OpenMS Tutorial

The OpenMS Developers

Contents

1 General remarks

- This handout will guide you through an introductory tutorial for the OpenMS/TOPP software package [1].
- OpenMS [2, 3] 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") [4] can be understood as small building blocks of arbitrarily complex data analysis workflows.
- In order to facilitate workflow construction, OpenMS was integrated into KNIME [5], 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 [6].
- 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 released online on our GitHub page OpenMS/Tutorials).

2 Getting started

2.1 Installation

Before we get started we will install OpenMS and KNIME. If you take part in a training session you will have likely received an USB stick from us that contains the required data and software. If we provide laptops with the software you may of course skip the installation process and continue reading the next section.

2.1.1 Installation from the OpenMS USB stick

Please choose the directory that matches your operating system and execute the installer.

For example for **Windows** you call

- the OpenMS installer: 🗀`Windows / OpenMS-2.1.0_Win64_setup.exe`
- the KNIME installer: 🗀`Windows / KNIME Full 3.3.1 Installer (64bit).exe`
- 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 or on the OpenMS USB stick (🗀`Windows / OpenMS-2.1-prerequisi`

on **macOS** you call

- the OpenMS installer: 🗀`Mac / OpenMS-2.1.0_Mac.dmg`
- the KNIME installer: 🗀`Mac / knime-full_3.3.1.app.macosx.cocoa.x86_64.dmg`

and follow the instructions. For the OpenMS installation on macOS, you need to accept the license drag and drop the OpenMS folder into your Applications folder. On **Linux** you can extract KNIME to a folder of your choice and for TOPPView you need to install OpenMS via your package manager or build it on your own with the instructions under www.openms.de/documentation.

2.1.2 Installation from the internet

If you are working through this tutorial at home you can get the installers under the following links:

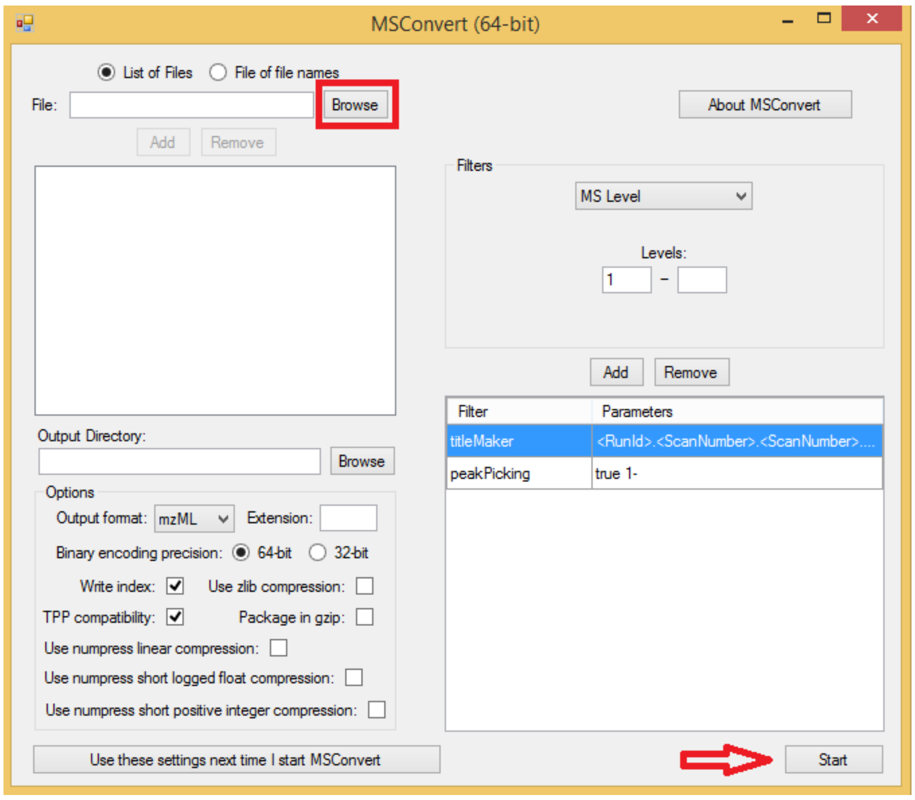
- OpenMS: https://www.openms.de/download/openms-binaries
- KNIME: https://www.knime.org/downloads/overview
- OpenMS prerequisites (Windows-only): After installation, before your first use of the OpenMS plugin in KNIME you will be asked to download it automatically if certain requirements are not found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

Choose the installers for the platform you are working on. We suggest using the full installers of KNIME. It includes all free community extensions so that you can skip the installation of the OpenMS plugin and most other dependencies for the example workflows later.

2.2 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 MSconvert, which is part of a ProteoWizard installation. All files used in this tutorial **have already been converted to mzML** by us, so you do not need to perform the data conversion yourself. However, we provide a small raw file so you can try the important step of raw data conversion for yourself.

> **Note:** The OpenMS installation package for Windows automatically installs ProteoWizard, so you do not need to download and install it separately. Due to restrictions from the instrument vendors, file format conversion for most formats is **only possible on Windows** systems. In practice, performing the conversion to mzML on the acquisition PC connected to the instrument is usually the most convenient option.



[✓ List of Files] File of file names

File:

MS Level

Levels: 1 –

Filter	Parameters
titleMaker	<RunId>.<ScanNumber>.<ScanNumber>....
peakPicking	true 1-

Output Directory:

Options

Output format: mzML Extension:

Binary encoding precision: [✓ 64-bit] 32-bit

Write index: ✓ Use zlib compression: ☐

TPP compatibility: ✓ Package in gzip: ☐

Use numpress linear compression: ☐

Use numpress short logged float compression: ☐

Use numpress short positive integer compression: ☐

Figure 1: MSConvertGUI (ProteoWizard), allows converting raw files to mzML. Select the raw files you want to convert by clicking on the browse button and then on "Add". Default parameters can usually be kept as-is. To reduce the initial data size, make sure that the peakPicking filter (converts profile data to centroided data) is listed, enabled (true) and applied to all MS levels (parameter "1-"). Start the conversion process by clicking on the "Start" button.

To convert raw data to mzML using ProteoWizard you can either use MSConvertGUI (a graphical user interface) or msconvert (a simple command line tool). Both tools are available in:
🗀`C: / Program Files / OpenMS-2.1.0 / share / OpenMS / THIRDPARTY / pwiz-bin`. You can find a small RAW file on the USB stick: 🗀`Example_Data`▸`Introduction`▸`datasets`▸`raw`.

2.2.1 MSConvertGUI

MSConvertGUI (see Figure 1) exposes the main parameters for data conversion in a convenient graphical user interface.

2.2.2 msconvert

The msconvert command line tool offers more options than the graphical tool MSConvertGUI. It allows converting large numbers of files and can be easier automatized.
To convert and pick the file raw_data_file.RAW you may write:

```
msconvert raw_data_file.RAW --filter "peakPicking true 1-"
```

in your command line.
To convert all RAW files in a folder may write:

```
msconvert *.RAW -o my_output_dir
```

