Building a qcML file per run

As a start, we will build a basic qcML file for each mzML file in the label-free analysis. We are already creating the two necessary analysis files to build a basic qcML file upon each mzML file, a feature file and an identification file. We use the QCCalculator node from [Community Nodes](#) > [OpenMS](#) > [Utilities](#) where also all other QC* nodes will be found. The QCCalculator will create a very basic qcML file in which it will store collected and calculated quality data.

- Copy your label-free quantitation workflow into a new lfq-qc workflow and open it.
- Place the QCCalculator node after the IDMapper node. Being inside the ZipLoop, it will execute for each of the three mzML files the Input node.
- Connect the first QCCalculator port to the first ZipLoopStart outlet port, which will carry the individual mzML files.
- Connect the last's ID outlet port (IDFilter or the ID metanode) to the second QCCalculator port for the identification file.
- Finally, connect the IDMapper outlet to the third QCCalculator port for the feature file.

The created qcML files will not have much to show for, basic as they are. So we will extend them with some basic plots.

- First, we will add an 2D overview image of the given mass spectrometry run as you may know it from TOPPView. Add the ImageCreator node from [Community Nodes](#) > [OpenMS](#) > [Utilities](#). Change the *width* and *height* parameters to 640x640 as we don't want it to be too big. Connect it to the first ZipLoopStart outlet port, so it will create an image file of the mzML's contained run.

- Now we have to embed this file into the qcML file, and attach it to the right Quality-Parameter. For this, place a QCEmbedder node behind the ImageCreator and connect that to its third inlet port. Connect its first inlet port to the outlet of the QC Calculator node to pass on the qcML file. Now change the parameter *cv_acc* to *QC:0000055* which designates the attached image to be of type *QC:0000055* - MS experiment heatmap. Finally, change the parameter *qp_att_acc* to *QC:0000004*, to attach the image to the QualityParameter *QC:0000004* - MS acquisition result details.
- For a reference of which CVs are already defined for qcML, have a look at <https://github.com/qcML/qcML-development/blob/master/cv/qc-cv.obo>.

There are two other basic plots which we almost always might want to look at before judging the quality of a mass spectrometry run and its identifications: the *total ion current* (TIC) and the *PSM mass error* (Mass accuracy), which we have available as pre-packaged QC metanodes.

Task



Import the workflow from Workflows ▶ Quality Control ▶ QC Metanodes.zip in KNIME: File > Import KNIME Workflow...

- Copy the Mass accuracy metanode into the workflow behind the QCEmbedder node and connect it. The qcML will be passed on and the Mass accuracy plots added. The information needed was already collected by the QC Calculator.
- Do the same with the TIC metanode so that your qcML file will get passed on and enriched on each step.

R Dependencies: This section requires that the R packages *ggplot2* and *scales* are both installed. This is the same procedure as in section 5.5.5. In case that you use an R installation where one or both of them are not yet installed, open the R Snippet nodes inside the metanodes you just used (double-click). Edit the script in the *R Script* text editor from:

```
#install.packages("ggplot2")
#install.packages("scales")
```

to

```
install.packages("ggplot2")
install.packages("scales")
```

Press to execute the script.

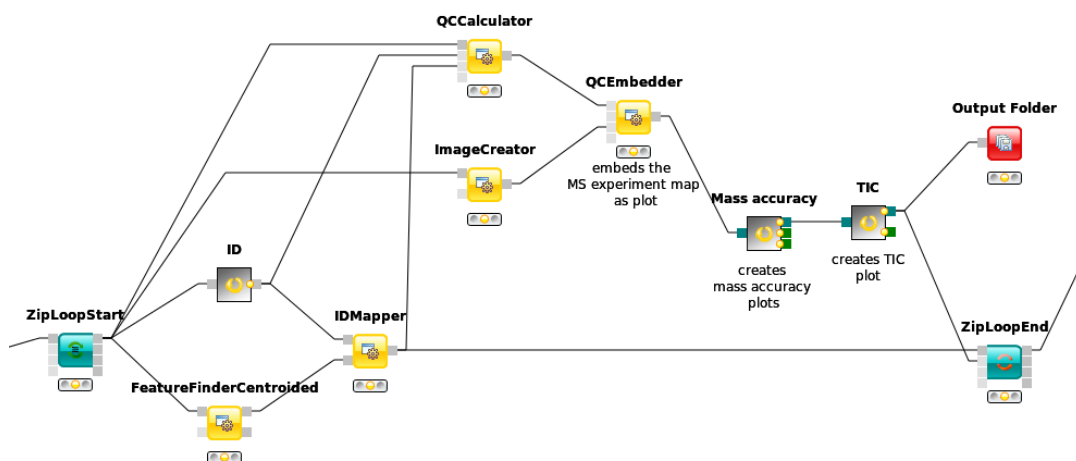


Figure 18: Basic QC setup within a LFQ workflow

Note: To have a peek into what our qcML now looks like for one of the ZipLoop iterations, we can add an `Output Folder` node from `Community Nodes` `GenericKnomeNodes` `IO` and set its destination parameter to somewhere we want to find our intermediate qcML files in, for example `tmp` `qc_lfq`. If we now connect the last metanode with the `Output Folder` and restart the workflow, we can start inspecting the qcML files.

Task






Find your first created qcML file and open it with the browser (not IE), and the contained QC parameters will be rendered for you.

8.3 Adding brand new QC metrics

We can also add brand new QC metrics to our qcML files. Remember the Histogram you added inside the ZipLoop during the label-free quantitation section? Let's imagine for a moment this was a brand new and utterly important metric and plot for the assessment of your analyses quality. There is an easy way to integrate such new metrics into your qcMLs. Though the Histogram node cannot pass its plot to an *image*, we can do so with a R View (table).

- Add an R View (table) next to the IDTextReader node and connect them.
- Edit the R View (table) by adding the *R Script* according to this:

```
#install.packages("ggplot2")
library("ggplot2")
ggplot(knime.in, aes(x=peptide_charge)) +
  geom_histogram(binwidth=1, origin =-0.5) +
  scale_x_discrete() +
  ggtitle("Identified peptides charge histogram") +
  ylab("Count")
```

- This will create a plot like the Histogram node on *peptide_charge* **and** pass it on as an *image*.
- Now add and connect a Image2FilePort node from    to the R View (table).
- We can now use a QCEmbedder node like before to add our new metric plot into the qcML.

- After looking for an appropriate target in <https://github.com/qcML/qcML-development/blob/master/cv/qc-cv.obo>, we found that we can attach our plot to the *MS identification result details* by setting the parameter *qp_att_acc* to *QC:0000025*, as we are plotting the charge histogram of our *identified* peptides.
- To have the plot later displayed properly, we assign it the parameter *cv_acc* of *QC:0000051*, a *generic plot*. Also we made sure in the *R Script*, that our plot carries a caption so that we know which is which, if we had more than one new plot.
- Now we redirect the QCEmbedders output to the Output Folder from before and can have a look at how our qcML is coming along after restarting the workflow.

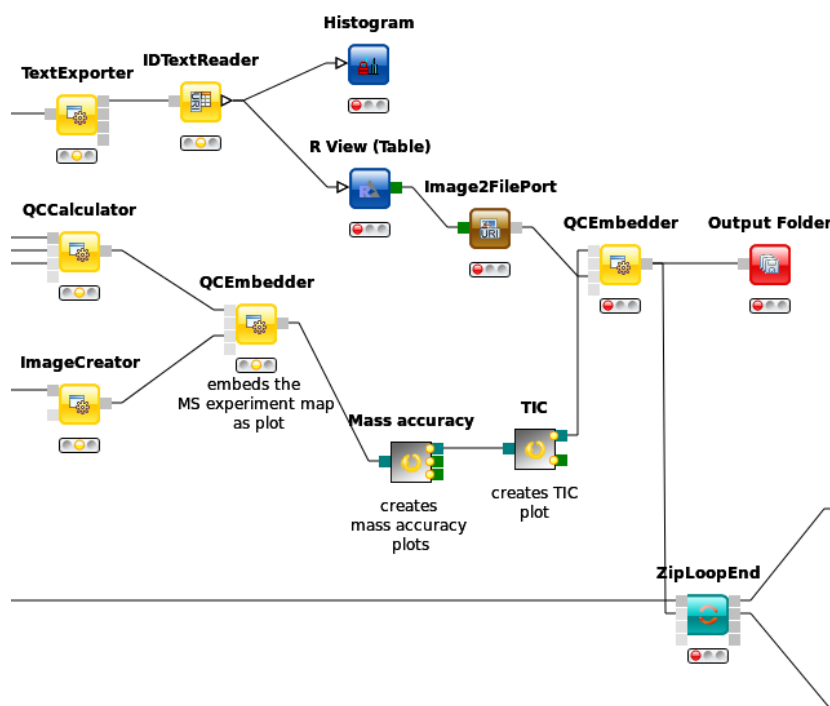


Figure 19: QC with new metric

8.4 Set QC metrics

Besides monitoring the quality of each individual mass spectrometry run analysis, another capability of QC with OpenMS and qcML is to monitor the complete set. The easiest control is to compare mass spectrometry runs which should be similar, e.g. technical replicates, to spot any aberrations in the set.

For this, we will first collect all created qcML files, merge them together and use the qcML onboard *set* QC properties to detect any outliers.

- connect the QCEmbedders output from last section to the ZipLoopEnds second input port.
- The corresponding output port will collect all qcML files from each ZipLoop iteration and pass them on as a list of files.
- Now we add a QCMerger node after the ZipLoopEnd and feed it that list of qcML files. In addition, we set its parameter *setname* to give our newly created set a name - say *spikein_replicates*.
- To inspect all the QCs next to each other in that created qcML file, we have to add a new Output Folder to which we can connect the QCMerger output.

When inspecting the set-qcML file in a **browser**, we will be presented another overview. After the set content listing, the basic QC parameters (like number of identifications) are each displayed in a graph. Each set member (or run) has its own section on the x-axis and each run is connected with that graph via a link in the mouseover on one of the QC parameter values.

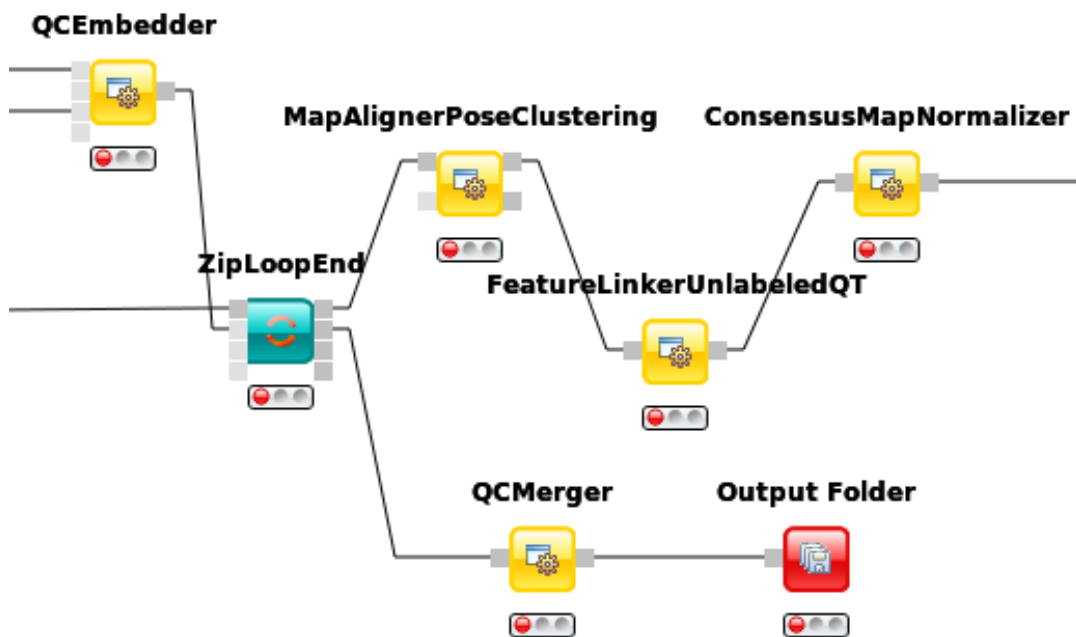


Figure 20: QC set creation from ZipLoop

Task



For ideas on new QC metrics and parameters -as you add them in your qcML files as generic parameters, feel free to contact us, so we can include them in the CV.

9 Troubleshooting guide

This section will show you where you can turn to when you encounter any problems with this tutorial or with our nodes in general. Please see the FAQ first. If your problem is not listed or the proposed solution does not work, feel free to leave us a message at the means of support that you see most fit.

9.1 FAQ

9.1.1 General

Q: Can I add my own modifications to the Unimod.xml?

A: Unfortunately not very easy. This is an open issue.

Q: I have problem XYZ but it also occurs with other nodes or generally in the KNIME environment/GUI, what should I do?

A: This sounds like a general KNIME bug and we advise to search help directly at the KNIME developers. They also provide a FAQ and a forum.

9.1.2 Platform-specific problems

Linux

Q: Whenever I try to execute an OpenMS node I get an error similar to these:

```
/usr/lib/x86_64-linux-gnu/libgomp.so.1: version `GOMP_4.0' not found  
/usr/lib/x86_64-linux-gnu/libstdc++.so.6: version `GLIBCXX_3.4.20' not found
```

A: We currently build the binaries shipped in the OpenMS KNIME plugin with gcc 4.8. We will try to extend our support for older compilers. Until then you either need to upgrade your gcc compiler or at least the library that the tool complained about or you need to build the binaries yourself (see OpenMS documentation) and replace them in your KNIME binary folder

(`YOURKNIMEFOLDER/plugins/de.openms.platform.architecture.version/payload/bin`).

Q: Why is my configuration dialog closing right away when I double-click or try to configure it? Or why is my GUI responding so slow?

A: If you have any problems with the KNIME GUI or the opening of dialogues under Linux you might be affected by a GTK bug. See the KNIME forum (e.g. [here](#) or [here](#)) for a discussion and a possible solution.

Windows

Q: KNIME has problems getting the requirements for some of the OpenMS nodes on Windows, what can I do?

A: Get the prerequisites installer [here](#) or install NET3.5, NET4 and VCRedist10.0 yourself.

9.1.3 Nodes

Q: Why is my XTandemAdapter printing empty results?

A: From OpenMS 2.0.1 the XTandemAdapter requires a default parameter file. Give it the default configuration in

```
YOURKNIMEFOLDER/plugins/de.openms.platform.architecture.version/payload/share/  
CHEMISTRY/XTandem_default_input.xml
```

 as a third input file.

Q: Do MSGFPlusAdapter and LuciphorAdapter generally behave different/unexpected?

A: These are Java processes that are started underneath. For example they can not be killed during cancellation of the node. This should not affect its performance, however.

