Advancing lipidomic bioinformatic technologies for the study of neurodegenerative diseases

Julie Fiala

A thesis submitted in partial fulfillment of the requirements for the
Master of Science degree in Biochemistry specialized in Bioinformatics

Department of Biochemistry, Immunology, and Microbiology
Faculty of Medicine
University of Ottawa

Table of contents

Abstract

As an emerging field, lipidomics still encounters methodological challenges. Indeed, as it is an –omics field, analysis of data is time and labour intensive, if the necessary and appropriate bioinformatics tools are not existent. Here I present two programs I developed to address the challenges of peak identification and peak annotation and filtering for a targeted lipidomics approach, which is not supported by available software. The first program, Lipid Identification Tool (LIT), consists of a stand-alone offline search engine, which provides a robust in silico database generated by linear equations based on lipid structures, containing information lacking on presently available online databases. The second program, Lipid Identification for Targeted Lipidomics (LITL), allows the annotation of HPLC-ESI-MS/MS MRM acquired spectral data in text-based format, comparing them to a library of identities, via retention time and mass-over-charge of detected ions, and to easily export the results to a statistical analysis software.

La lipidomique est un jeune domaine encore proie à des défis quant à sa méthodologie. En effet, l'analyse de données en lipidomique demeurera longue et ardue, tant que des outils bioinformatiques appropriés ne sont pas créés. Je décris ici deux de mes programmes développés pour remédier aux défis d'identification de pics et d'annotations et de sélection de pics provenant d'une approche lipidomique dénommée ciblée, données présentement incompatibles avec d'autres programmes. Lipid Identification Tool (LIT) est un moteur de recherche hors-ligne possédant une robuste base de données in silico générée mathématiquement, et contenant des informations sur des espèces lipidiques non décrites dans les bases de données disponibles. Lipid Identification for Targeted Lipidomics (LITL) permet l'annotation de données provenant de méthodes HPLC-ESI-MS/MS MRM, en comparant leur temps de rétention ainsi que leur rapport masse-sur-charge à une bibliothèque d'identités lipidiques, et permet de les exporter facilement vers un logiciel d'analyse statistique.

List of Tables

List of Figures

List of Abbreviations

amu	atomic mass units
Cer	ceramide
CID	collision induced dissociation
ESI	electrospray ionization
GPC	glycerophosphocholine
GPE	glycerophosphoethanolamine
GPS	glycerophosphoserine
HPLC	high performance liquid chromatography
LC	liquid chromatography
LIT	lipid identification tool
LITL	lipid identification for targeted lipidomics
m/z	mass-over-charge ratio
MRM-IDA-EPI	multiple reaction monitoring information dependent acquisition enhanced product ion
MRM	multiple reaction monitoring
MS	mass spectrometry
MS/MS	tandem mass-spectrometry
NL	neutral loss scan
PIS	precursor ion scan
PR	product ion scan
Q1	the first quadrupole in a triple quadrupole mass spectrometer
Q2	the second quadrupole in a triple quadrupole mass spectrometer
Q3	the third quadrupole in a triple quadrupole mass spectrometer
RT	retention time
SPH	sphingolipid
SSRT	standardized and scaled retention time

Acknowledgements

I never thought I'd have to write one of those acknowledgements sections, but here we are..! Excuse me in advance as I take this opportunity to write this page freely, while trying to shove a humongous and tear-able amount of jokes and references in there... But where are my manners!

Greetings to you, dear reader, and thank you for bestowing me with your mind's attention, as I present to you my Master's thesis! This work would have been impossible without all the people in the Bennett lab, past and present, especially the people who interviewed me back in 2016 and judged me sane enough to join, and who endured my unceasing puns as an aftermath. I also want to thank GT for the lab's website, because let's be honest that's how I found out about the lab in my last year of undergrad. Stef – I still remember one of my classmates warning me about how scary you were, and, well, what would have been scary is actually doing my grad studies without you as my supervisor..! And this of course applies to Ted as well (the "this would have been scary without you" part, not the "classmate thinks you are scary" part). You two are mentors that shaped up my life. So to everyone I had the pleasure to work with in the lab – Caitlyn, Danielle, Doan-Nghi, Fangshen, Graeme, GT, Hongbin, Irina, Jacob, Jayu, Jenna, Mark, Matthew, Sam, Shannon, Stef, Steph, Ted, Thao, Valérie – thank you. Thank you for providing me with your company, your data, your knowledge, your opinion, and your wisdom, no matter if we just spent lunch breaks together, or if we spent late nights in the lab, cursing at our computer screens together (no computers were armed, only what was left of our pride when finding a missing typographic symbol). Special thanks to Jayu for being an awesome graphic design teacher and designing logos for both LIT and LITL . You're the best! 정말 감사합니다!

Outside of the lab, I would not have been able to stay sane without my second family KKD, nor my actual family (Hi mom!). While I will let the definition of sanity up to the reader, I will proceed to thank my lovely brother, as my resourcefulness wouldn't be what it is without him.

I also need to thank the two professors who showed me what coding was, and made me fall (back) in love with computer science and statistics (respectfully): Benoît Pagé and Antoine Morin.

Discovering MATLAB and R during class was definitely the highlight of my undergraduate studies, and this probably shaped up my future more than any other courses (Japanese is up there too, 松下先生、どもありがとうございます).

Weirdly enough, shout out to the printemps érable too? Because I would never have considered studying in Ottawa without being forced to miss my exams in CÉGEP? Life is strange ahah especially since I also wish to thank the University of Ottawa and its Faculty of Graduate and Postdoctoral Studies for my scholarships, which made my studies possible, and contributed to making me decide on pursuing a Master's degree.

As you may start to question my English writing skills, I will immediately proceed to reassure you, since the following work finds itself in a totally different style and tone. I will probably end up embarrassing myself when I re-read this, years from now. Anyway, I've seen enough TellTales' Walking Dead, Until Dawn, Heavy Rain, and Detroit: Become Human playthroughs that at this point, I could even thank that random dog who tripped me so randomly, bestowing me with a glorious hole in the knee, since it would've somehow unlocked a storyline path that I took and led me to the day where I am writing this very sentence. Fun fact, the dog was used as a NPC during a gamejam, we named it Tod. So, before I derail too much, I will close this section with some cheesy motivational quote from one British person, which once said: "Do what you want, Be enthousiastic at what you do, And somewhere out there, There is a pancake for you" (yes, that was what was written on my bookmark this whole time). So to everyone out there, find that spark, whatever drives you to get up in the morning without snoozing that alarm. And to whomever made it this far, stay determined[*]! In the great word of Stephanie and Matthew Patrick: "Thank". And without further ado, enjoy the poetry of science.

[*] https://youtu.be/BEWOTdVnnZ8

Chapter 1: Introduction

1.1 Importance of lipids in biological systems

Lipids are essential biomolecules in living organisms. There are eight lipid categories (fatty acids, glycerolipids, glycerophospholipids, sphingolipids, sterol lipids, prenol lipids, saccharolipids, and polyketides) each defining lipids of different classes and subclasses that are organized structurally[1]. It is estimated that as many as 100,000 different lipids are represented in biological membranes[2]. These hydrophobic and amphipathic molecules play roles in multitudes of biological processes, as critical components of biological membranes[3], and as signaling molecules and biological regulators[2, 4]. Regulation can occur at the level of membrane curvature or lipids can interact with proteins and affect their activity[5, 6]. Because cell membranes incorporate integral membrane proteins at high densities[7], lipid-protein interactions represent one means of regulating protein activities[8-10]. Moreover, lipids act as ligands for G-protein coupled receptors[11, 12]. Further, as one concrete example of cell signaling, ceramides, from the sphingolipids lipid category, can signal cell cycle arrest[13-16]. Lipids therefore represent biomolecules of the utmost importance, impacting on energy storage, cellular and subcellular organization, and cell signaling.