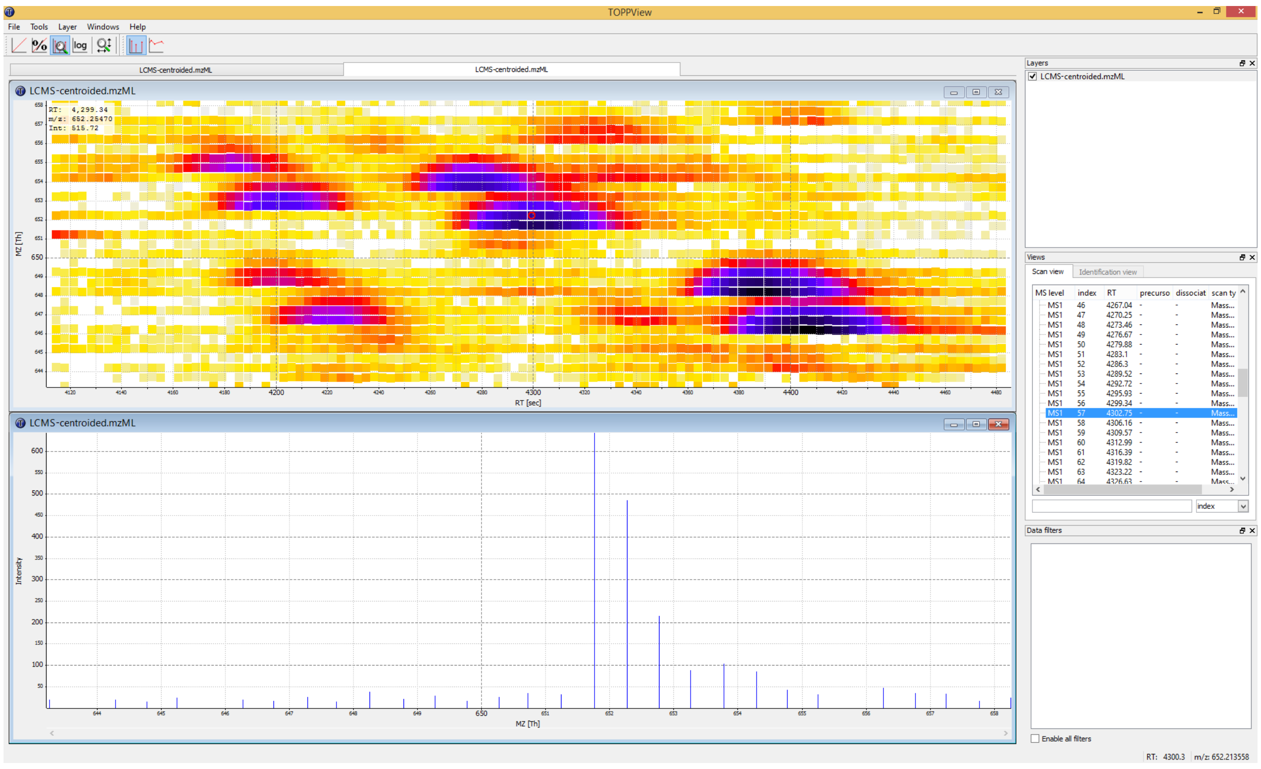
> **Note:** To display all options you may type `msconvert --help`. Additional information is available on the ProteoWizard web page.

2.3 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 Windows' Start-Menu or `Applications` ▸ `OpenMS` on macOS)
- Go to File › Open File, 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 Ok button.



MS level	index	RT	precurso	dissociat	scan ty
MS1	46	4267.04	-	-	Mass...
MS1	47	4270.25	-	-	Mass...
MS1	48	4273.46	-	-	Mass...
MS1	49	4276.67	-	-	Mass...
MS1	50	4279.88	-	-	Mass...
MS1	51	4283.1	-	-	Mass...
MS1	52	4286.3	-	-	Mass...
MS1	53	4289.52	-	-	Mass...
MS1	54	4292.72	-	-	Mass...
MS1	55	4295.93	-	-	Mass...
MS1	56	4299.34	-	-	Mass...
MS1	57	4302.75	-	-	Mass...
MS1	58	4306.16	-	-	Mass...
MS1	59	4309.57	-	-	Mass...
MS1	60	4312.99	-	-	Mass...
MS1	61	4316.39	-	-	Mass...
MS1	62	4319.82	-	-	Mass...
MS1	63	4323.22	-	-	Mass...
MS1	64	4326.63	-	-	Mass...

Figure 2: 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.

- Play around.
- Three basic modes allow you to interact with the displayed data: scrolling, zooming and measuring:
  - 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, you can scroll through the spectra. 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 backspace ⟵ zooms out to show the full LC-MS map (and also resets the 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.
- 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 macOS, 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.

2.4 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 parts of this tutorial you will use KNIME to create and execute workflows. The 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.

2.4.1 Plugin and dependency installation

> **Note:** If you chose the KNIME installer that contains all community contributions, or you received the full installer on our USB Stick you can skip this section. This is usually the preferred way but only works for stable releases that were available at the time the full release of KNIME was created. For bleeding edge functionality (as provided through the KNIME nightly community contributions or in the case we cover prereleases of OpenMS) you need to perform the actions below to download the plugin from official update sites.

Before we can start with the tutorial we need to install all the required extensions for KNIME. Since KNIME 3.2.1 the program automatically detects missing plugins when you open a workflow but to make sure that the right source for OpenMS is chosen, please follow the instructions here. 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 Help > Install New Software...
2. From the Work with: drop-down list select http://update.knime.org/analytics-platform/3.3
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) [Windows-only]
   - KNIME Report Designer
   - KNIME SVG Support
4. Click on "next" and follow the instructions (you may but don't need to restart KNIME now)
5. Click again on Help > Install New Software...
6. From the Work with: drop-down list select http://tech.knime.org/update/community-contributions/trusted/3.3
7. Now select the following plugin from the "KNIME Community Contributions - Cheminformatics" category
   - RDKit KNIME integration
8. Click on "next" and follow the instructions and after a restart of KNIME the dependencies will be installed.

You are now ready to install the OpenMS nodes.

- Open KNIME.

- Click on Help > Install New Software...

Now you need to decide which OpenMS nodes you want to install. You may choose between the stable, well-tested release or the unstable, nightly release with extended functionality.

Instructions for the stable release (recommended):

- From the Work with: drop-down list select the http://tech.knime.org/update/community-contributions/trusted/3.3
- Select the **OpenMS** nodes in the category:
  "KNIME Community Contributions - Bioinformatics & NGS" and click Next.
- Follow the instructions and after a restart of KNIME the OpenMS nodes will be available in the Node repository under "Community Nodes".

Instructions for the unstable, nightly release:

- In the now open dialog choose Add... (in the upper right corner of the dialog) to define a new update site. In the opening dialog enter the following details.
  Name: `Trunk Community Contributions`
  Location: http://tech.knime.org/update/community-contributions/trunk/
- After pressing OK KNIME will show you all the contents of the added Update Site.
- **Note:** From now on, you can use this repository for plugins in the Work with: drop-down list.
- Select the **OpenMS** nodes in the category:
  "KNIME Community Contributions - Bioinformatics & NGS" and click Next.
- Follow the instructions and after a restart of KNIME the OpenMS nodes will be available in the Node repository under "Community Nodes".

2.4.2 KNIME concepts

A **workflow** is a sequence of computational steps applied to a single or multiple input data 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 nodes, 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. The same holds for output ports, despite you can deactivate them in the configuration dialog (double-click on node) under the OutputTypes tab. After execution, deactivated ports will be marked with a red cross and downstream nodes will be inactive (not configurable).
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 typically longer for these types of nodes as they perform the main computations.
- Downstream statistical analysis and visualization. Here, tables are passed between nodes and mostly internal KNIME nodes or nodes from third-party statistics plugins are used. The transfer from files (produced by OpenMS) and tables usually happens with our provided Exporter and Reader nodes (e.g. MzTabExporter followed by MzTabReader).

Moreover, 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 fails, 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 above the traffic light. Most nodes will be configured as soon as all input ports are connected. Some nodes need to know about the output of the predecessor and may stay red until the predecessor was executed. If nodes still remain in a red state, probably additional parameters have to be provided in the configuration dialog that can neither be guessed from the data nor filled with sensible defaults. In this case, or if you want to customize the default configuration in general, 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 3.

