5 Metabolomics

5.1 Introduction

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

5.2 Quantifying metabolites across several experiments

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

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

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

parameter	**value**
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 $\rightarrow$ *distance_RT* $\rightarrow$ *max_difference*	40.0
algorithm $\rightarrow$ *distance_MZ* $\rightarrow$ *max_difference*	20.0
algorithm $\rightarrow$ *distance_MZ* $\rightarrow$ *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 13. In the following section we will try to identify those metabolites.

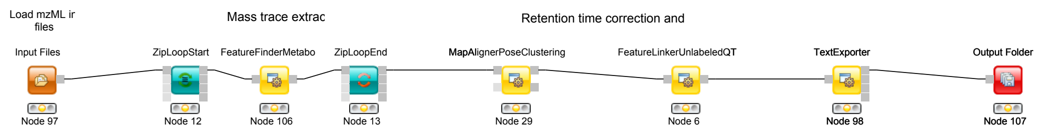

Figure 13: Label-free quantification workflow for metabolites

5.3 Identifying metabolites in LC-MS/MS samples

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

- Add a `FileConverter` node (Community Nodes › OpenMS › File Handling) and connect the output of the `FeatureLinkerUnlabeledQT` to the incoming port.
- Open the Configure dialog of the `FileConverter` and select the tab "OutputTypes". In the drop down list for FileConverter.1.out select "featureXML".
- Add an `AccurateMassSearch` node (Community Nodes › OpenMS › Utilities) and connect the output of the `FileConverter` to the first port of the `AccurateMassSearch`.
- Add four `Input File` nodes and configure them with the following files
  - `Example_Data▸Metabolomics▸databases▸PositiveAdducts.tsv`
    This file specifies the list of adducts that are considered in the positive mode. Each line contains the formula and charge of an adduct separated by a 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 [11] so you can easily open it in a text editor or import it into Excel or KNIME, which we will do in the next section. The complete workflow from this section is shown in Figure 14.