1.2 Structural classification of lipids

The structural diversity of lipids is a key determinant of the biophysical properties of organellar and subcellular organelle membranes[17, 18] as they dictate membrane curvature and fluidity, which in turn affect vesicular trafficking, amongst other physiological events[7, 19]. Thus, the ensemble of the lipids – the lipidome – needs to be profiled and classified in order for the wide spectrum of information inherent in lipid structure to be assessed properly.

In 2005, Fahy, E. et al.[1] standardized a classification system amendable to "omics" investigations, and later updated it in 2009[20]. This classification describes a nomenclature categorizing lipids by their structural composition, encompassing backbones, head groups, linkage types, and hydrocarbon chains.

These structures confer lipids with both subtle and dramatic biological differences. For my thesis, I will start by focusing on two overarching lipid categories or classes pertinent to the field of neurodegeneration: glycerophospholipids and sphingolipids. Their structure allows for a high occurrence of isobaric lipid species observable in biological samples, which represents one of the biggest challenges in lipidomic research.

1.2.1 Glycerophospholipids

Glycerophospholipids, as the name suggests, are composed of phospholipids defined by their glycerol backbone. In addition to possessing up to two hydrophobic hydrocarbon chains attached to the *sn*-1 and *sn*-2 carbons of their glycerol backbone, glycerophospholipids also have a hydrophilic headgroup attached to the *sn*-3 carbon of their backbone, as pictured in Figure 1.1. The hydrocarbon chains composing the hydrophobic component of glycerophospholipids can be of various lengths and possess varying degrees of unsaturation at different carbons (double bonds). They can also differ by being oxidized or hydroxylated at different positions. As the hydrocarbon chains are linked to the glycerol backbone by ester, ether, or vinyl ether linkages, glycerophospholipids also differ by these respective linkages at the *sn*-1 and *sn*-2 positions. On the other hand, the hydrophilic component, namely the headgroup linked to the *sn*-3 carbon, allows for further distinction between lipid species, as glycerophospholipid lipid subclasses are distinguished by their head group[1]. In the present thesis, I will focus on glycerophosphocholines (GPC), glycerophosphoethanolamines (GPE), and glycerophosphoserines (GPS).

1.2.2 Sphingolipids

Sphingolipids (SPH), again as the name suggests, are lipids possessing a sphingoid base as a backbone. This backbone can be composed of a hydrocarbon chain of varying length, with varying degree of unsaturation. It can also be deoxidized, dihydroxylated, or trihydroxylated[21]. SPH are deemed

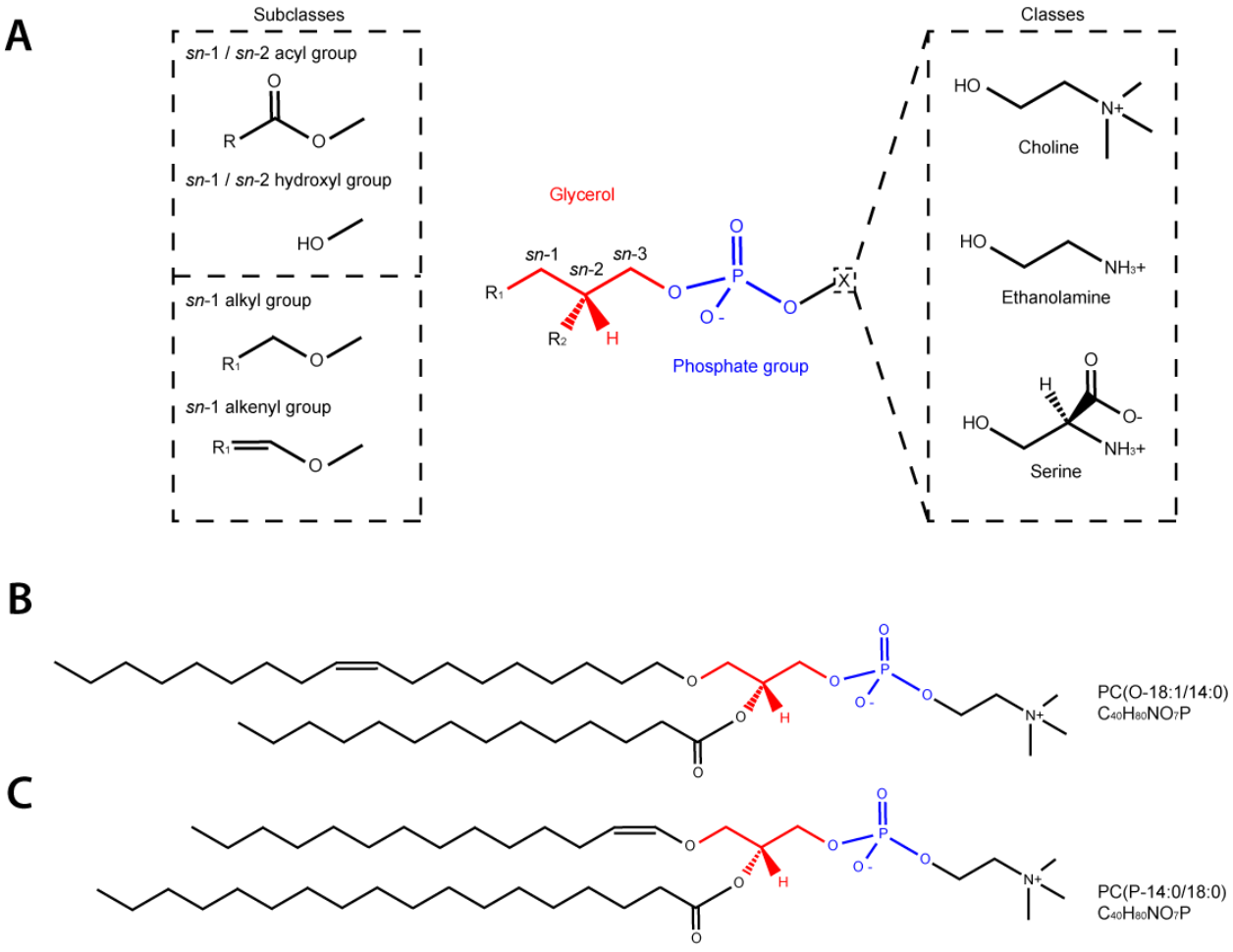
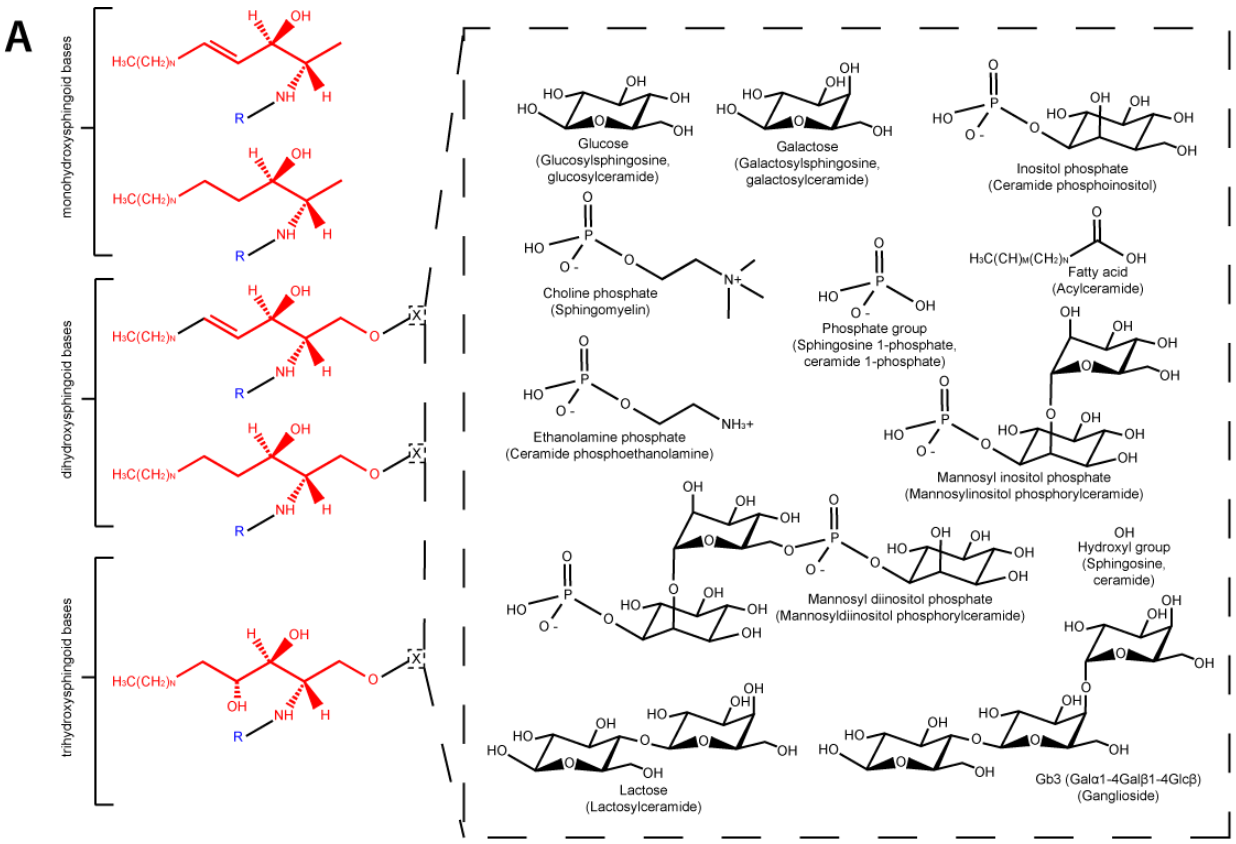

