



Method Article

Open source software toolchain for automated non-targeted screening for toxins in alternative foods



S.W. Breuer^a, L. Toppen^b, S.K. Schum^c, J.M. Pearce^{a,d,*}

^a Department of Electrical and Computer Engineering, Michigan Technological University, United States

^b Department of Environmental Engineering, Michigan Technological University, United States

^c ChARM lab, Michigan Technological University, United States

^d Department of Electrical and Computer Engineering, Western University, Canada

ABSTRACT

Previous published methods for non-targeted screening of toxins in alternative foods such as leaf concentrate, agricultural residues or plastic fed to biological consortia are time consuming and expensive and thus present accessibility, as well as, time-constraint issues for scientists from under resourced settings to identify safe alternative foods. The novel methodology presented here, utilizes a completely free and open source software toolchain for automatically screening unknown alternative foods for toxicity using experimental data from ultra-high-pressure liquid chromatography and mass spectrometry. The process uses three distinct tools (mass spectrometry analysis with MZmine 2, formula assignment with MFAssignR, and data filtering with ToxAssign) enabling it to be modular and easily upgradable in the future. MZmine 2 and MFAssignR have been previously described, while ToxAssign was developed here to match the formulas output by formula assignment to potentially toxic compounds in a local table, then look up toxic data on the Open Food Tox Database for the matched compounds. This process is designed to fill the gap between food safety analysis techniques and developing alternative food production techniques to allow for new methods of food production to be preliminarily tested before animal testing. The methodology was validated against a previous method using proprietary commercial software. The new process identifies all of the toxic elements the previous process identified with more detailed information than the previous process was able to provide automatically.

- Efficient analysis to find potentially toxic compounds in alternative foods and resilient foods.
- Identification of potentially unsafe products without the use of live animal testing.
- Modular free and open source design to allow for upgrading or fitting of user needs.

© 2021 The Author(s). Published by Elsevier B.V.

This is an open access article under the CC BY license (<http://creativecommons.org/licenses/by/4.0/>)

* Corresponding author at: Department of Electrical and Computer Engineering, Western University, Canada.
E-mail address: joshua.pearce@uwo.ca (J.M. Pearce).

ARTICLE INFO

Method name: Open Source Automated Non-Targeted Screening for Toxins in Alternative Foods
Keywords: Alternative food, Edible leaves, Edible plants, Existential risk, Global catastrophic risk, Leaf concentrate, Leaf protein, Non-target screening, Sustainable food systems, Toxins
Article history: Received 13 August 2021; Accepted 11 October 2021; Available online 14 October 2021

Specifications table

Subject Area:	Pharmacology, Toxicology and Pharmaceutical Science Pharmacology, Toxicology and Pharmaceutical Science
More specific subject area:	Toxic screening
Method name:	Open Source Automated Non-Targeted Screening for Toxins in Alternative Foods
Name and reference of original method:	Pearce, J.M., Khaksari, M. and Denkenberger, D., 2019. Preliminary automated determination of edibility of alternative foods: Non-targeted screening for toxins in red maple leaf concentrate. <i>Plants</i> , 8(5), 110; https://doi.org/10.3390/plants8050110 .
Resource availability:	https://osf.io/nh76z/

Method details

Introduction

Agricultural-loss-based global catastrophic risks (GCRs) have some of the greatest probabilities and impacts [1]. Previous work has shown alternative food supplies made from converting fossil fuels, wood and leaves to human-edible food could feed humanity even in the event of a GCR that eliminated all conventional agriculture [2–4]. Using a variety of alternative foods can provide a balanced diet of macronutrients [5] and micronutrients [6] to maintain human life. It is also cost effective to prepare for alternative food production both globally [6] and in the U.S. [7]. For alternative foods to be effective in GCRs they must be to maintain caloric intake consistently. This can be a challenge in the event of a sun-blocking GCR because there is a gap between the time that stored foods run out globally (~6 months) and the to ramp up production of alternative foods that do not rely on sunlight (~12 months) [2]. To date the best theoretical solution for this transition problem is to use leaves killed by the GCR [8]. It is possible to grind and press the leaves, and then coagulating the resultant liquid as leaf concentrate, which contains ~8% of the dry matter of the original leaves while the remaining liquid contains much of the toxins and is discarded [9,10]. Although technically, viable there is an enormous knowledge gap on the toxicity of leaf concentrate for humans from the most common tree leaves. Traditionally, the approach for detection of toxic compounds in a solution is to do targeted screening by purchasing the corresponding reference standards for identification [11]. This method is unrealistic for the potential toxic compounds in plant materials as there are over 1500 phytotoxins already identified in the Toxic Plants–Phytotoxin (TPPT) Database [12]. Fortunately, high-resolution mass spectrometry (HRMS) has allowed for non-targeted screenings where no prior information is available for identification of unknowns in a sample [14]. Further, a recent study, provided the preliminary steps for obtaining a rapid toxics screening process of common leaf concentrates to be used for alternative foods [13]. In non-targeted screening, experimental evidence is needed to confirm the identification using a suitable algorithm. This evidence includes accurate mass, isotope pattern, presence of additional adducts, retention time, fragmentation information, and other experimental evidence. The quantity and quality of evidence available for identification leads to a range of levels of confidence for compound identification [11,2,15]. Pearce et al.’s non-targeted approach used an ultra-high-resolution hybrid ion trap orbitrap mass spectrometer (MS) coupled to an ultra-high-pressure two-dimensional liquid chromatograph (LC) system on the most common leaf in North America, red maple (acer rubrum) [16], to provide the greatest potential for alternative food [13]. The data was analyzed using the ThermoFisher Scientific’s screening method of unwanted compounds in food [17] and relied on a proprietary software package - the Thermo Scientific Compound Discoverer software [18] for identification of unknown compounds. This software costs \$20,000 per seat [18]. Then the identified chemicals were cross-referenced pseudo-manually

among several databases to identify toxic and harmful chemicals. Although this process was effective, it was time consuming and expensive and thus presented both accessibility issues for scientists from under resourced countries to prepare their own lists of safe leaf concentrates for alternative foods as well as time barriers to screen all potential leaf sources and other alternative foods (e.g. agricultural residues or plastic fed to biological consortia). To overcome these limitations this methodology utilizes a completely free and open source software toolchain for automatically screening the experimental data from unknown alternative foods for toxicity.

Experimental

Leaf concentrate is prepared following standard procedures [9,10]. For LC/MS analysis, leaf concentrate was diluted 12 times in water–acetonitrile 80:20 (v:v) and filtered with a 0.2 μm quartz filter. A Thermo Scientific Dionex Ultimate 3000 standard system was then used as a high-pressure liquid chromatography (HPLC) system on the material. The instrument was calibrated externally with Thermo Pierce calibration solution before LC/MS runs. Following [13] the analytical column was Phenomenex reversed-phase Kinetex XB-C18, 150 \times 2.1 mm, 100 Å, with 1.7 μm particle size. Mobile phase A was 0.1% formic acid in 100% LC/MS grade water and mobile phase B was 0.1% formic acid in LC/MS grade acetonitrile–water 95:5 (v:v) solution. Using a constant flow rate of 0.2 mL/min (0.2 mL/min was used due to small particle size of the column (1.7 μm), which increases the back pressure), the mobile phase gradient was: 0 min; 5%B, 5 min; 5%B, 65 min; 90%B, 70 min; 90%B. The column was equilibrated with mobile A for 15 min before the next injection. The column oven was set at 35°C, and the full loop injection volume was set at 5 μL . The mass spectrometry instrument was a Thermo Scientific Orbitrap Elite equipped with electrospray ionization (ESI). The resolving power for accurate mass measurement during the LC/MS run was 120 K defined at m/z 400. The sample was run with both positive and negative ESI modes under two separate LC/MS runs. All the masses in the range of 100–600 m/z were recorded with full scan mode. In addition to the full scan, data-dependent MS/MS fragmentation was also recorded for the 5 tallest peaks on each spectral scan with a collision energy of 25 (arbitrary unit) to help identify co-eluting compounds.

Data analysis

This analysis method, which differs substantially from the original analysis by Pearce et al. as outlined in Fig. 1, consists of four parts, those being primary analysis by MZmine 2 [19,20], formula assignment by MFAssignR [21], filtering using PubChem [24] and a new open source python API caller known as ToxAssign [26], and final analysis by hand. Further confirmation of the toxic compounds identified can be performed using retention time and known pure samples following standard protocols [14].

MZmine analysis

The first section of this analysis relies on the software, outlined in the paper by Pluskal et al. [20], MZmine to convert the RAW output files from the mass spectrometer. This software uses multiple steps to filter and identify mass peaks as well as retention time data, including mass detection and chromatogram-based analysis. Fig. 1 shows this analysis first uses the software to import the raw data, then detect the mass peaks in the data, use those mass peaks to build a chromatogram, then perform chromatogram deconvolution before outputting data to a CSV file. The chromatogram builder and deconvolution step are of particular note because of their ADAP, or automated data analysis pipeline, method outlined in the paper by Myers et al. [27].

Raw Data Import:

To import a raw data file click on *Raw Data Methods* \rightarrow *Raw Data Import* as shown in Fig. 2. This will open a window where the files may be selected from computer or hard drive storage. The imported files will then appear on the left-hand window with a label *Raw data files*.

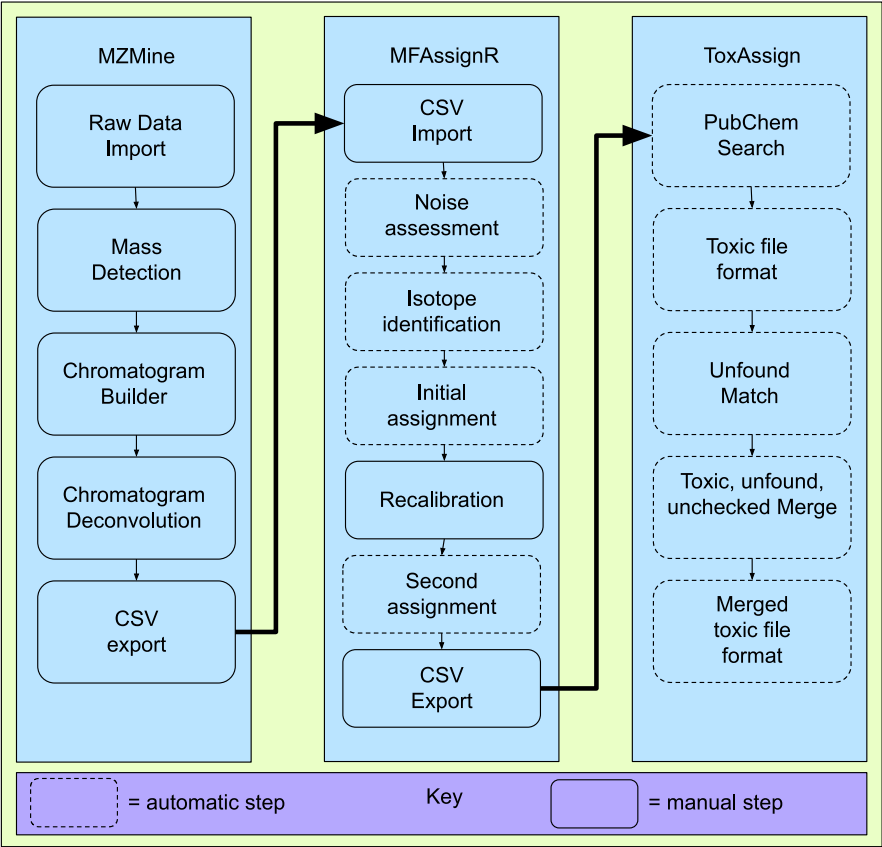


Fig. 1. Map of full open source process for screening toxicity of unknown alternative foods.