9.2 Sources of support

If your questions could not be answered by the FAQ, please feel free to turn to our developers via one of the following means:

- File an issue on GitHub
- Write to the Mailing List
- Open a thread on the KNIME Community Contributions forum for OpenMS

References

- [1] OpenMS, OpenMS home page [online]. 6
- [2] M. Sturm, A. Bertsch, C. Gröpl, A. Hildebrandt, R. Hussong, E. Lange, N. Pfeifer, O. Schulz-Trieglaff, A. Zerck, K. Reinert, and O. Kohlbacher, OpenMS - an open-source software framework for mass spectrometry., *BMC bioinformatics* **9**(1) (2008), doi:10.1186/1471-2105-9-163. 6, 51
- [3] O. Kohlbacher, K. Reinert, C. Gröpl, E. Lange, N. Pfeifer, O. Schulz-Trieglaff, and M. Sturm, TOPP--the OpenMS proteomics pipeline., *Bioinformatics* **23**(2) (Jan. 2007). 6, 51
- [4] M. R. Berthold, N. Cebron, F. Dill, T. R. Gabriel, T. Kötter, T. Meinel, P. Ohl, C. Sieb, K. Thiel, and B. Wiswedel, KNIME: The Konstanz Information Miner, in *Studies in Classification, Data Analysis, and Knowledge Organization (GfKL 2007)*, Springer, 2007. 6
- [5] M. Sturm and O. Kohlbacher, TOPPView: An Open-Source Viewer for Mass Spectrometry Data, *Journal of proteome research* **8**(7), 3760--3763 (July 2009), doi: 10.1021/pr900171m. 6
- [6] L. Y. Geer, S. P. Markey, J. A. Kowalak, L. Wagner, M. Xu, D. M. Maynard, X. Yang, W. Shi, and S. H. Bryant, Open mass spectrometry search algorithm, *Journal of Proteome Research* **3**(5), 958--964 (2004). 23
- [7] A. Chawade, M. Sandin, J. Teleman, J. Malmström, and F. Levander, Data Processing Has Major Impact on the Outcome of Quantitative Label-Free LC-MS Analysis, *Journal of Proteome Research* **14**(2), 676--687 (2015), PMID: 25407311, arXiv:http://dx.doi.org/10.1021/pr500665j, doi:10.1021/pr500665j. 23
- [8] D. S. Wishart, D. Tzur, C. Knox, et al., HMDB: the Human Metabolome Database, *Nucleic Acids Res* **35**(Database issue), D521--6 (Jan 2007), doi:10.1093/nar/gkl923. 42
- [9] D. S. Wishart, C. Knox, A. C. Guo, et al., HMDB: a knowledgebase for the human metabolome, *Nucleic Acids Res* **37**(Database issue), D603--10 (Jan 2009), doi:10.1093/nar/gkn810. 42

- [10] D. S. Wishart, T. Jewison, A. C. Guo, M. Wilson, C. Knox, et al., HMDB 3.0--The Human Metabolome Database in 2013, *Nucleic Acids Res* **41**(Database issue), D801--7 (Jan 2013), doi:10.1093/nar/gks1065. 42
- [11] J. Griss, A. R. Jones, T. Sachsenberg, M. Walzer, L. Gatto, J. Hartler, G. G. Thallinger, R. M. Salek, C. Steinbeck, N. Neuhauser, J. Cox, S. Neumann, J. Fan, F. Reisinger, Q.-W. Xu, N. Del Toro, Y. Perez-Riverol, F. Ghali, N. Bandeira, I. Xenarios, O. Kohlbacher, J. A. Vizcaino, and H. Hermjakob, The mzTab Data Exchange Format: communicating MS-based proteomics and metabolomics experimental results to a wider audience, *Mol Cell Proteomics* (Jun 2014), doi:10.1074/mcp.0113.036681. 43
- [12] H. L. Röst, G. Rosenberger, P. Navarro, L. Gillet, S. M. Miladinovic, O. T. Schubert, W. Wolski, B. C. Collins, J. Malmstrom, L. Malmström, and R. Aebersold, OpenSWATH enables automated, targeted analysis of data-independent acquisition MS data, *Nature Biotechnology* **32**(3), 219--223 (Mar. 2014). 51, 56
- [13] L. C. Gillet, P. Navarro, S. Tate, H. Röst, N. Selevsek, L. Reiter, R. Bonner, and R. Aebersold, Targeted Data Extraction of the MS/MS Spectra Generated by Data-independent Acquisition: A New Concept for Consistent and Accurate Proteome Analysis., *Molecular & Cellular Proteomics* **11**(6) (June 2012), doi:10.1074/mcp.0111.016717. 51
- [14] A. Bertsch, C. Gröpl, K. Reinert, and O. Kohlbacher, OpenMS and TOPP: open source software for LC-MS data analysis., *Methods in molecular biology (Clifton, N.J.)* **696**, 353--367 (2011), doi:10.1007/978-1-60761-987-1_23. 51
- [15] L. Reiter, O. Rinner, P. Picotti, R. Huttenhain, M. Beck, M.-Y. Brusniak, M. O. Hengartner, and R. Aebersold, mProphet: automated data processing and statistical validation for large-scale SRM experiments, *Nature Methods* **8**(5), 430--435 (May 2011), doi:10.1038/nmeth.1584. 51
- [16] E. W. Deutsch, M. Chambers, S. Neumann, F. Levander, P.-A. Binz, J. Shofstahl, D. S. Campbell, L. Mendoza, D. Ovelleiro, K. Helsens, L. Martens, R. Aebersold, R. L. Moritz, and M.-Y. Brusniak, TraML--A Standard Format for Exchange of Selected Reaction Monitoring Transition Lists, *Molecular & Cellular Proteomics* **11**(4) (Apr. 2012), doi:10.1074/mcp.R111.015040. 52

- [17] C. Escher, L. Reiter, B. MacLean, R. Ossola, F. Herzog, J. Chilton, M. J. MacCoss, and O. Rinner, Using iRT, a normalized retention time for more targeted measurement of peptides., *Proteomics* **12**(8), 1111--1121 (Apr. 2012), doi:10.1002/pmic.201100463.
52

TOPP and TOPPView Tutorial
Version: 2.0.0

Contents

1	Concepts	2
2	TOPPView main interface	3
2.1	Layers	3
2.2	Spectrum browser	4
2.3	Data filtering	4
2.4	Command line options	4
2.5	Looking for help?	4
2.6	Basic use	5
2.7	1D view	5
2.8	2D view	6
2.9	3D view	7
2.10	Display modes	9
2.11	View options	9
2.12	TOPP tools	10
2.13	Metadata	10
2.14	Statistics	11
3	Calling TOPP tools from TOPPView	14
4	Using TOPPAS to create and run TOPP pipelines	20
4.1	Introduction	21
4.2	Mouse and keyboard	23
4.3	Menus	24
4.4	Profile data processing	26
4.5	Identification of E. coli peptides	26
4.6	Quantitation of BSA runs	27
4.7	Merger and Collect nodes	29
5	Advanced Users: Tips & Tricks	29
6	Scripting with TOPP	31
6.1	File formats	31
6.2	Common arguments of the TOPP tools	31
6.3	TOPP INI files	32
6.4	General information about peak and feature maps	34
6.5	Problems with input files	34
6.6	Converting your files to mzML	34
6.7	Converting between DTA and mzML	34
6.8	Extracting part of the data from a file	35
6.9	Picking peaks with a PeakPicker	36
6.10	Finding the right parameters for the	37
6.11	Isotope-labeled quantitation	41
6.12	Label-free quantitation	41
6.13	Export of OpenMS XML formats	44
6.14	Import of feature data to OpenMS	44
6.15	Import of protein/peptide identification data to OpenMS	44

1 Concepts

TOPP, The OpenMS Proteomics Pipeline provides a set of computational tools which can be easily combined into analysis pipelines even by non-experts and then be used in proteomics workflows. These applications range from useful utilities (file format conversion, peak picking) over wrapper applications for known applications (e.g. Mascot) to completely new algorithmic techniques for data reduction and data analysis. TOPP is based on the OpenMS library and as more functionality is added to new OpenMS releases, TOPP will naturally contain new or updated tools.

In addition to offering a toolbox, TOPP contains TOPPView - the central graphical user interface - which allows the user to view, inspect and manipulate proteomics data. TOPPView reads standard formats and allows the user not only to view the data, but also to interactively call TOPP tools and display the results. As such it is a powerful tool for interactive data manipulation.

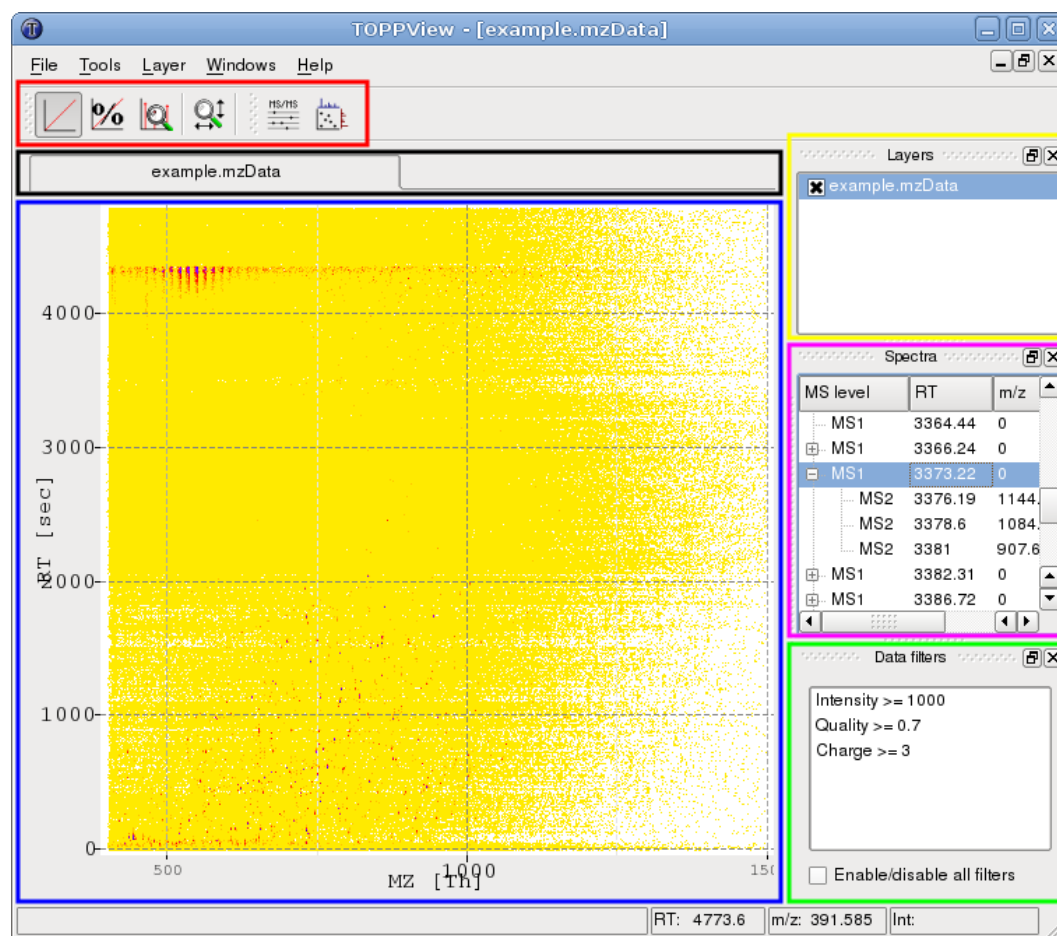
In this tutorial we will now in detail explain the capabilities of TOPPView and the TOPP tools using real data sets and standard analysis tasks.

2 TOPPView main interface

TOPPView is a viewer for MS and HPLC-MS data. It can be used to inspect files in mzML, mzData, mzXML and several other text-based file formats.

In each view, several datasets can be displayed using the layer concept. This allows visual comparison of several datasets as well as displaying input data and output data of an algorithm together.

TOPPView is intended for visual inspection of the data by experimentalists as well as for analysis software by developers.



The above example image shows a 2D view of TOPPView (blue rectangle) and the corresponding [Display modes and view options](#) (red rectangle). In the right dock area, you can see the [Layers](#) (yellow rectangle), the [Spectrum browser](#) (magenta rectangle) and the [Data filtering](#) tool (green rectangle). Switching between open windows can be done using the tab bar (black rectangle).

2.1 Layers

Each view of TOPPView supports several datasets, called layers. In the *layer manager* (dock window in the upper right corner), layers can be hidden and shown using the check box in front of each layer name.

By clicking on a layer, this layer is selected, which is indicated by a blue background. The selected layer can be manipulated using the *Tools* menu.

Layers can be copied by dragging them to the tab bar. If the layer is dropped on a tab, it is added to the

corresponding window. If the layer is dropped on the tab bar but not on a tab, a new window with that layer is added.

2.2 Spectrum browser

The spectra contained in a peak map can be browsed in the *spectrum browser*. It displays a tree view of all spectra in the current layer, where a spectrum with MS level n is a child of the spectrum with MS level $n - 1$ that contains its corresponding precursor peak. For each spectrum in the list, the retention time and (for spectra with MS level > 1) the m/z value of the precursor peak are also shown.

If a 2D or 3D window is active, double-clicking a spectrum will open it in a new 1D window. If the active window is a 1D view, the spectra can be browsed and the currently selected spectrum will be shown.

2.3 Data filtering

TOPPView allows filtering of the displayed peak data and feature data. Peak data can be filtered according to intensity and meta data. Meta data is arbitrary data the peak is annotated with. Feature data can be filtered according to intensity, charge, quality and meta data.

Data filters are managed by a dock window. Filters can be added, removed and edited through the context menu (right button mouse click) of the data filters window. For convenience, filters can also be edited by double-clicking them.

2.4 Command line options

Several files can be opened in one layer from the command line by putting a '+' character between the file names. The following command opens three files in three layers of the same window:

```
TOPPView file1.mzML + file2.mzML + file3.mzML
```

Without the '+' the files would be opened in three different windows.

The color gradient used to display a file can be changed by adding one of several '@' commands. The following command opens the file with a white-to-black gradient:

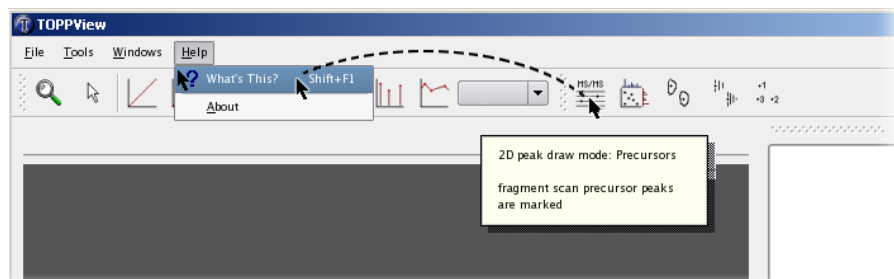
```
TOPPView file1.mzML @bw
```

For more information on command line parameters call:

```
TOPPView --help
```

2.5 Looking for help?

You can display a short help text for each button and dock window of TOPPView by clicking on it in *What's this?* mode. *What's this?* mode can be entered using the *Help* menu.



TOPPView offers three types of views -- a 1D view for spectra, a 2D view for peak maps and feature maps, and a 3D view for peak maps. All three views can be freely configured to suit the individual needs of the user.

