
untargeted-metabolomics-pipeline

Documentation

Release 0

Claire Duvallet

Aug 02, 2018

Contents:

1	Preparing your dataset for processing	3
2	Data processing code	7
3	Processing your data	9
4	Common confusing errors	11
5	Indices and tables	13

This documentation describes the process of going from open-source mzML mass spectrometry files to aligned feature table(s). This is orchestrated by the script *raw2feats.py*. Each dataset folder must contain a machine-readable text file called a summary file which gives instructions to *raw2feats.py*. The format of this summary file, and all files required for MS processing, are described in this documentation.

Preparing your dataset for processing

You should create a directory where you put all of the data and associated files described below.

1.1 Raw data

The pipeline does not currently support raw data, and begins instead with open-sourced mzML files.

1.2 mzML files

Currently, you need to convert your raw data files into mzML files manually. Use MSConvert for this. Each file should be run through MSConvert twice: once with a threshold of 1000, (named with the suffix `*.threshold1000.mzML`), and once with no threshold (named `*.mzML`). Negative mode peak-picking works better with the thresholded files, whereas positive-mode and MS2 processing don't need any of the pre-thresholding by MSConvert.

Note that the associated [github repository](#) contains a helper script, `msconvert_wrapper.py` which you can use to easily convert raw data into mzML files. Note that some raw data formats (e.g. Thermo Fisher) can only be converted with the Windows version of MSConvert. If this is your case, we recommend installing [Cygwin](#) and using `msconvert_wrapper.py` from that command line.

1.3 Sequence file

The sequence file should be provided to you by whoever ran the samples on the machine. The sequence file is essentially a mapping file containing the names of each data file and its corresponding metadata. Some metadata that is often present includes: instrument method, file path, injection volume, etc.

The required parts of the sequence file are as follows:

- **Sample ID:** The first column in your sequence file should be the sample IDs. This is how each sample will be labeled in all downstream processing.

- **File Name:** This column should contain the raw data file name (without any extensions). The processing code assumes that the mzML files are created from these file names. For example, if you have `mtab_alm_sample1` in this column, the code assumes that the corresponding mzML files are `mtab_alm_sample1.threshold1000.mzML` and `mtab_alm_sample1.mzML`.
- **Ion Mode:** This column contains the ion mode used for each sample. Accepted values are `negative` and `positive`.
- **Batches:** This column specifies the “batches” of samples you want to align. When aligning the picked features to create an aligned feature table, you may only want to consider a subset of your samples (e.g. all PPL samples in one batch, all direct injection samples in another). Each sample can be in many (or no) batches. If a sample is in multiple batches, the batch names should be comma-separated within the same cell. The order of batches in a cell doesn’t matter, but the case does. If a batch contains samples from multiple ionization modes, that batch will not be aligned. Each batch yields an aligned feature table with only the samples in that batch aligned.

The first row of the sequence file should contain at least the following case-insensitive column headers: `SampleID`, `File Name`, `Ion Mode`, and `batches`. If there is an additional line at the top of the file (above the column headers), delete it before providing to the pipeline. The sequence file is assumed to be comma-separated, but the delimitation can be specified in the summary file with the attribute `SEQUENCE_FILE_SEPARATOR`. To specify a different sequence file delimiter, include the Pythonic string representation of the separator. For example, a tab-delimited sequence file would have a `SEQUENCE_FILE_SEPARATOR` of `\t`.

1.3.1 Summary File

Once you have your data and sequence file all sorted, you need to create a summary file that “talks” to `raw2feats.py` (through the `SummaryParserMtab.py` module). Your summary file should be a tab-delimited file named `summary_file.txt` and placed in the same directory as your sequence file.

Required attributes

The following attributes are required to be specified in `summary_file.txt`:

<p><code>DATASET_ID</code> <code>MODE</code> <code>SEQUENCE_FILE</code></p>	<p>Identifier for this processing run. Output files will contain this string as an identifier.</p>
	<p>Ionization mode. Accepted values are <code>negative</code> or <code>positive</code>. If you have both positive and negative mode files to process, you will need to do them in two separate runs.</p>
	<p>Name of sequence file. Full path is not necessary, as sequence file is assumed to (and should) be in the same directory as the summary file.</p>

Optional attributes

The following attributes can be specified in the summary file, but are not required for processing:

DATA_DIRECTORY	If the data is in a different directory, you can provide the full path to the directory here. Otherwise the code assumes that all of your mzML files are in the input directory.
SEQUENCE_FILE	The sequence file is assumed to be comma-delimited. If this is not the case, specify the delimitation here. (i.e. if your sequence file is tab-delimited, this should be <code>\t</code>).
RIMAGE	If peak-picking has already been performed on this dataset, you may provide the full path to an Rimage file containing this results to load up and skip the peak-picking. This Rimage should contain an <code>xcmsSet</code> object named <code>xs</code> . If no Rimage file is specified, this attribute will be updated with the correct file once peak-picking has been run once, so you may re-use this summary file to run different alignments without necessarily re-picking peaks.
RAW_DATA	True if you are providing raw data that needs to be converted to mzML, False if you are directly providing mzML files. Note: the current code does NOT accept raw data. This functionality may be added in future iterations.