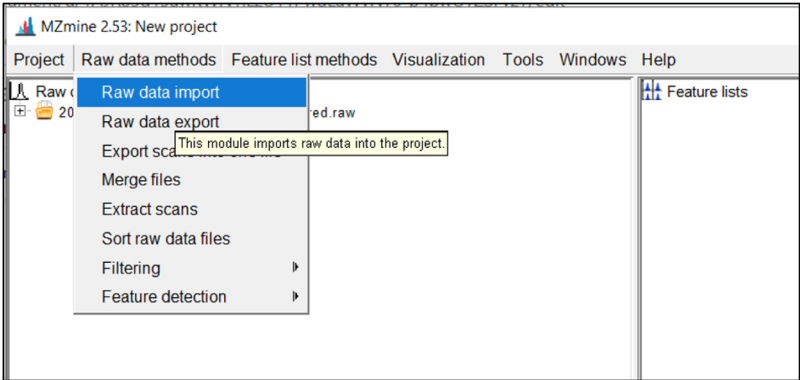


Fig. 2. Screenshot to import raw data files.

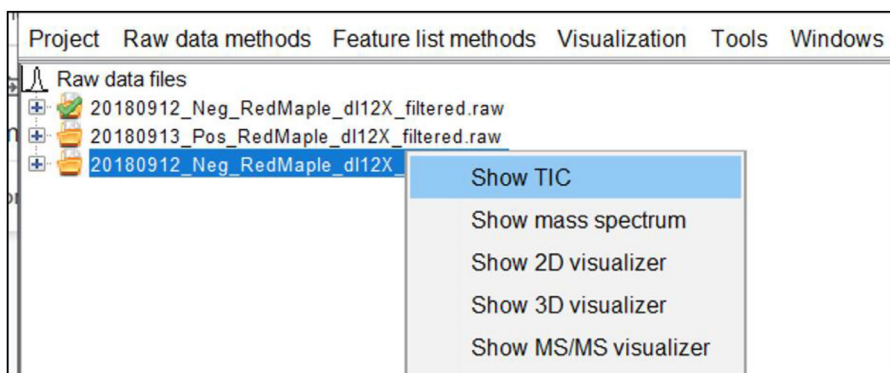


Fig. 3. Screenshot to show initial chromatogram of raw data.

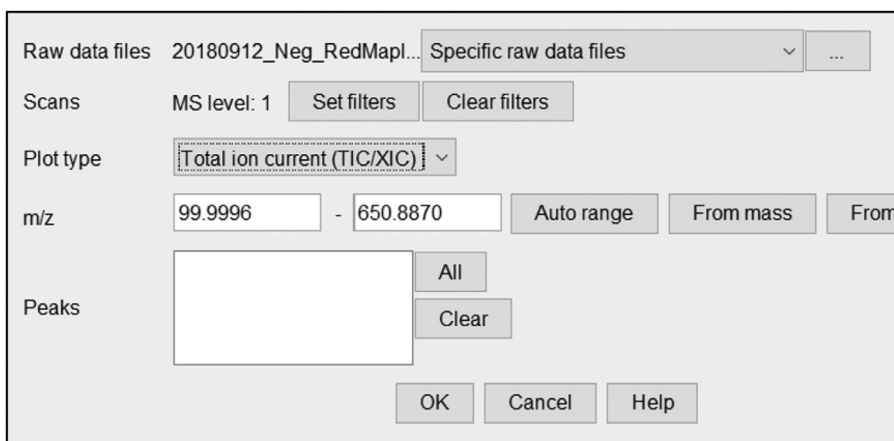


Fig. 4. Parameter window to view initial chromatogram of raw data load.

After the raw data is imported, it can be helpful to get an initial view of the chromatogram to determine the retention time where peaks are detected. To do so, hover the mouse over the desired raw data file on the left window and select show TIC on the dropdown menu as shown in Fig. 3.

This will bring up a parameter menu as shown in Fig. 3. Select Total ion current in the plot type dropdown menu and select Auto range for m/z, then click ok to display the chromatogram (Fig. 4).

An example of an initial chromatogram is shown in Fig. 5, this is an example of a data set that can have its retention time cut down because of the clarity that there are no detected peaks after 30 min of retention time. Taking this extra step to shave down the amount of data to analyze can save a lot of time and better concentrate desired data.

Mass detection:

The next step is to run the mass spectra through the peak detection feature to detect the masses from the data. Select the imported files by clicking on them in the left window, then click *Raw data methods* → *Peak detection* → *Mass detection* as shown in Fig. 6.

This will open a window with several parameters to be entered. The next section will outline each parameter section shown on the right-hand side of Fig. 7.

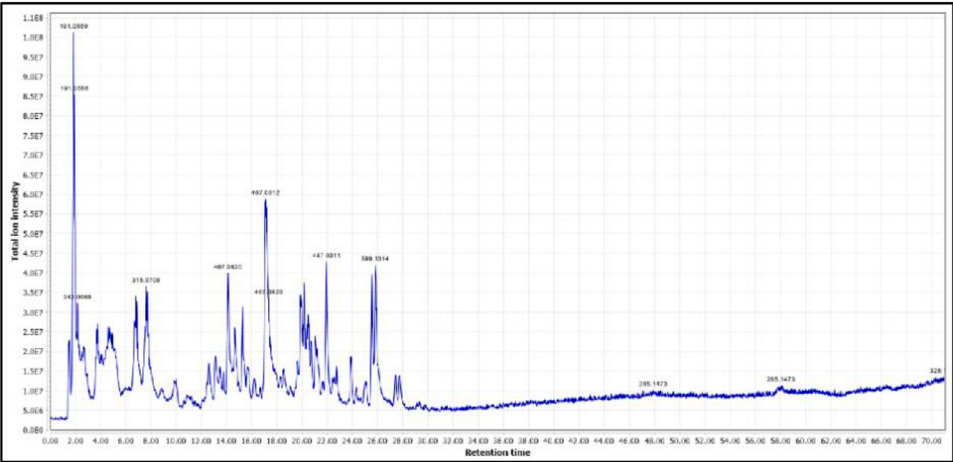


Fig. 5. Example chromatogram of raw data to cut down retention time in future steps of analysis.

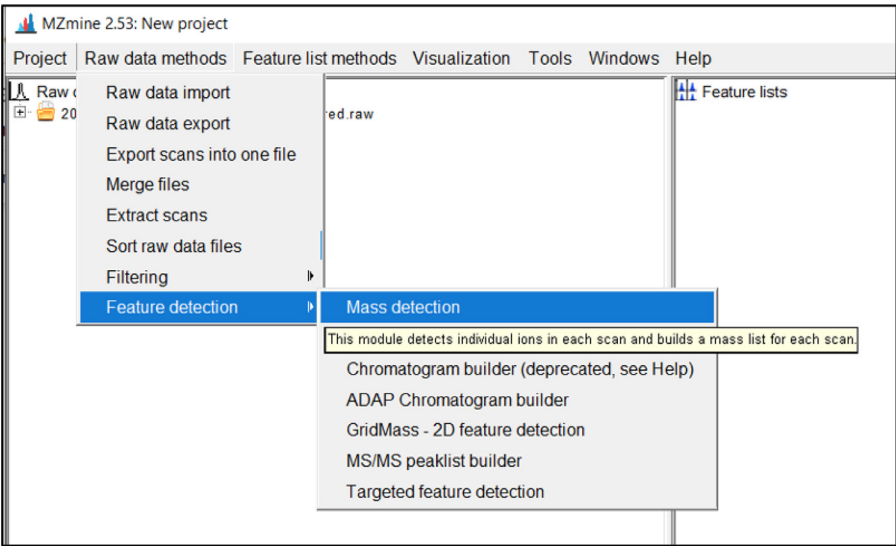


Fig. 6. Screenshot to run mass detection.

Mass detector:

First step is to set the mass detector, an algorithm to use for mass detection and its parameters. Click the drop-down box titled *mass detector* and select *Exact Mass*.

Then set the noise level to what matches the data, click the ellipses box to the right of the mass detector drop down box. This will open a parameter window where the noise can be set and previewed. The blue lines in Fig. 8 represent those that have not been selected and in red the ones that have. Previewing different noise levels can help determine what level is best to set the noise at. Noise should be roughly the same level across.

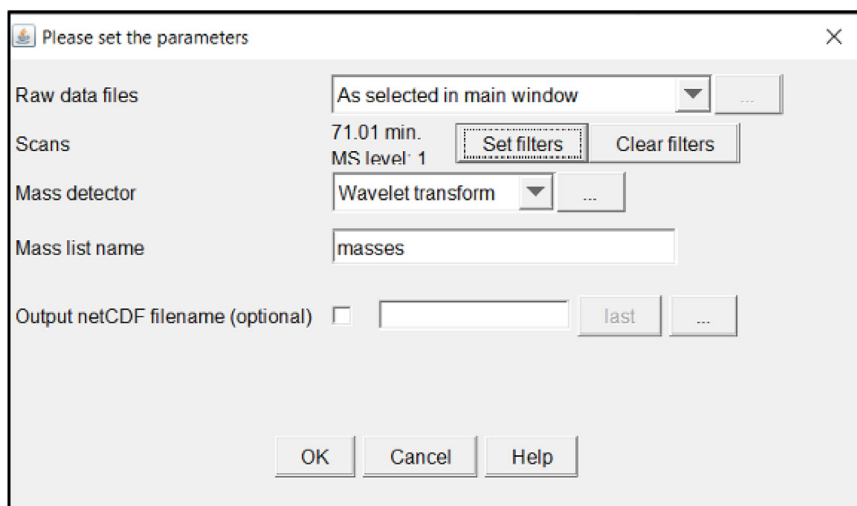


Fig. 7. Mass detection parameter window.