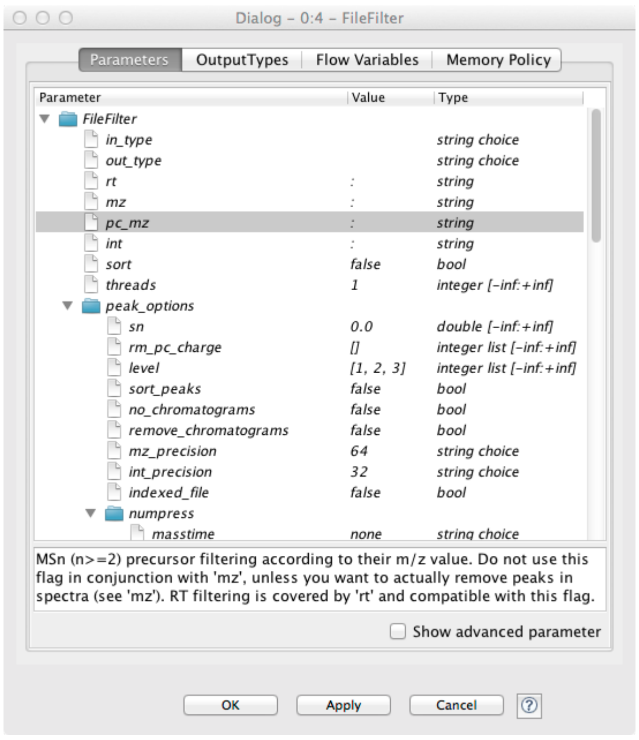
> **Note:** OpenMS distinguishes between normal parameters and advanced parameters. Advanced parameters are by default hidden from the users since they should only rarely be customized. In case you want to have a look at the parameters or need to customize them in one of the tutorials you can show them 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. Please also note the tabs on the top of the configuration dialog. In the case of OpenMS nodes, there will be another tab called OutputTypes. It contains dropdown menus for every output port that let you select the output filetype that you want the node to return (if the tool supports it). For optional output ports you can select Inactive such that the port is crossed out after execution and the associated generation of the file and possible additional computations are not performed. Note that this will deactivate potential downstream nodes connected to this port.

2.4.3 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 4. We will briefly introduce the individual panels and their 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”.



Parameter	Value	Type
FileFilter		
in_type		string choice
out_type		string choice
rt	:	string
mz	:	string
pc_mz	:	string
int	:	string
sort	false	bool
threads	1	integer [-inf:+inf]
peak_options		
sn	0.0	double [-inf:+inf]
rm_pc_charge	[]	integer list [-inf:+inf]
level	[1, 2, 3]	integer list [-inf:+inf]
sort_peaks	false	bool
no_chromatograms	false	bool
remove_chromatograms	false	bool
mz_precision	64	string choice
int_precision	32	string choice
indexed_file	false	bool
numpress		
masstime	none	string choice

MSn (n>=2) precursor filtering according to their m/z value. Do not use this flag in conjunction with 'mz', unless you want to actually remove peaks in spectra (see 'mz'). RT filtering is covered by 'rt' and compatible with this flag.

Figure 3: Node configuration dialog of an OpenMS node.

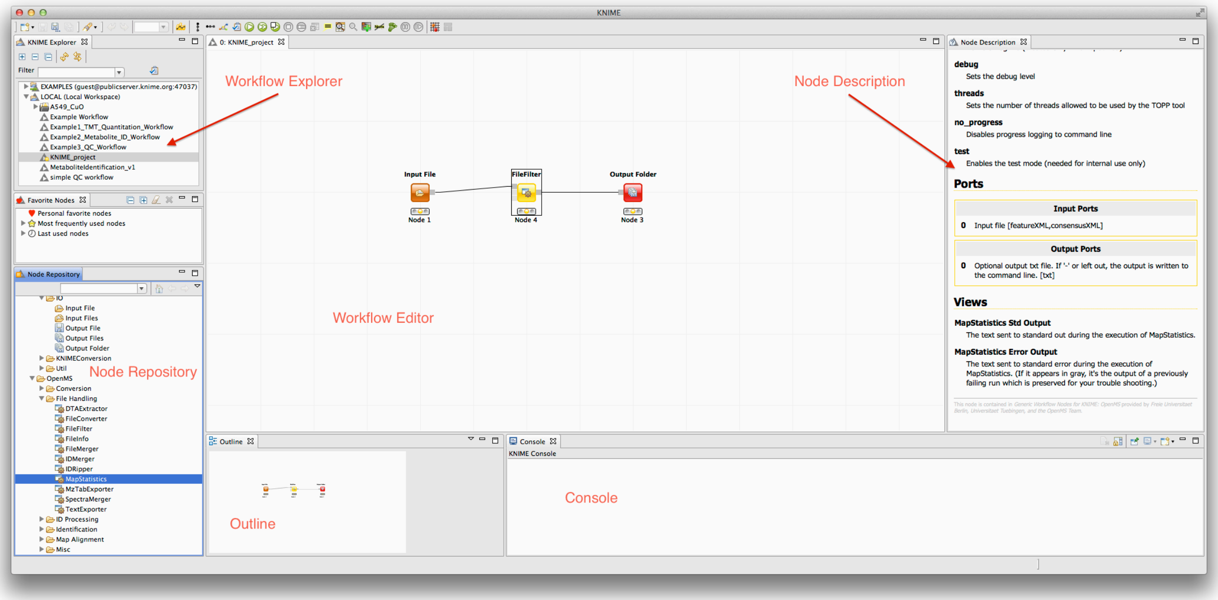


Figure 4: The KNIME workbench.

For quick creation of a workflow, note that double-clicking on a node in the repository automatically connects it to the selected node in the workbench (connecting all the inputs with as many fitting outputs of the last node). Manually, 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. Deletions are possible by selecting nodes and/or edges and pressing Del or (Fn+)Backspace depending on your OS and settings. Multiselection happens via dragging rectangles with the mouse or adding elements to the selection by clicking them while holding down Ctrl.

KNIME 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.

Workflow Coach (since KNIME 3.2.1): Shows a list of suggested following nodes, based on the last added/clicked nodes. When you are not sure which node to choose next, you have a reasonable suggestion based on other users behavior there. Connect them to the last node with a double-click.

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 > GenericKnimeNodes. 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. You can adjust the zoom level of the explorer by adjusting the percentage in the toolbar at the top of KNIME.

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 and especially their in- and outputs, such that you know what types of data nodes may produce or expect. For OpenMS nodes you will also find a link to the tool page of the online documentation.

2.4.4 Creating workflows

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

2.4.5 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 File > Export KNIME Workflow... . KNIME will export workflows as a knwf file containing all the information on nodes, their connections, and their parameter configuration. Those knwf files can again be imported by selecting File > Import KNIME Workflow... .

> **Note:** For your convenience we added all workflows discussed in this tutorial to the 🗀 `Workflows` folder on the USB Stick. Additionally, the workflow 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 knwf file.

2.4.6 Duplicating workflows

In this 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, save it, close it and 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, too.

2.4.7 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. This minimal workflow can also be used to check if all requirements are met and that your system is compatible.

- Create a new workflow.
- Add an `Input File` node and an `Output Folder` node (to be found in Community Nodes > GenericKnimeNodes > 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 5. 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.

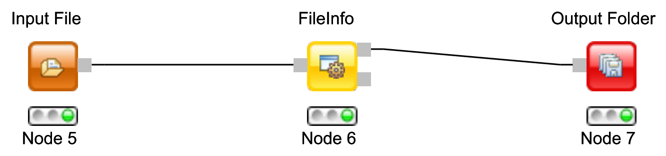


Figure 5: A minimal workflow calling FileInfo on a single file.

> **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 (shift key + F7; or the button with multiple green triangles in the KNIME Toolbar) 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.

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.

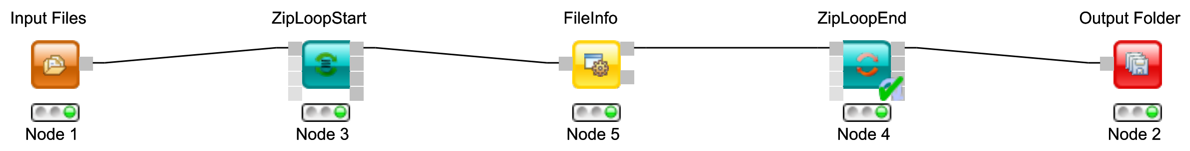


Figure 6: A minimal workflow calling FileInfo on multiple files in a loop.

- First we need to replace our single input file with multiple files. Therefore we add the `Input Files` node from the category Community Nodes › GenericKnimeNodes › 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 › GenericKnimeNodes › 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 6
- 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.
- 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.

2.4.8 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. **Pro-tip:** There is a Select Loop option when you right-click a node in a loop, that does exactly that for you. Then, 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

✓ Freeze/wrap the meta node to let it behave like an encapsulated single node. First select the `Meta Node`, open the context menu (right-click) and select Meta Node ⟩ Wrap. The differences between Meta Nodes and their wrapped counterparts are marginal (and only apply when exposing user inputs and workflow variables). Therefore we suggest to use standard meta nodes to clean up your workflow and cluster common subparts until you actually notice their limits.