2.6 Basic use

All three views share a similar interface. Three action modes are supported -- one for translation, one for zooming and one for measuring:

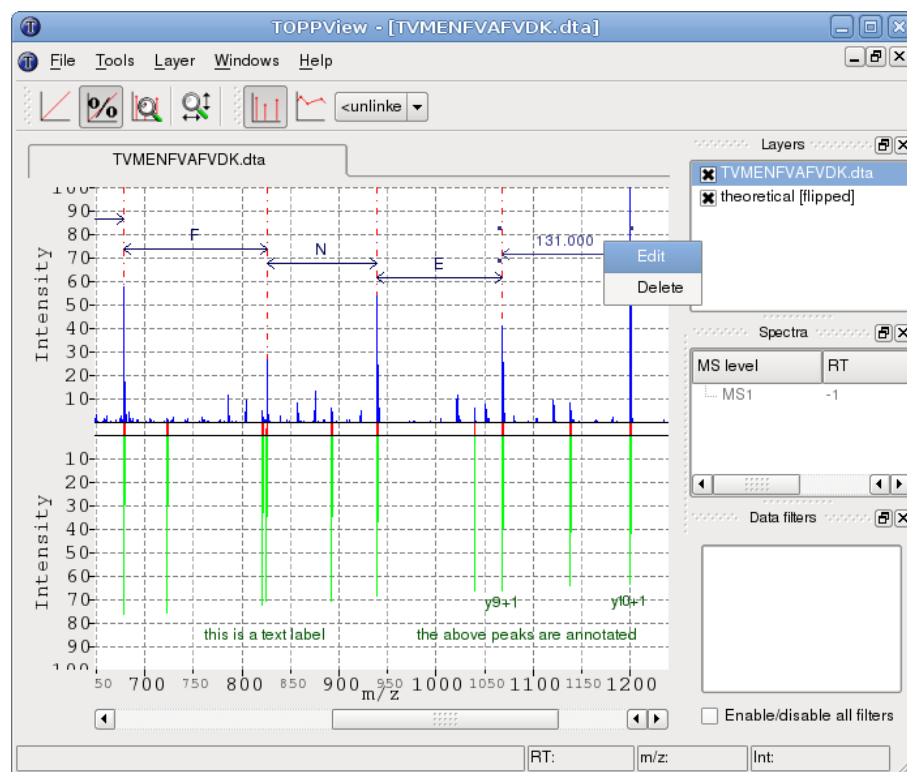
- Translate mode
 - It is activated by default
 - Move the mouse while holding the mouse button down to translate the current view
 - Arrow keys can be used to translate the view without entering translate mode (in 1D-View you can additionally use Shift-key to jump to the next peak)
- Zoom mode
 - All previous zoom levels are stored in a zoom history. The zoom history can be traversed using CTRL+/CTRL- or the mouse wheel (scroll up and down)
 - Zooming into the data:
 - * either mark an area in the current view with your mouse, while holding the left mouse button plus the CTRL key to zoom to this area.
 - * or use your mouse wheel to traverse the zoom history
 - * if you have reached the end of the history and keep on pressing CTRL+ or scroll up, the current area will be enlarged by a factor of 1.25
 - Pressing the *Backspace* key resets the zoom and zoom history
- Measure mode
 - It is activated using the *SHIFT* key
 - Press the left mouse button down while a peak is selected and drag the mouse to another peak to measure the distance between peaks
 - This mode is implemented in the 1D and 2D mode only

2.7 1D view

The 1D view is used to display raw spectra or peak spectra. Raw data is displayed using a continuous line. Peak data is displayed using one stick per peak. The color used for drawing the lines can be set for each layer individually. The 1D view offers a mirror mode, where the window is vertically divided in halves and individual layers can be displayed either above or below the "mirror" axis in order to facilitate quick visual comparison of spectra. When a mirror view is active, it is possible to perform a spectrum alignment of a spectrum in the upper and one in the lower half, respectively. Moreover, spectra can be annotated manually. Currently, distance annotations between peaks, peak annotations and simple text labels are provided.

The following example image shows a 1D view in mirror mode. A theoretical spectrum (lower half) has been generated using the theoretical spectrum generator (*Tools > Generate theoretical spectrum*). - The mirror mode has been activated by right-clicking the layer containing the theoretical spectrum and selecting *Flip downward* from the layer context menu. A spectrum alignment between the two spectra has been performed (*Tools > Align spectra*). It is visualized by the red lines connecting aligned peaks and can be reset through the context menu. Moreover, in the example, several distances between abundant peaks have been measured and subsequently replaced by their corresponding amino acid residue code. This is

done by right-clicking a distance annotation and selecting *Edit* from the context menu. Additionally, peak annotations and text labels have been added by right-clicking peaks and selecting *Add peak annotation* or by right-clicking anywhere and selecting *Add label*, respectively. Multiple annotations can be selected by holding down the CTRL key while clicking them. They can be moved around by dragging the mouse and deleted by pressing DEL.



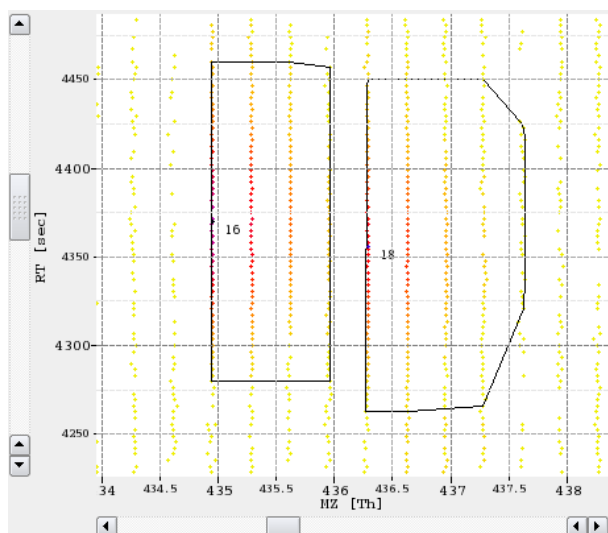
Through the **context menu**: of the 1D view you can:

- View/edit meta data
- Save the current layer data
- Change display settings
- Add peak annotations or arbitrary text labels
- Reset a performed alignment

2.8 2D view

The 2D view is used to display peak maps and feature maps in a top-down view with color-coded intensities. Peaks and feature centroids are displayed as dots. For features, also the overall convex hull and the convex hulls of individual mass traces can be displayed. The color gradient used to encode for peak and feature intensities can be set for each layer individually.

The following example image shows a small section of a peak map and the detected features in a second layer.



In addition to the normal top-down view, the 2D view can display the projections of the data to the m/z and RT axis. This feature is mainly used to assess the quality of a feature without opening the data region in 3D view.

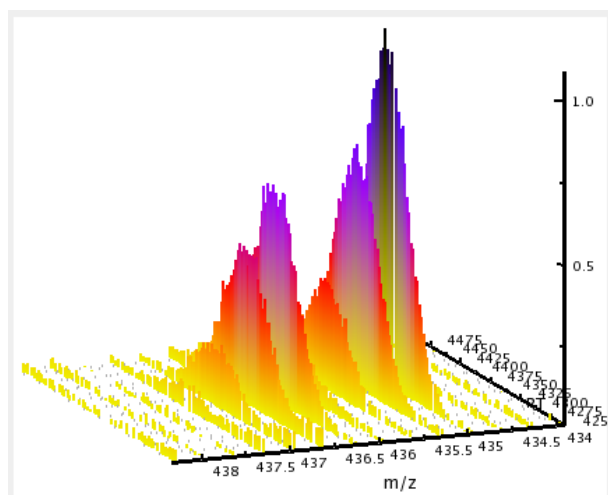
Through the **context menu**: of the 2D view you can:

- View/edit meta data
- View survey/fragment scans in 1D view
- View survey/fragment scans meta data
- View the currently selected area in 3D view
- Save the current layer data
- Change display settings

2.9 3D view

The 3D view can display peak maps only. Its primary use is the closer inspection of a small region of the map, e.g. a single feature. In the 3D view slight intensity differences are easier to recognize than in the 2D view. The color gradient used to encode peak intensities, the width of the lines and the coloring mode of the peaks can be set for each layer individually.

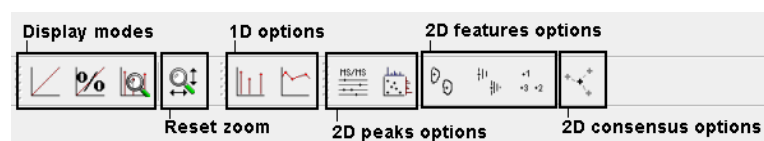
The following example image shows a small region of a peak map:



Through the **context menu**: of the 3D view you can:

- View/edit meta data
- Save the current layer data
- Change display settings

All of the views support several display modes and view options. Display modes determine how intensities are displayed. View options configure the view.



2.10 Display modes

Intensity display modes determine the way peak intensities are displayed.

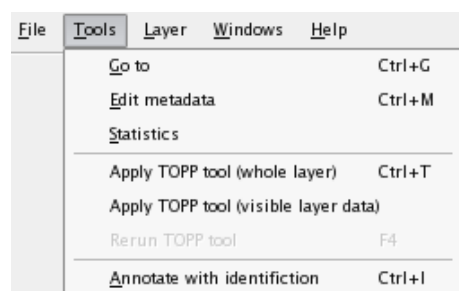
- **Linear:**
Normal display mode.
- **Percentage:**
In this display mode the intensities of each dataset are normalized with the maximum intensity of the dataset. This is especially useful in order to visualize several datasets that have large intensity differences. If only one dataset is opened it corresponds to the normal mode.
- **Snap to maximum intensity:**
In this mode the maximum currently displayed intensity is treated as if it was the maximum overall intensity.

2.11 View options

View options configure the view of the current layer.

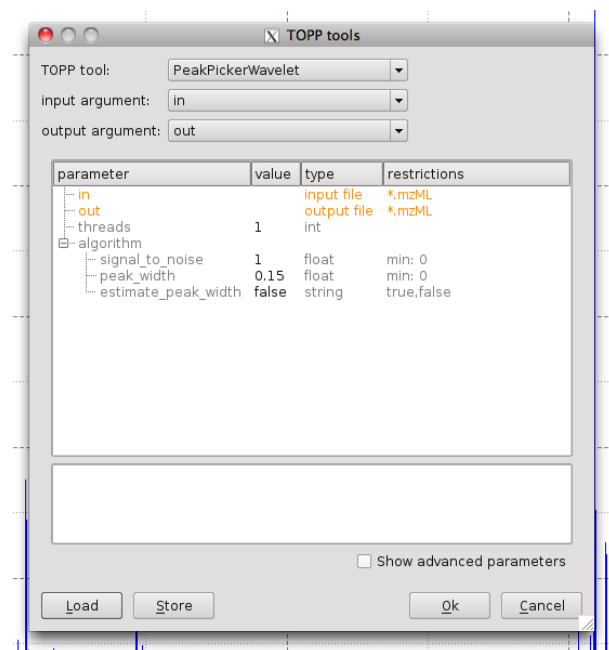
- **1D:**
Switching between raw data and peak mode.
- **2D (peaks):**
MS/MS precursor peaks can be highlighted.
Projections to m/z and RT axis can be shown.
- **2D (features):**
Overall convex hull of the features can be displayed.
Convex hulls of mass traces can be displayed (if available).
A feature identifier, consisting of the feature index and an optional label can be displayed.
- **2D (consensus):**
The elements of a consensus feature can be displayed.
- **3D:**
Currently there are no options for 3D view.

TOPPView also offers limited data analysis capabilities for single layers, which will be illustrated in the following sections. The functionality presented here can be found in the *Tools* menu:



2.12 TOPP tools

Single TOPP tools can be applied to the data of the currently selected layer or to the visible data of the current layer. The following example image shows the TOPP tool dialog:

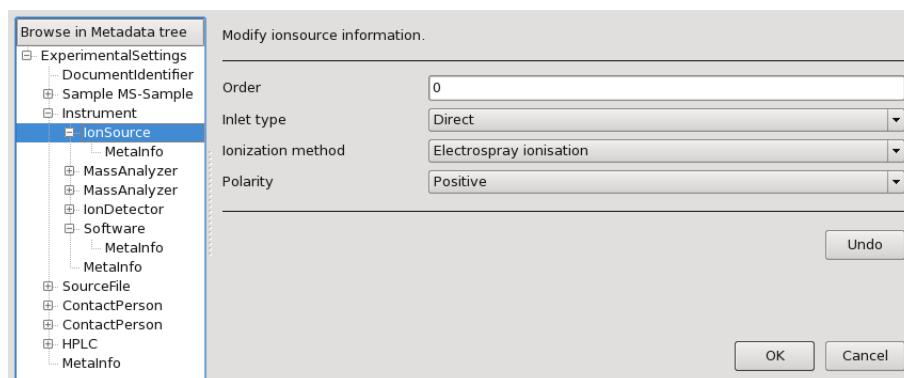


To apply a TOPP tool, do the following:

- Select a TOPP tool and if necessary a type.
- Specify the command line option of the tool, that takes the input file name.
- Specify the command line option of the tool, that takes the output file name.
- Set the algorithm parameters manually or load them from an INI file.

2.13 Metadata

One can access the meta data the layer is annotated with. This data comprises e.g. contact person, instrument description and sample description.

**Note**

Identification data, e.g. from a Mascot run, can be annotated to the spectra or features, too. After annotation, this data is listed in the meta data as well.

2.14 Statistics

Statistics about peak/feature intensities and peak meta information can be displayed. For intensities, it is possible to display an additional histogram view.

	Count	Min	Max	Average	Distribution
Intensity	-	10.00	7284.19	37.27	Show
area	1041134	0.40	1210.53	7.24	
charge	1041134	0.00	0.00	0.00	
fwhm	1041134	0.10	1.92	0.19	
leftWidth	1041134	0.48	84.78	10.32	
peakShape	1041134	1.00	1.00	1.00	
rValue	1041134	0.51	1.00	0.97	
rightWidth	1041134	0.70	84.89	10.31	
signalToNoise	1041134	2.00	1147.68	10.27	

TOPPView also offers editing functionality for feature layers.

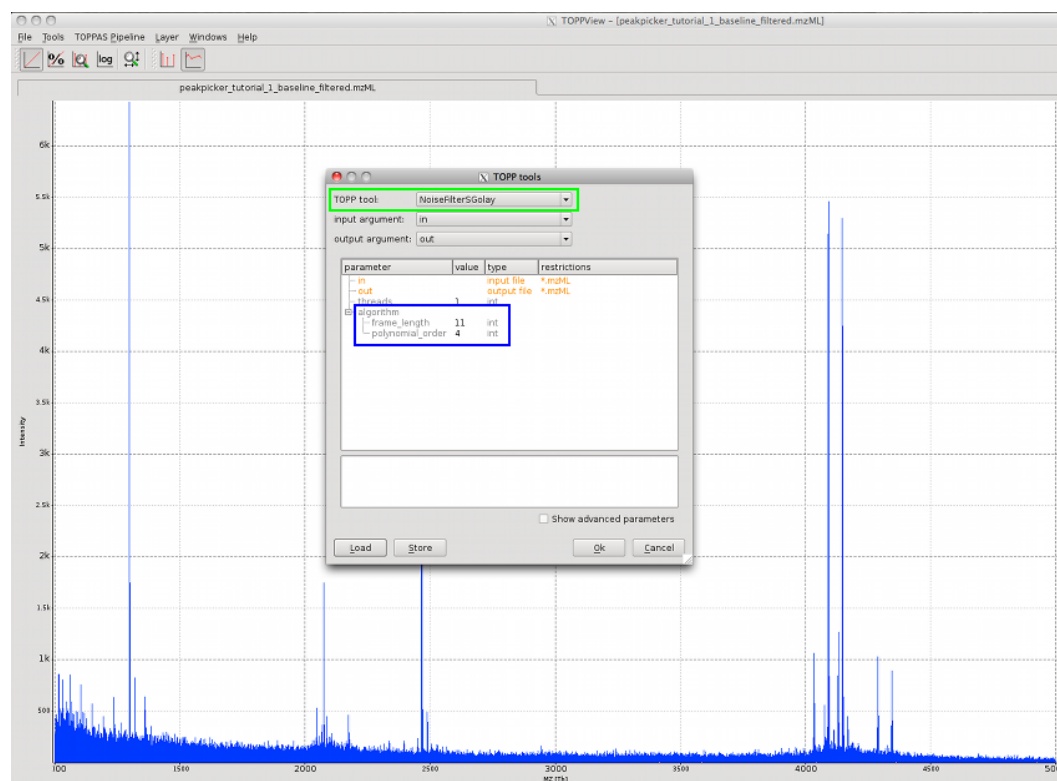
After enabling the feature editing mode in the context menu of the feature layer, the following actions can be performed:

- Features can be dragged with the mouse in order to change the m/z and RT position.
- The position, intensity and charge of a feature can be edited by double-clicking a feature.
- Features can be created by double-clicking the layer background.
- Features can be removed by selecting them and pressing the DEL key.