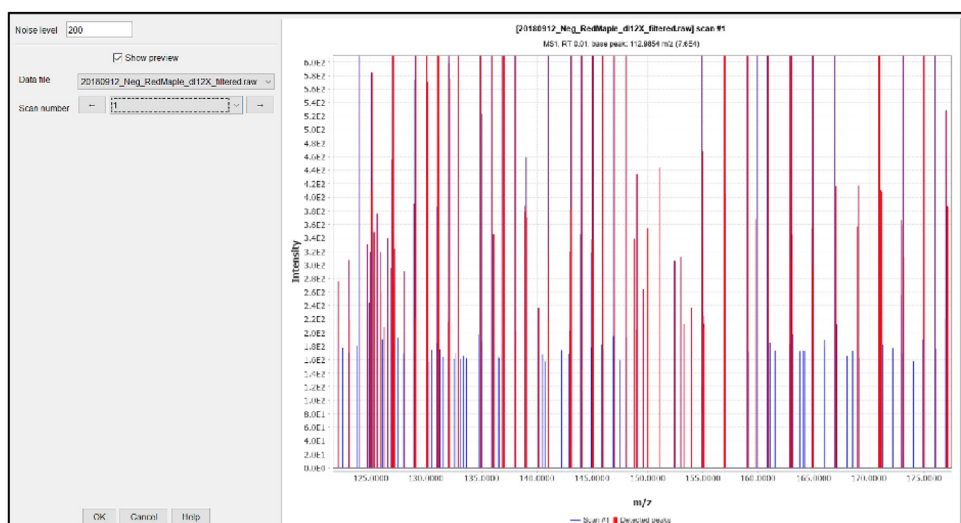


Fig. 8. Setting and previewing noise level.

Mass list name:

The mass list name is what the name of the detected mass file will be outputted as after the feature is run it will generate a list of points for each scan in the sample. The program auto inputs *masses* as the title.

Once all parameters are set appropriately for the data set, click the *OK* button to run the program. The program progress status is shown in the bottom right side of the window. When mass detection is complete a green check will appear over the file icon next to raw data on the left-hand side of the window. After the process is finished, click on the plus sign box to the left of each data file, this will display a list of the profile spectra. Then click on the plus sign box to the left of each profile spectrum and this will display the centroid spectrum labelled as the mass list name *masses*. Double click on the

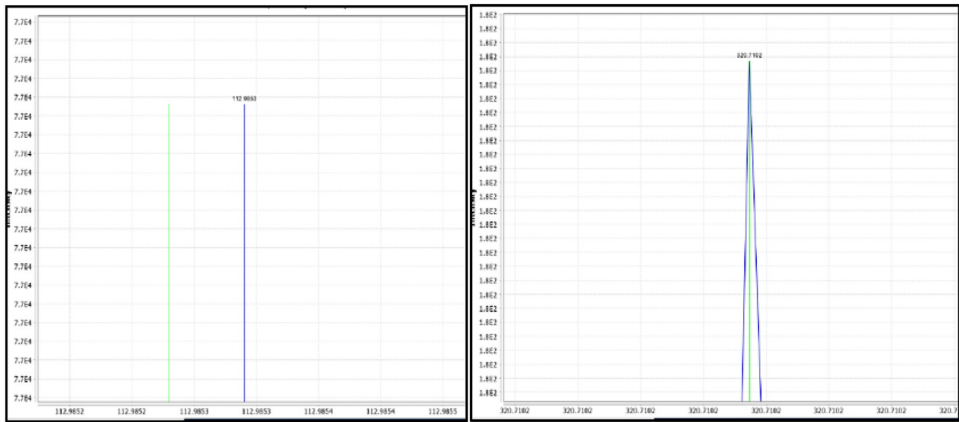


Fig. 9. Checking mass detection feature with raw data.

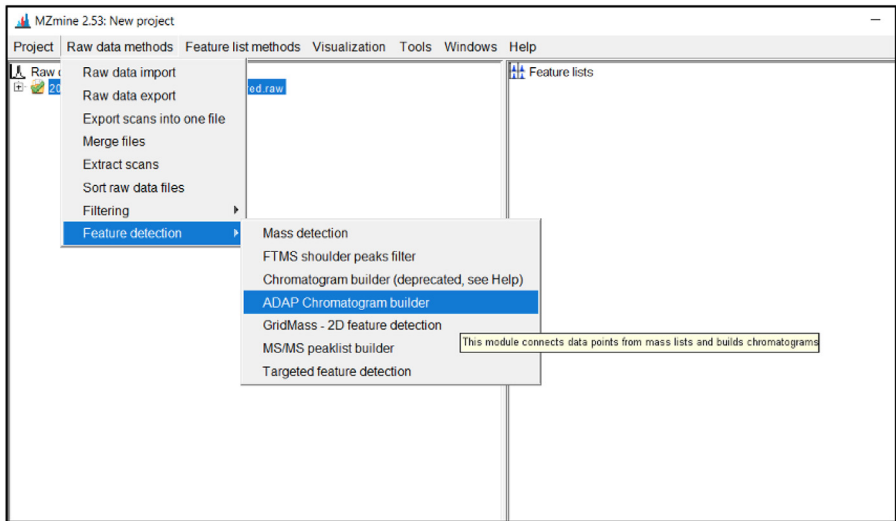


Fig. 10. Screenshot to run ADAP chromatogram builder.

masses brings up a window that shows a profile spectrum in blue and the centroid masses in green. The accuracy of the mass detection step can be evaluated by comparing where the centroid spectrum and the profile spectrum line up. The goal is to have the green centroid line intersect the peak of the profile spectra in blue, or if there is no peak, for the green line to run closely parallel to the blue line. An example is shown in Fig. 9.

ADAP chromatogram builder:

The next step is to use the ADAP chromatogram builder feature, which builds extracted ion chromatograms by taking the mass lists and builds chromatograms for each mass that can be detected continuously over spectrometry scans.

To run this feature, click *Raw data methods* → *peak detection* → *ADAP chromatogram builder* as shown in Fig. 10.

Raw data files: 20180912_Neg_RedMapl... As selected in main window

Scans: MS level: 1 Set filters Clear filters

Mass list: masses Choose...

Min group size in # of scans: 5

Group intensity threshold:

Min highest intensity: 25

m/z tolerance: 0.04 m/z or 5 ppm

Suffix: chromatograms

Fig. 11. Example of ADAP chromatogram building parameter.

This will display a parameter window for the ADAP chromatogram building. The window of parameters is shown in Fig. 11 with descriptions of each parameter section following.

Scans:

Enter MS level 1 and the scan filters will be the same from what was set in the mass detection step of the process, scan parameters are not required unless needed to manipulate the raw data set.

Mass list:

Enter “masses” or whatever was set as the title for the mass detection results to be used for this program

Min group size in # of scans:

Minimum scan span over which some peak in the chromatogram must have (continuous) points above the noise level to be recognized as a chromatogram. The chromatography system will determine the optimal value. By evaluating the raw data and it is possible to observe the typical time span of chromatographic peaks and select appropriate values. Typically, a value of 3 to 5 will provide accurate results, 5 will work in most situations. The min group size number of scans detects peaks, so a setting at 3 will detect more peaks but with less confidence than a selection of 5. Selecting 3 can be helpful for small peaks that are eluted very quickly and not observed in many scans.

Group intensity threshold:

This parameter is the intensity value for which intensities greater than this value can contribute to the minimumScanSpan count. This value should be set to match the noise level (determined by analyzing the chromatogram in the mass detection step) exactly to keep the data set consistent as the chromatogram is being constructed.

Min highest intensity:

Points below this intensity will not be considered in starting a new chromatogram. The minimum highest intensity should be higher than the noise level. If 200 is entered for noise and intensity threshold, 300–400 would be a good minimum highest intensity, this can be raised to as high as 20,000 based on the minimum intensity used at Michigan Tech in past toxicology projects. Again, this can be adjusted based on results. This parameter should stay consistent between ESI negative and ESI positive datasets. It should be noted that the noise levels can be different between the two ionization modes sometimes, so this may not always be the best option.

m/z tolerance:

Maximum allowed difference between two m/z values to be considered the same chromatogram. The value is specified both as absolute tolerance (in m/z) and relative tolerance (in ppm). The tolerance range is calculated using maximum of the absolute and relative tolerances. If there is greater confidence in using either m/z or ppm, any value set to 0.0 will not be used. Typical values for m/z tolerance are 0.015 m/z and 3 ppm.











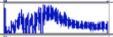











ID	Average		Identity	Comment	Peak shape	20180912_Neg_RedMaple_d112X_filtered.raw		
	m/z	RT				Status	Height	Area
1	100.0037	0.12					1.5E3	1.0E6
2	100.1465	1.90					8.5E2	7.7E5
3	100.9335	69.99					7.0E4	6.6E7
4	101.0246	1.77					5.2E4	1.5E6
5	101.9336	69.99					6.1E3	5.3E6
6	102.0561	1.73					2.0E4	8.5E5
7	102.9567	27.24					4.5E3	7.3E6
8	103.0038	2.02					1.1E4	2.8E6
9	103.9201	0.72					9.1E2	1.2E6
10	104.0353	1.70					1.1E4	8.2E5
11	104.9538	31.20					1.5E3	2.3E6

Fig. 12. Screenshot of sample output from ADAP Chromatogram builder.

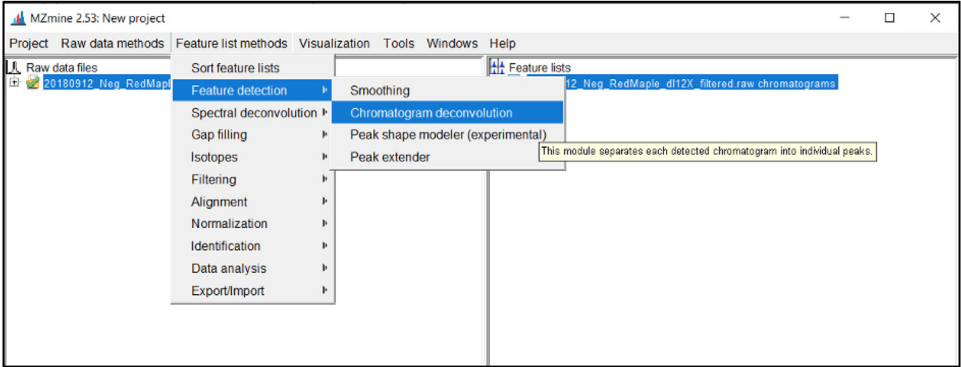


Fig. 13. Screenshot to run ADAP peak deconvolution.

Fig. 12 shows a sample output results after running the ADAP Chromatogram builder. It is important to note, each row represents a possible compound ID, however it is impossible to analyze when there are multiple peaks present in the “peak shape” column. The deconvolution process in the next step is critical to assign each ID row with a single peak.

ADAP peak deconvolution:

Next, the peaks from all chromatograms will be detected. Deconvolution is needed to separate the previously constructed chromatograms that span the entire duration. To run ADAP peak deconvolution select the chromatogram files on the left side of the screen and click *Peak list methods* → *peak detection* → *chromatogram deconvolution* as shown in Fig. 13.