Task

✓ Undo the packaging. First select the `(Wrapped) Meta Node`, open the context menu (right-click) and select (Wrapped) Meta Node ⟩ Expand.

2.4.9 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 four 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'." or "Execution failed: R Home is invalid.": 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 (e.g., `C:` ▸ `Program Files` ▸ `R` ▸ `R-3.3.1`).

If you get an error message like: "Execute failed: Could not find Rserve package. Please install it in your R installation by running "install.packages('Rserve')"." You may need to run your R binary as administrator (In windows explorer: right-click "Run as administrator") and enter install.packages('Rserve') to install the package.

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 of peptides

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 quantification workflow.

3.2 Peptide Identification

As a start, we will extend the minimal workflow so that it performs a peptide identification using the OMSSA [7] 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. [8]

- 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 2.4.6 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` node from Community Nodes > OpenMS > File Handling 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` node (from category 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 (all node results are discarded and need to be recalculated by executing the nodes again).

- 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 Add. Then select the newly added modification to open the drop down list where you can select the correct modification.

- A common step in analyis 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 Community Nodes > OpenMS > ID Processing. 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 decoy_string_position to prefix in the configuration dialog of `PeptideIndexer`.
- Now we can go for the FDR estimation, which the `FalseDiscoveryRate` node will calculate for us (you will find it in 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.

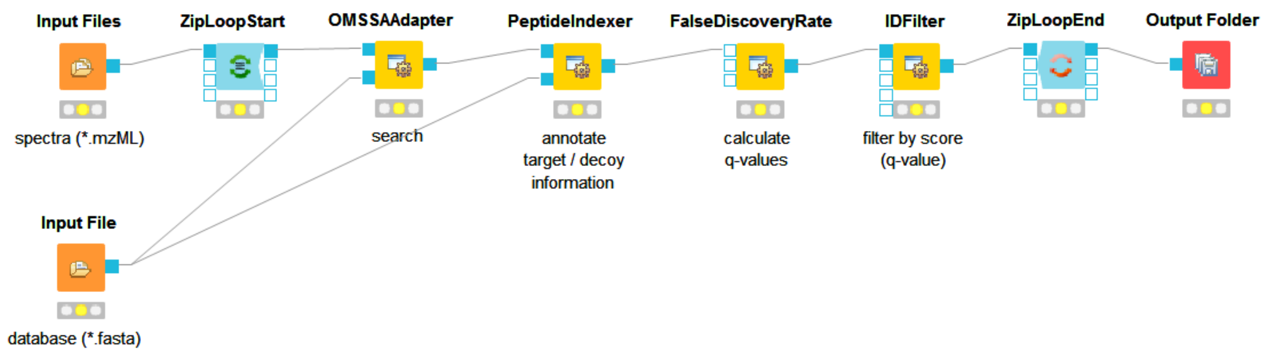


Figure 7: OMSSA ID pipeline including FDR filtering.

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` Community Nodes > OpenMS > Identification node and set its parameters and ports analogously to the `OMSSAAdapter`.
- To calculate the PEP, introduce each a `IDPosteriorErrorProbability` Community Nodes > OpenMS > ID Processing 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 Community Nodes > GenericKnimeNodes > Flow so we can then merge the corresponding IDs with a `IDMerger` Community Nodes > OpenMS > File Handling.
- Now we can create a consensus identification with the `ConsensusID` Community Nodes > OpenMS > ID Processing 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!Tandems non-tryptic identifications as well.

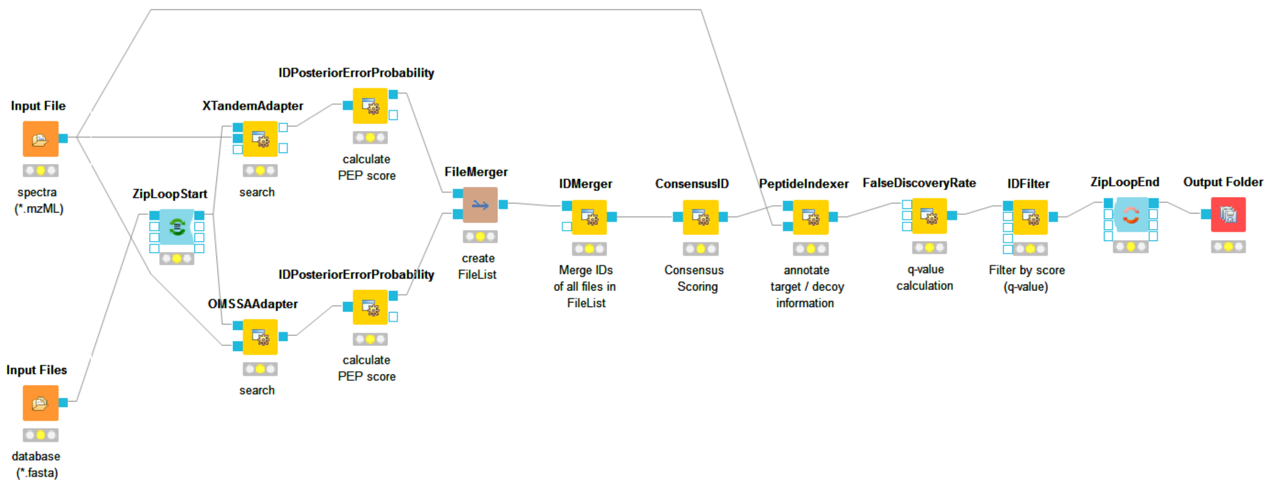


Figure 8: Complete consensus identification workflow.

3.3 Quantification

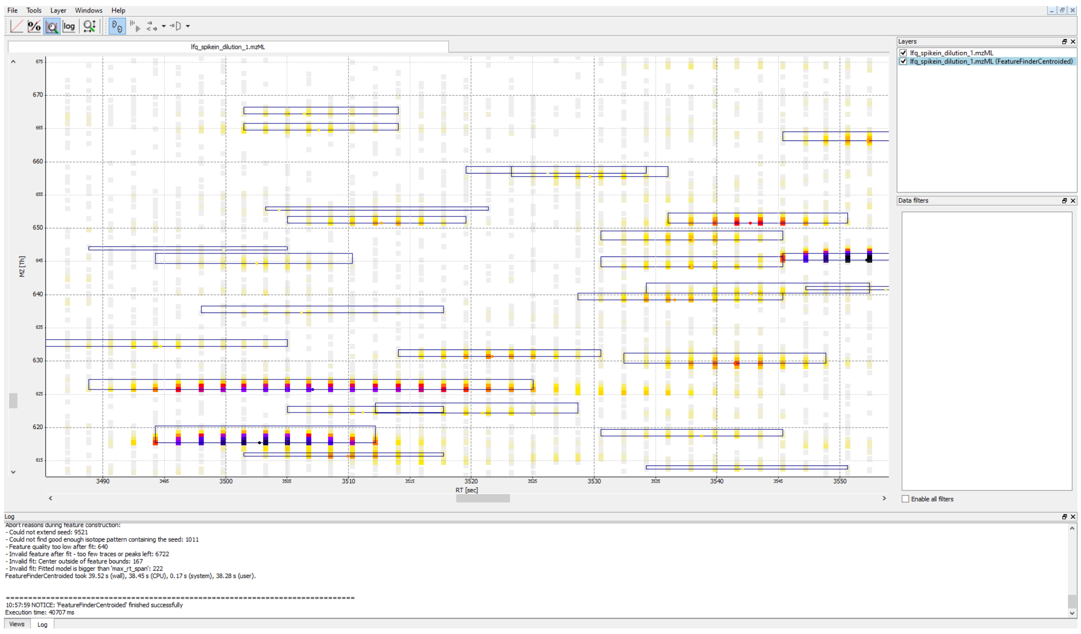
Now that we have successfully constructed a peptide identification pipeline, we can add quantification capabilities to our workflow.