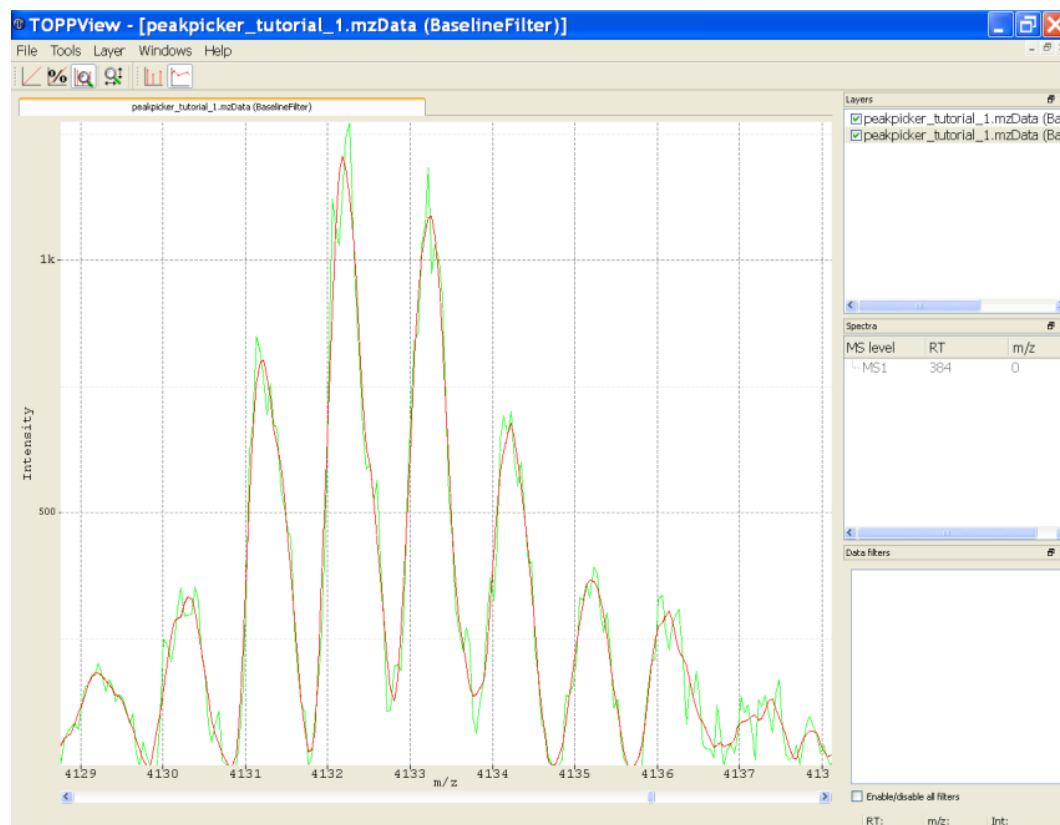
File handling	
CTRL+O	Open file
CTRL+W	Close current window
CTRL+S	Save current layer
CTRL+SHIFT+S	Save visible data of current layer
Navigation in the data	
CTRL	Activate zoom mode
SHIFT	Activate measurement mode
Arrow keys	Translate currently shown data (1D View: Shift + Left/Right moves to the next peak in sparse data)
CTRL+'+',CTRL+'-'	Move up and down in zoom history
mouse wheel	Move up and down in zoom history
CTRL+G	Goto dialog
Backspace	Reset zoom
PageUp	Select previous layer
PageDown	Select next layer
Visualization options	
CTRL+R	Show/hide grid lines
CTRL+L	Show/hide axis legends
N	Intensity mode: Normal
P	Intensity mode: Percentage
S	Intensity mode: Snap-to-maximum
I	1D draw mode: peaks
R	1D draw mode: raw data
CTRL+ALT+Home	2D draw mode: increase minimum canvas coverage threshold (for raw peak scaling). 'Home' on MacOSX keyboards is also 'Fn+ArrowLeft'
CTRL+ALT+End	2D draw mode: decrease minimum canvas coverage threshold (for raw peak scaling). 'End' on MacOSX keyboards is also 'Fn+ArrowRight'
CTRL+ALT+'+'	2D draw mode: increase maximum point size (for raw peak scaling)
CTRL+ALT+'-'	2D draw mode: decrease maximum point size (for raw peak scaling)
Annotations in 1D view	
CTRL+A	Select all annotations of the current layer
DEL	Delete all currently selected annotations
Advanced	
CTRL+T	Apply TOPP tool to the current layer
CTRL+SHIFT+T	Apply TOPP tool to the visible data of the current layer
F4	Rerun TOPP tool
CTRL+M	Show layer meta information
CTRL+I	Annotate with identification results
1	Show precursor peaks (2D peak layer)
2	Show projections (2D peak layer)
5	Show overall convex hull (2D feature layer)
6	Show all convex hulls (2D feature layer)
7	Show numbers and labels (2D feature layer)
9	Show consensus elements (2D consensus layer)
Help	
F1	Show TOPPView online tutorial
SHIFT+F1	Activate <i>What's this?</i> mode

3 Calling TOPP tools from TOPPView

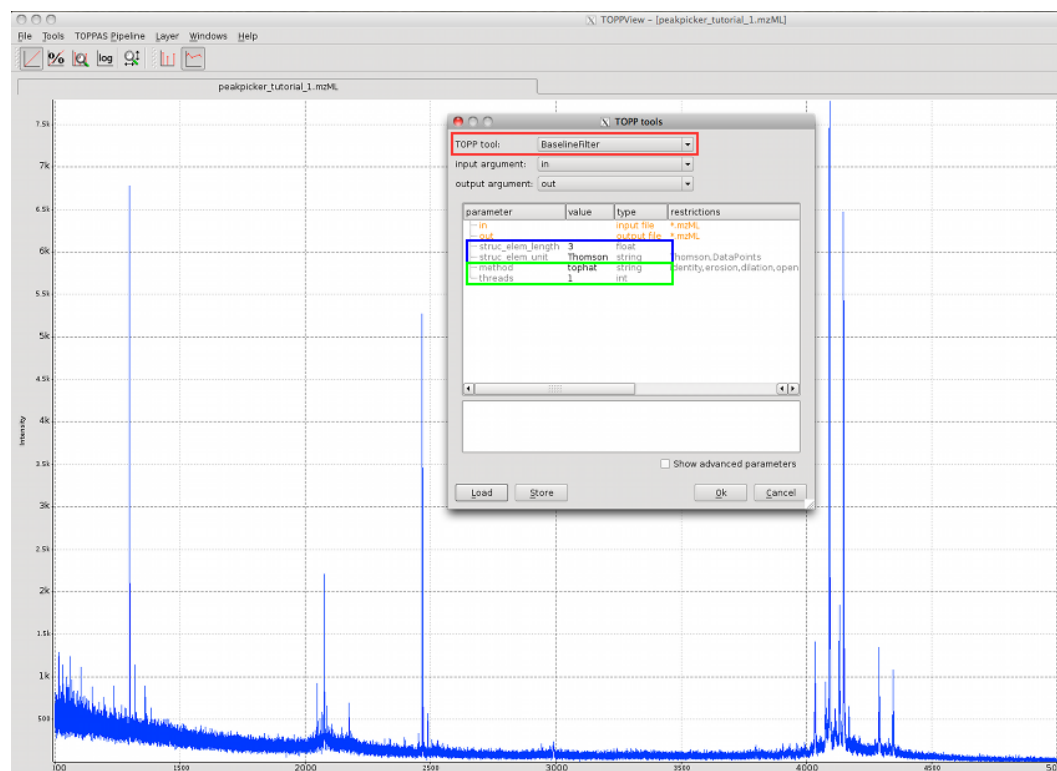
To smooth your raw data call one of the available NoiseFilters via the Tools-menu, (*Tools > Apply - TOPP tool*), then select NoiseFilterSGolay or NoiseFilterGaussian as TOPPtool (green rectangle). The parameters for the filter type can be adapted (blue rectangle). For the Savitzky-Golay filter you can set the frame length and the order of the polynomial that is fitted. For the Gaussian filter the gaussian width and the ppm tolerance for a flexible gaussian width depending on the m/z value can be adapted. Press Ok to run the selected NoiseFilter.



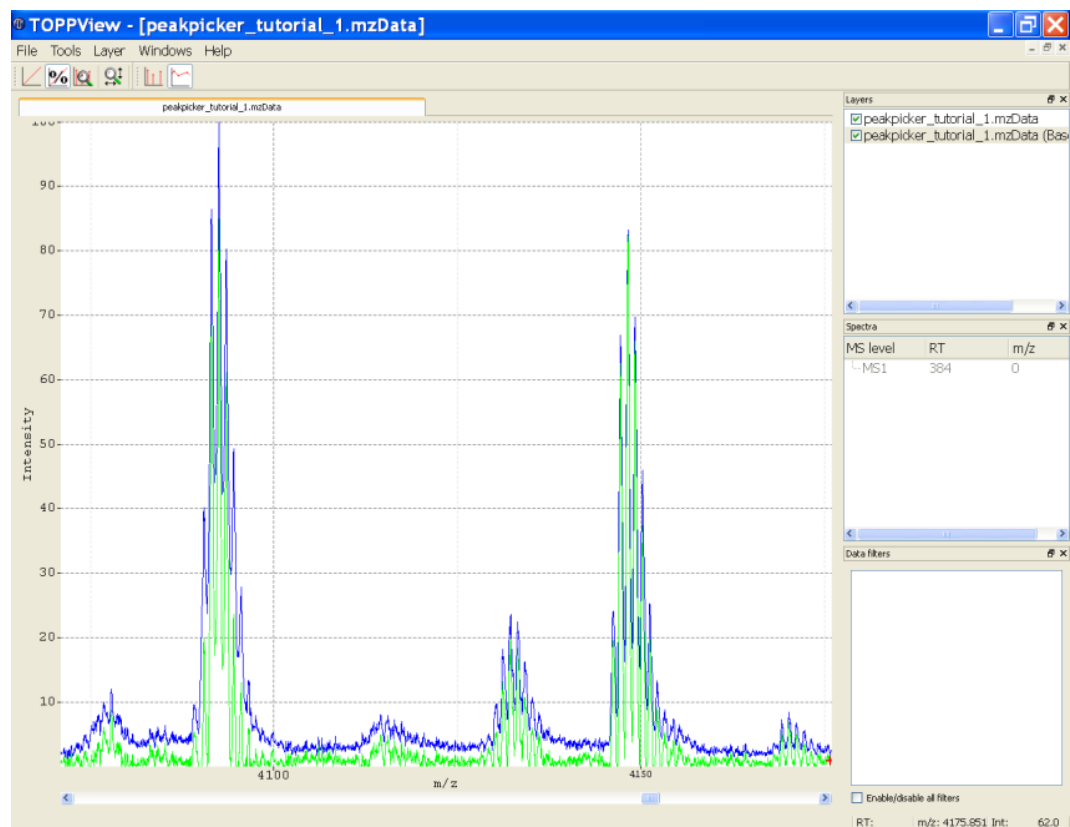
The following image shows a part of the spectrum after smoothing as red line with the un-smoothed data in green.



First we load the spectrum to be analyzed in TOPPView. If you want to test the described tools, you can open the tutorial data via the File-menu (*File > Open example file*, then select *peakpicker_tutorial_1.mzML*). The BaselineFilter can be called via the Tools-menu (*Tools > Apply TOPP tool*), then select BaselineFilter as TOPPtool (red rectangle). One can choose between different types of filters (green rectangle), the one mainly used is TopHat. The other important parameter is the length of the structuring element (blue rectangle). The default value is 3 Thomsson. Press OK to start the baseline subtraction.



The following image shows a part of the spectrum after baseline filtering as green line, the original raw data is shown by the blue line.



If you have low resolution data you may consider to smooth your data first ([Smoothing raw data](#)) and subtract the baseline ([Subtracting a baseline from a spectrum](#)) before peak picking.

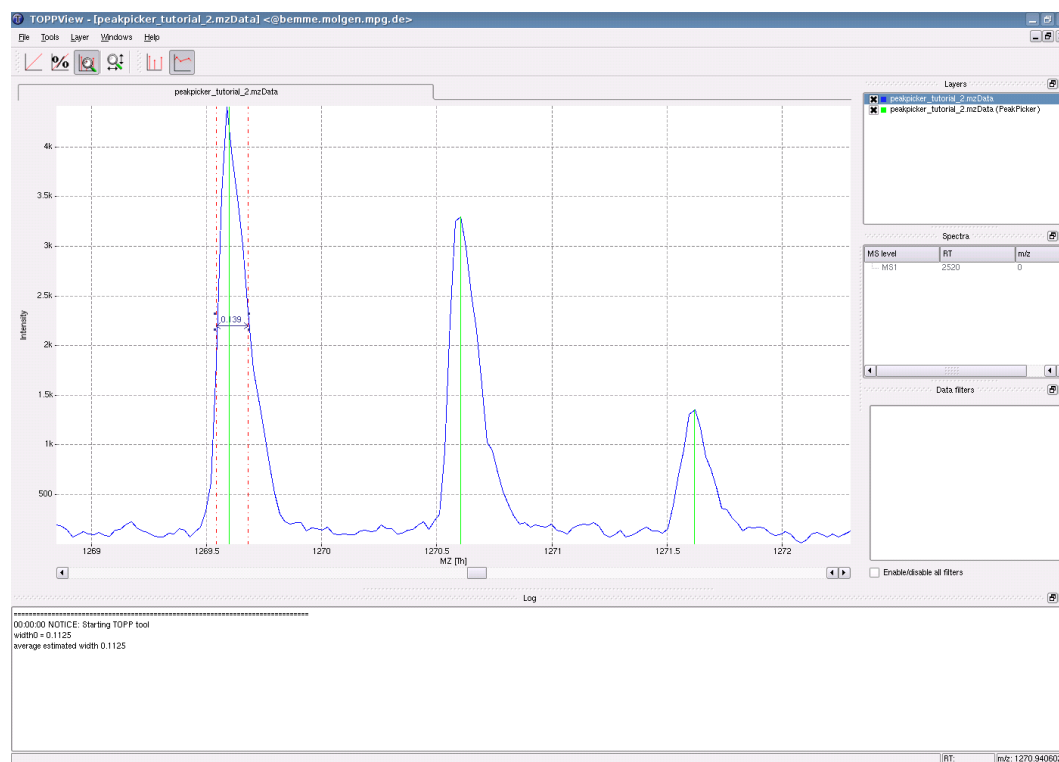
We have two types of PeakPickers, the PeakPickerWavelet and one especially suited for high resolution data (PeakPickerHiRes). In this tutorial we consider the PeakPickerWavelet. We use the file *peakpicker_tutorial_2.mzML* from the examples data (*File > Open example data*).

The main parameters are the peak width and the minimal signal to noise ratio for a peak to be picked. If you don't know the approximate fwhm of your peaks you can use the estimation included in the PeakPickerWavelet. Here you need to set the flag `estimate_peak_width` to true. After applying the PeakPickerWavelet you can see which peak width was estimated and used for peak picking in the log window.

If you want to estimate the peak width on your own, you may use the measuring tool ([Basic use](#)) to determine the fwhm of one or several representative peaks.

If the peak picker delivers only a few peaks even though the the `peak_width` and `signal_to_noise` parameters are set to good values, you may consider changing the advanced parameter `fwhm_lower_bound_factor` to a lower value. All peaks with a lower fwhm than `fwhm_lower_bound_factor * peak_width` are discarded.