A
Subclasses
sn-1 / sn-2 acyl group
sn-1 / sn-2 hydroxyl group
sn-1 alkyl group
sn-1 alkenyl group
Glycerol
sn-1
sn-2
sn-3
Phosphate group
Classes
Choline
Ethanolamine
Serine
B
PC(O-18:1/14:0)
C40H80NO7P
C
PC(P-14:0/18:0)
C40H80NO7P

Figure 1.1 Combinatorial structure of glycerophospholipids. (A) Acyl, alkyl, or alkenyl groups are linked to the glycerol backbone of glycerophospholipids (in red) at the *sn*-1 position by an ester (acyl), ether (alkyl), or vinyl ether (alkenyl) linkage, respectively. To date, only hydrocarbon chains esterified at *sn*-2 position have been found in biological samples. In addition, either *sn*-1 or *sn*-2 positions can be a hydroxyl group, defining the lysoglycerophospholipid subclass. The hydrocarbon chains (R_1 and R_2) vary in length and degree of unsaturation, defining molecular species of glycerophospholipids. Different polar head groups are linked to the glycerol backbone at the *sn*-3 position via a phosphodiester linkage, defining classes of glycerophospholipids. Depicted on the right are choline, ethanolamine, and serine head groups (from top to bottom), and the resulting lipid classes, which are glycerophosphocholine (GPC), glycerophosphoethanolamine (GPE) and glycerophosphoserine (GPS), respectively. (B) The chemical structure of glycerophosphocholine PC(O-18:1/14:0) is shown with an *sn*-1 ether linked (O) 18-carbon mono-unsaturated alkyl chain (18:1) and an *sn*-2 ester linked 14-carbon acyl group (14:0). The numbers following the colon (:) indicate the degree of unsaturation (number of double bonds) of each chain. (C) The chemical structure of glycerophosphocholine PC(P-14:1/18:0) is shown with an *sn*-1 vinyl ether linked (P) 14-carbon chain (14:0) and a saturated 18-carbon acyl group (18:0) at the *sn*-2 position. The lipid species depicted in (B) and (C) are isobars, having identical mass yet different structures.


A
monohydroxysphingoid bases
dihydroxysphingoid bases
trihydroxysphingoid bases
Glucose
(Glucosylsphingosine,
glucosylceramide)
Galactose
(Galactosylsphingosine,
galactosylceramide)
Inositol phosphate
(Ceramide phosphoinositol)
Choline phosphate
(Sphingomyelin)
Fatty acid
(Acylceramide)
Phosphate group
(Sphingosine 1-phosphate,
ceramide 1-phosphate)
Ethanolamine phosphate
(Ceramide phosphoethanolamine)
Mannosyl inositol phosphate
(Mannosylinositol phosphorylceramide)
Hydroxyl group
(Sphingosine,
ceramide)
Mannosyl diinositol phosphate
(Mannosyldiinositol phosphorylceramide)
Lactose
(Lactosylceramide)
Gb3 (Galα1-4Galβ1-4Glcβ)
(Ganglioside)

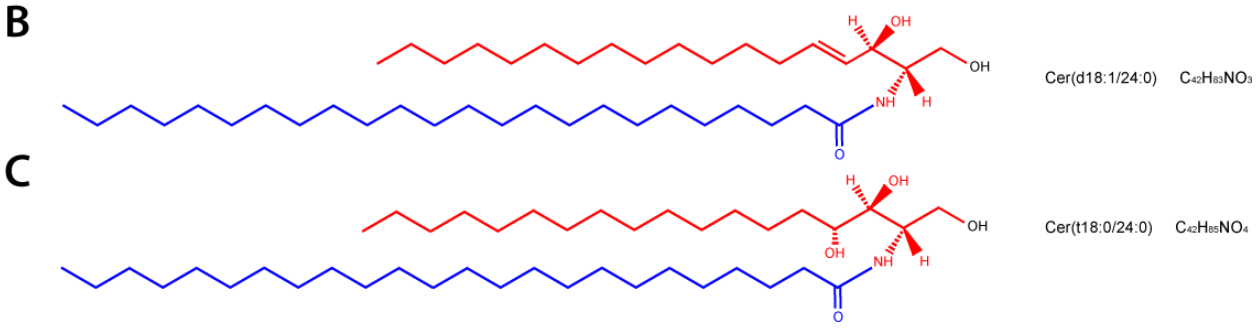

B
Cer(d18:1/24:0) C42H83NO3
C
Cer(t18:0/24:0) C42H85NO4

Figure 1.2 Combinatorial structure of sphingolipids. (A) Long-chain sphingoid bases are the defining structural unit of sphingolipids. Sphingoid bases (in red) can be monohydroxylated (top group), dihydroxylated (middle group), or trihydroxylated (bottom group), and they vary in length, as depicted by the $(CH_2)_n$. Fatty acids of varying chain length and degree of unsaturation (R, in blue) are N-acylated to sphingoid bases via an amide bond, resulting in ceramides. The C1 hydroxyl group (X) of sphingoid bases (except for monohydroxysphingoid bases) can be modified by a number of polar head groups. The structures of some of the most common head groups are shown on the right in the dotted box, and the resulting sphingolipids are shown in parenthesis under each head group name. (B) The chemical structure of ceramide Cer(d18:1/24:0) is shown, with a sphingosine backbone of 18 carbons and one unsaturation (d18:1), and a N-linked saturated hydrocarbon chain of 24 carbons (24:0). (C) The structure of phytoceramide Cer(t18:0/24:0) is shown, with a phytosphingosine of 18 carbons (t18:0) and a N-linked saturated hydrocarbon chain (24:0).