- Add a `FeatureFinderCentroided` node from Community Nodes > OpenMS > Quantitation which gets input from the first output port of the `ZipLoopStart` node. Also, add an `IDMapper` node (from Community Nodes > OpenMS > ID Processing) which receives 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. Open the mzML file of your data to display the raw peak intensities.
- 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. To this end, open the featureXML file in TOPPView by clicking on File › Open file and add it to a new layer (Open in › New layer). The features are now visualized on top of your raw data. If you zoom in on a small region, you should be able to see the individual boxes around features that have been detected (see Figure 9). If you hover over the the feature centroid (small circle indicating the chromatographic apex of monoisotopic trace) additional information of the feature is displayed.


File
Tools
Layer
Windows
Help
lfq_spikein_dilution_1.mzML
Layers
lfq_spikein_dilution_1.mzML
lfq_spikein_dilution_1.mzML (FeatureFinderCentroided)
Data filters
Enable all filters
MZ [Th]
RT [sec]
Log
Views

```
Abort reasons during feature construction:
 - Could not extend seed: 9521
 - Could not find good enough isotope pattern containing the seed: 1011
 - Feature quality too low after fit: 640
 - Invalid feature after fit - too few traces or peaks left: 6722
 - Invalid fit: Center outside of feature bounds: 167
 - Invalid fit: Fitted model is bigger than 'max_rt_span': 222
FeatureFinderCentroided took 39.52 s (wall), 38.45 s (CPU), 0.17 s (system), 38.28 s (user).


================================================================================
10:57:59 NOTICE: 'FeatureFinderCentroided' finished successfully
Execution time: 40707 ms
```

Figure 9: Visualization of detected features (boxes) in TOPPView.

> **Note:** The chromatographic RT range of a feature is about 30-60 s and its m/z range around 2.5 m/z in this dataset. If you have trouble zooming in on a feature, select the full RT range and zoom only into the m/z dimension by holding down Ctrl (cmd ⌘ on macOS) 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`.

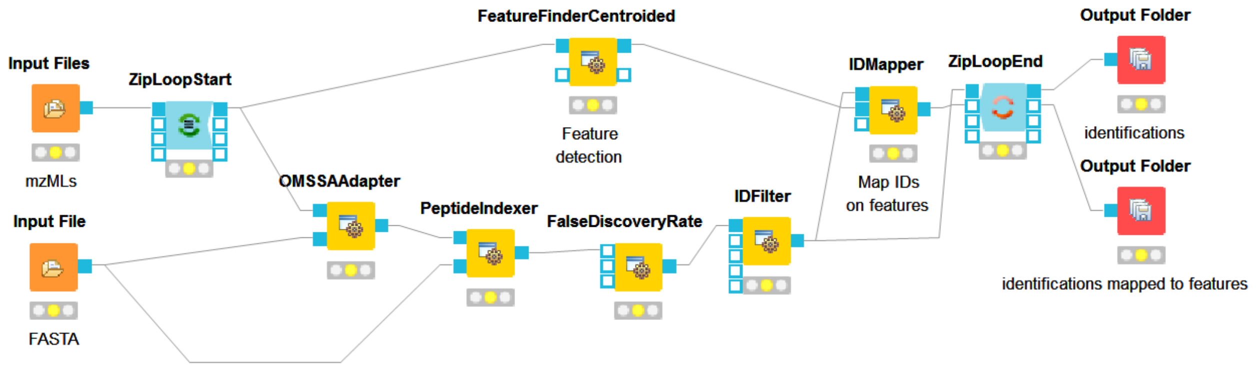


Figure 10: Extended workflow featuring peptide identification and quantification.

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` in 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.

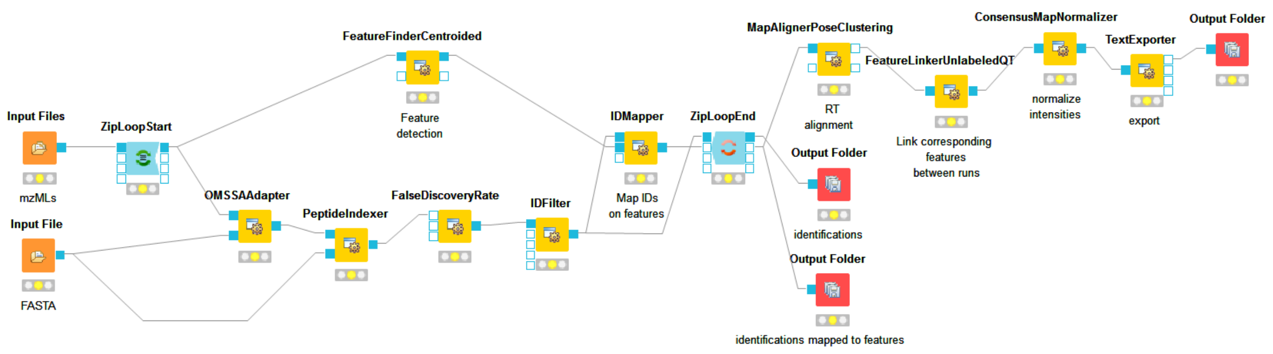


Figure 11: Complete identification and label-free quantification workflow.

> **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 in 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` in 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 in 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.

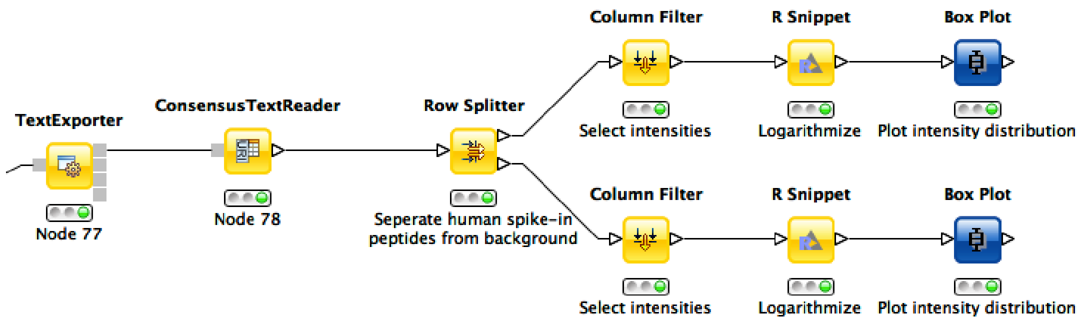


Figure 12: Simple KNIME data analysis example for LFQ.

- 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. 12, 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 apply the 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 only contain 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 Data Manipulation > Column > Filter, 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 R 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 Views 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. 12. 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 > labelfree_with_protein_quant` into KNIME via File > Import KNIME workflow > Select 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 by executing all nodes at once.

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. In fact, we also used a greedy method in FidoAdapter (parameter `greedy_group_resolut` to uniquely assign the peptides of a group to the most probable protein(s) in the respective group. This boosts the number of quantifications but slightly raises the chances to yield distorted protein quantities.

- As a prerequisite for using FidoAdapter, we have added an `IDPosteriorErrorProbability` node within the `ID` meta node, between the XTandemAdapter (note the replacement of OMSSA because of ill-calibrated scores) 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. Also note that we excluded FDR filtering from the standard meta node. Harsh filtering before inference impacts the calibration of the results. Since we filter peptides before quantification though, no potentially random peptides will be included in the results anyway.

- 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. This is done to get a unique assignment of peptides to proteins over all samples.

- 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 (e.g. (pseudo) receiver operating curves; ROCs) of the protein inference results using some of the more advanced KNIME and R nodes. The meta node also shows how to use MzTabExporter and MzTabReader.

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.

- We use the `MzTabExporter` to convert the inference results from `FidoAdapter` to a human readable, tab-separated mzTab file. mzTab contains multiple sections, that are all exported by default, if applicable. This file, with its different sections can again be read by the `MzTabReader` that produces one output in KNIME table format (triangle ports) for each section. Some ports might be empty if a section did not exist. Of course, we continue by connecting the downstream nodes with the protein section output (second port).

- Since the protein section contains single proteins as well as protein groups, we filter them for single proteins with the standard `Row Filter`.

