Chromatography Documentation

Stephen Moreno, William Bahureksa, Simeon Schum,

Nathaneal Gonzalez, Daniel Moreno, Emalyn Howard,

Austin Martin and Jason Ovalle

New Mexico State University

Fall 2025

Chemical Project Documentation

The automated analysis workflow project is an ongoing partnership with multiple campus entities such as the Physical Science Laboratory, Chemical Analysis Instrumentation Lab, and upcoming Computer Science students. This partnership allows students to gain real world experience before graduating while also helping the Chemical lab gain a more efficient workflow for their data analysis of different chemical markers.

Introduction

Project Background

The Chemical Analysis Instrumentation Lab (CAIL) is a research facility that supports agriculture, engineering and other disciplines to provide liquid and gas chromatography for optical and thermal measurements. The Physical Science Laboratory (PSL) is a multi disciplined primarily defense oriented and technical organization throughout the campus. The PSL team also partners with students, laboratories and other national platforms for the advancement of technical solutions to real world applications to their customers. The project you will be working on is developing analysis software to support mass spectrometry research for CAIL. The motivation for this project is to advance the research of water resources in New Mexico while maintaining the cleanliness for the communities this project will support.

Struggles with Original Workflow

Mass spectrometry analysis traditionally relies on MZmine, a powerful but slow and often unstable GUI, especially when importing large files or configuring complex settings. Sometimes the scientists at CAIL only need a simple scatter plot, yet are forced to navigate a time consuming application. This left a bottle neck in the workflow that needed to be fixed and adapted for future projects.

Helpful Definitions

- mass - refers to compound/chemical/molecule mass in respect to atomic mass units (amu), and in this context, is interchangeable with daltons (Da), molar mass, and molecular weight.
- m/z - mass-to-charge, defined as mass divided by charge present on the molecule (molecules must be charged for measurement). We are ONLY considered singly charged ($z = 1$) compounds for the current study.
- Base peak - refers to the largest peak in the mass spectrum
- Intensity - a unit-less value that refers to the measured signal from the molecule of interest.

Scope of the Project

For this project it will be completed over the course of multiple semesters with multiple different teams. This part of the project was scoped for an entire semester with the intention of going further into the original pipeline if time allowed. With this in mind we were mainly focused on the front end of the project with the ideas and focuses constantly changing. We started with the original pipeline that will be provided below. There were some original changes that were made to tailor it to the scope of a full semester. As we progressed we noticed that we needed to continue to tailor it due to some unexpected challenges we were facing at the time, since we wanted it to be completed we did not want to promise a completion of a section if it did not work out. We are glad that most of our goals were completed even with the unexpected challenges we faced.

High Level Goals

If we had time to continue developing this project we would have liked to implement the moving lines that change with each generated graph. The lines are critical to helping our users analyze chemical noise projections, while they can be added in manually it would slow down the efficiency we were trying to bring to our users. This automated feature would have enhanced user experience but were unable to complete it within the timeline.

We also would have liked to implement the API's needed for the program to communicate effectively with other tools being used in the workflow. This feature would have streamlined the process while also playing a key role in customization and automation that would have exceeded expectations. Unfortunately it was a stretch goal for us within the time of a semester.

Things we Would've Liked to Know

For the next team that is going to take this on we would like to share some of the knowledge we have gained through our experience. We highly recommend reviewing this section closely as we believe it would have saved us significant time while also helping our team navigate new challenges during the semester.

First, utilize your advisors (Stephen, William, and Simeon). They are here to help the group and respond in a timely manner. No question is too small for the group to ask, each advisor is very knowledgeable in their field making them the perfect ideal resources to give you clarifying concepts, suggesting alternative solutions, and advice on what is the best path forward.

Before any coding we strongly recommend you team to have in-depth discussions about the applications you're going to use to make sure everyone is on the same page. Group cohesion

early on ensures that there is less misunderstanding and frustrations later on in the development process.

To save you time we've left chemical descriptions in the introduction section to save you time searching for basic definitions. If any definition is missed the advisors are happy to help at any time but these provide clarity on some chemical formulas especially when explaining aspects of the project during interviews.my

One major oversight was practice presenting and explaining different components of the project to prepare for the interview processes. Prioritizing these skills will improve the communication skills with the entire class but also helps you track the pipeline more accurately. Staying aware of what has been changed, what has remained consistent, and what still needs to be updated to maintain a clear understanding of the project's progress.