The following image shows a part of the spectrum with the picked peaks shown in green, the estimated peak width in the log window and the measured spectrum width.



To quantify peptide features, TOPP offers the **FeatureFinder** tools. In this section the **FeatureFinder-Centroided** is used, which works on centroided data only. There are other FeatureFinders available that also work on profile data.

For this example the file *LCMS-centroided.mzML* from the examples data is used (*File > Open example data*). In order to adapt the algorithm to our data, some parameters have to be set.

Intensity

The algorithm estimates the significance of peak intensities in a local environment. Therefore, the HPLC--MS map is divided into n times n regions. Set the *intensity:bins* parameter to 10 if you are working on a whole map. If you are working on a small region only, you should set it to 1.

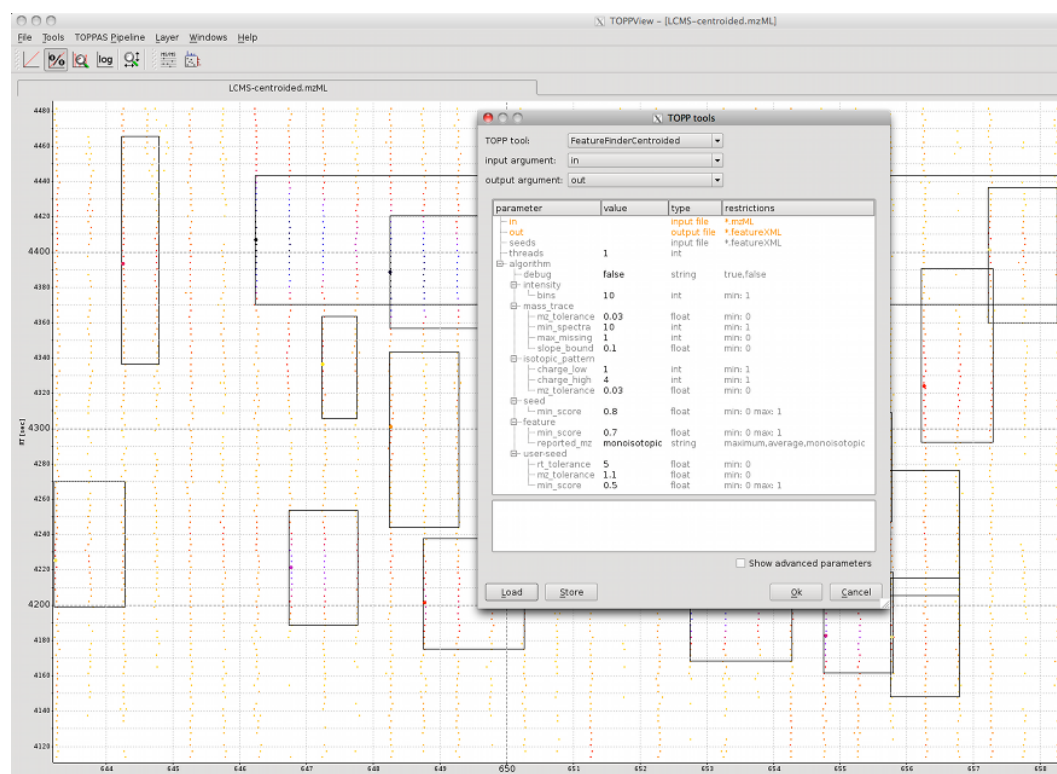
Mass trace

For the mass traces, you have to define the number of adjacent spectra in which a mass has to occur (*mass_trace:min_spectra*). In order to compensate for peak picking errors, missing peaks can be allowed (*mass_trace:max_missing*) and a tolerated mass deviation must be set (*mass_trace:mz_tolerance*).

Isotope pattern

The expected isotopic intensity pattern is estimated from an average amino acid composition. The algorithm searches all charge states in a defined range (*isotopic_pattern:charge_min* to *isotopic_pattern:charge_max*). Just as for mass traces, a tolerated mass deviation between isotopic peaks has to be set (*isotopic_pattern:mz_tolerance*).

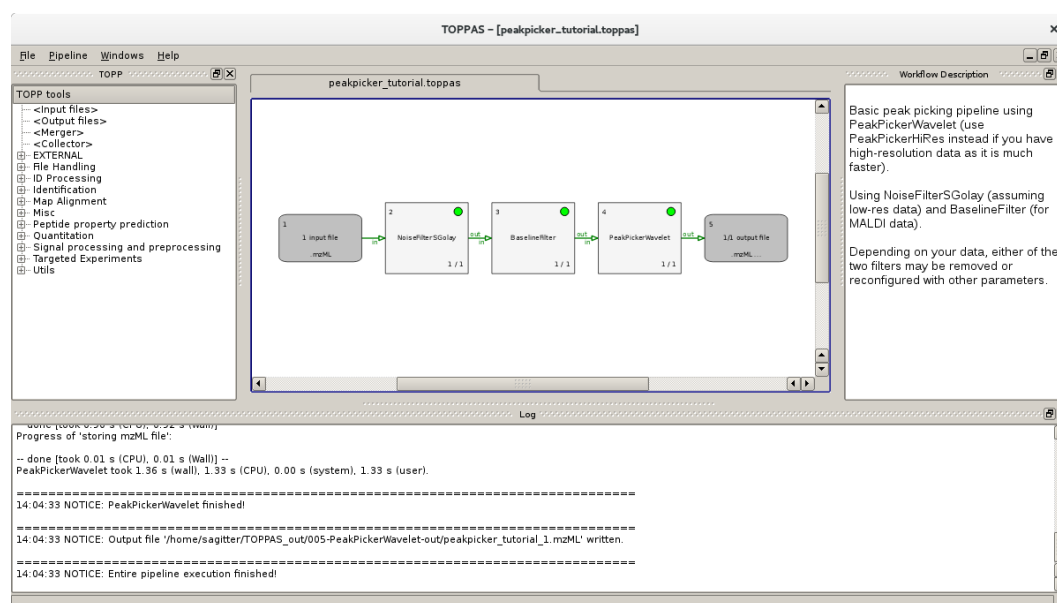
The image shows the centroided peak data and the found peptide features. The used parameters can be found in the TOPP tools dialog.



4 Using TOPPAS to create and run TOPP pipelines

TOPPAS allows to create, edit, open, save, and run TOPP workflows. Pipelines can be created conveniently in a GUI. The parameters of all involved tools can be edited within TOPPAS and are also saved as part of the pipeline definition in the .toppas file. Furthermore, **TOPPAS** interactively performs validity checks during the pipeline editing process and before execution (i.e., a dry run of the entire pipeline), in order to prevent the creation of invalid workflows. Once set up and saved, a workflow can also be run without the GUI using *ExecutePipeline -in <file>*.

The following figure shows a simple example pipeline that has just been created and executed successfully:



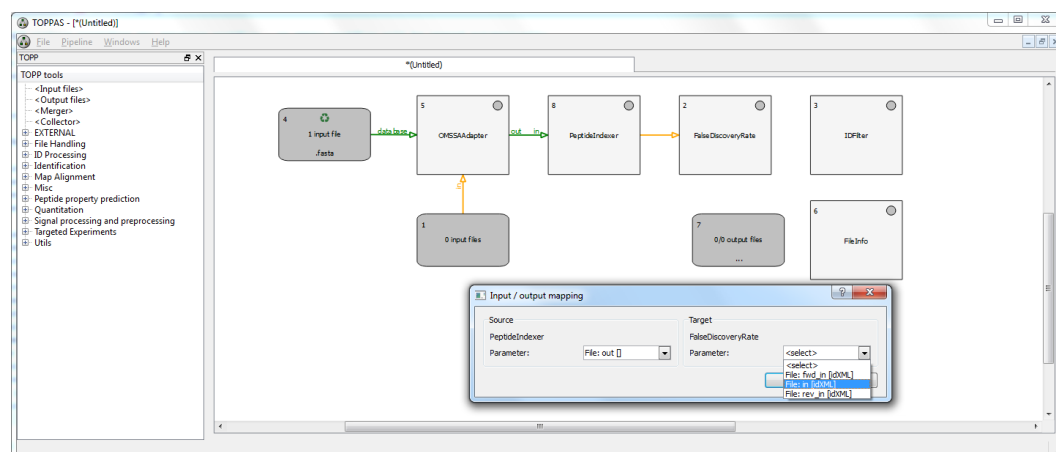
To create a new TOPPAS file, you can either:

- open TOPPAS without providing any existing workflow - an empty workflow will be opened automatically
- in a running TOPPAS program choose: *File > New*
- create an empty file in your file browser (explorer) with the suffix .toppas and double-click it (on Windows systems all .toppas files are associated with TOPPAS automatically during installation of OpenMS, on Linux and MacOS you might need to manually associate the extension)

4.1 Introduction

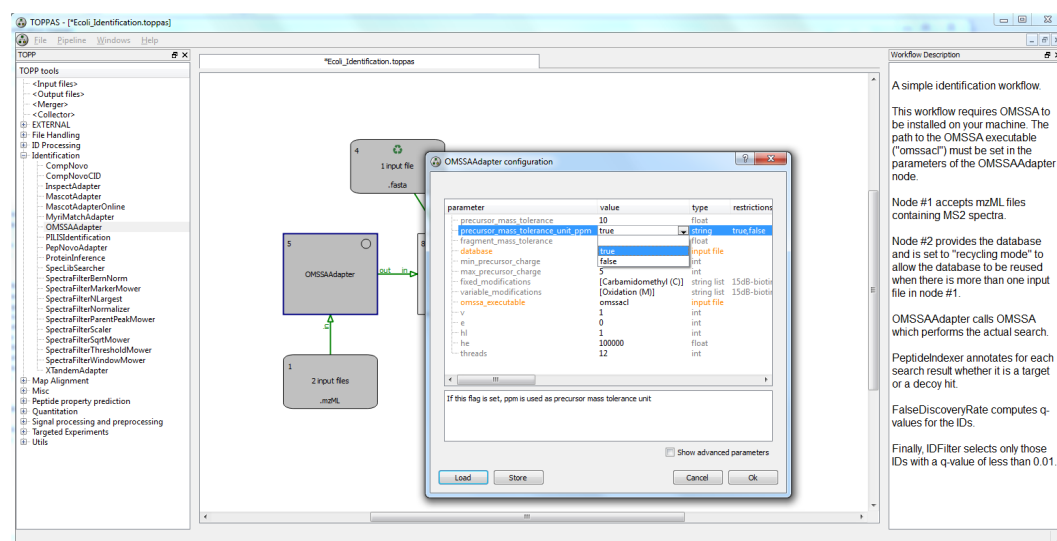
The following figure shows the **TOPPAS** main window and a pipeline which is just being created. The user has added some tools by drag&dropping them from the TOPP tool list on the left onto the central window. Additionally, the user has added nodes for input and output files.

Next, the user has drawn some connections between the tools which determine the data flow of the pipeline. Connections can be drawn by first *deselecting* the desired source node (by clicking anywhere but not on another node) and then dragging the mouse from the source to the target node (if a *selected* node is dragged, it is moved on the canvas instead). When a connection is created and the source (the target) has more than one output (input) parameter, an input/output parameter mapping dialog shows up and lets the user select the output parameter of the source node and the input parameter of the target node for this data flow - shown here for the connection between PeptideIndexer and FalseDiscoveryRate. If the file types of the selected input and output parameters are not compatible with each other, **TOPPAS** will refuse to add the connection. It will also refuse to add a connection if it would create a cycle in the workflow, or if it just would not make sense, e.g., if its target is an input file node. The connection between the input file node (#1) and the OMSSAAdapter (#5) tool is painted yellow which indicates it is not ready yet, because no input files have been specified.



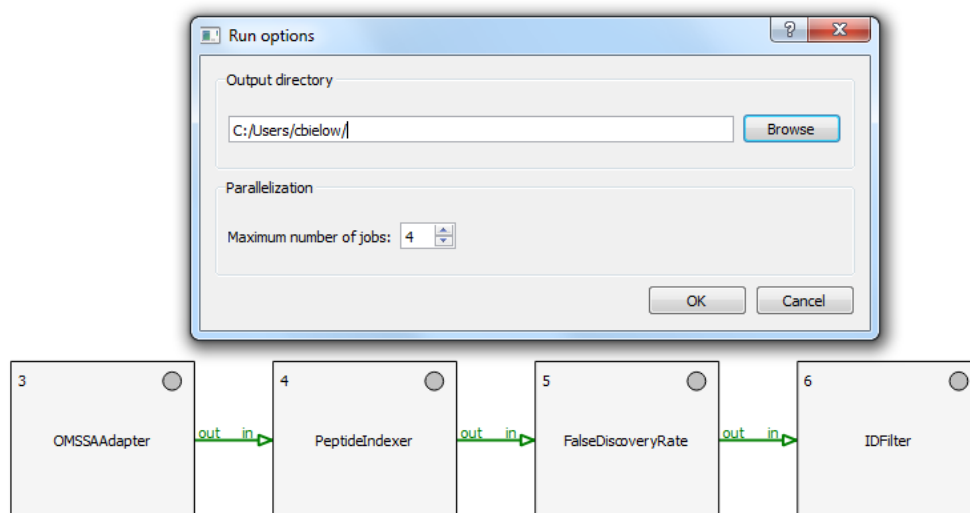
The input/output mapping of connections can be changed at any time during the editing process by double-clicking an connections or by selecting *Edit I/O mapping* from the context menu which appears when a connection is right-clicked. All visible items (i.e. connections and the different kinds of nodes) have such a context menu. For a detailed list of the different menus and their entries, see [Menus](#) .

The following figure shows a possible next step: the user has double-clicked one of the tool nodes in order to configure its parameters. By default, the standard parameters are used for each tool. Again, this can also be done by selecting *Edit parameters* from the context menu of the tool.



Once the pipeline has been set up, the input files have to be specified before the pipeline can be executed. This is done by double-clicking an input node and selecting the desired files in the dialog that appears. Input nodes have a special mode named "recycling mode", i.e., if the input node has fewer files than the following node has rounds (as it might have two incoming connections) then the files are recycled until all rounds are satisfied. This might be useful if one input node specifies a single database file (for a Search-Adapter like Mascot) and another input node has the actual MS2 experiments (which is usually more than one). Then the database input node would be set to "recycle" the database file, i.e. use it for every run of the MascotAdapter node. The input list can be recycled an arbitrary number of times, but the recycling has to be *complete*, i.e. the number of rounds of the downstream node have to be a multiple of the number of input files. Recycling mode can be activated by right-clicking the input node and selecting the according entry from the context menu.

Finally, if you have input and output nodes at every end of your pipeline and all connections are green, you can select *Pipeline > Run* in the menu bar or just press *F5*.



You will be asked for an output file directory where a sub-directory, *TOPPAS_out*, will be created. This directory will contain your output files. You can also specify the number of jobs TOPPAS is allowed to run in parallel. If a number greater than 1 is selected, TOPPAS will parallelize the pipeline execution in the following scenarios:

- A tool has to process more than one file. In this case, multiple files can be processed in parallel.
- The pipeline contains multiple branches that are independent of each other. In this case, multiple tools can run in parallel.

Be careful with this setting, however, as some of the TOPP tools require large amounts of RAM (depending on the size of your dataset). Running too many parallel jobs on a machine with not enough memory will cause problems. Also, do not confuse this setting with the *threads* parameter of the individual TOPP tools: every TOPP tool has this parameter specifying the maximum number of threads the tool is allowed to use (although only a subset of the TOPP tools make use of this parameter, since there are tasks that cannot be computed in parallel). Be especially careful with combinations of both parameters! If you have a pipeline containing the *FeatureFinderCentroided*, for example, and set its *threads* parameter to 8, and you additionally set the number of parallel jobs in **TOPPAS** to 8, then you end up using 64 threads in parallel, which might not be what you intended to do.