ceramides (Cer) when a hydrocarbon is N-acylated to a sphingoid base by an amide linkage. Like glycerophospholipids, SPH can also be distinguished by different head groups, resulting in glycosphingolipids and phosphosphingolipids[1], as shown in Figure 1.2.

1.3 Lipidomics

It is only at the beginning of the 21st century that the term "lipidomics" was first used[22, 23]. Lipidomics refers to the characterization, identification, and quantification of lipid species at the molecular level in order to assess their role in a system's biology via their interactions and the pathways that they affect. Lipidomics research encompasses discovery of biomarkers of lipid metabolic disruptions and potential therapeutic targets for prevention and treatment of diseases[24]. This emerging field is challenged by bioinformatic limitations requiring new technologies if the promise of clinical application is to be realized[2, 4, 22, 25-29]. In the following section, I will review the experimental and bioinformatics methods used in a targeted lipidomics workflow to highlight some of the challenges that are addressed in this thesis.

1.3.1 Sample preparation

A lipidomics approach first requires the isolation of lipids through meticulous sample preparation ("sample preparation" step in Figure 1.3). Various lipid extraction methods exist, such as the modified Bligh and Dyer and the modified Folch method[26, 28, 30, 31]. These techniques exploit the high solubility of lipids in organic solvents to achieve separation of lipids from other components of biological samples, such as proteins, carbohydrates, and nucleic acids. When preparing lipid samples, it is of utmost importance to properly handle them as their collection, processing, and storage procedures are all prey to experimental variables. Artefactual lipid oxidation, peroxidation, and hydrolysis can be reduced and ideally prevented by keeping solvents and extracts under nitrogen, storing extracts at -80°C, processing samples at 4°C, and minimizing exposure to ultraviolet light[26, 30]. Contamination is also avoided by using

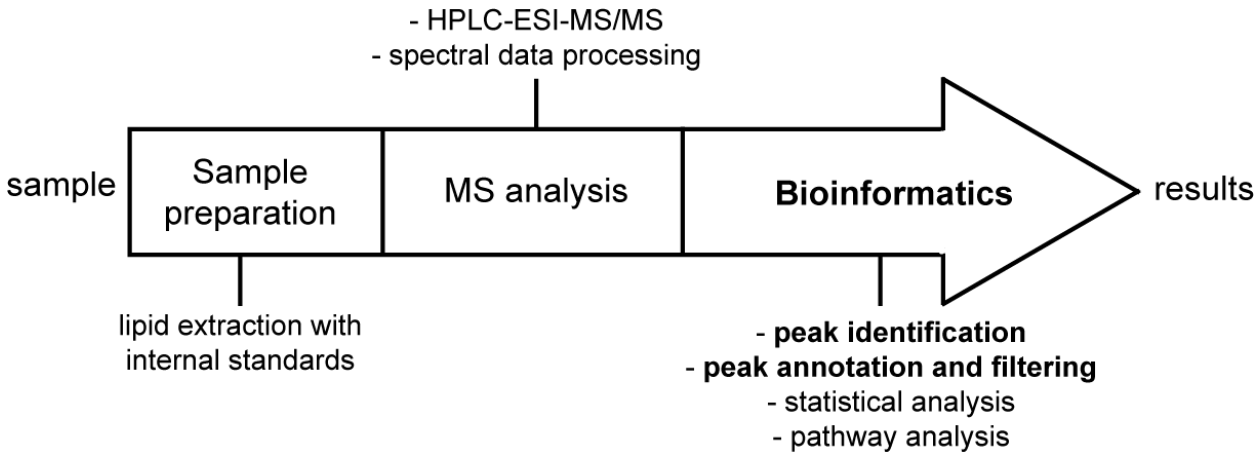

- HPLC-ESI-MS/MS
- spectral data processing
sample
Sample preparation
MS analysis
Bioinformatics
results
lipid extraction with internal standards
- peak identification
- peak annotation and filtering
- statistical analysis
- pathway analysis

Figure 1.3 Targeted lipidomics pipeline. **Sample preparation**: Lipidomics analysis of biological samples starts with lipid extraction. Appropriate extraction methods are chosen based on the lipid of interest to be analyzed. Internal lipid standards are added at the time of extraction for quantification, accounting for both extraction efficiency as well as instrument response variability. **MS analysis**: The prepared samples are analyzed using a HPLC-ESI-MS/MS method, which consists of front-end separation by liquid chromatography, followed by tandem MS analysis. The spectral data is then processed by appropriate software, most commonly the software shipped with the instrument. **Bioinformatics**: The resulting initial data output from the software needs to be converted and formatted appropriately before it can be assessed bioinformatically. This data is usually in a table format containing information of detected peak intensity (peak area or height) and retention time. This spectral data must be first filtered and annotated before meaningful statistical analysis and pathway analysis can be conducted. A major bottleneck in current lipidomics pipeline is in the bioinformatics, specifically peak identification, annotation and filtering.

glassware instead of plastic pipette tips, tubes, and other containers susceptible to degradation in organic solvents[30, 31].

For assessment of extraction efficiency and subsequent normalization, internal exogeneous standards are added. Internal standards are used in order to enable, in later steps, correction and normalization of different variables in lipid analysis experimental protocols, such as extraction or ionization efficiency. They can be added at different steps of the sample preparation in order to address the experimental variables to be corrected (e.g., extraction efficiency, normalization, etc.)[26, 32].