This will open a window with parameters for the deconvolution step as shown in Fig. 14. Make sure the desired data lists are selected and add *deconvoluted* as the suffix. From the drop-down box select *Wavelets (ADAP)*. The methodology used for this project follows the Wavelet ADAP process which is the most recent and comprehensive method developed [27].

After selecting Wavelets from the dropdown box, click on the ellipses to the right. This will open the algorithm parameter window shown in Fig. 15.

Below are descriptions of each algorithm parameter shown in Fig. 16.

Feature lists	20180912_Neg_RedMapl...	As selected in main window
Suffix	deconvoluted	
Algorithm	Wavelets (ADAP)	
m/z center calculation	MEDIAN	
m/z range for MS2 scan pairing (Da)	<input type="checkbox"/>	
RT range for MS2 scan pairing (min)	<input type="checkbox"/>	
Remove original feature list	<input type="checkbox"/>	

Fig. 14. Parameter list for ADAP peak deconvolution.

S/N threshold	10	
S/N estimator	Intensity window SN	...
min feature height	10	
coefficient/area threshold	110	
Peak duration range	0.00	- 10.00
RT wavelet range	0.00	- 0.10

Fig. 15. Algorithm parameter window.

ID	Average		Identity	Comment	Peak shape	20180912_Neg_RedMaple_dl12X_filtered.raw		
	m/z	RT				Status	Height	Area
1	100.1282	36.61					1.8E2	6.1E2
2	100.1729	49.47					1.5E2	5.4E2
3	100.9333	14.86					4.1E2	1.1E3
4	101.0245	1.77					5.2E4	2.4E5
5	101.0286	67.17					2.1E2	8.3E2
6	102.0548	1.73					2.0E4	7.6E4
7	102.0640	65.74					1.7E2	5.2E2
8	102.0250	34.49					1.9E2	7.9E2

Fig. 16. Example deconvoluted peaks of chromatogram.

S/N threshold:

Signal to noise ratio threshold value greater than or equal to 7 will detect a small number of false peaks, this value compares peaks as 10x that of noise. 10 is a typical value to use for the Signal to noise threshold but can be adjusted based on results desired.

S/N estimator:

User can choose one of two estimators of the signal-to-noise ratio

1. The intensity window SN, which is tested on LC-MS datasets, utilizes peak height as signal level and standard deviation of intensities around the peak as the noise level;
This is the Signal to Noise estimator used for this project because we are working with LC-MS data and not GC-MS.

2. Wavelet Coeff. SN, which is tested on the GC-MS datasets, utilizes continuous wavelet transform coefficients in order to estimate the signal and noise ratio. There is an analogous approach implemented in the R-package wmtsa [28,27].

Min feature height:

Minimum height of a feature. Should be the same, or similar to, the value - min start intensity- set in the chromatogram building. This value should be set to match the noise level and group intensity threshold used throughout running the program. Again, this value will change depending on the noise determined for a certain data set.

Coefficient/Area threshold:

This is a threshold for the maximum coefficient (inner product) divided by the area under the curve of the feature. Filters out bad peaks. This number must be chosen by looking at examples using the show preview button at the bottom of the window. This is the best coefficient found by taking the inner product of the wavelet at the best scale and the peak, and then dividing by the area under the peak. This value should also be similar to the RT Wavelet range set below. Somewhere around 1 or 2 should work for most data sets but can range up to 100.

Peak duration range:

Peak duration range is the range of acceptable peak lengths. Analyze chromatogram to determine how long peaks last. Auto entered at 0–10 adjusted down to 0.0–1.0 after analyzing the chromatogram, this will vary from dataset to dataset. The smaller the range the more confidence in the identified peaks.

RT wavelet range:

Upper and lower bounds of retention times to be used for setting the wavelet scales. Choose a range that is similar to the range of peak widths expected to be found from the data. Auto entered at 0-0.1, the auto entered range typically works well but can be increased to 1 or 2 when the data calls for it.

Once all algorithm parameters are set, click OK to return to the deconvolution parameter page. Then finish inputting those parameters shown below.

m/z center calculation:

Select "AUTO"

m/z range for MS2 scan pairing (Da):

Not entered/needed

RT range for MS2 scan pairing (min):

Not entered/needed

After the peak deconvolution program has run, a list of the chromatographic peaks will appear below the list of chromatograms in the *Peak lists* window on the right side of the screen. Fig. 16 shows an example of the deconvoluted peaks, this is updated from the detection step because peaks are now isolated and there is one clear peak for each row ID.

Export data as csv file

The data is now at the point where it can be exported to a .csv file so that it may be used in MFAssign R for formula assignment, identification, and interpretation. To export a csv, select the desired data and click *Feature List Methods* → *Export/Import* → *Export .csv*

This will display a parameter setting screen shown in Fig. 17.

Feature lists: this is the same in all parameter windows, make sure the desired set of peaks/data is selected.

Filename: This is where the .csv will be stored; it is recommended to attach it to a blank excel sheet so it can be viewed and manipulated. It will still be stored as a .csv file. Make sure it is saved in a hard drive folder if using it in MFAssignR

Field separator: It will auto enter a comma in this field, change if needed.

Export common elements: The MFAssignR code requires row m/z, and optionally, row retention time data (useful for LC-MS analysis), select these items under the common elements section.

Fig. 17. Export sample parameter page.

Export data file elements: The MFAssignR code also requires the Peak area data, select this item under the data file elements section.

Export quantitation results and other information: Unnecessary, leave this box unchecked.

Filter rows: Select “ALL”

Preparing data for MFAssignR:

Once the data from MZMine is successfully outputted into a .csv file, view the data in an Libre Office Calc [29] spreadsheet. Ensure the column order is as follows from left to right: row m/z, peak area, row retention time. This is the order needed for MFAssignR to process the data properly. Now ensure that the data is saved in a .csv file *not* .ods or .xlsx and save the data on a hard drive in a folder that has a defined pathway, this will be needed for the MFAssignR code.

MFAssignR assignment

The next section utilizes MFAssignR [21]. To properly run this software the user will need to install both the programming interpreter for the R language [22] as well as the SDK “R Studio” [23], for both of which installation guides can be found online. This software uses multiple steps to assign formulas to the filtered mass spectrometry data created in the last step by MZMine. These steps, outlined in Fig. 1, consist of noise assessment, isotope identification, assignment, and recalibration. Blue text in this paper indicates direct code line from the MFAssignR package [21].

Install packages:

Installing packages is necessary to run the program. This only needs to be done once, can be commented out after they have been installed once. Lines can be “commented out” by inserting “#” in front of the desired line of code. A list of the packages needed for this program are listed below.

```
install.packages("dplyr")
install.packages("tidyr")
```

```
install.packages("colorRamps")
install.packages("devtools")
install.packages("ggplot2")
install.packages("rmarkdown")
```

The next step is to set the working directory, this will be the folder that the MFAssignR package was saved to on the user's computer. Will be different to what is shown below.

```
setwd("C:/Users/name/Drive/MFAssignR-master")
```

The next step is to write the call to install the MFAssignR program.

```
devtools::install("MFAssignR")
```

Data Loading:

Now loading the different necessary packages to do the formula assignment.

```
library(MFAssignR)
library(ggplot2)
library(dplyr)
library(tidyr)
```

Now set the working directory to where the .csv file produced from MZmine was stored and load the datafile. Must attach the .csv file produced from MZmine to the word "data" to be called on throughout the rest of the code.

```
setwd("C:/Users/lucyt/OneDrive/MFA Data")
Data <- read.csv("-ESI Final Compounds, Red Maple Leaf.csv")
```

Next step is to set the signal to noise ratio, this value allows the user to change the signal-to-noise ratio that will be multiplied by the estimated noise to determine the noise removal threshold. The SNRatio can be set from 0-10, but will likely be around 3. For the toxicity analysis project, the SNRatio was set to zero to ensure no legitimate peaks are mistaken for noise, since a majority of the noise was removed in the MZmine steps.

```
SNRatio <- 0
print("SNRatio")
0
```

Signal to noise assessment:

This is the signal to noise assessment section of the R markdown, demonstrating how to use the function KMDNoise()

```
Noise <-KMDNoise(Data, upper.y=0.3, lower.y=0.1)
```

This code shows how to extract the results of the KMDNoise() function so that they can be used

```
Noise [ ["KMD"]] #Plot showing the signal to noise estimation plot
KMDN <-Noise [ ["Noise"]] #Saving the estimated noise as a global variable in the environment
KMDN #Printing the noise so that it can be seen in the final report.
SNRatio = 0
SNplot(Data, cut = SNRatio * KMDN, mass = 319.1, window.x = 20, window.y = 10)
```

This will produce 3 output screens that are attached below. Fig. 18 shows the KMD signal to noise determination plot. This plot is used just to look at the noise threshold relative to the mass peaks and their intensities. The goal here is to have most of the light to dark blue dots to be between the red lines whose location is specified by the code in line with (upper.y = 0.3, lower.y = 0.1).

After analyzing Fig. 18 it is clear the noise settings (red lines) do not contain the clearly separated blue points, this is remedied by adjusting the upper and lower y limits. After adjusting, a noise determination plot such as the one in Fig. 19 is produced.

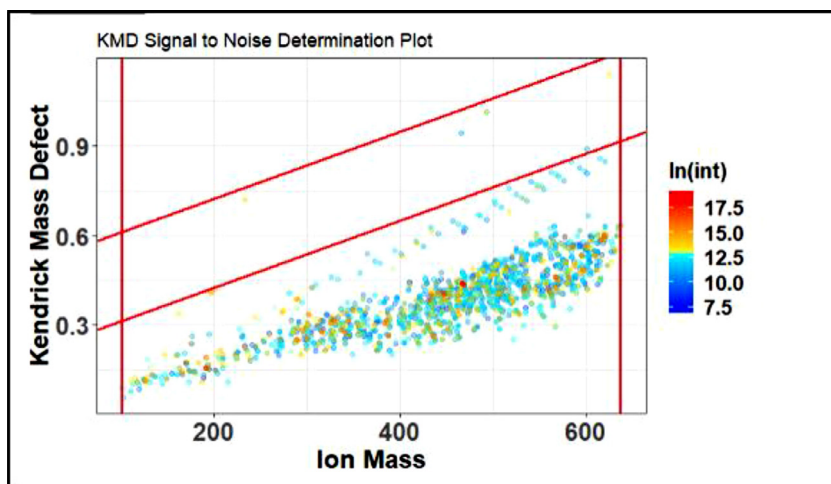


Fig. 18. Improper noise determination plot.