System Overview

This software provides scientists at CAIL with a fast and streamlined way to analyze mass spectrometry data by generating a scatter plot. This enables researchers to quickly assess signal-to-noise patterns and gain deeper insight into the chemical composition of their samples. This project eliminates that inefficiency by offering a lightweight, reliable, and quicker way to visualize the data without depending on MZmine.

Specifically designed to support chemists, analysts, and researchers at CAIL who routinely work with mass spectrometry data and need quick access to visualizations of mass spectrometry. It simplifies exploratory analysis and enhances productivity by reducing reliance on outside applications.The system is built using Python, with PyQt providing the graphical interface and

Matplotlib powering the scientific plotting. Data processing is handled through JSON parsing, and Kendrick Mass Defect computations. The final application is packaged with PyInstaller to ensure easy deployment across different machines.

Requirements Specification

Non-Functional Requirements

The nonfunctional requirements we focused on included was creating a GUI that was seamless, intuitive, and easy to use. We also wanted to make sure that our software remained efficient while producing quality results for the chemist to analyze further. Although these were requested by the stakeholders we aimed to make sure that this ensured more of a positive experience and ensuring the tool would be both practical and enjoyable for them to us.

Functional Requirements

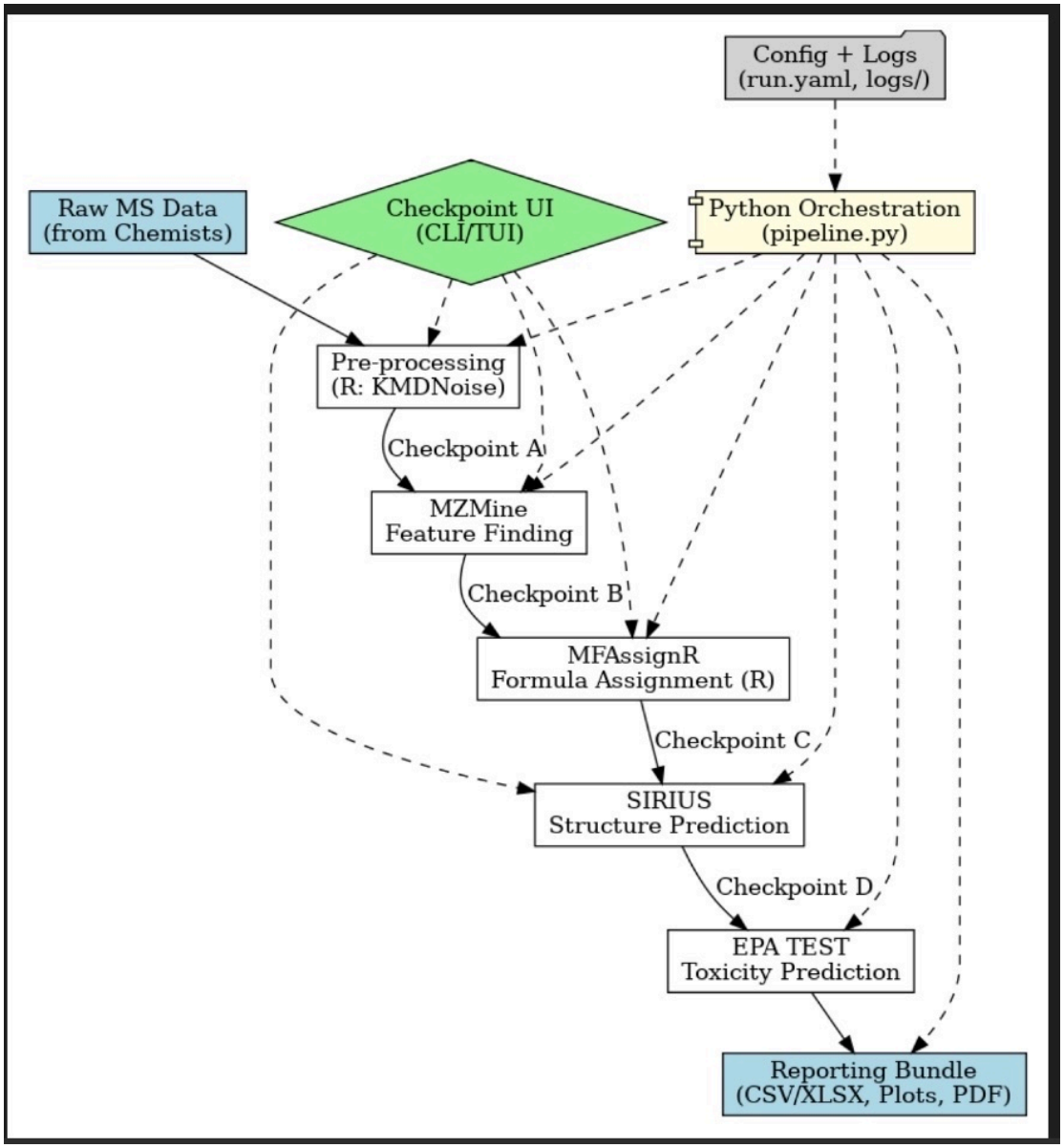
The stakeholders primary request was to make the data processing part of the application automated faster, and less reliant on graphical user interfaces. They also requested that we develop the application for easy integration with the tools in existence like SIRIUS and MZmine. Another key requirement is to change the file type from a .json to .csv after the data was processed. These functional requirements made it easier for us as a group to help guide our planning and communication as a team, allowing us to better understand how to implement their needs and divide tasks based on a team member's strength.