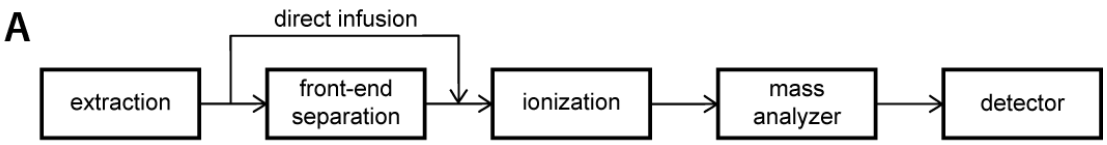
1.3.2 Mass spectrometry (MS) analysis

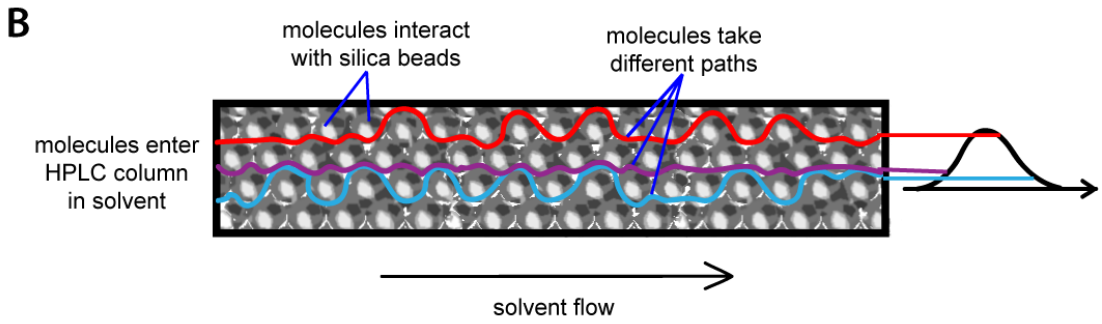
Lipids are identified by reconstructing their structure using high performance liquid chromatography (HPLC) eletrospray ionization (ESI) tandem mass spectrometry (MS/MS). First, following extraction, lipids can be separated according to their biophysical properties using HPLC and their molecular identities and abundances can be assessed by MS, as pictured in Figure 1.4A (which corresponds to the "MS analysis" step in Figure 1.3). Further characterization of the acquired data is then achieved through bioinformatics.

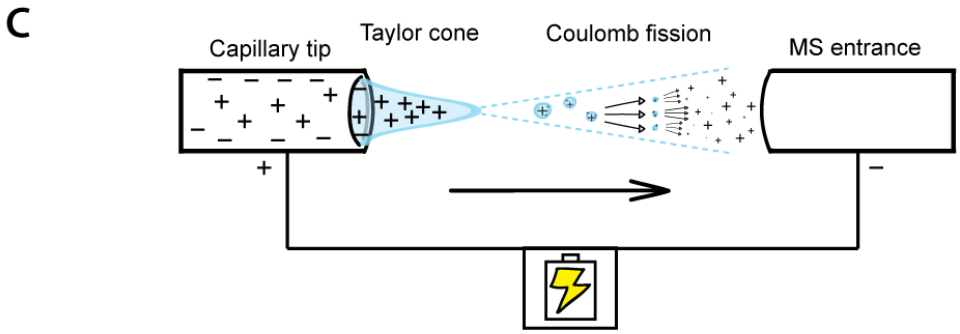
1.3.2.1 Front-end chromatographic separation versus direct infusion lipidomics

A direct infusion approach is said to be used if the lipid samples are directly infused in the mass spectrometer without any prior separation, as shown in Figure 1.4A. By contrast, a front-end chromatographic separation is said to be used when a chromatography technique is applied to the lipid sample before it enters the mass spectrometer[26]. With direct infusion, a constant ratio of ions enters the mass spectrometer over a defined period of time, whereas in front-end separation, lipids are separated according to their hydrophobicity or hydrophilicity depending on the type of HPLC separation[33].

Direct infusion is not ideal for characterizing low abundance lipids, compared to front-end separation[26], nor is it ideal for differentiating lipid species with identical chemical composition and thus


A
direct infusion
extraction
front-end separation
ionization
mass analyzer
detector


B
molecules interact with silica beads
molecules take different paths
molecules enter HPLC column in solvent
solvent flow


C
Capillary tip
Taylor cone
Coulomb fission
MS entrance

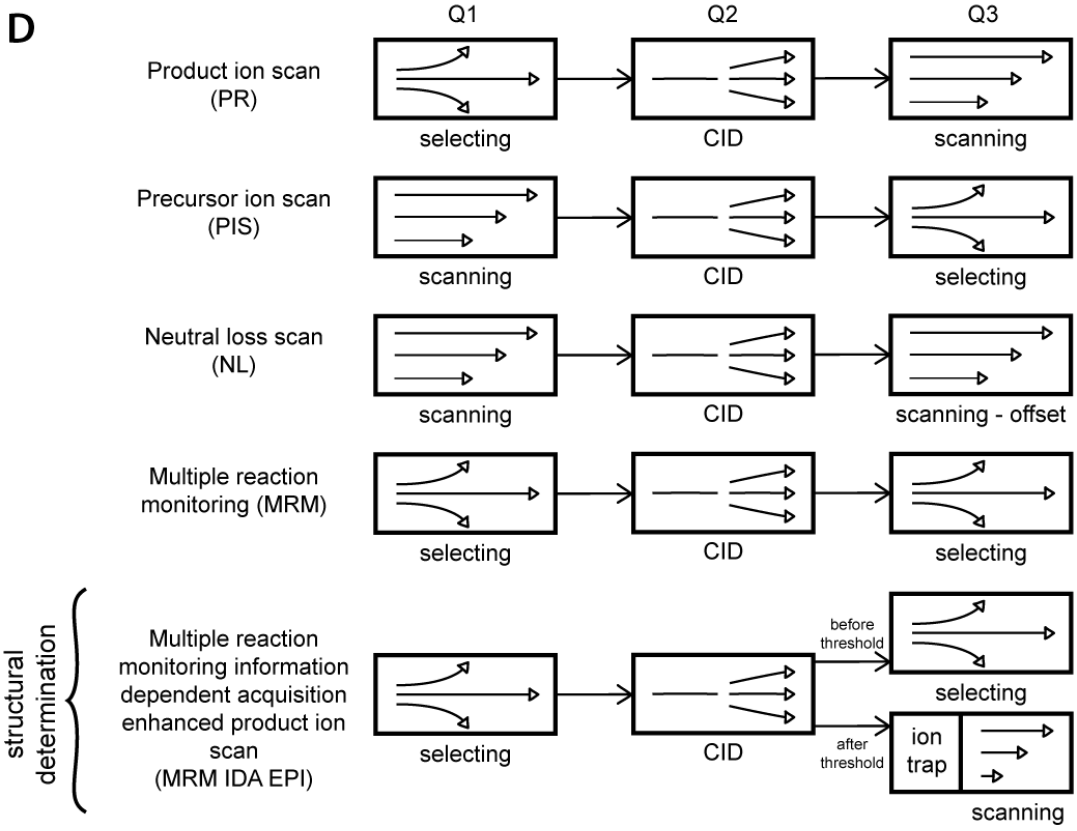

D
Q1
Q2
Q3
Product ion scan (PR)
selecting
CID
scanning
Precursor ion scan (PIS)
scanning
CID
selecting
Neutral loss scan (NL)
scanning
CID
scanning - offset
Multiple reaction monitoring (MRM)
selecting
CID
selecting
structural determination
Multiple reaction monitoring information dependent acquisition enhanced product ion scan (MRM IDA EPI)
selecting
CID
before threshold
selecting
after threshold
ion trap
scanning

Figure 1.4 Experimental workflow with a triple quadrupole mass spectrometer in a targeted lipidomics pipeline. (A) Lipids extracted from biological samples are analyzed by MS either with or without the coupling of the front end LC separation. The latter method is called direct infusion. (B) High performance liquid chromatography in reversed phase. Molecules will elute and produce Gaussian shaped peak signals mostly due to eddy diffusion, as they take different paths and different channels between the beads of the chromatography column as depicted by the red, purple, and blue paths. (C) Electrospray ionization. (D) Scan modes of a triple quadrupole to further select ions through the mass analyzer. Depending on the scan mode selected, ions are scanned or selected in the first quadrupole (Q1), fragmented in the collision cell (Q2) by collision induced dissociation (CID), then scanned/filtered in the third quadrupole (Q3).

with exact mass but different structural compositions (isobars)[33]. Indeed, distinguishing isobars only becomes possible using HPLC, considering their elution times, also known as retention times (RTs). Different lipid species will elute at different RTs, depending on features like the lengths, unsaturations, and linkages of their hydrocarbon chains to their respective backbones, as well as their head groups.