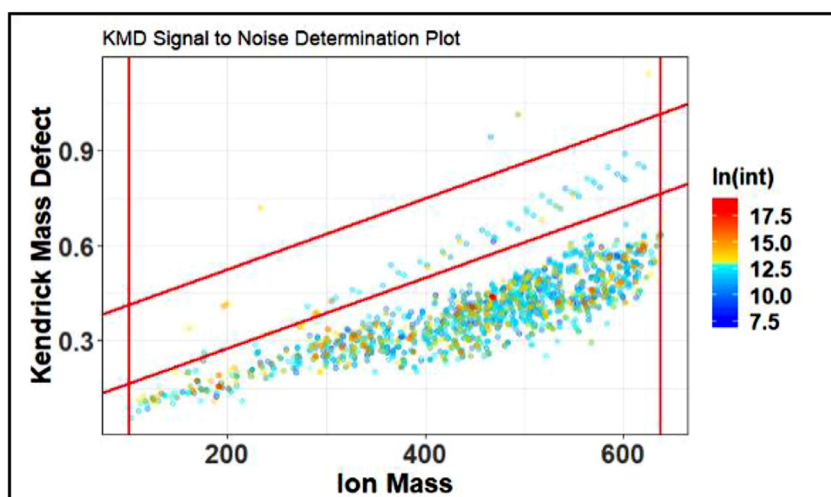


Fig. 19. Proper noise determination plot.

The next figure window that will be produced from the Signal to Noise Assessment step is the KMDN value. This is a numerical representation of the estimated noise. This value will be saved in the global environment and used for future steps of the code.

The last figure window that will be produced from this step is the noise index. This figure will change depending on what the SN Ratio is set to. If the SNRatio is set to 0, the visual will look like the one displayed in Fig. 20. There is nothing to analyze in the index if the SNRatio is set to zero since no peaks will be defined as "bad". Most of the noise has already been removed in MZMine2 in this case so it is not necessary to have a non-zero SNRatio, but in other situations (predominately direct infusion) a ratio of 3–10 is generally appropriate.

However, if the SNRatio is set to 1–10, the noise index will look more like the visual in Fig. 21. This figure will display the blue peaks as "good" and the red ones as "bad", the code will remove the peaks that are deemed "bad" from future steps of molecular formula assignments, so it is important

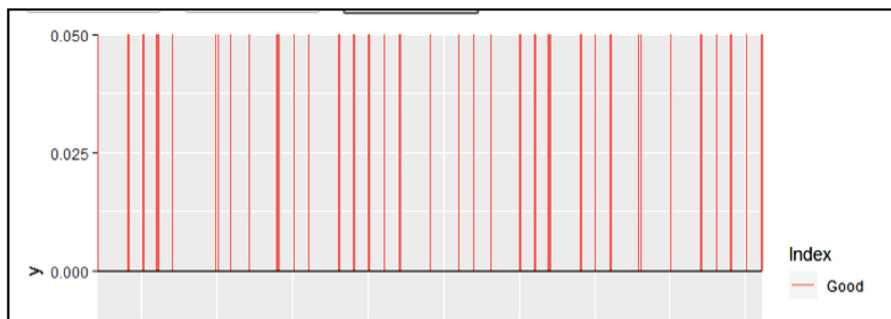


Fig. 20. Noise index with SNRatio set to zero.

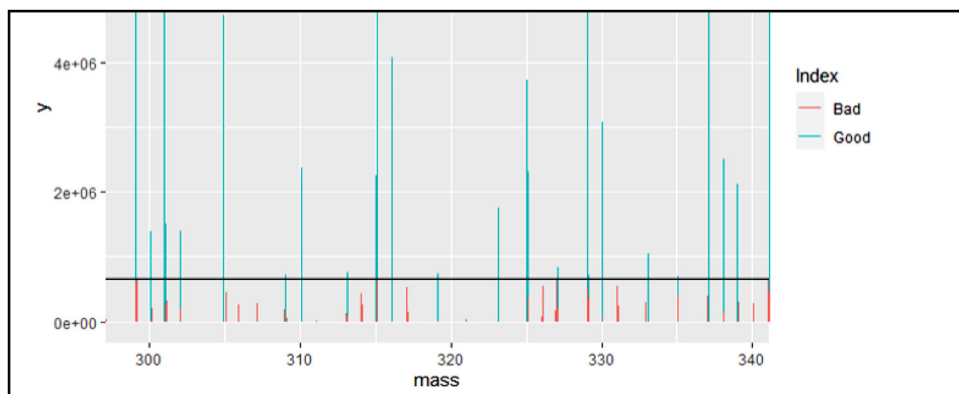


Fig. 21. Noise index with SNRatio set to three.

to ensure these peaks truly represent noise. Ensure this by looking at the level of the “noisy” peaks, they should all be close to the same level and remember, more peaks that are removed can mean less confidence in the final toxin results.

Isotope identification:

This section shows the usage of the `IsoFiltR()` function, which separates the single raw mass list into a list of likely monoisotopic masses (“Mono”) and likely poly isotopic masses (“Iso”). The first line of code sets “Isotopes” as the results of the `IsoFiltR` function. Parameters for this include “Sulferr” and “Carberr”. This is the allowed error level when filtering sulfur and carbon. Typically, 3 is a good value for this but can be changed based on the mass accuracy of the instrument being used. The following two lines simply extract the two resultant mass lists from `IsoFiltR` and label them “Mono” and “Iso” so that they can be used in later steps. There are no visual outputs for this section but extracted mass lists can be viewed in the upper right window/ the global environment as shown in [Fig. 22](#).

```
Isotopes <- IsoFiltR(Data, SN = SNRatio * KMDN, Sulferr = 3, Carberr = 3)
Mono <- Isotopes [ ["Mono"]]
Iso <- Isotopes [ ["Iso"]]
```

Prelim assignment:

Next is the preliminary assignment step, this will assign molecular formulas to the initial peaks before any recalibration is performed. Be sure this section of code is updated based on what the target assignments are. The following line shows how to use the CHO only version of formula assignment. It is typically done to find molecular formula series to be used in recalibration. The parameters given

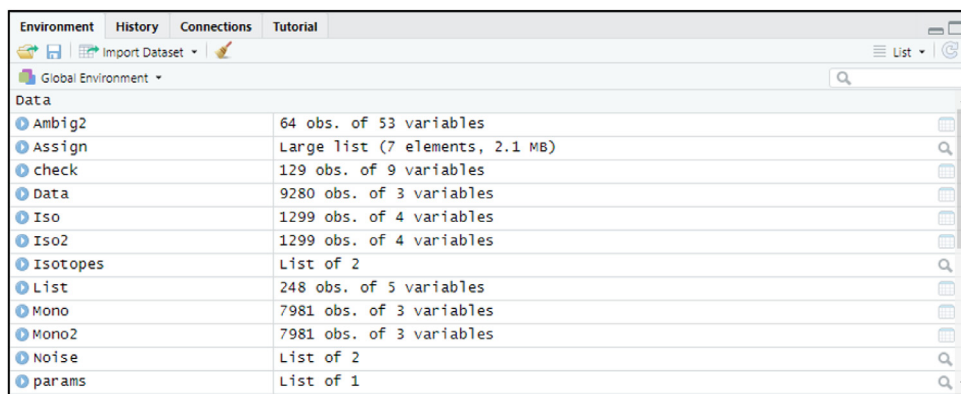


Fig. 22. Screenshot of global environment in RStudio.

Table 1

Preliminary assignment step parameters for negative and positive mode.

	Standard Parameters for Neg. Mode	Standard Parameters for Pos. Mode
ionMode	"neg"	"pos"
lowMW	50	50
highMW	1000	1000
ppm_err	3	3
H_Cmin	0.3	0.3
Omin	1	0
Mx	Not Included	1

"Mx" represents the sodium adduct that will be detected only in ESI Positive data and therefore is not necessary to include in the negative data processing. The following lines of code extract the outputs from the MFAssignCHO_RMD() function in the step above. This includes unambiguous and ambiguous molecular formula assignments as well as indexing the unassigned mass. The outputs will include 4 plots and 3 data frames.

in the code are standard for ESI negative mode. [Table 1](#) shows standard parameters for both negative and positive modes.

```
Assign <- MFAssignCHO_RMD(Mono, Iso, ionMode = "neg", lowMW = 50, highMW = 1000,
  ppm_err = 3, H_Cmin = 0.3, Omin = 1,
  HetCut = "off", NMScut = "on", SN = SNRatio*KMDN)
Unambig1 <- Assign [ ["Unambig"]] #Unambiguous molecular formula assignments
Ambig1 <- Assign [ ["Ambig"]] #Ambiguous molecular formula assignments
Unassigned <- Assign [ ["None"]] #Unassigned masses
Plot1 <- Assign [ ["MSAssign"]] #Mass spectrum showing which peaks are assigned and
unassigned in the spectrum
Plot2 <- Assign [ ["Error"]] #Plot showing the error trend relative to mass for
assignments
Plot3 <- Assign [ ["MSgroups"]] #Mass spectrum showing the assigned molecular formulas
Plot4 <- Assign [ ["VK"]] #O/C vs H/C plot showing the assigned molecular formulas
Plot1
Plot2
Plot3
Plot4
```

The next lines of code are included to clear up some of the memory to keep the markdown running as fast as possible.

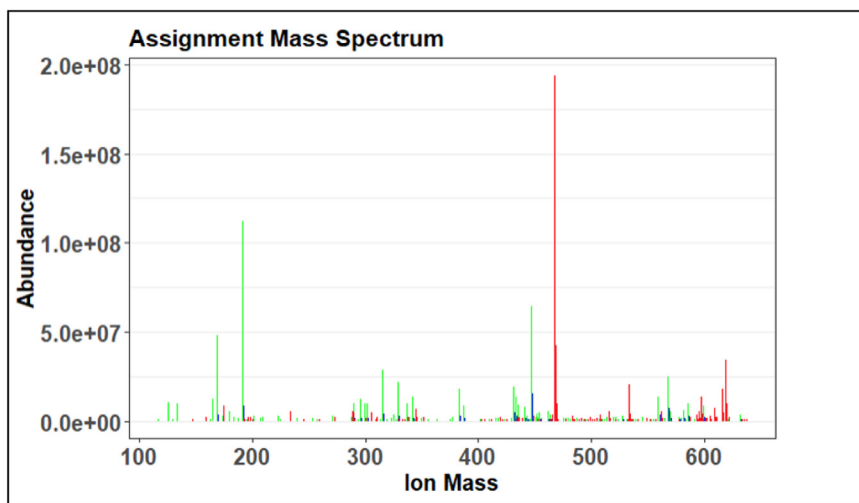


Fig. 23. Example of an assignment mass spectrum.

```
rm(Plot1)
rm(Plot2)
rm(Plot3)
rm(Plot4)
rm(Unassigned)
rm(Ambig1)
gc()
```

The first visual produced by the preliminary assignment code is the assignment mass spectrum, an example is shown in Fig. 23. On this mass spectrum, red indicates an ambiguous or undefined assignment, green indicates an unambiguous or defined molecular formula assignment, and blue indicates a poly isotope. The goal of this visual would be to see as many green peaks as possible, however this is the preliminary step so it is expected to have more red peaks than will be in the final assignment step.

