Software Implementation for the Unsupervised Identification of Isotope Labeled Peptides

Joshua E. Goldford, Igor G. Libourel

April 14, 2016

Abstract

This document is meant to guide you through implementation of the procedure required to recover isotope labeled peptides fROM high resolution mass spectrometry datasets.

1 Introduction

High-throughput peptide-based metabolic flux analysis requires the identification of non-uniformly labeled peptides in large-scale proteomics datasets [1, 2]. Peptides generated from metabolic isotope labeling experiments are generated from amino acids with mass distributions (AMDs) that are functions of the metabolic activity within the original culture. In metabolic flux analysis (MFA), AMDs are used to recover information on the metabolic state of a biological sample. However, if metabolic heterogeneity existed in a complex biological sample (either from compartmentalization or microbial species within a community), traditional MFA would only be able to report an average metabolic state of the culture. By using peptides instead of amino acids as the principle measurement to infer a metabolic state, an additional dimension is added to the measurement: sequence. Thus, if a peptide is only synthesized in a single species or within a single compartment within a heterogenous sample, measuring AMDs for only amino acids within that peptide would allow for the study of the metabolic state within that single compartment or species.

To leverage additional information encoded by peptides, we used a mathematical model to describe how AMDs should manifest within the mass isotope distributions of peptides (PMDs), using discrete convolution. For example, if x and y represents the iosotope distribution of alanine and glycine, the PMD of the dipeptide Ala-Gly, z could be calculated using the following expression:

$$z_n = (x * y)(n) = \sum_{i+j=n} x(i)y(j)$$

where n represents the n^{th} mass isotope of the dipeptide, and the operation $*$ represents polynomial multiplication. Thus, the original AMDs that give rise to this PMD can be solved using the gradient-based fitting approach called *deconvolution*.

Currently, the mass isotope distributions within peptides are used as key selection criteria for peptide identification in mass spectrometry software. Because AAMDs are a function of the unknown flux distribution, *a priori* information about the PMDs is not available. To enable rapid identification to of isotope-labeled peptides from arbitrary labeling schemes we developed an unsupervised approach that learns amino acid label distributions from identified peptides, and subsequently refines identification by updating the underlying parameters in the convolution model in an iterative approach.

This document provides an outline of our algorithm, as well as a description of the code developed to aid in our analysis of paired unlabeled and flux-labeled biological samples. The first section describes the acquisition of a peptide database from an unlabeled sample using MaxQuant. The second section describes the use of a custom MATLAB library to import raw LCMS data obtained from Thermo-Fisher.RAW files. The third section describes the use of OpenMS to acquire mass traces. The forth section describes the software used for peptide mass distribution identification.

To follow along with this tutorial, follow in the MATLAB script **main.m**.

2 Unlabeled data

An unlabeled experiment is performed and MaxQuant is used to generate a list of peptides. The following section describes the MATLAB code used to parse output files from MaxQuant.

2.1 MaxQuant

MaxQuant is a software suite designed for identification within high-resolution proteomics datasets. Optimized parameter choice for natural abundantly labeled samples are beyond the scope of this tutorial and described elsewhere [3]. Thus the starting point of this section the aquizition of the output data provided by MaxQuant.

All output files are stored in a directory **/combined/txt**. Make sure the this the path to this directory is called in the MATLAB script. Call the function **maxQuantOutput** using the following line in the MATLAB terminal:

```
% construct protein database
protein = maxQuantOutput2protein(PATH_TO_MAXQUANT_TXT,PATH_T0_FASTA_FILE);
```

The protein output structure contains a lot of useful information. It contains a field called **peptides** which is a structure array containing information for each peptide matched to the protein. Of which, the most important will be the following fields:

1. **seq**: amino acid sequence (string)

2. **charge**: charge state (integer)
3. **modseq**: sequences with protein modification (string)
4. **mzRef**: mass over charge ratio used as a mass reference if peptide matching (float)
5. **rt**: retention time of the peptide ion (float)

3 Labeled data

3.1 MSFileReader - MATLAB API

For our analysis, we needed information that was stored in the .RAW file like noise and raw intensity values. Before use, it is important that the following are installed on your 64 bit* Windows machine:

1. **MATLAB MEX** and a C++ compiler
2. **MS File Reader** from Thermo Fisher Scientific.