In addition to *TOPPAS_out*, a *TOPPAS_tmp* directory will be created in the OpenMS temp path (call the *OpenMSInfo* tool to see where exactly). It will contain all temporary files that are passed from tool to tool within the pipeline. Both folders contain further sub-directories which are named after the number in the top-left corner of the node they belong to (plus the name of the tool for temporary files). During pipeline execution, the status lights in the top-right corner of the tools indicate if the tool has finished successfully (green), is currently running (yellow), has not done anything so far (gray), is scheduled to run next (blue), or has crashed (red). The numbers in the bottom-right corner of every tool show how many files have already been processed and the overall number of files to be processed by this tool. When the execution has finished, you can check the generated output files of every node quickly by selecting "@a Open @a files @a in @a TOPPView" or "@a Open @a containing @a folder" from the context menu (right click on the node).

4.2 Mouse and keyboard

Using the mouse, you can

- drag&drop tools from the TOPP tool list onto the workflow window (you can also double-click them instead)
- select items (by clicking)
- select multiple items (by holding down *CTRL* while clicking)
- select multiple items (by holding down *CTRL* and dragging the mouse in order to "catch" items with a selection rectangle)
- move all selected items (by dragging one of them)
- draw a new connection from one node to another (by dragging; source must be deselected first)
- specify input files (by double-clicking an input node)
- configure parameters of tools (by double-clicking a tool node)
- specify the input/output mapping of connections (by double-clicking a connection)
- translate the view (by dragging anywhere but on an item)
- zoom in and out (using the mouse wheel)
- make the context menu of an item appear (by right-clicking it)

Using the keyboard, you can

- delete all selected items (*DEL* or *BACKSPACE*)
- zoom in and out (+ / -)
- run the pipeline (*F5*)
- open this tutorial (*F1*)

Using the mouse+keyboard, you can

- copy a node's parameters to another node (only parameters with identical names will be copied, e.g., 'fixed_modifications') (*CTRL* while creating an edge) The edge will be colored as dark magenta to indicate parameter copying.

4.3 Menus

Menu bar:

In the *File* menu, you can

- create a new, empty workflow (*New*)
- open an existing one (*Open*)
- open an example file (*Open example file*)
 - include an existing workflow to the current workflow (*Include*)
 - visit the online workflow repository (*Online repository*)
- save a workflow (*Save / Save as*)
 - export the workflow as image (*Export as image*)
 - refresh the parameter definitions of all tools contained in the workflow (*Refresh parameters*)
- close the current window (*Close*)
 - load and save TOPPAS resource files (.trf) (*Load / Save TOPPAS resource file*)

In the *Pipeline* menu, you can

- run a pipeline (*Run*)
- abort a currently running pipeline (*Abort*)

In the *Windows* menu, you can

- make the TOPP tool list window on the left, the description window on the right, and the log message at the bottom (in)visible.

In the *Help* menu, you can

- go to the OpenMS website (*OpenMS website*)

- open this tutorial (*TOPPAS tutorial*)

Context menus:

In the context menu of an *input node*, you can

- specify the input files
- open the specified files in TOPPView
- open the input files' folder in the window manager (explorer)
- toggle the "recycling" mode
- copy, cut, and remove the node

In the context menu of a *tool*, you can

- configure the parameters of the tool
- resume the pipeline at this node
- open its temporary output files in TOPPView
- open the temporary output folder in the file manager (explorer)
- toggle the "recycling" mode
- copy, cut, and remove the node

In the context menu of a *Merger* or *Collector*, you can

- toggle the "recycling" mode
- copy, cut, and remove the node

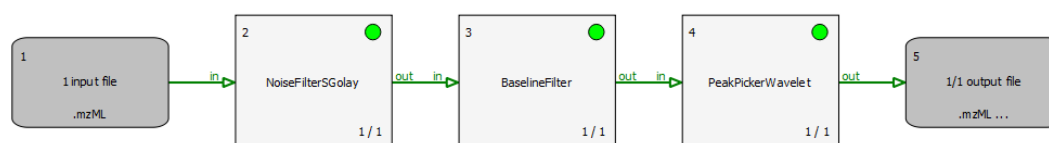
In the context menu of an *output node*, you can

- open the output files in TOPPView
- open the output files' folder in the window manager (explorer)
- copy, cut, and remove the node

The following sections explain the example pipelines TOPPAS comes with. You can open them by selecting *File > Open example file*. All input files and parameters are already specified, so you can just hit *Pipeline > Run* (or press *F5*) and see what happens.

4.4 Profile data processing

The file *peakpicker_tutorial.toppas* contains a simple pipeline representing a common use case: starting with profile data, the noise is eliminated and the baseline is subtracted. Then, *PeakPickerWavelet* is used to find all peaks in the noise-filtered and baseline-reduced profile data. This workflow is also described in the section [Profile data processing](#). The individual steps are explained in more detail in the TOPPView tutorial: [Smoothing raw data](#), [Subtracting a baseline from a spectrum](#), and [Picking peaks](#).

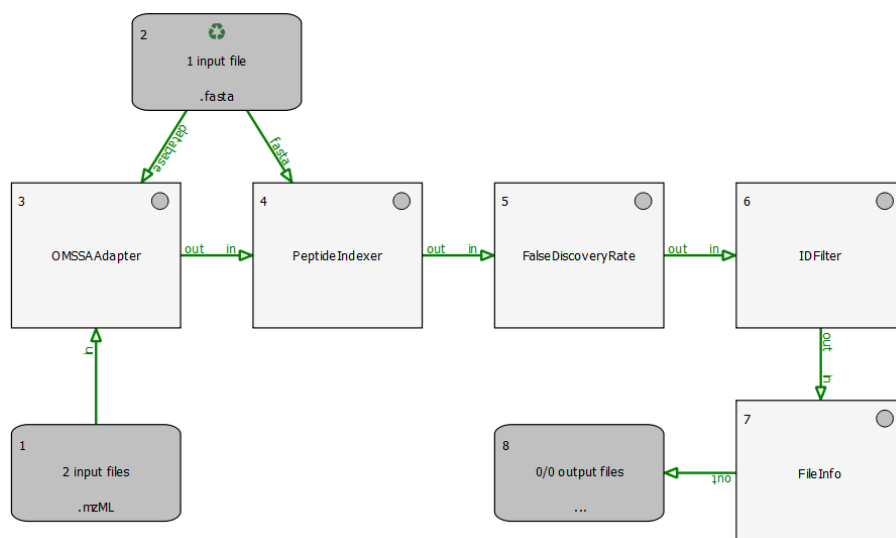


4.5 Identification of E. coli peptides

This section describes an example identification pipeline contained in the example directory, *Ecoli_Identification.toppas*. It is shipped together with a reduced example mzML file containing 139 MS2 spectra from an E. coli run on an Orbitrap instrument as well as an E. coli target-decoy database.

We use the search engine OMSSA (Geer et al., 2004) for peptide identification. Therefore, OMSSA must be installed and the path to the OMSSA executable (*omssacl*) must be set in the parameters of the OMSSAAdapter node.

- Node #1 accepts mzML files containing MS2 spectra.
- Node #2 provides the database and is set to "recycling mode" to allow the database to be reused when there is more than one input file in node #1.
- OMSSAAdapter calls OMSSA which performs the actual search.
- PeptideIndexer annotates for each search result whether it is a target or a decoy hit.
- FalseDiscoveryRate computes q-values for the IDs.
- Finally, IDFilter selects only those IDs with a q-value of less than 1%.

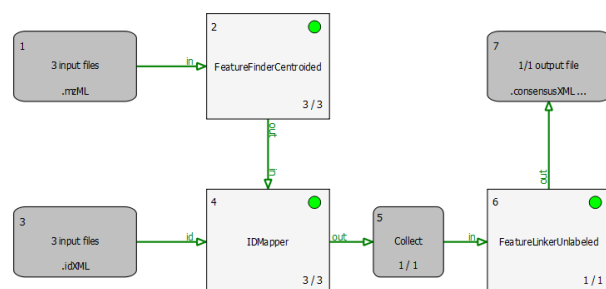


Extensions to this pipeline would be to do the annotation of the spectra with multiple search engines and combine the results afterwards, using the ConsensusID TOPP tool.

The results may be exported using the TextExporter tool, for further analysis with different tools.

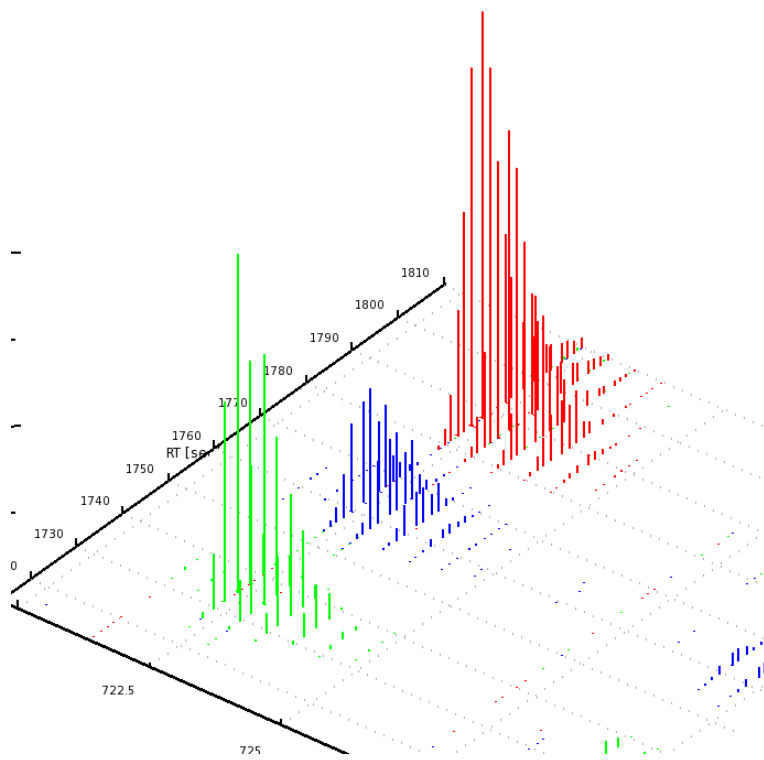
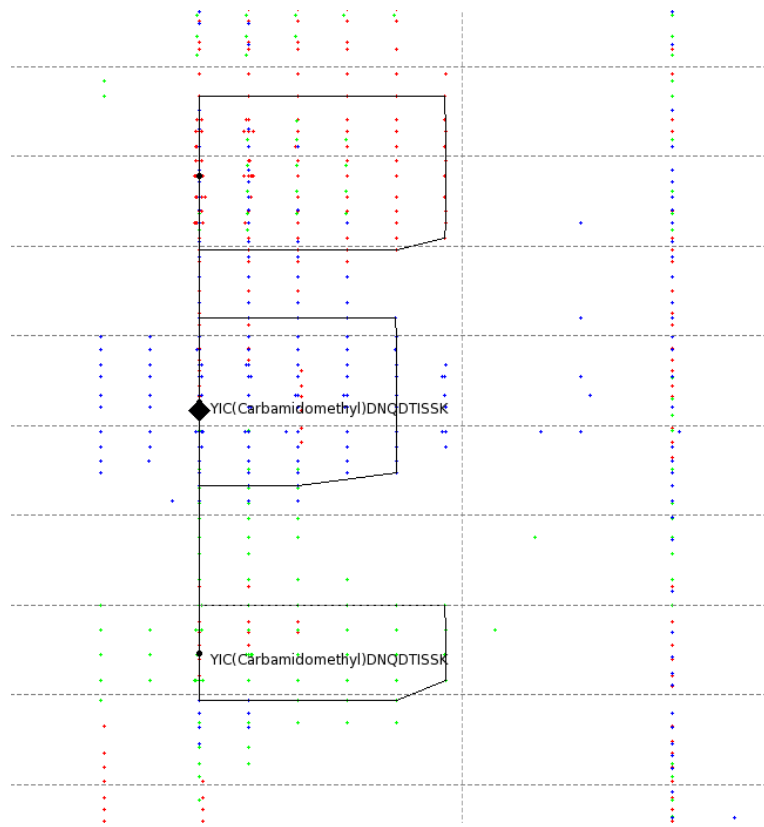
4.6 Quantitation of BSA runs

The simple pipeline described in this section (*BSA_Quantitation.toppas*) can be used to quantify peptides that occur on different runs. The example dataset contains three different bovine serum albumin (BSA) runs. First, FeatureFinderCentroided is called since the dataset is centroided. The results of the feature finding are then annotated with (existing) identification results. For convenience, we provide these search results from OMSSA with peptides with an FDR of 5% in the BSA directory.



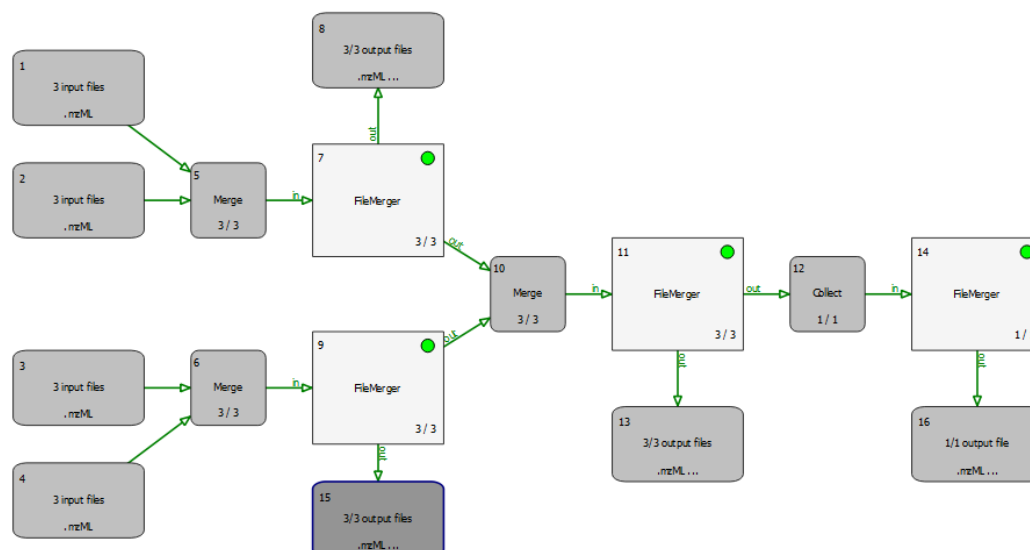
Identifications are mapped to features by the IDMapper. The last step is performed by FeatureLinker-Unlabeled which links corresponding features. The results can be used to calculate ratios, for example. The data could also be exported to a text based format using the TextExporter for further processing (e.g., in Microsoft Excel).

The results can be opened in TOPPView. The next figures show the results in 2D and 3D view, together with the feature intermediate results. One can see that the intensities and retention times are slightly different between the runs. To correct for retention times shift, a map alignment could be done, either on the spectral data or on the feature data.



4.7 Merger and Collect nodes

The following example is actually not a useful workflow but is supposed to demonstrate how merger and collector nodes can be used in a pipeline. Have a look at *merger_tutorial.toppas*:



As its name suggests, a merger merges its incoming file lists, i.e., files of all incoming edges are appended into new lists (which have as many elements as the merger has incoming connections). All tools this merger has outgoing connections to are called with these merged lists as input files. All incoming connections should pass the same number of files (unless the corresponding preceding tool is in recycling mode).