As a chromatography technique, HPLC has a stationary phase and a mobile phase separating molecules according to their biophysical properties, as seen in Figure 1.4B. The mobile phase is composed of a solvent. Samples, in mobile phase, are applied, under high pressure under computationally controlled flow, to a column packed with synthetic silica beads, forming the stationary phase. Lipid molecules interact with both the synthetic beads (stationary phase) and the solvent (mobile phase), and thus their elution time differs according to these interactions.

HPLC is either done in normal or reversed-phase. Normal phase is used to separate lipids in relation to their hydrophilicity, due to their polar head group. Thus the mobile phase used is hydrophobic and the stationary phase, polar. In reversed-phase HPLC, lipids are separated in relation to their hydrophobicity, which is more closely related to the structure of their hydrophobic components, their hydrocarbon chains. Thus, the mobile phase is polar, and the stationary phase, hydrophobic: the silica beads used in reversed-phase HPLC are coated with immobilized hydrocarbon chains of fixed lengths, which also defines the type of HPLC column. For example, a C8 column contains a silica substrate to which hydrocarbon chains of 8 carbons in length are affixed, while a C18 column contains a silica substrate to which hydrocarbon chains of 18 carbons in length are affixed. Therefore, separating lipids by class is better achieved with normal phase HPLC, while separating molecular species within a lipid class is better achieved with reversed-phase HPLC[26, 33].

As they are eluted, the molecules interact with both the mobile and stationary phase. Thus, their elution time through the column will differ depending on their size and on their hydrophobicity – a property depending on the composition and linkages of their hydrocarbon chains to their respective backbone. Lipid species that are identical do not however emerge at the exact same RT, but within a RT frame, giving their signal in the data the shape of Gaussian peaks[34]. This is due to diffusion in the column.

Without diffusion, the molecules eluting migrate together, forming sharp chromatography bands. Broadening of bands to form wider peaks is mostly due to eddy diffusion[35], according to which a set of molecules will have distinct elution times based on the different paths and different channels between the beads of the chromatography column[34] as seen in Figure 1.4B.

1.3.2.2 ESI

Mass spectrometers are composed of three main components: an ion source, one or multiple mass analyzers, and one or multiple detectors. Molecules can only be detected by a mass spectrometer if they are charged and in gas phase. Soft ionization techniques allow for the formation of gas-phase ions from non-volatile analytes, enabling the analysis of non-volatiles components such as proteins and lipids, without the extensive fragmentation characteristic of hard ionization techniques[26, 36]. The most popular soft ionization technique in lipidomics is ESI[26].

In ESI, analytes are introduced in solution into a fine capillary to which a high voltage is applied. The electrode of opposite charge is located at the other end of the ionization chamber, where the entrance of the mass spectrometer is also located (Figure 1.4C). Due to the high electrical current, the solution forms a Taylor cone at the tip of the capillary before being sprayed in the ionization chamber under the form of small droplets. In positive-ion mode, positive current is applied at the end of the capillary and negative current at the entrance of the mass analyzer, causing the droplets to be positively charged and vice-versa in negative-ion mode. The ions formed encompass an ion adduct that was present in solution. For example, the ion adduct will be [M+H]+ if the molecules are protonated in positive-ion mode. In the chamber, an axial flow of nitrogen contributes as a drying gas to the evaporation of the solvent of the droplets, with the high temperatures or vacuum conditions at which the ionization chamber is maintained. This evaporation decreases the size of the droplets as they are sprayed through the chamber. The charged droplets shrink in size until reaching their Rayleigh limit, at which their charge is the maximum their size can withhold. Then, the surface tension of the droplets is surpassed by the repulsion force of the charges within themselves, causing their deformation, followed by their Coulomb fission, causing the droplets to

divide into smaller and more stable droplets. These will then shrink until reaching their Rayleigh limit and undergo Coulomb fission themselves. The lipid ions, now desorbed from the solvent droplets, reach the end of the ionization chamber and enter the mass spectrometer[23, 28, 37-39].

ESI produces ions with insufficient energy for fragmentation, therefore minimizing fragmentation and in-source decay, resulting in a high ion intensity turnover[23, 38, 39]. However, it must be acknowledged that in-source fragmentation of lipid ions can still occur and create lipid artifacts[40]. This additional fragmentation can cause the non-specific loss of a water molecule in a sphingolipid, or of a sugar residue of a glycosphingolipid, resulting in what I will now define as dehydrated and deglycosylated lipid ions. These artifacts are indistinguishable in terms of their RT, as they are only separated after HPLC. Their distinction remains however possible through their characteristic fragmentation patterns, and defined mass-over-charge (m/z) offset. For example, an in-source dehydrated sphingolipid will be detected with an identical RT but an offset of m/z 18 compared to its non-dehydrated counterpart, corresponding to the loss of the 18 atomic mass units (amu) of a molecule of water (H_2O). The same is applicable to deglycosylations, as their m/z would be offset by the m/z of their glycan.

1.3.2.3 Mass analyzers

Mass analyzers are able to select ions based on their m/z, which, as the name suggests, is the unitless ratio of an ion's mass over its charge. Hence, the ionization of molecules prior to their entrance in the mass spectrometer is required as to render them detectable by the mass analyzers, achieved as described above by ESI. Because of this, non-charged particles cannot be selected by the mass analyzers. Different approaches exist to select the m/z of ions transmitted to the mass analyzer. The type of mass analyzer dictates the nature of ion detection. For example, an orbitrap selects ions based on the frequency of their axial oscillation, whereas a quadrupole bases its selection on the stability of the trajectory of an ion across an oscillating electric field[37, 39, 41].

In triple quadrupoles mass spectrometers, three quadrupoles are arranged in tandem, as seen in Figure 1.4D. The first and third quadrupoles, referred to as Q1 and Q3 respectively, serve as filters while

the second quadrupole, referred to as Q2, serves as a collision cell. In order to filter ions, an oscillating electrical field is created within a quadrupole between its four rods placed in parallel. Ions transmitted through the resulting radio frequency quadrupole field have their trajectory either stabilized (selected) or destabilized (filtered). Quadrupole cells can also be used as collision cells, like Q2. In that case, transmitted ions collide with a non-reactive gas like nitrogen or helium contained in the collision cell, resulting in their fragmentation. Thus, ions entering a triple quadrupole mass spectrometer can be selected by m/z in Q1, fragmented in Q2, then selected again by m/z in Q3, before a detector records their electric signal, allowing assessment of their intensity[39].

1.3.2.4 Scan modes for targeted lipidomics

Lipidomics approaches using mass spectrometry are either targeted or un-targeted (often referred to as "shotgun", which should not be confused with "shotgun" proteomics). While shotgun lipidomics use a full scan of the ions of different m/z present in the sample analyzed with high resolution mass accuracy, with direct infusion, they cannot distinguish ions of identical mass[42]. Even when coupled with a front-end separation technique, further selection of the ions in the mass spectrometer is essential to properly characterize isobars. This selection is offered through various scan modes used in targeted lipidomics. These scan modes allow different methods of selection of the ions before and/or after a certain number of fragmentations through multiple mass analyzers, like the triple quadrupoles, in order to identify different lipid classes and subclasses[27]. Thus, the advantage of shotgun lipidomics is that it can profile multiple lipid families simultaneously. The disadvantages are that, given the complexity of the matrix, only the most abundant lipid species can be chromatographically separated and detected. While targeted lipidomics does not benefit from the high-throughput capacity of shotgun lipidomics, since each lipid subclass must be analyzed separately, it does offer comprehensive coverage of lipid species within given classes and subclasses, notably lower abundant species. By targeting lipid families, the complexity of the matrix is reduced and more precise separation of isobars within a given family can be achieved.