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. When configuring this node, select the opt_global_target_decoy column as the class (i.e. target outcome) column. We want to find out, how good our inferred protein probability discriminates between them, therefore add best_search_engine_score[1] (the inference engine score is treated like a peptide search engine score) to the list of "Columns containing positive class probabilities". View the plot by right-clicking and selecting View: ROC Curves. A perfect classifier has an area under the curve (AUC) of 1.0 and its curve touches the upper left of the plot. However, 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) or its protein-centric counterpart, the q-value. 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 have a 5% probability of being false. This implies that the estimated number of false positives can be computed as the sum of posterior error probabilities (= 1 - posterior probability) in a set, divided by the number of proteins in the set. Thereby a posterior-probability-estimated FDR is computed which can be compared to the actual target-decoy FDR. 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, although a non-correspondence does not necessarily indicate wrong results.

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 for each protein: the q-value of a certain protein ID is the minimum FDR 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 14).

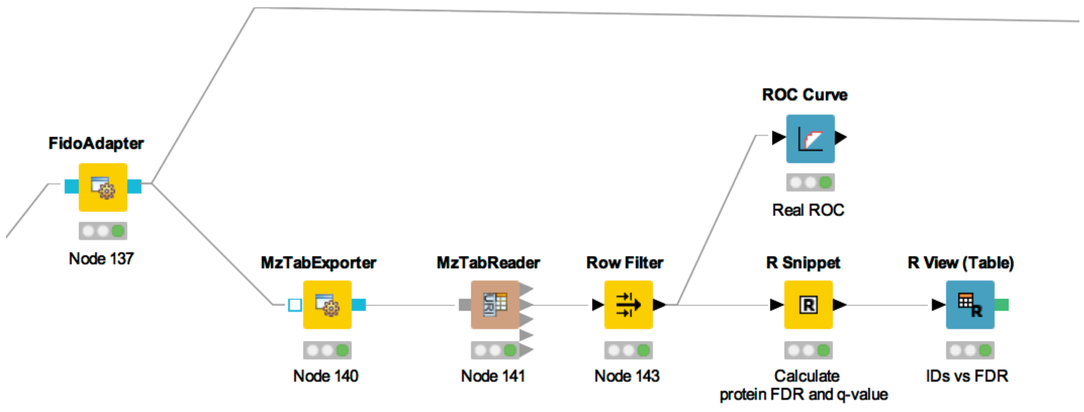


Figure 13: The workflow of statistical analysis of protein inference results

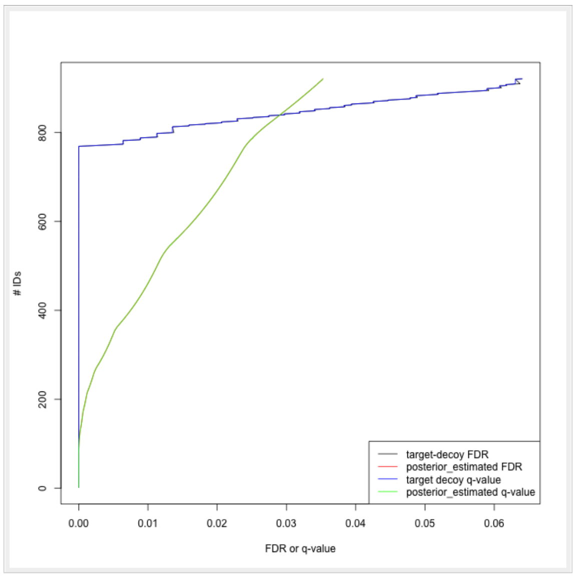


Figure 14: the pseudo-ROC Curve of protein IDs. The accumulated number of protein IDs is plotted on two kinds of scales: target-decoy protein FDR and Fido posterior probability estimated FDR. The largest value of posterior probability estimated FDR is already smaller than 0.04, this is because the posterior probability output from Fido is generally very high.

5 Label-free quantification of metabolites

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**
algorithm → common → chrom_fwhm	8.0
algorithm → mtd → trace_termination_criterion	sample_rate
algorithm → mtd → min_trace_length	3.0
algorithm → mtd → max_trace_length	600.0
algorithm → epd → width_filtering	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**
algorithm → max_num_peaks_considered	-1
algorithm → superimposer → mz_pair_max_distance	0.005
algorithm → superimposer → num_used_points	10000
algorithm → pairfinder → distance_RT → max_difference	20.0
algorithm → pairfinder → distance_MZ → max_difference	20.0
algorithm → pairfinder → distance_MZ → unit	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
algorithm → distance_RT → max_difference	40.0
algorithm → distance_MZ → max_difference	20.0
algorithm → distance_MZ → unit	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 15. In the following section we will try to identify those metabolites.

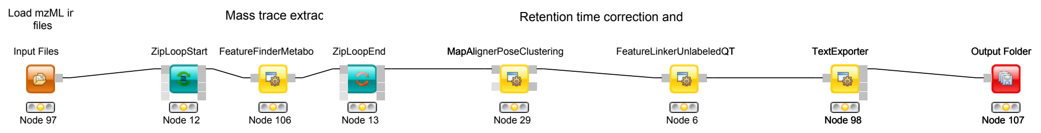


Figure 15: Label-free quantification workflow for metabolites

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)[9, 10, 11]. We start with the workflow from the previous section (see Figure 15).

- 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 semicolon (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 [12] 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 16.

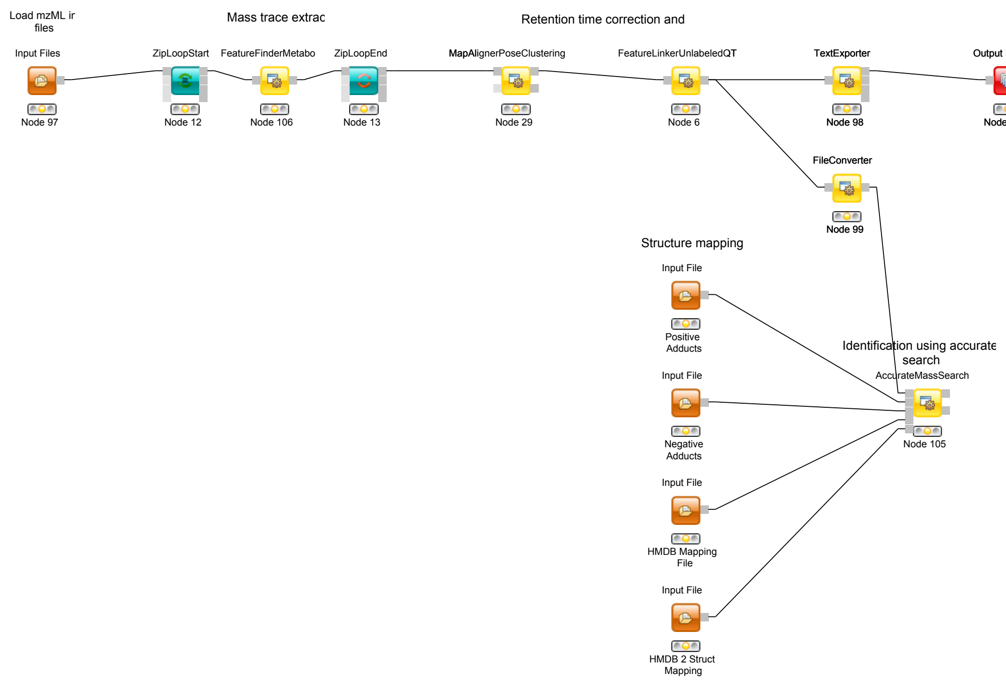


Figure 16: Label-free quantification and identification workflow for metabolites

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 17.