The next visual produced from this step of preliminary assignment is of the error plot, an example is shown in Fig. 24. The error should demonstrate some sort of trend that may vary depending on the instrument being used, for the preliminary error plot, trends are good. This example has a reasonable trend, though there are a couple point where the trend may be exceeding the error limits placed on the assignment. The error is raised by altering parameters within the KMDN step described above.

The following two visuals shown in Figs. 25 and 26 are alternate views of the assignment mass spectrum and are useful to distinguish different species groups such as CH, CHO, CHNO, etc. When more chemicals are added to the assignment code, these figures can determine if the added chemicals are causing more peaks to be defined as ambiguous.

Recalibration

The recalibration step requires the most user input. If available, representative samples should be run first to determine the set function “recalibrants” with confidence. The first two lines of code before the “check” step are optional but are useful for setting up recalibration in LC-MS runs because they remove duplicate mass/formula combinations (from isomers) that cause issues in recalibration calculations. Typically, these lines are set up to retain the most abundant example of each mass/formula combination. The goal of this is to remove all series that may have a Series Score less than 1, because a Series Score less than one causes issues for recalibration. This filtering step only impacts the selection of recalibrant series and the calculation of recalibration correction terms,

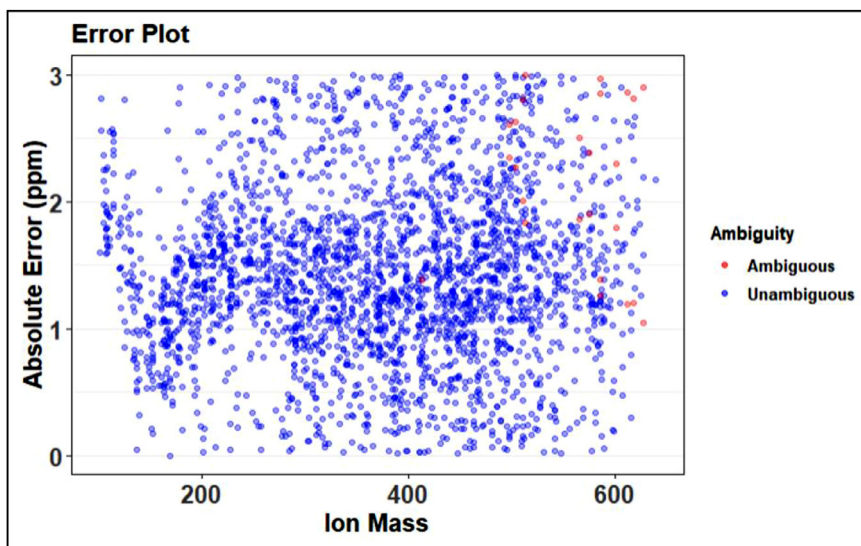


Fig. 24. Example of error plot from preliminary assignment.

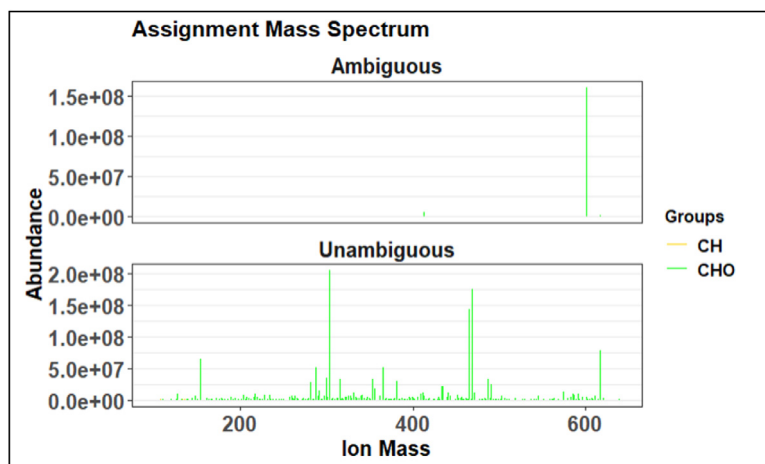


Fig. 25. Example of the second view of the assignment mass spectrum from preliminary assignment.

all masses will be recalibrated in the function. This makes it easier to select formulas that cover the entire mass range spectrum.

```

"{r, echo = FALSE, message = FALSE, warning = FALSE}
Unambig1 <- Unambig1 [order(-Unambig1$abundance),]
Unambig1 <- Unambig1 %>% distinct(formula, .keep_all=TRUE)
check <- RecallList(Unambig1)

```

After running the above section of code, the user must run a qualitative check of "recalibrant" series and mass recalibration. This is done by examining the "check" page that can be found in the upper right-hand global workspace, also found in Fig. 22. After selecting the check, the table comes up in the upper right-hand window as shown in Fig. 27.

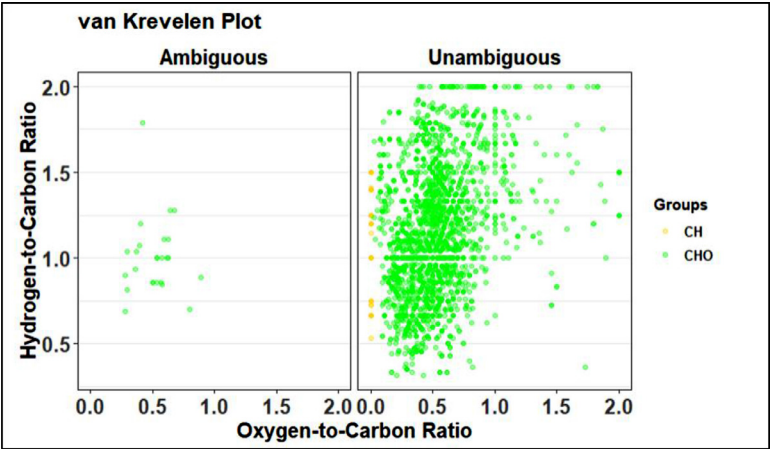


Fig. 26. Example of a van Krevelen plot from preliminary assignment step.

	Series	Number Observed	Series Index	Mass Range	Tall Peak	Abundance Score	Peak Score	Peak Distance	Series Score
54	O10_Na_6	11	1	313.017-467.188	355.0632	230.930148	0.64034455	6.006688	1.092019
43	O11_Na_7	9	3	369.042-551.246	369.0425	-24.491824	0.37770629	5.005590	1.557175
68	O12_Na_6	9	2	373.037-541.225	387.0532	-3.937070	0.17170577	1.001120	1.445937
16	O3_H_7	8	9	163.039-317.211	219.1012	0.000000	0.56465450	3.003337	1.501536
31	O7_Na_1	8	4	233.063-387.235	233.0631	31.996652	1.08918637	2.002209	1.501536
46	O10_Na_7	8	8	339.032-465.173	353.0476	941.623606	0.04270674	2.002249	1.251259
48	O9_Na_6	8	7	311.037-409.146	339.0684	128.669065	0.20033632	4.004465	1.000973
99	O6_Na_2	8	5	247.042-415.23	415.2297	43.384899	0.15375944	3.003347	1.626679
100	O6_Na_3	8	6	231.011-441.245	413.2143	159.067595	0.79910256	5.005603	2.002089
7	O4_H_7	7	14	179.034-263.127	179.0337	-34.520674	0.34758085	5.005585	1.000949
10	O4_H_6	7	12	153.018-265.143	153.0181	810.978808	1.31652027	8.008922	1.286990
23	O5_H_10	7	18	231.028-343.154	273.0754	1.171872	0.35859106	5.005583	1.287000
38	O6_H_9	7	17	235.023-361.164	291.0860	219.069436	0.33180356	2.002245	1.430010
45	O7_Na_3	7	11	215.016-425.251	355.1722	149.846357	0.24614682	10.011156	2.288112
58	O8_Na_1	7	10	221.027-389.216	347.1671	62.583461	0.20741481	4.004416	1.859071
67	O5_H_11	7	20	243.028-537.357	341.1378	-67.278238	0.47207547	14.015679	3.146214
78	O13_H_12	7	21	411.019-509.128	467.0815	14.347345	0.47505563	1.001110	1.143969
91	O10_Na_8	7	15	337.017-505.204	463.1571	-37.077041	0.18113263	2.002274	1.859051

Fig. 27. View of the “check” table within the recalibration step of code.

This is the data that will be analyzed to determine the series used to create the recalibrated mass spectrum. The overall goal is to get the longest continuous spectrum of ion masses covered by the series selected. This can be visually checked in Fig. 28, the selected spectrum is indicated by the blue color.

Selecting series is determined on a set of parameters found in the column section of the “check” table.

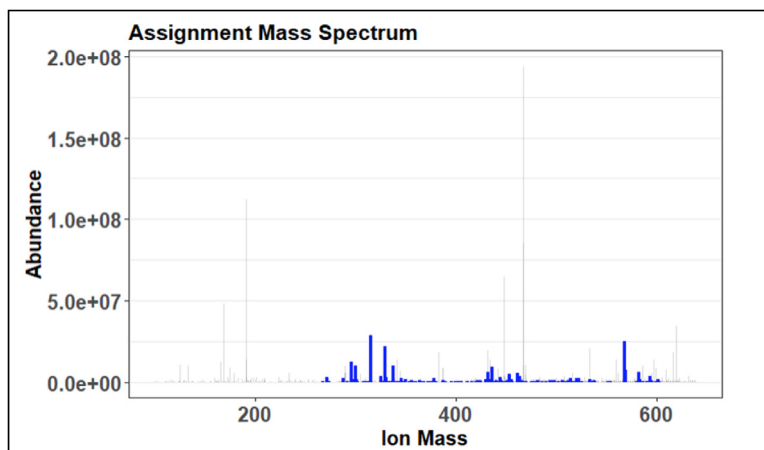


Fig. 28. Sample Assignment mass spectrum showing range of ion mass assignment for the recalibration step.

Table 2

Parameters for selecting series in recalibration.

Parameter Name	Ideal Value	Notes
Number Observed	Higher	*sort rows highest to lowest
Mass Range	Largest	*if able, select larger mass ranges instead of smaller
Abundance Score	Positive	*optimally positive
Peak Score	Close to 0	*some room (can be above or below)
Peak Distance	Close to 1	*Least Important, can be closer to 2
Series Score	Close to 1	*Critical, do not go below 1

```

138     tox_names = set()
139     esi = pd.read_csv(f'./{sign} {compound}.csv', encoding='unicode_escape')
140     tox = pd.read_csv('./OpenFoodTox.csv', encoding='unicode_escape')\
141           .dropna(axis=0, subset=['MOLECULARFORMULA', 'COM_NAME']).drop_duplicates()
142     tox_names.update(tox.COM_NAME[tox.MOLECULARFORMULA.isin(esi.formula)])

```

Fig. 29. View of the “check” table within the recalibration step of code.