A collector node, on the other hand, waits for all rounds to finish before concatenating all files from all incoming connections into one single list. It then calls the next tool with this list of files as input. This will happen exactly once during the entire pipeline run.

In order to track what is happening, you can just open the example file and run it. When the pipeline execution has finished, have a look at all input and output files (e.g., select *Open in TOPPView* in the context menu of the input/output nodes). The input files are named *rt_1.mzML*, *rt_2.mzML*, ... and each contains a single spectrum with RT as indicated by the filename, so you can easily see which files have been merged together.

5 Advanced Users: Tips & Tricks

We will not introduce you to some advanced concepts of TOPP, which will increase your productivity and easy usage.

Global database for search engine adapters

In your \$HOME directory you will find an *OpenMS.ini* file in the *.OpenMS* subfolder. This INI file contains global parameters which are honored by many/all TOPP tools - depending on what the parameters refer to. The *id_db_dir* parameter allows you to specify one or more directories where you have placed FASTA files (or related, e.g., *.psq* files). If you now specify just a filename (without path) in an ID engine adapter, the filename will be looked up in the current working directory. If its not found, the directories specified in *id_db_dir* will be searched. This allows you to build scripts and/or TOPPAS pipelines which are portable across several computers - you just have to adapt the *OpenMS.ini* once on each machine.

Note when using TOPPAS: You can use an "input file" node to specify the FASTA file for several engines

simultaneously. However, when selecting the file, TOPPAS will use the absolute pathname and the dialog will not allow you to name a non-existing file. After you've selected the file, you can however edit the filename and remove the path (this will issue a warning which you can ignore).

Note

This approach does not work for our MascotAdapters, as each Mascot instance has its own database managed internally. You can however, make sure that the database is present on all mascot servers you are going to use, thus making the INI settings portable.

Using External tools in your workflows

OpenMS supports the wrapping of external tools (like msconvert from ProteoWizard), thus allowing you to build scripts and/or TOPPAS pipelines containing external tools.

6 Scripting with TOPP

This tutorial will give you a brief overview of the most important TOPP tools. First, we explain some basics that you will need for every TOPP tool, then we show several example pipelines.

6.1 File formats

The TOPP tools use the HUPO-PSI standard format mzML 1.1.0 as input format. In order to convert other open formats (mzData, mzXML, DTA, ANDI/MS) to mzML, a file converter is provided by TOPP.

Proprietary MS machine formats are not supported. If you need to convert these formats to mzML, mzData or mzXML, please have a look at the [SASHIMI project page](#) or contact your MS machine vendor.

mzML covers only the output of a mass spectrometry experiment. For further analysis of this data several other file formats are needed. The main file formats used by TOPP are:

- **mzML** The HUPO-PSI standard format for mass spectrometry data.
- **featureXML** The OpenMS format for quantitation results.
- **consensusXML** The OpenMS format for grouping features in one map or across several maps.
- **idXML** The OpenMS format for protein and peptide identification.

Documented schemas of the OpenMS formats can be found at <http://www.openms.de/schemas/>.

idXML files and *consensusXML* files created by OpenMS can be visualized in a web browser directly. - XSLT stylesheets are used to transform the XML to HTML code. The stylesheets are contained in the *OpenMS/share/OpenMS/XSLT/* folder of your OpenMS installation.

If you want to view the file on the computer with the OpenMS installation, you can just open it in your browser.

If you copy the file to another computer, you have must copy the XSLT stylesheet to that computer and change the second line in the XML file. The following example shows how to change the stylesheet location for an idXML file. You simply have to change the *PATH* in the line

```
<?xml-stylesheet type="text/xsl" href="file:/// *PATH*idXML.xsl"?>
```

to the folder where the stylesheet resides.

6.2 Common arguments of the TOPP tools

The command line and INI file parameters of the TOPP tools vary due to the different tasks of the TOPP tools. However, all TOPP tools share this common interface:

- **-ini <file>** Use the given TOPP INI file
- **-log <file>** Location of the log file (default: 'TOPP.log')
- **-instance <n>** Instance number in the TOPP INI file (default: '1')
- **-debug <n>** Sets the debug level (default: '0')
- **-write_ini <file>** Writes an example INI file
- **-no_progress** Disables progress logging to command line
- **--help** Shows a help page for the command line and INI file options

6.3 TOPP INI files

Each TOPP tool has its own set of parameters which can be specified at the command line. However, a more convenient (and persistent) way to handle larger sets of parameters is to use TOPP INI files. TOPP INI files are XML-based and can contain the configuration of one or several TOPP tools.

The following examples will give an overview of how TOPP tools can be chained in order to create analysis pipelines. INI files are the recommended way to store all settings of such a pipeline in a single place.

Note that the issue of finding suitable parameters for the tools is not addressed here. If you encounter problems during the execution of the example pipelines on your data, you probably have to adapt the parameters. Have a look at the documentation of the corresponding TOPP tool in that case.

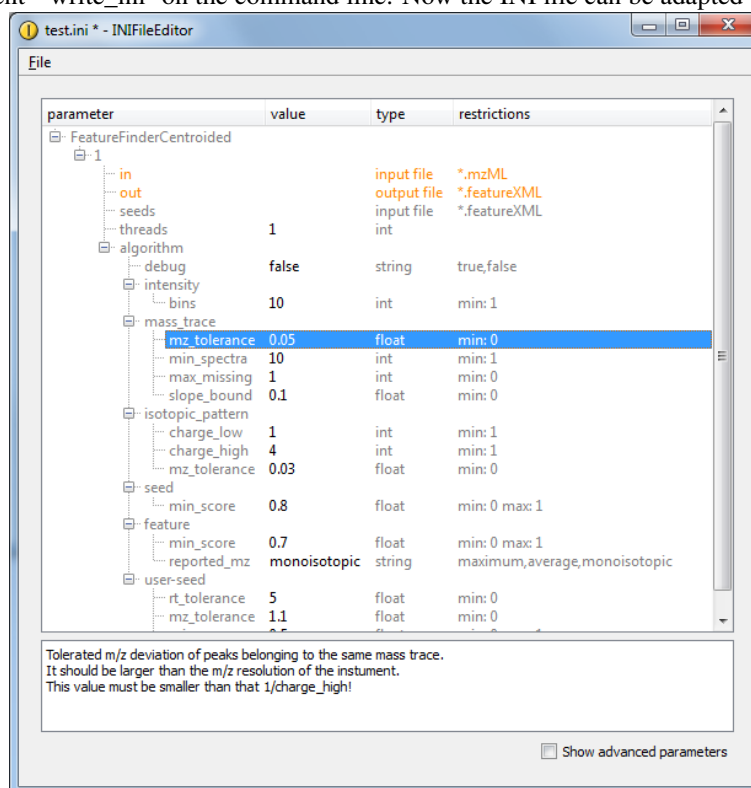
6.3.1 Parameter documentation

General documentation of a TOPP tool and documentation for the command line parameters, can be displayed using the command line argument `--help`.

Some TOPP tools also have subsections of parameters that are internally handed to an algorithm. The documentation of these subsections is not displayed with `--help`. It is however displayed in **INIFileEditor** (see next section).

6.3.2 Creating an INI file for a TOPP tool

The easiest way of creating an INI file is to advise the corresponding TOPP tool to write its default configuration file using the argument `'-write_ini'` on the command line. Now the INI file can be adapted to your



needs using **INIFileEditor**.

In the TOPP_INIFileEditor, the documentation of the parameters is displayed in the window at the bottom, once you click on the respective parameter.

6.3.3 Updating an INI file for a TOPP tool or a whole TOPPAS pipeline

If you have an old INI file which does not work for a newer OpenMS version (due to renamed/removed or new) parameters, you can rescue parameter's whose name did not change into the new version by using our UTILS_INIUpdater tool by calling it with (a list of) outdated INI and/or TOPPAS files. See the I-NIUpdater tool description for details. This will remove invalid parameters and add new parameters (if available) while retaining values for unchanged parameters.

6.3.4 General structure of an INI file

An INI file is always enclosed by the `<PARAMETERS>` tag. Inside this tag, a tree-like hierarchy is created with `<NODE>` tags that represent sections and `<ITEM>` tags, each of which stores one of the parameters. The first two level of the hierarchy have a special meaning.

Example: Below is the content of an INI file for **FileFilter**.

Several parameter sets for a TOPP tool can be specified in a *tool section*. The tool section is always named after the program itself, in this case "FileFilter".

- In order to make storing several parameter sets for the same tool in one INI file possible, the tool section contains one or several *numbered instance subsections* ('1', '2', ...). These numbers are the instance numbers which can be specified using the '-instance' command line argument. (Remember the default is '1'.)
- Within each instance section, the actual parameters of the TOPP tool are given. INI files for complex tools can contain nested subsections in order to group related parameters.
- If a parameter is not found in the instance section, the *tool-specific common section* is considered.
- Finally, we look if the *general common section* contains a value for the parameter.

Imagine we call the **FileFilter** tool with the INI file given below and instance number '2'. The FileFilter parameters *rt* and *mz* are looked up by the tool. *mz* can be found in section **FileFilter - 2**. *rt* is not specified in this section, thus the *common - FileFilter* section is checked first, where it is found in our example. - When looking up the *debug* parameter, the tool would search the instance section and tool-specific common section without finding a value. Finally, the general *common* section would be checked, where the debug level is specified.

```
<PARAMETERS>

  <NODE name="FileFilter">
    <NODE name="1">
      <ITEM name="rt" value="0:1200" type="string"/>
    </NODE>
    <NODE name="2">
      <ITEM name="mz" value="700:1000" type="string"/>
    </NODE>
  </NODE>

  <NODE name="common">
    <NODE name="FileFilter">
      <ITEM name="rt" value=":" type="string"/>
      <ITEM name="mz" value=":" type="string"/>
    </NODE>
    <ITEM name="debug" value="2" type="int"/>
  </NODE>
</PARAMETERS>
```

6.4 General information about peak and feature maps

If you want some general information about a peak or feature map, use the **FileInfo** tool.

- It can print RT, m/z and intensity ranges, the overall number of peaks, and the distribution of MS levels
- It can print a statistical summary of intensities
- It can print some meta information
- It can validate XML files against their schema
- It can check for corrupt data in peak files See the 'FileInfo --help' for details.

6.5 Problems with input files

If you are experiencing problems while processing an XML file you can check if the file does validate against the XML schema:

```
FileInfo -v -in infile.mzML
```

Validation is available for several file formats including mzML, mzData, mzXML, featureXML and idXML.

Another frequently-occurring problem is corrupt data. You can check for corrupt data in peak files with **FileInfo** as well:

```
FileInfo -c -in infile.mzML
```

6.6 Converting your files to mzML

The TOPP tools work only on the HUPO-PSI *mzML* format. If you need to convert *mzData*, *mzXML* or *ANDI/MS* data to *mzML*, you can do that using the **FileConverter**, e.g.

```
FileConverter -in infile.mzXML -out outfile.mzML
```

If you use the format names as file extension, the tool derives the format from the extension. For other extensions, the file formats of the input and output file can be given explicitly.

6.7 Converting between DTA and mzML

Sequest DTA files can be extracted from a mzML file using the **DTAExtractor**:

```
DTAExtractor -in infile.mzML -out outfile
```

The retention time of a scan, the precursor mass-to-charge ratio (for MS/MS scans) and the file extension are appended to the output file name.

To combine several files (e.g. DTA files) to an mzML file use the **FileMerger**:

```
FileMerger -in infile_list.txt -out outfile.mzML
```

The retention times of the scans can be generated, taken from the *infile_list.txt* or can be extracted from the DTA file names. See the FileMerger documentation for details.

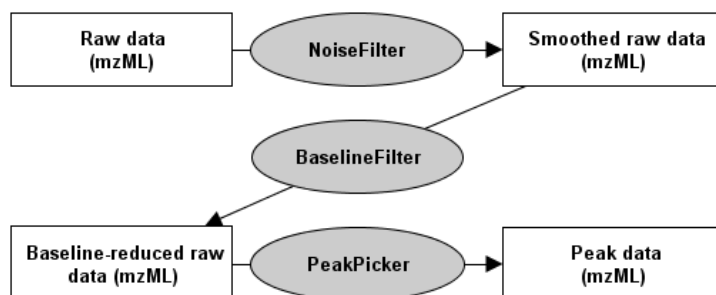
6.8 Extracting part of the data from a file

If you want to extract part of the data from an mzML file, you can use the **FileFilter** tool. It allows filtering for RT, m/z and intensity range or for MS level. To extract the MS/MS scans between retention time 100 and 1500, you would use the following command:

```
FileFilter -in infile.mzML -levels 2 -rt 100:1500 -out outfile.mzML
```

Goal: You want to find all peaks in your profile data.

The first step shown here is the elimination of noise using a **NoiseFilter**. The now smoothed profile data can be further processed by subtracting the baseline with the **BaselineFilter**. Then use one of the **PeakPickers** to find all peaks in the baseline-reduced profile data.



We offer two different smoothing filters: NoiseFilterGaussian and NoiseFilterSGolay. If you want to use the Savitzky Golay filter, or our **BaselineFilter** with non equally spaced profile data, e.g. TOF data, you have to generate equally spaced data using the **Resampler** tool.

6.9 Picking peaks with a PeakPicker

The **PeakPicker** tools allow for picking peaks in profile data. Currently, there are two different TOPP tools available, PeakPickerWavelet and PeakPickerHiRes.

PeakPickerWavelet **Input data:** profile data (low/medium resolution)

Description:

This peak picking algorithm uses the continuous wavelet transform of a raw data signal to detect mass peaks. Afterwards a given asymmetric peak function is fitted to the raw data and important peak parameters (e.g. fwhm) are extracted. In an optional step these parameters can be optimized using a non-linear optimization method.

The algorithm is described in detail in Lange et al. (2006) Proc. PSB-06.

Application:

This algorithm was designed for low and medium resolution data. It can also be applied to high-resolution data, but can be slow on large datasets.

See the PeakPickerCWT class documentation for a parameter list.

PeakPickerHiRes **Input data:** profile data (high resolution)

Description:

This peak-picking algorithm detects ion signals in raw data and reconstructs the corresponding peak shape by cubic spline interpolation. Signal detection depends on the signal-to-noise ratio which is adjustable by the user (see parameter *signal_to_noise*). A picked peak's *m/z* and intensity value is given by the maximum of the underlying peak spline. Please notice that this method is still **experimental** since it has not been tested thoroughly yet.

Application:

The algorithm is best suited for high-resolution MS data (FT-ICR-MS, Orbitrap). In high-resolution data, the signals of ions with similar mass-to-charge ratios (*m/z*) exhibit little or no overlapping and therefore allow for a clear separation. Furthermore, ion signals tend to show well-defined peak shapes with narrow peak width. These properties facilitate a fast computation of picked peaks so that even large data sets can be processed very quickly.

See the PeakPickerHiRes class documentation for a parameter list.

6.10 Finding the right parameters for the

NoiseFilters, the BaselineFilter and the PeakPickers

Finding the right parameters is not trivial. The default parameters will not work on most datasets. In order to find good parameters, we propose the following procedure:

1. Load the data in TOPPView
 2. Extract a single scan from the middle of the HPLC gradient (Right click on scan)
 3. Experiment with the parameters until you have found the proper settings
- You can find the **NoiseFilters**, the **BaselineFilter**, and the **PeakPickers** in **TOPPView** in the menu 'Layer' - 'Apply TOPP tool'

We offer two calibration methods: an internal and an external calibration. Both can handle peak data as well as profile data. If you want to calibrate profile data, a peak picking step is necessary, the important parameters can be set via the ini-file. If you have already picked data, don't forget the '-peak_data' flag.

