Workflow Automation in Liquid Chromatography Mass Spectrometry

Reinhard Gentz
Lawrence Berkeley National Lab
Berkeley, CA USA
rgentz@lbl.gov

Héctor García Martin
Joint BioEnergy Institute
Emeryville, USA
hgmartin@lbl.gov

Edward Baidoo
Joint BioEnergy Institute
Emeryville, USA
eebaidoo@lbl.gov

Sean Peisert
Lawrence Berkeley National Lab
Berkeley, CA USA
sppeisert@lbl.gov

***Abstract*—In this document, we describe the fully automated workflow path that was developed for the ingest and analysis of liquid chromatography mass spectrometry (LCMS) data. With the help of this computational workflow, we were able to replace two human work days to analyze data with two hours of unsupervised computation time. In addition, this tool also can compute confidence intervals for all its results, based on the noise level present in the data. We leverage only open source tools and libraries.**

***Index Terms*—liquid chromatography mass spectrometry, LCMS, automation, workflow, noise detector, analysis**

I. Introduction

Since the industrial revolution, humans have replaced manual labor with automation and machines. In this paper, we describe the toolset that we have delveoped for automating the workflow to ingest, analyze, and store liquid chromatography mass spectrometry (LCMS) data. This process contains of several steps that are linked together to achieve the desired automation goal. The input to the workflow are proprietary '.d' files from an Agilent LC-MS machine and the outputs of the workflow are mass distribution vectors (MDVs) and chemical concentrations. MDVs comprise the fractions of carbon 13 within a molecule group, while chemical concentration is defined as the concentration of each molecule per liter.

The most important programming library utilized by this workflow is *Mzmine* [1], which is used to extract signal peaks and signal areas from the data stream. This signal will be, along with the noise level detected, used to compute the desired outputs. Alternative tools include the also open source software XCMS [2] and the proprietary Agilent Masshunter software. We chose Mzmine over XCMS for its simple workflow with batch processing. The Agilent Masshunter software was not a consideration since it is proprietary software that cannot be freely redistributed with our workflow.

This work was part of the DOE Joint BioEnergy Institute (https://www.jbei.org) and the DOE Joint Genome Institute (https://jgi.doe.gov) supported by the U.S. Department of Energy, Office of Science, Office of Biological and Environmental Research, and was part of the Agile BioFoundry (http://agilebiofoundry.org) supported by the U.S. Department of Energy, Energy Efficiency and Renewable Energy, Bioenergy Technologies Office, through contract DE-AC02- 05CH11231 between Lawrence Berkeley National Laboratory and the U.S. Department of Energy. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsors of this work.

II. Steps of the workflow

In this section we describe the workflow steps necessary to go from the input, the LC-MS machine, to the state of having the fully-analyzed output stored in the database. Our goal is complete automation of each task, with the human in the loop only being present for quality assurance (QA) spot-checking. We therefore have automated each step completely, and generate several graphs on the way for manual QA.

A. Fill the LC-MS machine with samples

First the user is required to fill the LC-MS machine with the samples, blanks, and standards to be analyzed. In the future we plan to use an automated robot to perform this step.

B. Run the analysis

To run the analysis, we generate a configuration script for the LC-MS that defines how the samples are to be processed. For example, this script determines the naming of each data output and the order of samples and blanks, and defines the parameters of the LC-MS sensors. This configuration script is produced from a CSV file in which the user enters the desired parameters, that are then converted with a Python script to machine-usable code for the LCMS. This configuration file defines all the manual inputs needed by following steps.

1) Detection of new files on the LC-MS machine: The LS-MS machine is running on a Windows Computer that is co-located with the sensing elements and controls it. We are running a powershell script that is monitoring the sensors output folder for new files using the Windows build in "File System Watcher". This path is very efficient as the operating system will notify our code of any changes and we do not actively have to poll the directory. Once a new file is detected it is copied over to a remote compute server where the computing resources do not compete with the LC-MS sensor's operation. On the computer server the script doing the next steps are triggered as we describe next.

2) Proprietary data conversion: The data files output by our LC-MS machine are in a proprietary Agilent '.d' format. These files have to be converted to a standard '.mzML' format that can be used by our remaining tools. For this purpose we use the 'ProteoWizard MSconvert' [3] software, which can be invoked via command line and can convert one file per CPU core. We have written a Python wrapper around this software

to invoke the binary in parallel fro the command line. The number of parallel executions is dependent on the number of cores on the system and the available amount of memory, as each '.d' file conversion needs about 2GB of memory.

This conversion tool is the only tool that needs to be run on a Windows machine—all other tools used are platform-agnostic and use either Python or Java.