Examine the “check” table, in the environment to select 5 or 6 different chemicals to use in full assignment step. When selecting formulas, want mass ranges that cover a large range of the total mass spectrum (need to know highest and lowest mass range) while also examining the following parameters outlined in Table 2.

When formulas are selected, insert them in the following code of the Recal function. Depending on confidence of formulas selected and peak and series scores, the `mzRange` is typically around 15 but can be increased if selected series are not completing the necessary range coverage. After the following lines of code are run the plot in Fig. 29 will be produced and analyzed, in some cases the selected series will not provide a strong enough range and the code will error. In this case new series must be selected.

```

Test <- Recal(Unambig1, peaks = Mono, isopeaks = Iso, mzRange = 15, mode = "neg",
  SN = SNRatio*KMDN, series1 = "O7_H_6", series2 = "O9_H_5", series3 = "O6_H_7",
  series4 = "O9_H_10", series5 = "O9_H_12", series6 = "O12_H_11", series7 = "O5_H_5")
Plot <- Test [ ["Plot"]]
Plot #This plot can take time to generate

```

```

Mono2 <- Test [ ["Mono"]] #Recalibrated monoisotopic mass list
Iso2 <- Test [ ["Iso"]] #Recalibrated isotopic mass lists
List <- Test [ ["RecalList"]] #The list of formulas/masses used as recalibrants
print("mzRange")
15

```

Full assignment

It is important that this section of code is updated with all necessary elements depending on assignment goals.

Use of MFAssign_RMD() for molecular formula assignment. The notable differences between the full and preliminary assignment steps are the option to add chemicals (Sx, Nx, Px, Clx, etc.) to the assignment process, and the isotope error (set to 3), and having Sulfur check and MSMS set to "on". Following the code below, besides what was indicated in [Table 1](#), everything remains the same in negative and positive modes.

```

Assign <- MFAssign_RMD(Mono2, Iso2, ionMode = "neg", lowMW = 50, highMW = 1000, Sx = 1,
  Nx = 3, ppm_err = 1, iso_err = 1, H_Cmin = 0.3, Omin = 1, SulfCheck = "on",
  HetCut = "off", NMScut = "on", SN = SNRatio*KMDN, MSMS = "on")

```

It is important to note that the addition of more elements to formula assignment beyond C, H, and O can lead to increased ambiguity in the formula assignments due to more chemically feasible combinations of elements being possible, particularly at high masses [34]. MFAssign has been written to include data-dependent pathways to decrease the ambiguity in formula assignments using formula extension "spiderwebs", which decreases the likelihood of ambiguous assignments, but care should be taken when including multiple non-oxygen heteroatoms to be confident in the formula assignments. To mitigate this response, the user can run the same data set through the code more than once, adding different chemicals (of increasing error) to the full assignment step each time. It is recommended to run through the data once with only CHO and no user added chemicals, this should result in no ambiguous assignments and offers a solid baseline of confirmed formulas that can be compared to the formulas produced when more chemicals are added. A good work progress for this method is to run through CHO first, then add nitrogen for the CHNO assignment, then add Sulfur and chlorine. When checking chlorine assignments, check for the isotope Cl37 and confirm there is a good peak at the correct retention time.

Once as many formulas as possible are assigned and verified, the data can be run through once more with the addition of phosphorus. Phosphorus is difficult to assign because there are no isotopes, so it can swing the formula assignment results. When analyzing phosphorus assignments some unreasonable groups such as PNO2 will be present, these can be excluded from the confirmed formula assignment. This is because of the lack of enough oxygen to support what should likely be a phosphate group given the presence of phosphorus. This "staging" method may not be necessary for all data sets, some will produce satisfactory results with all chemicals included in the first run. Overall, when the output formula assignments are reviewed if any seem unreasonable they can be confirmed by comparing the mass in the data with the theoretical "exact mass" of that compound with an error range of around 3 ppm. The key is to always be critical of formula assignments from a chemical feasibility angle based on knowledge of the sample and analytical method, particularly when users are first getting started.

Extraction of data from the MFAssign_RMD() function. It has the same format as the MFAssign_CHO_RMD() function. The code for outputting the final visuals and molecular formula assignments is below:

```

Unambig2 <- Assign [ ["Unambig"]]
Ambig2 <- Assign [ ["Ambig"]]
Unassigned2 <- Assign [ ["None"]]
Plot1 <- Assign [ ["MSAssign"]]
Plot2 <- Assign [ ["Error"]]
Plot3 <- Assign [ ["MSgroups"]]

```

```
Plot4 <- Assign [ ["VK"]]
Plot1
Plot2
Plot3
Plot4
```

Plots 1 through 4 match the ones produced in the preliminary assignment step shown in [Figs. 24–27](#). Progress made in the recalibration step can be seen by comparing two of the same plots in prelim and full assignment steps. Changes that are expected are less peaks in the ambiguous graphs and more in the unambiguous. The error plot should have no trend and the van Krevelen plot should be defined.

Final step is to write the output csv code. This can be done by the following line of code for the desired data. Typically, the final results will be stored under the name “Unambig2” if the results call for it or if the user would like to analyze the ambiguous assignments as well, “Ambig2” can also be outputted as long as “Ambig=On” is specified in the MFAssign_RMD section of code. These object names can be changed by the user as desired.

```
write.csv(Unambig2, "MFARedMapleLeafprelim.csv")
```

ToxAssign filtering process

ToxAssign [\[26\]](#) is a tool developed by that utilizes python data mining tools, the NIH’s PubChem API tool [\[30\]](#), and the Open Food Tox Database (OpenFoodTox) published by European Food Safety Authority [\[31,32\]](#). The European Food Safety Authority’s scientists have produced risk assessments for more than 4950 substances. Each substance, which has been evaluated has a summary of its effect on human health. In addition, based on the relevant legislation and intended uses, there can also be animal health and ecological hazard assessments provided as well. All of this information is collected and structured into the European Food Safety Authority’s chemical hazards database: OpenFoodTox. OpenFoodTox is fully open source data meant for substance characterization. For researchers it also provides links to other output including background European legislation. Finally, OpenFoodTox provides a summary of the available critical toxicological endpoints and any reference values.

ToxAssign is broken into five modules, one module that automates execution and the other four that perform specific operations. The automation tool creates a directory for each sample, calls the PubChem module, toxic filter module, match module, the merge module three times, then the toxic filter module again. The PubChem module first matches potential toxic formulas from the list of data from MFAssignR against the local toxic table, then retrieves the full records from PubChem, then sorts them based on chosen records and writes them to files. The toxic filter module takes the toxic records and filters the records by acute toxicity and writes them to file. The match module matches any records not found on PubChem against a local table of records and writes the found records and still unfound records to file. Finally, the merge module takes the files from each sample and merges them together then writes them to file.

To install this tool, the user will need to go to the Github page for this project [\[26\]](#), where the instructions for installation, operation, and output are detailed. As well the user may need to download and install the python language [\[25\]](#), for which guides can be found in the reference below. To tell whether it is needed or not, open a terminal and type “python3”, if there is no error then python is already installed. Otherwise, go to the reference below, navigate to the downloads page, and download the proper install for the computer operating system the operator is using.

PubChem:

As is demonstrated in [Fig. 1](#), the first module to be called is the PubChem module. This module first matches the formulae given out by MFAssignR and matches them against a local table of potentially toxic compounds as shown in [Fig. 29](#).

Then, in the same way as [Fig. 31](#), the module queries PubChem with the compound name to get the CID or list of CIDs that match that compound. Next, as demonstrated in [Fig. 30](#), the module uses these retrieved CIDs to get the full records for each compound from the PubChem database, waiting a fifth of a second between each to not overload PubChem’s servers.


```
response = req.get(
    "https://pubchem.ncbi.nlm.nih.gov/rest/pug/compound/name/{compound}/cids/JSON".format(compound=value))
time.sleep(.2)
cids = json.loads(response.text)
```

Fig. 30. View of the “check” table within the recalibration step of code.

```
if record["Record"]["Section"][i]["TOCHHeading"] == "Toxicity":
    toxicrecordcheck(record, toxic)
elif record["Record"]["Section"][i]["TOCHHeading"] == "Safety and Hazards":
    checked = safetyhazardcheck(record, toxic, i)
```

Fig. 31. View of the “check” table within the recalibration step of code.

```
22 data = json.loads(value["toxicity"])
23 for i in range(20):
24     if data[i]["String"][0:12] == "Acute Tox. 1":
25         set1.add(value["name"])
26         toxfound = 1
27     elif data[i]["String"][0:12] == "Acute Tox. 2" and toxfound > 1:
28         set2.add(value["name"])
29         toxfound = 2
30     elif data[i]["String"][0:12] == "Acute Tox. 3" and toxfound > 2:
31         set3.add(value["name"])
32         toxfound = 3
33     elif data[i]["String"][0:12] == "Acute Tox. 4" and toxfound > 3:
34         set4.add(value["name"])
35         toxfound = 4
```

Fig. 32. View of the “check” table within the recalibration step of code.

Next the module sorts the records by different fields that may or may not show up. To do so, the module loops through the fields of the record looking for either ones relating to toxicity or to food safety, checking in such a way as is shown in Fig. 31. When it finds one of these fields it then writes it to a data object that then gets written to file.

Toxic filter:

The next module to be ran by the automation, as shown in Fig. 1, is the toxic filter. This module takes the toxic record output by the PubChem module and filters it into five categories then writes them to a file to make understanding the data much easier. This is done by looping through each classification of the record and matching it against different acute toxic levels as in Fig. 32. The highest matching toxicity is then stored to a data object and finally written to a file.

Match:

Next, the match module matches all records that were not found in the PubChem database against a local table of previously unfound records. Any that are matched are outputted to a file along with their safety class and any that are not are put back into the unfound file to be added to the local table (Fig. 33).

Merge:

The final module is the merge module, which takes the files from individual samples and merges them together for help in decisions about further analysis of the set, as well as easier analysis for repeated analysis of the same sample. This is done first as shown in Fig. 34 by looping over all of the


```

11     unfound.name = unfound.name.astype(str)
12     found.compound = found.compound.astype(str)
13     found_comp = unfound.merge(found, right_on="compound", left_on="name", how="inner")
14     set_safe = found_comp
15     set_safe = set_safe[(set_safe.Safety == "safe") | (set_safe.Safety == "flavoring agent")
16                       | (set_safe.Safety == "fragrance") | (set_safe.Safety == "supplement")]
17     found_comp = found_comp[~found_comp.compound.isin(set_safe.compound)]
18     set_unfound = set()
19     set_unfound.update(unfound[~unfound.name.isin(found.compound)])

```

Fig. 33. View of the "check" table within the recalibration step of code.

```

60     for subdir in os.scandir("."): # merge files into one object
61         try:
62             os.chdir(f"./{subdir}") # move into data directory
63             # read in toxic file
64             unfound = pd.read_csv(f"./Set{infile}.txt", delimiter="\t", engine='python', names=["Compound"])
65             checked = unfound.merge(checked, how="outer") # merge unfound into checked

```

Fig. 34. View of the "check" table within the recalibration step of code.

```

66     unfound = pd.read_csv(f"./Set{infile}.txt", delimiter="\t", engine='python', names=["Compound"])
67     checked = unfound.merge(checked, how="outer")
68     finally:
69         # move out of dir at end
70         os.chdir("../src")

```

Fig. 35. View of the "check" table within the recalibration step of code.

directories and first moving into them, then merging the positive data set into a data object within the code.