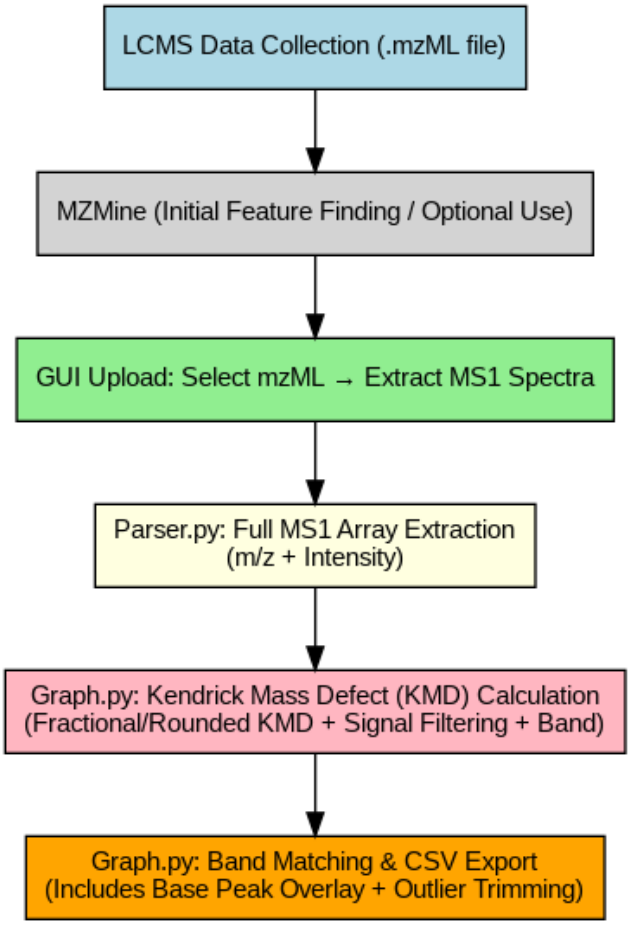
System Design

The software is built using a modular architecture composed of three primary layers: a PyQt based graphical user interface, a parsing engine responsible for extracting MS1 spectral data

from .mzML files, and a computational analysis engine that generates a Kendrick Mass Defect (KMD) plot. The parser converts raw mzML data into JSON for compatibility with Python's dictionaries that store scan identifiers, base peak characteristics, and full m/z and intensity arrays for each MS1 event. Once the parser outputs the data, the graph function uses the JSON objects and computes the per-ion KMD metrics, performs signal to noise filtering and generates a plot using Matplotlib which is shown in the interface.

The frontend has a three step workflow consisting of Upload -> Configure -> Visualize, implemented through stacked page widgets that maintain the state across the screens. Shared state is stored in a central controller dictionary which allows each page to read user selections like the chemical basis and update or display analysis results. Overall, the system design emphasizes modularity, data abstraction through JSON, and a clean frontend to backend pipeline where user actions trigger computations and show the results through a plot.