3) Peak detection: Peak detection accuracy is critical for the automated workflow. As described in the introduction, we settled on using the MZmine tool. This tool can process files from command line with a matching configuration file. This configuration file is simple XML that we generate from a user-defined CSV file with a Python script. An additional input file contains the masses and retention times of the molecules of interest, which are calculated from the chemical formula and experimentally found retention times. Prior to automation, it was this process that was the most human-labor intensive—a trained human could have taken about 2 days for a batch of 30 samples. With the automated process, we can reduce this time to under two hours on a two-core laptop. We note that the computation is not bound by the compute speed, as the process runs effectively in real time, as the LC-MS sensor is slower in acquiring the data than the computer is in processing it.

C. Noise detector

We wrote our own noise level detector to find the signal-to-noise level for each peak found in the previous step. This noise level is not constant for each peak, as each 'mass' has its own unique noise level. To compute the noise level, we decode the '.mzML' file with the pymzML [4] library. pymzML decodes results in raw samples which our workflow immediately filters, while decoding is in processing, for the 'masses' of interest to reduce the memory footprint of the computation. To find the noise, we use a window of one minute (in retention time space) before the peak with an offset of 0.2 minutes. To ensure that we do not include the signal of another peak for our noise calculations, we exclude other peaks with a z-score.

D. Normalize MDVs and compute concentration

The output form the peak detection and the noise detector is a 'count' of the element. MDVs comprise the fractions of carbon 13 within a molecule group and can be directly computed from the 'count.' In order to find the confidence interval for our computations, we take the noise signal that was found and make a 'worst case,' respective best case analysis. For the worst/best case we subtract/add the noise from the signal of the molecule isotope in question while for other isotopes we add/subtract the l2norm of the noises to their signal.

To compute the concentration of each molecule we compare the 'count' each sample against a known standard's 'count' for which we do know the concentration.

E. Upload to database

Finally, our workflow formats and uploads our results to our "Experiment Data Depot" database [5], so data can be accessed by the experimenter, shared, or computed on. We also ensure that all chemical compounds comply with the international chemical identifier for portability and easy data sharing.

III. Results

In Fig. 1 we show a comparison of the accuracy of automated MDV generation to manual detection by a trained person. We can see that the results match very well. In addition, the automated results provide a confidence interval on how trustworthy each peak is, given the noise present in the data (which is lost in the manual detection).

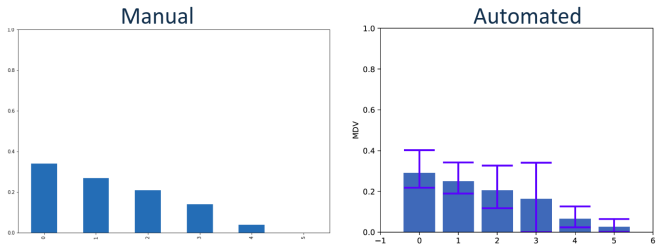

Fig. 1. Manual and automated MDV generation of glutamic acid.

We also compute the concentrations of various chemicals present in the samples and we can see that the system can detect the concentration with a tight confidence interval.

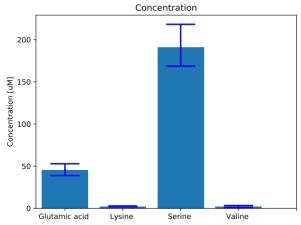

Fig. 2. Concentrations of various chemical compounds.

References

[1] T. Pluskal, S. Castillo, A. Villar-Briones, and M. Orešič, "Mzmine 2: Modular framework for processing, visualizing, and analyzing mass spectrometry-based molecular profile data," *BMC Bioinformatics*, vol. 11, no. 1, p. 395, 2010.

[2] C. A. Smith, E. J. Want, G. O'Maille, R. Abagyan, and G. Siuzdak, "XCMS: Processing Mass Spectrometry Data for Metabolite Profiling Using Nonlinear Peak Alignment, Matching, and Identification," *Analytical Chemistry*, vol. 78, no. 3, pp. 779–787, 2006. PMID: 16448051.

[3] R. Adusumilli and P. Mallick, "Data conversion with ProteoWizard msConvert," in *Proteomics*, pp. 339–368, Springer, 2017.

[4] T. Bald, J. Barth, A. Niehues, M. Specht, M. Hippler, and C. Fufezan, "pymzMLPython module for high-throughput bioinformatics on mass spectrometry data," *Bioinformatics*, vol. 28, no. 7, pp. 1052–1053, 2012.

[5] W. C. Morrell, G. W. Birkel, M. Forrer, T. Lopez, T. W. H. Backman, M. Dussault, C. J. Petzold, E. E. K. Baidoo, Z. Costello, D. Ando, J. Alonso-Gutierrez, K. W. George, A. Mukhopadhyay, I. Vaino, J. D. Keasling, P. D. Adams, N. J. Hillson, and H. Garcia Martin, "The Experiment Data Depot: A Web-Based Software Tool for Biological Experimental Data Storage, Sharing, and Visualization," *ACS Synthetic Biology*, vol. 6, no. 12, pp. 2248–2259, 2017. PMID: 28826210.