Automation of the Liquid Chromatography-Mass Spectrometry (LC-MS) Workflow Using a Run Sequence Wizard

Key Points

- Software computation resulted in **99.995% reduction in time and therefore increase in efficiency for scientists**, excluding user time on software.
- Regardless of valid user parameters entered, software **yielded 100% accuracy rate.**
- Repeated testing of same user parameters yielded **100% precision of software**.

Introduction

This was the main project that I was tasked with during my internship at Merck Research Laboratories (MRL) Pharmacodynamics, Pharmacokinetics and Drug Metabolism (PPDM) Discovery Bioanalytical (DBA) group. For a better understanding of concepts discussed in this document, check out Python, LC-MS, LabKey, Visual Studio Code and Graphical User Interface (GUI). As of **January 2019, a publication related to the work completed is being prepared**. My supervisor was mass spectrometry expert, Kristin Geddes, Associate Principal Scientist at Merck. The group director was Dr. Daniel Spellman.

Problem Statement

The problem that this research group had was with the creation of Run-Sequence Files for Liquid Chromatography-Mass Spectrometry (LC-MS) instruments.

Run-Sequence Files are documents that direct LC-MS instruments to use specific samples, at specific volumes, with specific methods, to analyze from a 96-well plate. They are complex, tricky, error-prone and time-consuming to formulate, let alone by hand-preparation. Run-Sequence Files are also problematic to create since this group employs four different LC-MS manufacturers that each have their own requirements for file preparation. This has lead to a more wide-spread problem of not having standardization between files. **The goal was to create an immaculate, high-performance, scalable, vendor-agnostic GUI application that could be deployed on the LabKey server, used by this group, to ultimately automate the tedious but imperative task of Run-Sequence File creation. This is all while making sure the software standardizes the file-creation process and does not add any separate work for users.** The software had to consider engineering and biological constraints for the instrument and user input, respectively. Using these considerations, the software optimized a solution that allowed the instrument to perform its function in an accurate and precise manner. The software needed to be perfect since I was the only computational person in my department and so modifying software would not be feasible.

Methods

Since this project is proprietary work of Merck & Co. I cannot go into details about ALL the optimizations and steps of this project.

I decided to use Python 3.7 as the programming language since it is commonly used in Bioinformatics and is an easy programming language to code with (in case the

code needs to be modified in the future by someone else). It is also easy to download and has an in-built GUI creation package, tkinter.

I used tkinter to create an elaborate GUI that allowed users to have extreme control over the final file without spending much time using the software. The type and order of biological species that went into the final file was determined by a simplistic 96-well plate map file. Scientists must make these files anyways so that they understand which species/reagents must go into which bore on a LC-MS 96-well plate before they load this plate onto the instrument.

The software involved 2 consecutive GUIs. These GUI's allowed plate map matrix upload, options for adding biological samples, settings for each instrument manufacturer and a required input key indicating the Merck user using the software. There were multiple sub-algorithms to process/verify input files, to create/authenticate output file and to handle all the previous inputs mentioned.

Optimizing memory usage and speed of the software was imperative. Increasing the speed aligned with my goal to get the software computation time to under one second. There were many steps in the program workflow that involved string concatenation which were achieved using the "join" method built into Python which is optimized for speed as opposed to "adding" strings together. Another optimization involved the usage of multiple "helper" functions for simple tasks throughout the program. This avoids code redundancy and takes advantage of local variable vs global variable storage. Variables inside functions are local and have quicker storage and lookup times in Python.

There was no set numerical upper-limit for memory usage since the variability of user inputs makes this unfeasible. Regardless of all possible inputs, ideal software should minimize memory usage. This was achieved in two ways. There were many Python data structures including lists and dictionaries that contained large data sets (1000's of elements) which were deleted, for memory collection, once they were extraneous. The other approach was to eliminate single-use variables and reduce the number of data instances that Python had to maintain.

Once the program created a Run-Sequence File using data mentioned previously as well as information that cannot be disclosed publicly, it would use the Merck user key to create a directory (if necessary). The directory was created on a network drive accessible to most Merck employees. The final file was deposited in the directory of the Merck user. The directory on the network drive that housed all these finished Run-Sequence Files had folders for documentation, user history, plate map templates, and folders for each user of the software (where the files went). Plate map templates are just simplistic Microsoft Excel file sheets that standardizes software user entry and improves readability of filled plate maps between scientists.

Network drives were used as an alternative to the LabKey server due to difficulties with LabKey module deployment. **I used XML to create my own modules in the server but these did not ultimately work even though the XML code was correct.** Due to time constraints, we chose to go down the route of using a network drive. The final software, Python installation executable and documentation from the network drive folder was uploaded to LabKey server and was also available in the network drive folder.

The final software was dubbed as the Run Sequence Wizard. The reason for calling it a “Wizard” is due to the software definition of Wizard. Software Wizards involve user interfaces that lead people through a set of clear steps to ultimately aid them with a tedious or complex task.

Results

All in all, the software took approximately 2500 lines to write and contains approximately 30 functions. All code was developed in the free software, Microsoft Visual Studio Code, and is encapsulated into one class. The GUI application was made into a class in case future work would like to instantiate multiple GUI applications at once. The software calls the class once hence instantiating one GUI application (as expected).

One major goal of this software was to eliminate all possible error that may occur had scientists manually prepared files. The software was tested with approximately 50 different user test cases. The test cases contained varied plate map files, instrument vendor choices among other parameters that cannot be disclosed due to confidentiality. The final output files from the software were compared with correct files as well as reviewed by scientists working in the department. **All in all, the software achieved a 100% accuracy rate.**

Afterwards, all user input test cases mentioned in the previous paragraph were put through the software again to verify that they are the same as output files from previous runs. This process was repeated multiple times and **ultimately yielded a 100% precision rate for the software.**

As a reminder, the original goal was to have the software create files in under a second, excluding the time users need to insert files/inputs. During accuracy and precision testing of software, computation time was measured and saved to get an average time metric. The **software on average, took 0.09 seconds or 9 milliseconds,** to generate the two GUI's, read in files, load data structures, create directories on the network drive and produce final output files. Due to confidentiality, there are many smaller or equivalent-sized computational steps that are not listed in the previous sentence. Scientists in this group explained that the **average time for hand-preparation of Run-Sequence Files is around 30 minutes, after preparation of plate map.** Coupling this fact with trivial calculation, we can see that the **software reduces scientists file preparation times by 99.995%!** The average time to create plate maps (using templates) and upload all inputs is approximately 5 minutes.