Targeted lipidomics enables a selection of lipids of interest to be monitored through various filtering methods. By possessing three consecutive mass analyzers, triple quadrupole mass spectrometers offer the ion selection desired in targeted lipidomics. Indeed, the set-up of Q1, Q2, and Q3 allows for a multitude of filtering methods, called scan modes (Figure 1.4D), as Q1 and Q3 can be either scanning or parked on a pre-set m/z, filtering ions with a given m/z[39]. As they gather molecular information on different subsets of ions, these different scan modes all have their own application.

With a product ion scan (PR), the ions entering the mass spectrometer (called precursor ions) are selected in Q1, as Q1 is parked on a predetermined m/z. This enables a filtering of the precursor ions based on their m/z. After selection, the selected ions undergo collision induced dissociation (CID) in Q2, their fragments (called product ions) are scanned in Q3 and then detected. As the fragments are scanned and not selected, a product ion scan gathers information about precursor ions of a given m/z and their fragmentation pattern. Product ion scan is thus useful for identification of an analyte at a pre-set m/z through its product ions[23, 39].

With a precursor ion scan (PIS), the precursor ions are not selected in Q1, but rather in Q3, which is parked on a predetermined m/z. Due to the CID in Q2, and selection in Q3, the ions detected are product ions of identical m/z from various precursor ions. This scan mode is particularly useful for characterizing ions of the same class, as their fragmentation will yield identical product ions. For example, GPC and sphingomyelin species yield a product ion of m/z 184, due to fragmentation and detection of their phosphocholine component[43].

With a neutral loss scan (NL), neither Q1 nor Q3 is selecting. The key of neutral loss scans is the offset of Q3 from the precursor ion in Q1 due to the loss of an uncharged fragment that cannot be detected. The precursor ions are fragmented in Q2, but because only charged fragments are detectable by Q3, it is the precursor ion minus the loss of the uncharged ion (neutral loss) that is detected in Q3. Therefore, if a lipid class is characterized by the same fragment, which does not hold a charge, the offset between Q1 and Q3 allows for its detection[23, 27, 39]. For example, GPE species yield a loss of a neutral fragment of m/z 141 corresponding to the loss of their polar head group after CID in Q2[44, 45].

With a multiple reaction monitoring (MRM) scan, both Q1 and Q3 are selecting for given m/z. Therefore, both precursor and product ions are targeted. Thus, this scan mode yields high specificity and high sensitivity required for accurate quantification, as ions of interest are pre-selected, and both quadrupoles are selecting for a small range of m/z (due to a high duty cycle). The term duty cycle refers to the ratio of ions of a given m/z passing through the mass spectrometer that are detected and analyzed. Moreover, multiple transitions, defined as the paired m/z Q1 and Q3 selected, can be investigated. Thus, MRM scan mode allows the monitoring of multiple pre-set and precise fragmentation outcomes[23, 27, 39]. However, due to the selection in Q1, the spectral data resulting from MRM has a discontinuous m/z dimension.

To obtain the full fragmentation patterns of precursor ions fragmented and selected in Q3 during an MRM scan, it is possible to couple MRM with an information dependent acquisition (IDA) enhanced product ion scan (EPI). In the resulting scan mode, MRM-IDA-EPI, ions are transmitted through the quadrupoles in a similar fashion as during an MRM scan, however, when the signal detected after Q3 reaches a certain ion intensity threshold, Q3 is switched to act as an ion trap. It can then gather and scan all the fragmented ions, enabling their further molecular characterization[46]. Here, ions can be both quantified by MRM albeit without the precision of a standard MRM given application of an intensity threshold and simultaneously structurally confirmed by MRM-IDA-EPI[47-49].

1.3.2.5 Data acquisition

Spectral data from HPLC-ESI-MS/MS are represented as peaks in a three-dimensional space shown in Figure 1.5, of RT, m/z, and ion intensity. Their selection, also known as peak picking, allows for their separation in individual peaks. These peaks can then be assessed individually through multiple features in addition to the three dimensions of RT, m/z, and ion intensity.

The RT is the dimension of data enabled by front-end HPLC. As the sample is separated, the elution time of its molecules differ. Evaluation of RT enables further characterization of the spectral data allowing researchers to infer on varying biophysical properties of each lipid species. Notably, this

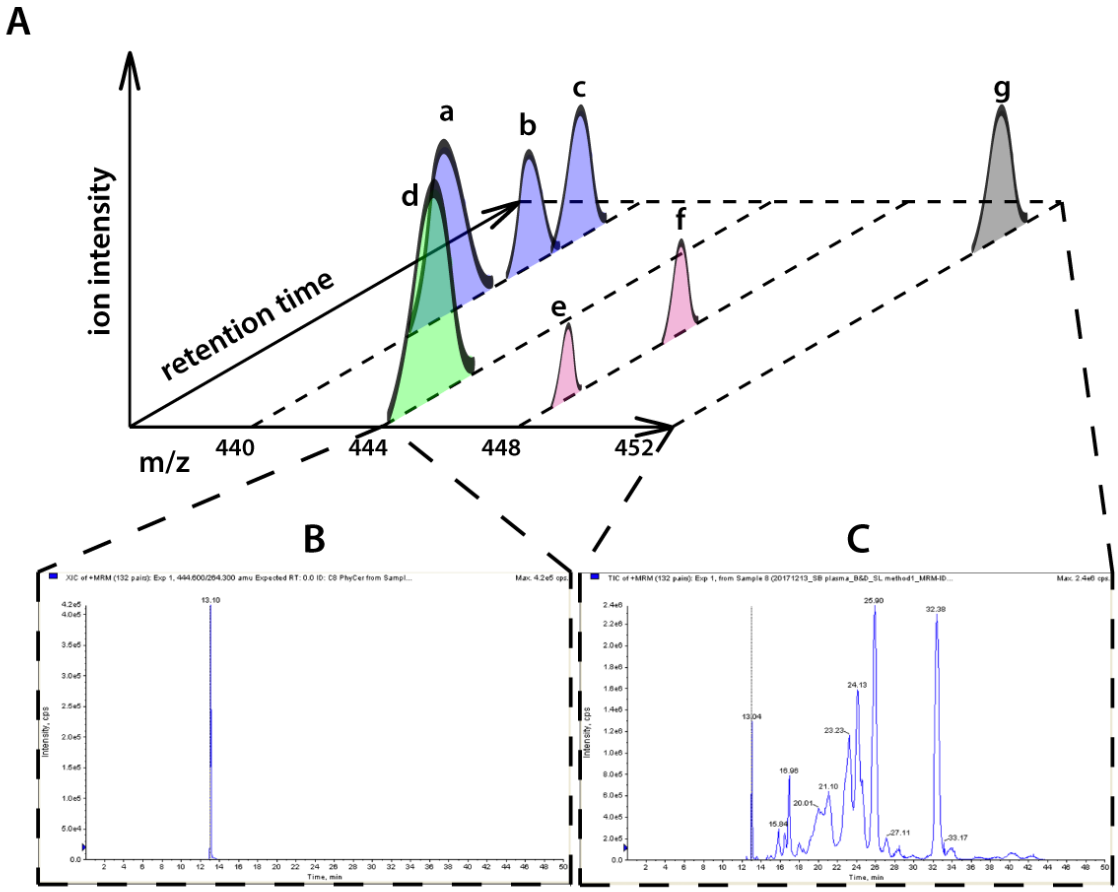

A
ion intensity
retention time
m/z
440
444
448
452
a
b
c
d
e
f
g
B
C
13.10
13.04
15.84
16.96
20.01
21.10
23.23
24.13
25.90
27.11
32.38
33.17