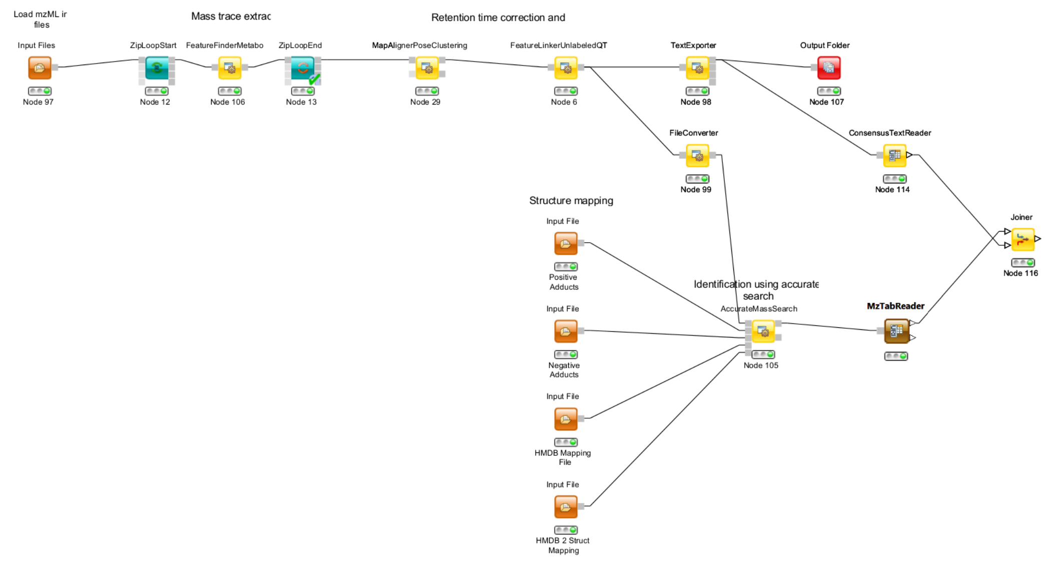


Figure 17: Label-free quantification and identification workflow for metabolites that loads the results into KNIME and joins the tables.

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.knwf` in KNIME: File > Import KNIME Workflow...

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 for identification

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

5.5.2 Data preparation for quantification

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(knime.in) or cat(knime.in) 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 (grepl) 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 Eval script 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 libary, which is by default part of the R Windows binaries (for full KNIME version 3.1.1 or higher). 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.

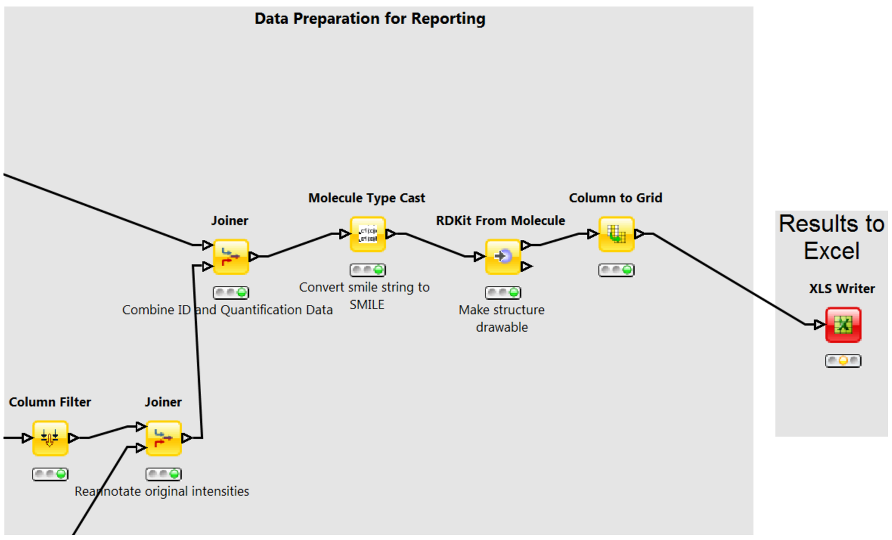


Figure 18: Data preparation for reporting

Question

What happens if we use a 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`).

5.6 Spectral library search

Identifying metabolites using only the accurate mass may result in ambiguous results. In practice, additional information (e.g., the retention time) is used to further narrow down potential candidates. Apart from MS1-based features, tandem mass spectra (MS2) of metabolites provide additional information. In this part of the tutorial, we take a look on how metabolite spectra can be identified using a library of previously identified spectra.

Because these libraries tend to be large we don't distribute them with OpenMS.

Task

☑ Please copy the spectral library from
`Example_Data▸Metabolomics▸databases▸MetaboliteSpectralDB.mzML`
on the USB stick into the following directory of your KNIME installation (version numbers and dates may differ slightly):

- Windows
  `C:▸Program Files▸KNIME3.3▸plugins▸de.openms.win32.x86_64_2.1.0.201611191105▸payload▸share▸OpenMS▸CHEMISTRY`

- macOS
  🗀 Applications ▸ KNIME 3.3.app ▸ Contents ▸ Eclipse ▸ plugins ▸ de.openms.macosx.x86_64_2.1.0.201611191105 ▸ payload ▸ share ▸ OpenMS ▸ CHEMISTRY

Now construct the workflow as shown in Figure 19. Use the file 🗀 Example_Data ▸ Metabolomics ▸ datasets ▸ Metabolite_ID_SpectraDB_positive.mzML as input for your workflow. It contains tandem spectra that are identified by the MetaboliteSpectralMatcher. The resulting mzTab file is read back into a KNIME table and stored in an Excel table. Make sure that you connect the MzTabReader port corresponding to the Small Molecule Section to the Excel writer (XLS)).

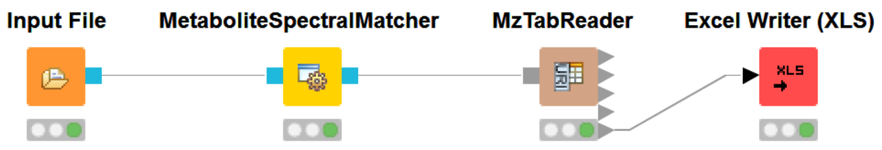


Figure 19: Spectral library identification workflow

Run the workflow and inspect the output.

5.6.1 Manual validation

In metabolomics, matches between tandem spectra and spectral libraries are manually validated. Several commercial and free online resources exist which help in that task. Some examples are:

- mzCloud contains only spectra from Thermo Orbitrap instruments. The webpage requires Microsoft Silverlight which currently does not work in modern browsers (see https://www.mzcloud.org/DataViewer).
- MassBank North America (MoNA) has spectra from different instruments but falls short in in number of spectra (compared to Metlin and mzCloud) http://mona.fiehnlab.ucdavis.edu/spectra/display/KNA00122

- METLIN includes 961,829 molecules ranging from lipids, steroids, metabolites, small peptides, carbohydrates, exogenous drugs and toxicants. In total over 14,000 metabolites.

Here, we will use METLIN to manually validate metabolites.

Task

✓ Open the mzML file in TOPPView and inspect some of your top hits. Select the tandem spectrum of Glutathione, but do not close TOPPView, yet.

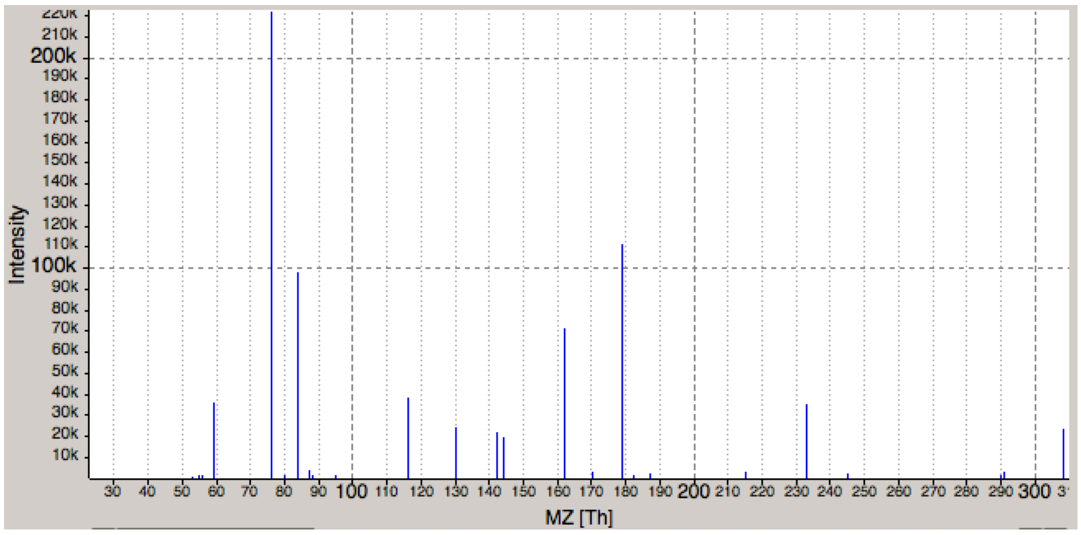


Figure 20: Tandem spectrum of glutathione. Visualized in TOPPView.