Find and copy the full path to **Xrawfile2_x64.dll**. Paste this path into line 10 in each .cpp file in the **source** subdirectory. Compile .cpp files using the**mex** command and place **.mex64** files in the **mex** subdirectory. Make sure this is included in your MATLAB path. To import profile scans into MATLAB, run the function **getRawData.m** with the path to the .RAW file as an input:

```
% using the mex compiled c++ functions
raw = getRawData(PATH_TO_RAW_DATA_FILE);
```

The output is a structure, containing information regarding the intensity, mass-over-charge, and noise per measurement for each scan event. These data are not necessarily essential for the identification of the isotope labeled peptides, but become important for quantification during flux analysis. These data are then matched to measurements from mass traces acquired via Open MS. A description of generating mass trace data is provided next.

3.2 OpenMS Mass Trace Extraction

Mass traces are collections of measurements with similar *m/z* observed in contiguous scan events. These mass traces are clustered using no prior knowledge of heavly label in the next section, and are acquired using the open source software Open MS. Open MS is a suite of software tools for the processing of mass spectrometry data. Labeled .RAW files have to be converted to .MZXML format using a one of many platforms, including msconvert or ReAdW, both of witch require access to the XCalibur .dll library provided by MSFileReader.

Download the TOPPAS graphical workflow [4, 5] GUI provided by OpenMS and use **/modules/openms/lib/toppas/mass_traces_extract.toppas** to convert the MZXML into a .featureXML file containing grouped mass traces. Parameter choices for this procedure are described elsewhere [6]. The featureXML data can be parsed using **/modules/openms/lib/python/openms_exporter.py** into a MATLAB readable text file.

```
>> python openms_exporter.py --input [IN.featureXML] --output [OUT.TSV] --type feature_hull
```

The text file can be read into MATLAB and matched to raw data using the the following function:

```
  % using the structure containing the raw data (raw)
mass_traces = getMassTraces(PATH_TO_MASS_TRACE_TEXT_FILE,raw);
```

An example output structure for mass traces matched to raw data are in the file: **/data/ms/labeled/mass_traces.mat**.

3.3 Peptide MID Identification

The following section describes the use of MATLAB code used to (1) filter mass trace data, (2) cluster mass traces into candidate features for each peptide and (3) iteratively deconvolve PMDs to increase precision and accuracy of peptide-feature assignments.

3.3.1 Mass Trace Data Reduction

All putative mass traces belonging to a target peptide are collected and the remaining unassigned traces are filtered from the larger dataset. Two parameters are used for this procedure, a mass accuracy cutoff (MAC) and a retention time window (RTW). The MAC and RTW define the maximinim allowable absolute deviation of the mass (ppm) and elution time (s), respectively. The wrapper **filterMassTraceForProtein.m** achieves this and the following is example call in MATLAB:

```
% only take one protein from database for example
pro_idx = 73;
% filter using a MAC of 5 ppm and a RTW of 1 min
mtf.mac5rtw1 = filterMassTraceForProtein(mass_traces,protein(pro_idx),5,1);
% filter using a MAC of 10 ppm and a RTW of 2 min
mtf.mac10rtw2 = filterMassTraceForProtein(mass_traces,protein(pro_idx),10,2);
```

Note that for this example, only one protein is being investigated (this will take some time on a PC). Both filtered mass trace sets are required; we first learn the amino acid distributions from the more strict filtering criteria and look for additional matches using a less constrained dataset.