The external calibration (**TOFCalibration**) is used to convert flight times into m/z- values with the help of external calibrant spectra containing e.g. a polymer like polylysine. For the calibrant spectra, the calibration constants the machine uses need to be known as well as the expected masses. Then a quadratic function is fitted to convert the flight times into m/z-values.

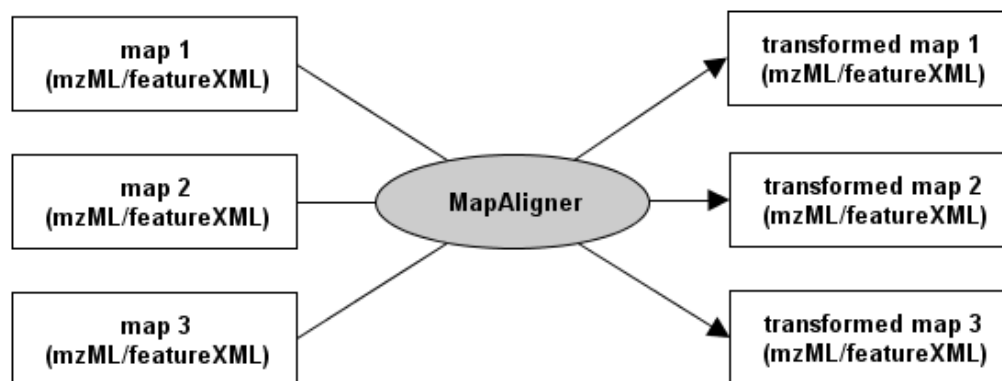
The internal calibration (**InternalCalibration**) uses reference masses in the spectra to correct the m/z-values using a linear function.

In a typical setting one would first pick the TOF-data, then perform the TOFCalibration and then the InternalCalibration:

```
PeakPickerWavelet -in raw_tof.mzML -out picked_tof.mzML -ini pp.ini
TOFCalibration -in picked_tof.mzML -out picked.mzML -ext_calibrants ext_cal.
                mzML
                -ref_masses ext_cal_masses
                -tof_const tof_conv_consts -peak_data
InternalCalibration -in picked.mzML -out picked_calibrated.mzML
                  -ref_masses internal_calibrant_masses -peak_data
```

The goal of map alignment is to transform different HPLC-MS maps (or derived maps) to a common retention time axis. It corrects for shifted and scaled retention times, which may result from changes of the chromatography.

The different **MapAligner** tools take n input maps, de-warp them and store the n de-warped maps. The following image shows the general procedure:



There are different map alignment tools available. The following table gives a rough overview of them:

Application: MapAlignerPoseClustering	Applicable to: feature maps, peak maps
Description: This algorithm does a star-wise alignment of the input data. The center of the star is the map with most data points. All other maps are then aligned to the center map by estimating a linear transformation (shift and scaling) of retention times. The transformation is estimated using a pose clustering approach as described in doi:10.1093/bioinformatics/btm209	
Application: MapAlignerIdentification	Applicable to: feature maps, consensus maps, identifications
Description: This algorithm utilizes peptide identifications, and is thus applicable to files containing peptide IDs (idXML, annotated featureXML/consensusXML). It finds peptide sequences that different input files have in common and uses them as points of correspondence. From the retention times of these peptides, transformations are computed that convert each file to a consensus time scale.	
Application: MapAlignerSpectrum	Applicable to: peak maps
Description: This <i>experimental</i> algorithm uses a dynamic-programming approach based on spectrum similarity for the alignment. The resulting retention time mapping of dynamic-programming is then smoothed by fitting a spline to the retention time pairs.	
Application: MapRTTransformer	Applicable to: peak maps, feature maps, consensus maps, identifications
Description: This algorithm merely <i>applies</i> a set of transformations that are read from files (in TransformationXML format). These transformations might have been generated by a previous invocation of a MapAligner tool. For example, you might compute a transformation based on identifications and then apply it to the features or raw data. The transformation file format is not very complicated, so it is relatively easy to write (or generate) your own transformation files.	

For quantitation, the **FeatureFinder** tools are used. They extract the features from profile data or centroided data. TOPP offers different types of **FeatureFinders**:

<p>FeatureFinderIsotopeWavelet Input data: profile data</p> <p>Description: The algorithm has been designed to detect features in raw MS data sets. The current implementation is only able to handle MS1 data. An extension handling also tandem MS spectra is under development. The method is based on the <i>isotope wavelet</i>, which has been tailored to the detection of isotopic patterns following the averagine model. For more information about the theory behind this technique, please refer to Hussong et al.: "Efficient Analysis of Mass Spectrometry Data Using the Isotope Wavelet" (2007). Please note that this algorithm features no "modelling stage", since the structure of the isotopic pattern is explicitly coded by the wavelet itself. The algorithm also works for 2D maps (in combination with the so-called <i>sweep-line</i> technique (Schulz-Trieglaff et al.: "A Fast and Accurate Algorithm for the Quantification of Peptides from Mass Spectrometry Data" (2007))). The algorithm can be executed on (several) high-speed CUDA graphics cards. Tests on real-world data sets revealed potential speedups beyond factors of 200 (using 2 NVIDIA Tesla cards in parallel). Please refer to Hussong et al.: "Highly accelerated feature detection in proteomics data sets using modern graphics processing units" (2009) for more details on the implementation.</p> <p>Seeding: Identification of regions of interest by convolving the signal with the wavelet function. A score, measuring the closeness of the transform to a theoretically determined output function, finally distinguishes potential features from noise.</p> <p>Extension: The extension is based on the sweep-line paradigm and is done on the fly after the wavelet transform.</p> <p>Modelling: None (explicitly done by the wavelet). See the <code>FeatureFinderAlgorithmIsotopeWavelet</code> class documentation for a parameter list.</p>
<p>FeatureFinderCentroided Input data: peak data</p> <p>Description: This is an algorithm for feature detection based on peak data. In contrast to the other algorithms, it is based on peak/stick data, which makes it applicable even if no profile data is available. Another advantage is its speed due to the reduced amount of data after peak picking.</p> <p>Seeding: It identifies interesting regions by calculating a score for each peak based on</p> <ul style="list-style-type: none"> • the significance of the intensity in the local environment • RT dimension: the quality of the mass trace in a local RT window • m/z dimension: the quality of fit to an averagine isotope model <p>Extension: The extension is based on a heuristics -- the average slope of the mass trace for RT dimension, the best fit to averagine model in m/z dimension.</p> <p>Modelling: In model fitting, the retention time profile (Gaussian) of all mass traces is fitted to the data at the same time. After fitting, the data is truncated in RT and m/z dimension. The reported feature intensity is based on the fitted model, rather than on the (noisy) data. See the <code>FeatureFinderAlgorithmPicked</code> class documentation for a parameter list.</p>

In order to quantify differences across maps (label-free) or within a map (isotope-labeled), groups of corresponding features have to be found. The **FeatureLinker** TOPP tools support both approaches. These groups are represented by consensus features, which contain information about the constituting features in the maps as well as average position, intensity, and charge.

6.11 Isotope-labeled quantitation

Goal: You want to differentially quantify the features of an isotope-labeled HPLC-MS map.

The first step in this pipeline is to find the features of the HPLC-MS map. The **FeatureFinder** applications calculate the features from profile data or centroided data.

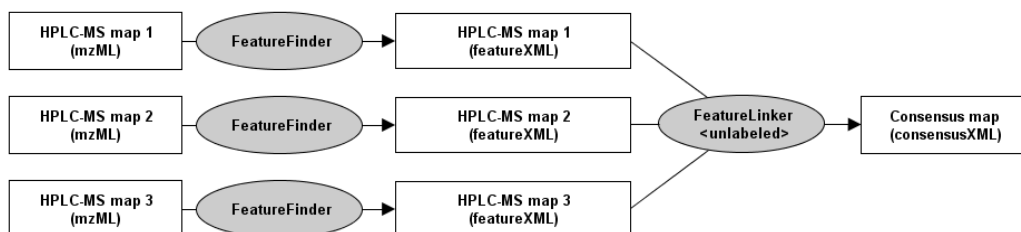
In the second step, the labeled pairs (e.g. light/heavy labels of ICAT) are determined by the **FeatureLinkerLabeled** application. **FeatureLinkerLabeled** first determines all possible pairs according to a given optimal shift and deviations in RT and m/z. Then it resolves ambiguous pairs using a greedy-algorithm that prefers pairs with a higher score. The score of a pair is the product of:

- feature quality of feature 1
- feature quality of feature 2
- quality measure for the shift (how near is it to the optimal shift)



6.12 Label-free quantitation

Goal: You want to differentially quantify the features of two or more label-free HPLC-MS map.



Note

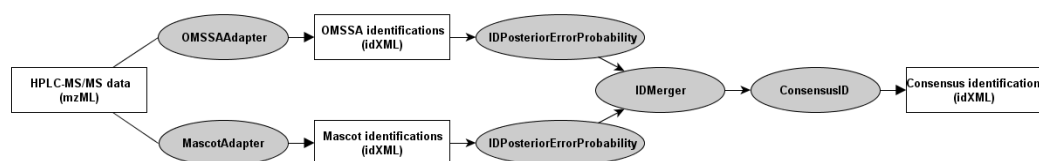
This algorithm assumes that the retention time axes of all input maps are very similar. If you need to correct for retention time distortions, please have a look at [Map alignment](#).

Goal: Use several identification engines in order to compute a consensus identification for a HPLC-MS\MS experiment.

OpenMS offers adapters for the following commercial and free peptide identification engines: Sequest, Mascot, OMSSA, PepNovo, XTandem and Inspect.

The adapters allow setting the input parameters and data for the identification engine and return the result in the OpenMS idXML format.

In order to improve the identification accuracy, several identification engines can be used and a consensus identification can be calculated from the results. The image below shows an example where Mascot and OMSSA results are fed to the **ConsensusID** tool (ConsensusID is currently usable for Mascot, OMSSA and XTandem).



Goal: Combine quantitation and identification results.

Protein/peptide identifications can be annotated to quantitation results (featureXML, consensusXML) by the **IDMapper** tool. The combined results can then be exported by the **TextExporter** tool: [Conversion between OpenMS XML formats and text formats](#). You can train a model for retention time prediction as

well as for the prediction of proteotypic peptides.

Two applications has been described in the following publications: Nico Pfeifer, Andreas Leinenbach, Christian G. Huber and Oliver Kohlbacher Statistical learning of peptide retention behavior in chromatographic separations: A new kernel-based approach for computational proteomics. *BMC Bioinformatics* 2007, 8:468 Nico Pfeifer, Andreas Leinenbach, Christian G. Huber and Oliver Kohlbacher Improving - Peptide Identification in Proteome Analysis by a Two-Dimensional Retention Time Filtering Approach *J. Proteome Res.* 2009, 8(8):4109-15

The predicted retention time can be used in IDFilter to filter out false identifications. Assume you have data from several identification runs. You should first align the data using MapAligner. Then you can use the various identification wrappers like MascotAdapter, OMSSAAdapter, ... to get the identifications. To train a model using RTModel you can now use IDFilter for one of the runs to get the high scoring identifications (40 to 200 distinct peptides should be enough). Then you use RTModel as described in the documentation to train a model for these spectra. With this model you can use RTPredict to predict the retention times for the remaining runs. The predicted retention times are stored in the idXML files. These predicted retention times can then be used to filter out false identifications using the IDFilter tool.

A typical sequence of TOPP tools would look like this:

```
MapAligner -in Run1.mzML,...,Run4.mzML -out Run1_aligned.mzML,...,
  Run4_aligned.mzML
MascotAdapter -in Run1_aligned.mzML -out Run1_aligned.idXML -ini Mascot.ini
MascotAdapter -in Run2_aligned.mzML -out Run2_aligned.idXML -ini Mascot.ini
MascotAdapter -in Run3_aligned.mzML -out Run3_aligned.idXML -ini Mascot.ini
MascotAdapter -in Run4_aligned.mzML -out Run4_aligned.idXML -ini Mascot.ini
IDFilter -in Run1_aligned.idXML -out Run1_best_hits.idXML -pep_fraction 1 -
  best_hits
RTModel -in Run1_best_hits.idXML -out Run1.model -ini RT.ini
RTPredict -in Run2_aligned.idXML -out Run2_predicted.idXML -svm_model Run1.
  model
RTPredict -in Run3_aligned.idXML -out Run3_predicted.idXML -svm_model Run1.
  model
RTPredict -in Run4_aligned.idXML -out Run4_predicted.idXML -svm_model Run1.
  model
IDFilter -in Run2_predicted.mzML -out Run2_filtered.mzML -rt_filtering
IDFilter -in Run3_predicted.mzML -out Run3_filtered.mzML -rt_filtering
IDFilter -in Run4_predicted.mzML -out Run4_filtered.mzML -rt_filtering
```

If you have a file with certainly identified peptides and want to train a model for RT prediction, you can also directly use the IDs. Therefore, the file has to have one peptide sequence together with the RT per line (separated by one tab or space). This can then be loaded by RTModel using the -textfile_input flag:

```
RTModel -in IDs_with_RTs.txt -out IDs_with_RTs.model -ini RT.ini -
  textfile_input
```

The likelihood of a peptide to be proteotypic can be predicted using PTModel and PTPredict. Assume we have a file PT.idXML which contains all proteotypic peptides of a set of proteins. Lets also assume, we have a fasta file containing the amino acid sequences of these proteins called mixture.fasta. To be able to train PTPredict, we need negative peptides (peptides, which are not proteotypic). Therefore, one can use the Digestor, which is located in the APPLICATIONS/UTILS/ folder together with the IDFilter:

```
Digestor -in mixture.fasta -out all.idXML
IDFilter -in all.idXML -out NonPT.idXML -exclusion_peptides_file PT.idXML
```

In this example the proteins are digested in silico and the non proteotypic peptides set is created by subtracting all proteotypic peptides from the set of all possible peptides. Then, one can train PTModel:

```
PTModel -in_positive PT.idXML -in_negative NonPT.idXML -out PT.model -ini PT.
  ini
```

6.13 Export of OpenMS XML formats

As TOPP offers no functionality for statistical analysis, this step is normally done using external statistics packages.

In order to export the OpenMS XML formats into an appropriate format for these packages the TOPP **TextExporter** can be used.

It converts the the following OpenMS XML formats to text files:

- featureXML
- idXML
- consensusXML

The use of the **TextExporter** is very simple:

```
TextExporter -in infile.idXML -out outfile.txt
```

6.14 Import of feature data to OpenMS

OpenMS offers a lot of visualization and analysis functionality for feature data.

Feature data in text format, e.g. from other analysis tools, can be imported using the **TextImporter**. - The default mode accepts comma separated values containing the following columns: RT, m/z, intensity. Additionally meta data columns may follow. If meta data is used, meta data column names have to be specified in a header line. Without headers:

```
1201 503.123 1435000
1201 1006.246 1235200
```

Or with headers:

```
RT m/z Int isHeavy myMeta
1201 503.123 1435000 true 2
1201 1006.246 1235200 maybe 1
```

Example invocation:

```
TextImporter -in infile.txt -out outfile.featureXML
```

The tool also supports data from msInspect,SpecArray and Kroenik(Hardkloer sibling), just specify the **-mode** option accordingly.

6.15 Import of protein/peptide identification data to OpenMS

Peptide/protein identification data from several identification engines can be converted to idXML format using the **IDFileConverter** tool.

It can currently read the following formats:

- Sequest output folder
- pepXML file
- idXML file

It can currently write the following formats:

- pepXML
- idXML

This example shows how to convert pepXML to idXML:

```
IDFileConverter -in infile.pepXML -out outfile.idXML
```