Metabolomics summary file attributes should be located between `#mtab_start` and `#mtab_end` in the summary file.

Sample Summary File

Note: all blank spaces are tab characters.

```

DATASET_ID test

#mtab_start
MODE negative
SEQUENCE_FILE test_sequence_file.csv
#mtab_end

```

Sample Summary File, with optional attributes

```

DATASET_ID test

#mtab_start
MODE negative
SEQUENCE_FILE testsequencefile.csv
RIMAGE full/path/to/file.Rimage
SEQUENCE_FILE_DELIMITER \t
#mtab_end

```

Data processing code

2.1 raw2feats.py

`raw2feats.py` is the main workhorse of the MS processing code. It reads in the summary file, picks peaks (if applicable), and aligns peaks for all batches specified in the sequence file. It basically coordinates all of the inputs and outputs and calls wrappers to R functions as necessary. Most of the functions that actually do work (i.e. pick and align peaks) are found in `preprocessing_mtab.py`.

2.2 Parsing the summary file

The `SummaryParserMtab.py` module simply reads in `summary_file.txt` and stores its attributes in a dictionary. `SummaryParserMtab.py` looks for the `summary_file.txt` in the input directory.

2.3 Picking peaks

If an Rimage file is specified in `summary_file.txt`, this part is skipped. If not, `raw2feats.py` calls the `pick_peaks` function in the `preprocessing_mtab.py` module. `pick_peaks()` calls `pick_peaks.R` and saves the PDF and Rimage files resulting from the call to `xcmsSet`. The Rimage file contains an `xcmsSet` object called `xs`. Once peaks are picked, `summary_file.txt` is updated with the respective RIMAGE file so that future processing calls skip the time-consuming peak picking step and go straight to aligning.

2.4 Aligning peaks

After peaks are picked, `raw2feats.py` reads in all of the specified batches in the `batches` column in the sequence file. One sample may be in multiple batches - batch names should be comma-separated in the sample's cell in the `batches` column. If a batch contains samples of multiple ionization modes, that batch is thrown out and never processed.

`align_peaks.R` first loads in the Rimage file that was either specified in the summary file or created by picking peaks. It identifies which samples to align and uses the `xcms` functions to align peaks across samples, group these peaks together, and fill in any peaks that weren't found in individual samples but are considered real peaks in some other samples. `align_peaks.R` then finds isotopes and adducts (for the specified mode) using `CAMERA`.

Back in `preprocessing_mtab.py`, the sample IDs in the aligned table, which are currently the mzML file names, are replaced by their sample ID in the sequence file.

Processing your data

3.1 Running `raw2feats.py`

To process your data, navigate to the directory containing your summary and sequence files using the command line. If this directory does not also contain your data, make sure to specify that in the `DATA_DIRECTORY` attribute in the summary file. Then, type

```
python /path/to/raw2feats.py -i /path/to/input/directory
```

Alternatively, if you want to specify a different output directory than the default, you can type

```
python /path/to/raw2feats.py -i /path/to/input/directory -o /full/path/to/output/  
↪directory
```

Note that you must have `raw2feats.py`, `SummaryParserMtab.py`, and `preprocessingmtab.py` all added to your `PYTHONPATH` or in the same folder.

3.2 Knowing when it's finished

When the data is finished processing, you should see `[[Processing mzML files]] Done.` print to screen.

3.3 Saving the outputs to files

Using Windows DOS command line, you can save the `stdout` (i.e. anything that is spit out by any of the python codes) by piping your command to a file name using `>`. You can also save the outputs of the calls to the R files using `2>`. For example, the following command saves the python outputs to `out1.txt` and the R outputs to `out2.txt`

```
python /path/to/raw2feats.py -i /path/to/input/directory > out1.txt 2> out2.txt
```

Saving outputs on UNIX command line should be straightforward (but to be honest I wrote most of this code back in the dark ages before I got a Mac so I haven't tested it out).

Common confusing errors

Error in “phenoDataFromPaths(files)”: Directory tree must be level

This error means that there is probably a mismatch between the files specified in your sequence file and those present in your data folder. The `pick_peaks` code goes through all files it finds in the sequence file’s `File Name` column and tries to pick peaks for them.

A value is trying to be set on a copy of a slice from a DataFrame

This is a non-fatal error. Don’t worry about it.

A subdirectory or file already exists, or other copying files errors

Errors related to making directories and copying files should also be non-fatal. As long as the processing continues after them, you can ignore them.

IOError: [Errno 13] Permission denied: file_name

This error means that one of the files that the code is trying to edit, like the `processing_tracker` or `summary_file` are open in another program. Close them and try again.

Rscript, pickpeaks.R, or alignpeaks.R not found

Make sure that the paths to the respective files are correct. These are in the `preprocessingmtab.py` module, in either (or both, in the case of the call to `Rscript`) `pickpeaks()` or `alignpeaks()`. If there are spaces in the file path and you are using a DOS command prompt, enclose the string that has spaces in quotation marks (“”).

CHAPTER 5

Indices and tables

- `genindex`
- `modindex`
- `search`