Next, the module combines the negative data into the same object and finally moves out of the directory to begin again. As demonstrated in Fig. 35, this uses the pandas utility's merge function, which combines two sets of data that match along a given key. Because there is only one column in both datasets the key does not need to be specified.

At the end the data object is written to an output file with the shared name of the in file as its name. This operation is performed three times: once for the toxic data, once for the unfound data, and once for the unchecked data. These are all merged to help choices for which standards to use to sample the entire data set, to help with adding to the remove data file, and to help with checking records that did not have any fields in the PubChem database.

Toxic filter:

As is shown in Fig. 1, the toxic format module is run one more time, and this is to format the merged toxic file to make it easier to understand, again for the use of picking standards for further analysis through comparing the sample to chosen standards.

Validation against Compound Discoverer method

A goal of this paper was to replicate a procedure performed previously using an expensive and hard to obtain piece of software, Compound Discoverer, in an open source and easy to access and use format. This is to make the process not only less expensive but also more automated and easier to use, and thus, more accessible to under-resourced labs and more attractive to researchers to build upon this method in the future. To that end a part of this paper will be devoted to comparing this analysis to the previous analysis done using Compound Discoverer.

The findings of this paper in regard to the volume of toxic compounds, while larger, are also still lacking one step of validation in their analysis. Also, it should be pointed out to this end this process finds only toxic compounds in the sample, while Compound Discoverer works to find all compounds

matches as there is no structural information for these formula assignments included in the current comparison.

Future work

This method still needs further validation and automation to overcome the gaps in the PubChem database as well as differences in naming conventions between the European Food Safety Authority's database and the NIH's database. For now, this is resolved with a manual search of records that do not match and a local storage of those records in a csv. Future extensions to this free software could work to use other services and online databases to match the European Food Safety Authority's recorded name to the NIH's recorded name and store the record in local JSON storage. This could also be used to store information about the compounds that have already been searched, allowing for faster processing and less strain on PubChem. Another potential upgrade includes building in functionality to analyze fracturing runs and retention time of compounds against standards using the MZCloud database [33] to automate the verification and certification of compounds to make the process fully automated. Finally, although the purpose of this free and open source software toolchain was to identify potential toxic substances in potential alternative or resilient foods [35,36], however the approach has a much larger potential set of applications including identifying toxicity in botanical substances that could be used for food or additives, feed for animals or for cosmetic ingredients.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

Funding for this work was provided by the Defense Advanced Research Projects Agency ReSource program cooperative agreement HR00112020033, the NSF SBIR Phase II grant number: 1746480, and the Thompson Endowment. The content of the information does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred.

References

- [1] A. Turchin, D. Denkenberger, Global catastrophic and existential risks communication scale, *Futures* 102 (2018) 27–38.
- [2] D. Denkenberger, J.M. Pearce, *Feeding Everyone No Matter What: Managing Food Security After Global Catastrophe*, Academic Press, Cambridge, MA, USA, 2014 ISBN 978-0-12-802358-7.
- [3] D. Denkenberger, J.M. Pearce, Feeding everyone: Solving the food crisis in event of global catastrophes that kill crops or obscure the sun, *Futures* 72 (2015) 57–68.
- [4] S. Baum, D. Denkenberger, J.M. Pearce, Alternative foods as a solution to global food supply catastrophes, *Solutions* 7 (2016) 31–35.
- [5] D. Denkenberger, J. Pearce, Micronutrient availability in alternative foods during agricultural catastrophes, *Agriculture* 8 (2018) 169.
- [6] D.C. Denkenberger, J.M. Pearce, Cost-effectiveness of interventions for alternate food to address agricultural catastrophes globally, *Int. J. Disaster Risk Sci.* 7 (2016) 205–215.
- [7] D.C. Denkenberger, J.M. Pearce, Cost-effectiveness of interventions for alternate food in the United States to address agricultural catastrophes, *Int. J. Disaster Risk Reduct.* 27 (2018) 278–289.
- [8] D. Denkenberger, J. Pearce, A.R. Taylor, R. Black, Food without sun: Price and life-saving potential, *Foresight* 21 (2018) 118–129.
- [9] Leaf for life. Industrial leaf concentrate process (France). Available online: <https://www.leafforlife.org/PAGES/INDUSTRIALHTM> (accessed on 28 November 2018).
- [10] D. Kennedy, *Leaf Concentrate: A Field Guide for Small Scale Programs*; Leaf for Life, Interlachen, FL, USA, 1993.
- [11] E.L. Schymanski, H.P. Singer, J. Slobodnik, I.M. Ipolyi, P. Oswald, M. Krauss, T. Schulze, P. Haglund, T. Letzel, S. Grosse, et al., Non-target screening with high-resolution mass spectrometry: Critical review using a collaborative trial on water analysis, *Anal. Bioanal. Chem.* 407 (2015) 6237–6255.
- [12] Agroscope Toxic Plants-Phytotoxin (TPPT) Database. Available online: <https://www.agroscope.admin.ch/agroscope/en/home/publikationen/apps/tppt.html>.
- [13] J.M. Pearce, M. Khaksari, D. Denkenberger, Preliminary automated determination of edibility of alternative foods: non-targeted screening for toxins in red maple leaf concentrate, *Plants* 8 (5) (2019) 110.

- [14] E.L. Schymanski, J. Jeon, R. Gulde, K. Fenner, M. Ruff, H.P. Singer, J. Hollender, Identifying small molecules via high resolution mass spectrometry: communicating confidence, *Environ. Sci. Technol.* 48 (2014) 2097–2098.
- [15] L.W. Sumner, A. Amberg, D. Barrett, M.H. Beale, R. Beger, C.A. Daykin, T.W.M. Fan, O. Fiehn, R. Goodacre, J.L. Griffin, et al., Proposed minimum reporting standards for chemical analysis, *Metabol* 3 (2007) 211–221.
- [16] FIA; Forest Inventory and Analysis National Program-National Assessment-RPA; <https://www.fia.fs.fed.us/program-features/rpa/index.php>.
- [17] M. Johansson, S. Ekroth, O. Scheibner, M. Bromirski, How to screen and identify unexpected and unwanted compounds in food, Thermo Scientific: APPLICATION NOTE 65144. Available online: <https://assets.thermofisher.com/TFS-Assets/CMD/Application-Notes/an-65144-lc-ms-unexpected-unwanted-compounds-food-an65144-en.pdf>.
- [18] Thermo fisher scientific; US. Compound discoverer software; <https://www.thermofisher.com/us/en/home/industrial/mass-spectrometry/liquid-chromatography-mass-spectrometry-lc-ms/lc-ms-software/multi-omics-data-analysis/compound-discoverer-software.html>.
- [19] MZMine team; MZmine 2; <https://mzmine.github.io/>.
- [20] T. Pluskal, S. Castillo, A. Villar-Briones, M. Orešič, MZmine 2: Modular framework for processing, visualizing, and analyzing mass spectrometry-based molecular profile data, *BMC Bioinform.* 11 (2010) 395.
- [21] S.K. Schum, L.E. Brown, L.R. Mazzoleni, MFAssignR: molecular formula assignment software for ultrahigh resolution mass spectrometry analysis of environmental complex mixtures, *Environ. Res.* 191 (2020). <https://github.com/skschum/MFAssignR>.
- [22] R Core TeamR: A Language and Environment for Statistical Computing, R Foundation for Statistical Computing, Vienna, Austria, 2017 <https://www.R-project.org/>.
- [23] RStudio TeamRStudio: Integrated Development Environment for R, RStudio, PBC, Boston, MA, 2021. <http://www.rstudio.com/>.
- [24] NIH; PubChem; <https://pubchem.ncbi.nlm.nih.gov/>.
- [25] Rossum, G.V.; Python tutorial, Technical Report CS-R9526, Centrum voor Wiskunde en Informatica (CWI), Amsterdam, May 1995.”; <https://www.python.org/>.
- [26] Breuer, S.; ToxAssign; <https://github.com/swbreuer/ToxAssign>.
- [27] O.D. Myers, S.J. Sumner, S. Li, S. Barnes, X. Du, One step forward for reducing false positive and false negative compound identifications from mass spectrometry metabolomics data: new algorithms for constructing extracted ion chromatograms and detecting chromatographic peaks, *Anal. Chem.* 89 (17) (2017) 8696–8703.
- [28] Constantine, W.; Percival, D.; wmtsa: wavelet methods for time series analysis; R project <https://cran.r-project.org/src/contrib/Archive/wmtsa/>.
- [29] Libre Office Team; Libre office; <https://www.libreoffice.org/>.
- [30] S. Kim, J. Chen, T. Cheng, A. Gindulyte, J. He, S. He, Q. Li, B.A. Shoemaker, P.A. Thiessen, B. Yu, L. Zaslavsky, J. Zhang, E. Bolton, PubChem in 2021: new data content and improved web interfaces, *Nucleic Acids Res.* 49 (D1) (2019) D1388–D1395.
- [31] Kovarich; S.; Ciacci; A.; Baldin; R.; Carnesecchi; E.; Roncaglioni; A.; Mosttrag; A.; Dorne; J.L.C.M.; OpenFoodTox: EFSA's chemical hazards database (2021). (Version 4) [Data set]
- [32] Chemical Hazards Database (OpenFoodTox) | European Food Safety Authority (europa.eu) <https://www.efsa.europa.eu/en/data-report/chemical-hazards-database-openfoodtox>.
- [33] Mzcloud team; mzcloud; <https://www.mzcloud.org/>.
- [34] B.P. Koch, T. Dittmar, M. Witt, G. Kattner, Fundamentals of molecular formula assignment to ultrahigh resolution mass data of natural organic matter, *Anal. Chem.* 79 (2007) 6, doi:10.1021/ac061949s.
- [35] D. Denckenberger, A. Sandberg, R.J. Tieman, J.M. Pearce, Long-term cost-effectiveness of interventions for loss of electricity/industry compared to artificial general intelligence safety, *Eur. J. Futures Res.* 9 (1) (2021) 1–24.
- [36] A Tzachor, CE Richards, L. Holt, Future foods for risk-resilient diets, *Nat. Food* 2 (5) (2021) 326–329.