Task

✓ On the METLIN homepage search for Compound Name Glutathione using the Advanced Search (`https://metlin.scripps.edu/landing_page.php?pgcontent=advanced_search`). Note that free registration is required. Which collision energy (and polarity) gives the best (visual) match to your experimental spectrum in TOPPView?

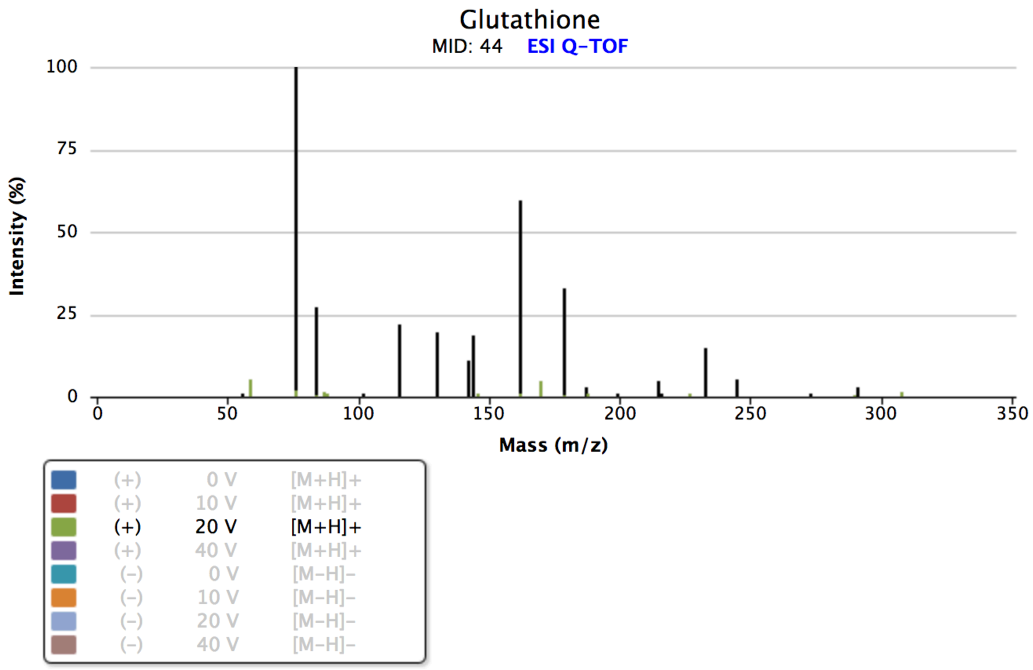


Figure 21: Tandem spectrum of glutathione. Visualized in Metlin. Note that several fragment spectra from varying collision energies are available.

6 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. If that is the case, please provide us with as much information as you can. In an ideal case, that would be:

- Your operating system and its version (e.g. Windows 8, Ubuntu 14.04)
- Your KNIME version (e.g. KNIME 3.1.2 full, KNIME 3.1.1 core)
- If not full: Which update site did you use for the OpenMS plugin? Trunk (nightly-builds) or Stable?
- Your OpenMS plugin version found under Help > Install New Software > What is already installed?
- Other installations of OpenMS on your computer (e.g. from the independent OpenMS installer, another KNIME instance etc.)
- The log of the error in KNIME and the standard output of the tool (see FAQ: How to debug)
- Your description of what you tried to do and experienced instead

6.1 FAQ

6.1.1 How to debug KNIME and/or the OpenMS nodes?

- **KNIME:** Start with the normal log on the bottom right of KNIME. In general all warnings and errors will be listed there. If the output is not helpful enough, try to set the logging verbosity to the highest (DEBUG) under Preferences -> KNIME -> Log file log level.
- **OpenMS nodes:** The first step should also be the log of KNIME. You can view the output and the errors of our tools by right-clicking on the node and selecting

View: NODENAME Std Output/Error. This shows you the output of the OpenMS executable that was called by that node. For advanced users, you can try to execute the underlying executable in your
KNIME/plugins/de.openms.platform.arch.version/payload/bin folder, to see if the error is reproducible outside of KNIME.
You can look up temporary files that are created by OpenMS nodes not connected to an Output or Viewer Node by right- clicking on a node and selecting the corresponding output view for the output you want to have a look at. The output views are located on the bottom of the menu that shows up after right-clicking. Their icon is a magnifying glass on top of a data table. The names of the output views in that menu may vary from node to node (usually a combination of "file","out","output" and optionally its possible extensions). For example for the Input File node you can open the information on the output files by clicking on "loaded file". In any case, a hierarchy of file descriptions will show up. If there are multiple files on that port they will be numbered (usually beginning from 0). Expand the information for the file you want to see and copy its URI (you might need to erase the "file:" prefix). Now open it with an editor of your choice. Be aware that temporary files are subject to deletion and are usually only stored as long as they are actually needed. There is also a Debug mode for the GKN nodes that keeps temporary files that can be activated under Preferences -> KNIME -> Generic KNIME Nodes -> Debug mode. For the single nodes you can also increase the debug level in the configuration dialog under the advanced parameters. You can also specify a log file there, to save the log output of a specific node on your file system.

6.1.2 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.

Q: After exporting and reading in results into a KNIME table (e.g. with a MzTabExporter and MzTabReader combination) numeric values get rounded (e.g. from scientific notation 4.5e-10 to zero) or are in a different representation than in the underlying exported file!
A: Please try a different table column renderer in KNIME. Open the table in question, right-click on the header of an affected column and select another Available Renderer by hovering and finally left-clicking.

Q: I have checked all the configurations but KNIME complains that it can not find certain output Files (FileStoreObjects).
A: Sometimes KNIME/GKN has hiccups with multiple nodes with a same name, executed at the same time in the same loop. We have seen that a simple save and restart of KNIME usually solves the problem.

6.1.3 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. In short: set environment variable by calling `export SWT_GTK3=0` or edit knime.ini to make Eclipse use GTK2 by adding the following two lines:

```
-launcher.GTK_version
2
```

macOS

Q: I have problems installing RServe in my local R installation for the R KNIME Extension:
A: If you encounter linker errors while running install.packages("Rserve") when using an R installation from homebrew, make sure gettext is installed via homebrew and you pass flags to its lib directory. See StackOverflow question 21370363.

Q: Although I Ctrl+Leftclick TOPPAS.app or TOPPView.app and accept the risk of a downloaded application, the icon only shortly blinks and nothing happens:
A: It seems like your OS is not able to remove the quarantine flag. If you trust us, please remove it yourself by typing the following command in your Terminal.app:

```
xattr -r -d com.apple.quarantine /Applications/OpenMS-2.1.0
```

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 (potentially also 12.0) yourself.

6.1.4 Nodes

Q: Why is my XTandemAdapter printing empty or VERY few results, although I did not use an e-value cutoff?
A: Due to a bug in 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. This should be resolved in newer versions though, such that it automatically uses this file if the optional inputs is empty.

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. In rare cases they might require changes to the configuration under which your Java VM is running. Also MSGFPlus is creating several auxiliary files and accesses them during execution. Some users therefore experienced problems when executing several instances at the same time.

6.2 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]. 5

[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. 5

[3] H. L. Röst, T. Sachsenberg, S. Aiche, C. Bielow, H. Weisser, F. Aicheler, S. Andreotti, H.-C. Ehrlich, P. Gutenbrunner, E. Kenar, et al., OpenMS: a flexible open-source software platform for mass spectrometry data analysis, Nature Methods **13**(9), 741–748 (2016). 5

[4] 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). 5

[5] M. R. Berthold, N. Cebron, F. Dill, T. R. Gabriel, T. Kötter, T. Meinl, 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. 5

[6] 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. 5

[7] 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). 26

[8] 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. 26

[9] 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. 47

[10] 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. 47

[11] 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. 47

[12] 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.O113.036681. 48