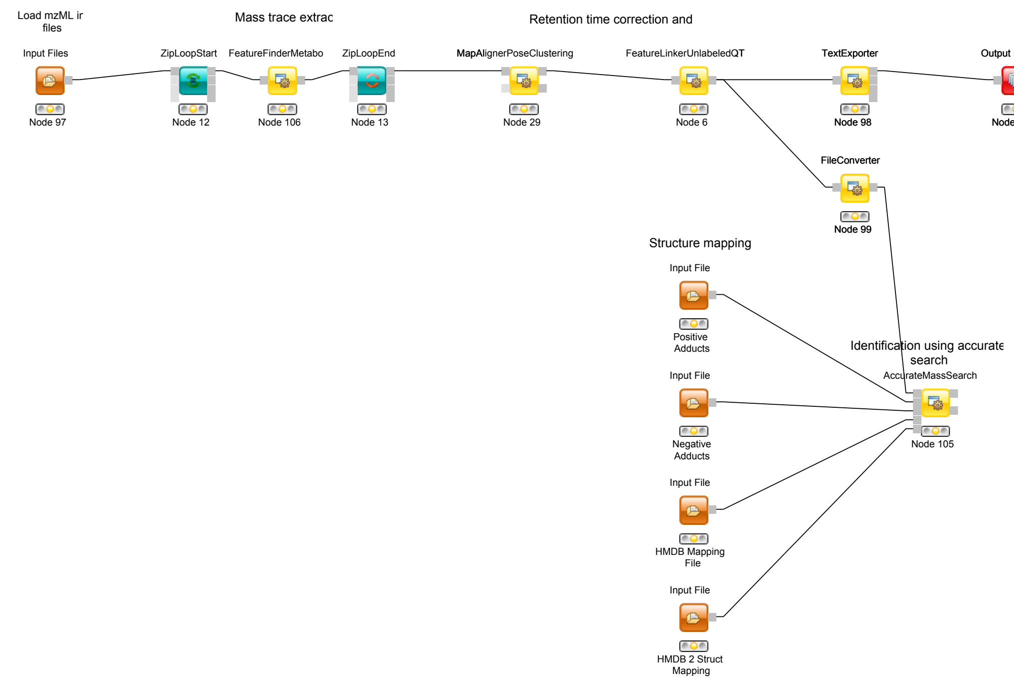

Figure 14: Label-free quantification and identification workflow for metabolites

5.4 Convert your data into a KNIME table

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

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

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

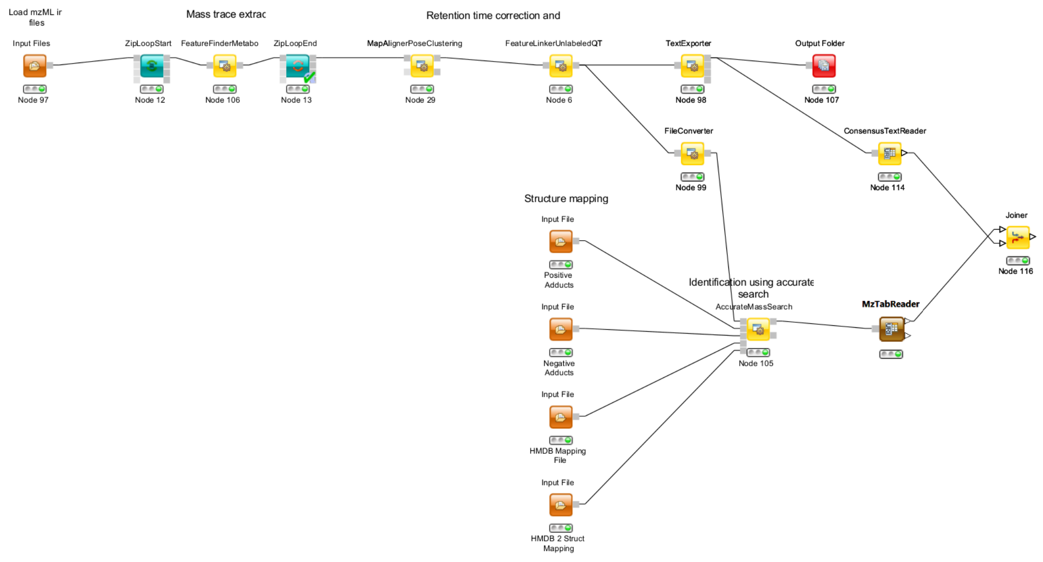

Figure 15: Label-free quantification and identification workflow for metabolites that loads the results into KNIME and joins the tables.

5.4.1 Bonus task: Visualizing data

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

Task

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

Task

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

Task

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

5.5 Downstream data analysis and reporting

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

Task

✓ Import the workflow from `Workflows` ▸ `metabolite_ID.zip` in KNIME: 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 ID

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

5.5.2 Data preparation Quant

The first part is identical to what you did for the simple metabolomics pipeline. Additionally, we convert zero intensities into NA values and remove all rows that contain at least one NA value from the analysis. We do this using a very simple `R Snippet` and subsequent `Missing Value filter` node.

Task

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

5.5.3 Statistical analysis

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

- Double-click on the first `R Snippet` node labeled "log scaling" to open the `R Snippet` dialog. In the middle you will see a short R script that performs the log scaling. To perform the log scaling we use a so-called regular expression (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:

```r
#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 the KNIME 3.1.1 full installation. We again use regular expressions to extract all measured intensity columns for plotting. For clarity, feature names are only shown in the heatmap after filtering by fold changes.

5.5.6 Data preparation for Reporting

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

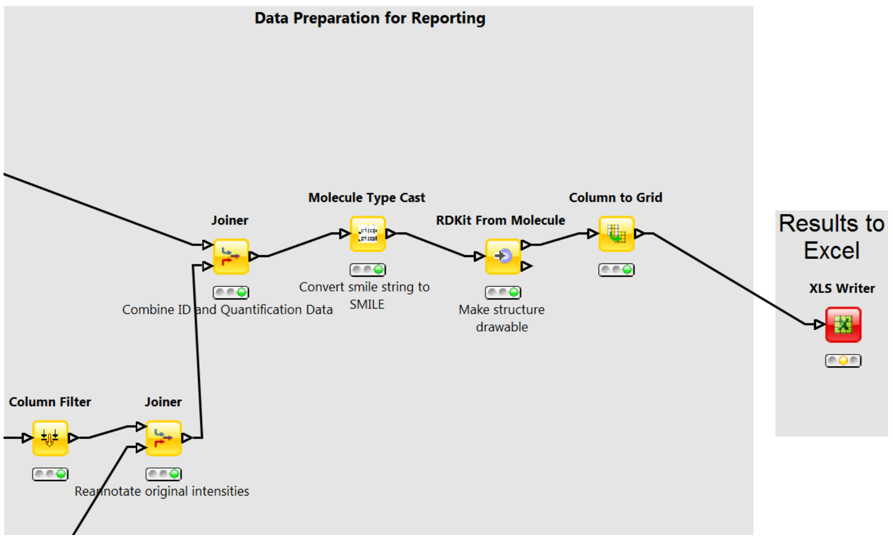

Figure 16: Data preparation for reporting

Question

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

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

Task

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

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