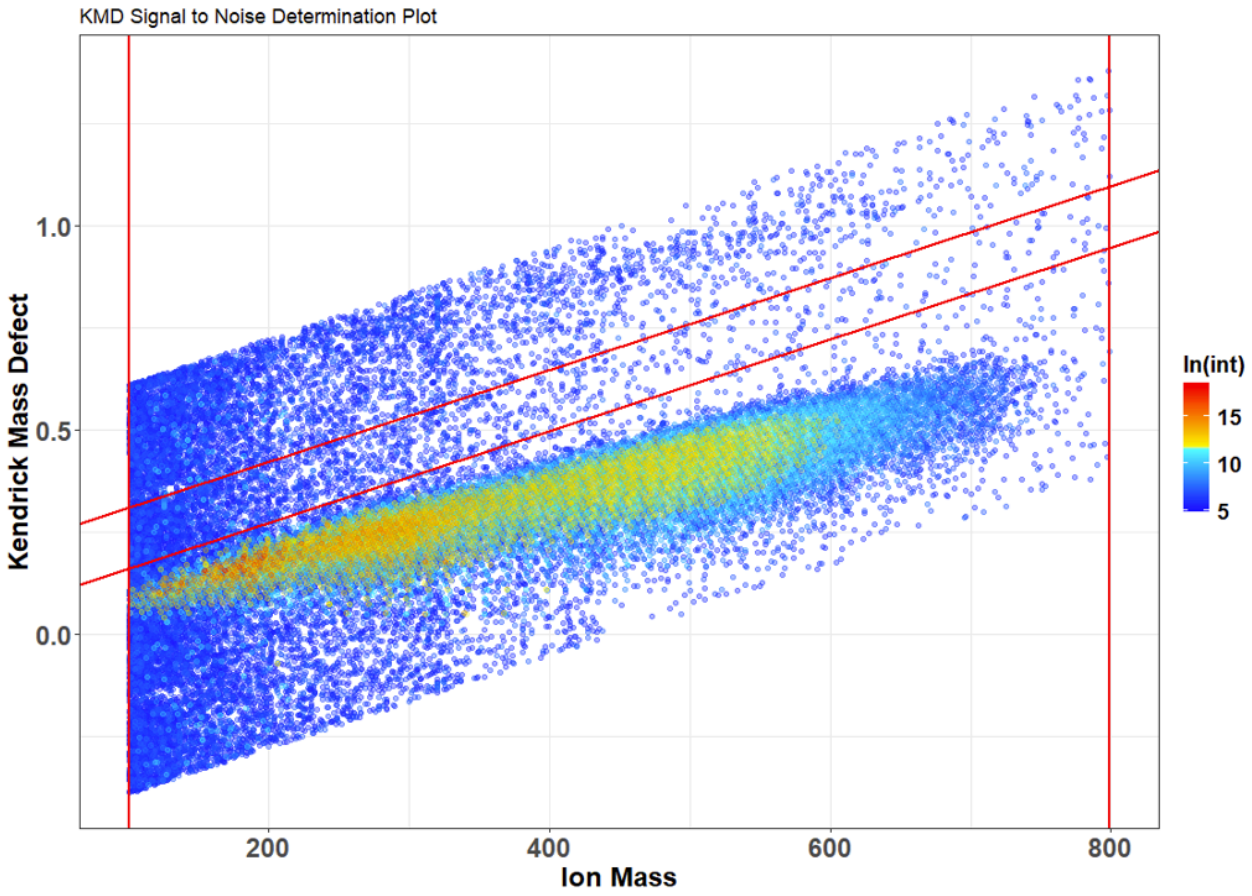

Config + Logs
(run.yaml, logs/)
Raw MS Data
(from Chemists)
Checkpoint UI
(CLI/TUI)
Python Orchestration
(pipeline.py)
Pre-processing
(R: KMDNoise)
Checkpoint A
MZMine
Feature Finding
Checkpoint B
MFAssignR
Formula Assignment (R)
Checkpoint C
SIRIUS
Structure Prediction
Checkpoint D
EPA TEST
Toxicity Prediction
Reporting Bundle
(CSV/XLSX, Plots, PDF)



Implementation

Math Formulas

William provided us with the math formulas that are used to calculate the MS1 values that are going to be plotted on the graph. It helps us calculate the intensities that are used on the color scale and the lines that buffers out the outliers to help us produce correct results. In these formulas we are finding the Kedrick Mass and the Kendrick Mass Defect (KMD). To find the Kendrick Mass you will need to use this formula “m/z x (Nominal Mass/Exact Mass)”. You will then take the number you get from the kendrick mass formula and subtract it from the base MS1 which will give you the Kendrick Mass Defect.