3.3.2 Feature Reconstruction

Mass traces are grouped into features using a hierarchical clustering algorithm presented in our accompanying manuscript. Briefly, mass traces are grouped based on similarity in both mass (isotopic precision) and time-correlation. Additional steps are taken to ensure that candidate features are not false positives (see main text). The function **cluter_mass_traces.m** is wrapper for this entire functionality. The following is example code:

```
% cluster mass traces after filtering (strict conditions)
peptides.mac5rtw1= cluster_mass_traces(protein(pro_idx),mtf.mac5rtw1{1});
% cluster mass traces after filtering (liberal conditions)
peptides.mac10rtw2= cluster_mass_traces(protein(pro_idx),mtf.mac10rtw2{1});
```

3.3.3 Iterative Deconvolution

This section describes the process of learning AMDs to allow for high quality features assignment to peptides using simulated isotope distributions. This is achieved by first solving for the AMDs that best explain PMDs from peptides with a single candidate feature, which is done using the strict mass and time cutoff set from the previous section. The function **getPeptidesForDeconvolution.m** retains all peptides with exactly one feature. Then the function assigns the feature's relative isotope distribution as the measured PMD for that peptide. The function is called

```
% keep only peptides with exactly one feature
deconvPeptideSet = getPeptidesForDeconvolution(peptides.mac5rtw1);
```

The peptide set is then used to learn the underlying AMDs using gradient-based deonvolution, which is performed in the **automated_deconv.m** subroutine. Here is the function call:

```
% perform deconvolution of unique features to get an amino acid MID
% estimate from the initial mass trace cutoffs
amino_acid_mid = automated_deconv(deconvPeptideSet,1,[]);
```

Here the second argument defines the number of random starting amino acid distributions. These estimates are then used to simulate an expected PMD for all peptides in the dataset with one or more candidate features. This is done using the filtered peptide set using much less strict filtering criteria (MAC:10 and RTW:2). The function **simulate_peptide_MIDS.m** updates the current estimate for each PMD. Here is the example:

```
% update feature simulated peptide MIDS and percent agreement
peptides.mac10rtw2 = simulate_peptide_MIDS(peptides.mac10rtw2,amino_acid_mids);
```

Features with relative isotope distributions that are highly dissimilar to expectation are filtered out using a predefined threshold, leaving only candidate features with a high degree of similarity. The initial threshold in this example is 0.5 and increases for subsequent iterations. The function **reduce_features.m** reduces candidate features below the percent agreement cutoff.

```
  % reduce the number of peptide candidate features based on a percent agreement cutoff of 0.5
peptides_temp = reduce_features(peptides.mac10rtw2,0.5);
```

This structure is then used to generate a new peptide set with exactly one unique feature per peptide, and is subsequently used for deconvolution. This entire process is repeated to get increasingly better estimates of the the underlying AMDs by narrowing the cutoff used for feature assignment. The cutoffs in the main script are 0.75 followed by 0.9. The final peptide set is provided as a MATLAB structure with the following key fields:

1. feature: field containing all mass, isotope, ion abundance information for the peptide
2. measured: relative isotope abundance to be used for flux analysis
3. isotope: isotope index $(m + i)$ corresponding to each measurement in the measured vector

These data can be then output into various formats suitable for flux analysis.

References

[1] Doug K Allen, Joshua Goldford, James K Gierse, Dominic Mandy, Christine Diepenbrock, and Igor G L Libourel. Quantification of peptide m/z distributions from 13C-labeled cultures with high-resolution mass spectrometry. *Analytical chemistry*, 86(3):1894–901, feb 2014.

[2] Dominic E Mandy, Joshua E Goldford, Hong Yang, Doug K Allen, and Igor G L Libourel. Metabolic flux analysis using C peptide label measurements. *The Plant journal : for cell and molecular biology*, 77(3):476–86, feb 2014.

[3] Jürgen Cox, Ivan Matic, Maximiliane Hilger, Nagarjuna Nagaraj, Matthias Selbach, Jesper V Olsen, and Matthias Mann. A practical guide to the MaxQuant computational platform for SILAC-based quantitative proteomics. *Nature protocols*, 4(5):698–705, jan 2009.

[4] Knut Reinert and Oliver Kohlbacher. OpenMS and TOPP: open source software for LC-MS data analysis. *Methods in molecular biology (Clifton, N.J.)*, 604:201–11, jan 2010.

[5] Johannes Junker, Chris Bielow, Andreas Bertsch, Marc Sturm, Knut Reinert, and Oliver Kohlbacher. TOPPAS: a graphical workflow editor for the analysis of high-throughput proteomics data. *Journal of proteome research*, 11(7):3914–20, jul 2012.

[6] Joshua Elliot Goldford. Automated quantification of 13C labeled peptides, aug 2013.