Testing & Validation

We used the server to test the code by uploading the most updated source code from the project into the virtual environment from 'https://sirius-dev1.psl.nmsu.edu/dashboard'. The program require multiple Python packages to run the program, they are PyQt5, Matplotlib, pandas and numpy. These were all installed using pip with the command *pip install <package name>*. Please note that PyQt does require additional configuration or else it will not run the program. Make sure before you run the program you'll need to set this setting in the terminal *QT_QPA_PLATFORM=offscreen.* After this, additional libraries are needed for PyQt, install these using *sudo apt install libxcb-xinerama0 libxkbcommon-x11-0 libxcb1 libxcb-render0 libxcb-render-util0 libxcb-keysyms1 libxrender1 libqt5gui5 libxcb-icccm4 libxcb-image0*

libxcb-shm0. After all the dependencies are installed, the program should run using the *python3 -m gui.gui* command. Dependencies are in the pip-requirements folder in the github. All dependencies can be installed using the pip install commands that are found in the read me.

For our testing methodology, we conducted User Acceptance Testing (UAT), evaluating the software from the perspective of a real end user. The goal was to verify that the system behaved correctly in practice, that the workflow was intuitive, features produced their expected results, and that no defects appeared during typical use.

We executed core user scenarios including uploading an mzML file with validation ensuring only .mzML files were accepted, selecting a chemical basis, generating plots, exporting results as PNG images and CSV files, and navigating smoothly through the three page workflow from start to finish, including initiating new analyses. Throughout this process, we monitored usability, responsiveness, and system stability.

UAT also identified several areas for enhancement, such as incorporating a progress indicator during file parsing to improve user feedback. Overall, this approach confirmed the software's functional correctness while highlighting meaningful opportunities for refinement and improved user experience.

Deployment

Installation Instructions

To run the SIRIUS software you must create an account from the SIRIUS website to ensure secure access. Once the website is on your screen you will need to go where it tells you to log in

and create a new account that includes your name, username and password. New Mexico State University should have their own license so you will not need to gain access to that if they have one the only requirement to use the institutional license is that you must create an account with your NMSU email address.

To distribute the application, PyInstaller must first be installed using pip install pyinstaller. PyInstaller packages the project's source code and dependencies into a standalone executable, eliminating the need for users to manually run the software with python3. The application can be built by running pyinstaller --windowed --name CAILAnalyzer --add-data "Pipeline:Pipeline" gui/gui.py, which generates the executable inside /home/astro_sirius/workspace/PSLGang/dist/. This file can then be relocated anywhere on the system and launched directly. Whenever changes are made to the codebase, such as new features, improvements, or bug fixes, the software should be rebuilt using the same command to produce an updated deployment version for CAIL.

Limitations and Future Work

Known Bugs

The primary known bug in this version is that this application only supports one chemical compound. To address this, additional files need to be added to produce accurate calculations of each extra chemical compound to ensure that the program continues to process them correctly and consistently when graphing them. Another issue is that the slope calculations used in the graphing component still require extensive testing to verify that the results remain accurate for multiple datasets. Lastly, the ranges for the x-values need to be improved, particularly for

compounds that have not been fully integrated into the system. Cleaning up the ranges will be essential for producing reliable graphs and supporting the future expansion of the application.

Planned Improvements

Planned improvements for the next phase(s) of this project included creating and enhancing the red reference lines used on the graphs to identify and correct outliers. Our goal is to make these lines adjustable across multiple graphs, creating a more flexible and user friendly experience. Additionally, the stakeholders expressed interest in integrating the application with relevant APIs to strengthen compatibility with existing analytical tools and to support potential future applications for future groups.

Conclusion

In conclusion, there is a great deal of potential in this project and are confident that future senior groups will continue to expand its capabilities in meaningful ways. The foundation built this semester opens the door for improvements in the automation, data processing, integration with existing tools and usability that future teams can refine and elevate the project even further. We are extremely grateful for the partnership we had with the CAIL and PSL teams. Their expertise, guidance and undeniable support throughout the process made a significant difference when navigating challenges. They were constantly knowledgeable, helpful, and responsive to our questions making the difficult parts more manageable. We hope future teams will benefit from this collaborative relationship while continuing to strengthen the project through their contributions.

References

Sirius-Ms. (n.d.). *Sirius-MS/Sirius: Sirius is a software for discovering a landscape of de-novo identification of metabolites using tandem mass spectrometry. this repository contains the code of the sirius software (GUI and CLI)*. GitHub. https://github.com/sirius-ms/sirius

Korf, A. (2025, November 25). *Welcome to the MZMINE documentation!*¶. mzmine documentation. https://mzmine.github.io/mzmine_documentation/index.html

Name, Y. (n.d.). *Welcome*. SIRIUS Documentation. https://v6.docs.sirius-ms.io/

Jayxvalle. (n.d.). *Jayxvalle/PSLGang*. GitHub. https://github.com/Natgonza99/PSLGang#