Figure 1.5 Three dimensional spectral data produced by HPLC-ESI-MS/MS. (A) A HPLC-ESI-MS/MS spectrum contains 3 dimensional information composed of m/z, retention time and ion intensity. Species a, b and c are isobars. That is they are of the same m/z but of different retention times, like species e and f. Peak (e) is an isotopic peak of peak (d), as identified by the identical retention time, characteristic m/z difference and relative isotopic abundance. However, peak (g) is not an isotopic peak of peak (c), as their m/z difference is too large. (B and C) Example spectra from MRM-IDA-EPI of sphingolipids in human plasma. Extracted ion chromatogram (XIC) of ions at m/z 444.6 with 444.6 to 264.3 transition is shown in (B). Total ion chromatogram (TIC) showing all ions detected along the retention time dimension is shown in (C).

dimension allows the separation of isobars, as molecules of the same molecular weight but different molecular structure will interact with the mobile and stationary phases in a different fashion. The RT of peaks in spectral data is recorded, as a continuous variable, at the point where the highest abundance of ions is detected – the peak apex.

The second dimension, m/z, represents the characteristic mass-over-charge ratio of a particular ion. In the context of this thesis, I will address the m/z of a peak as the m/z of its precursor ion and the abundance of lipid as the peak area of its product ion. In MRM scans, the m/z of a peak would then refer to the m/z selected in Q1. However, it is important to note that, as m/z transitions are selected and not scanned, the m/z dimension of spectral data acquired by MRM is of a discontinuous nature. Moreover, it is binned as the m/z plus and minus the mass accuracy of the mass spectrometer employed. In brief, the m/z dimension is another key dimension to assess lipid composition, as it holds information about the mass of each analyte observed.

Ion intensity is the third dimension of spectral data, which is recorded over time once ions are transmitted to Q3. This recorded signal is amplified relative to the ion intensity and represents relative abundance, indicative of the relative abundance of a specific lipid. The abundance of a selected peak is further assessed by its integration. The integrated region, which will now be described as peak area, is a direct measure of ion intensity of that particular ion, allowing relative quantification when compared to those of internal exogeneous standards added during sample preparation[32].

This quantification can only be relative, unless labour- and time-intensive standard addition methods are employed for each lipid species under interrogation. As loss of lipid can occur during HPLC, and as ion suppression can occur[33, 50], any measurement of ion intensity is indicative of the amount of a detected ion detectable under given analytical conditions, compared to an internal standard detected simultaneously under the same experimental conditions. Comparing ion intensities of analytes of interest and internal standards allows for molar equivalents to be determined. The chemical standard chosen must, however, fall within the same dynamic range of detection of the analytes to which it is compared to and must have the same response, that is to say a comparable ionization efficiency[32]. In order to achieve this,

an exogeneous standard from each lipid class and subclass of the targeted lipids is added during sample preparation. As for the dynamic range, it can be assessed for each standard, and analyte, *a priori* by serial dilution experiments[32]. For the purpose of this thesis, incorporation of dynamic range and limits of quantification will only be incorporated into the bioinformatic tools developed here following thesis submission.

Peak selecting, referring to the selection of *bona fide* lipid species in a data spectrum, also allows for other features to be defined – features that will be used further as part of the data analysis workflow to filter peaks that cannot be accurately discriminated, such as the full width at half maximal height of peaks. This feature is calculated once maximum ion intensity is reached, describing the spread of the peak over time. It then can be used to assess if peaks can be separated in the RT dimension.

The spectral data thus acquired by industry or open-source software is then ready for identification and further analysis. Software like AB SCIEX MultiQuant (AB SCIEX, Concord, Ontario) export this information as a .txt text file. This exported data contains the list of the peaks and their features, which is required in order to annotate the detected ions comprising the lipid sample with their predicted molecular structure, and to quantify the detected ions.

1.3.3 Bioinformatics challenges

Further analysis of acquired LC-MS/MS MRM data has its own bioinformatics challenges. First, the detected peaks must be identified based on their m/z and RT ("Bioinformatics" step in Figure 1.3). Here, one requires reference points, a database of spectral data, and validated lipid identities[33]. Moreover, artifacts must also be identified. Second, correspondence between samples must be established in order to compare their lipid profile[50]. Correspondence refers to establishing whether lipids identified in samples assessed at different experimental or clinical time points, extracted from different biological tissues or fluids, analyzed by different instruments, etc. are, in fact, the same species, and therefore enables comparisons between matrices (provenances of samples). Assessment of correspondence is not trivial as different lipids can have similar RTs yet only be detected in a given biological sample type or replicate. It

is a bioinformatic challenge to discriminate between two different species with close but not identical RTs when they do not co-exist across biological samples. Third, as spectral data is acquired in multiple proprietary formats and file types, parsing the structural and abundance information required for lipid identification and quantification is sometimes hindered by format incompatibility, as depicted in Figure 1.**6**. Thus, another bioinformatics challenge in lipidomics remains the platform dependence of the data.

1.3.3.1 Available programs

In order to identify lipids, databases of structures were developed, like LIPID MAPS[1]. In addition to proposing a standard for lipid nomenclature, LIPID MAPS introduced a database storing lipid information using their structural components, and assigning unique lipid identifiers based on that classification schema. This database allows the storage and retrieval of curated information on lipids via a query-able web-API (application program interface).

Other databases were also developed with the goal of querying what possible lipid candidates would fall within a given m/z range. However, these databases would be created and populated based on the calculation of the chemical formula of lipid species, allowing for theoretical structures to be queried, like the calculator of ALEX[51], LipidHome[52], and VaLID[53]. The calculator of ALEX, available online or as an executable program, queries a SQLite database listing lipids whose mass was calculated using their chemical formula and adjusted according to a user-specified tolerance window. The database of LipidHome, accessible online through the web, was developed around glycerophospholipids and glycerolipids, and was populated according to a set of rules for lipid structure generation. Thus, its database spans the chemical space for these two classes with a programming approach, also offering cross-references to other databases like LIPID MAPS when possible. VaLID, a Java web-based application, also took a programming approach to span the chemical space of given lipid classes, and also to produce 2D and 3D structural representations. Its search engine capability lets users query its Excel database for lipid species based on m/z and experimental MS conditions. Furthermore, it uses the

prevalence of fatty acids in mammalian cells to predict which returned lipid species are more likely to be the lipid species of interest.

Programs using the m/z of MS data peaks to identify lipids were also developed in order to identify and quantify MS spectral data. Identification by using a query-able database was a nice concept, which LIMSA[54], for instance, maintained. This program was created to assign and quantify experimental peaks from NetCDF formatted MS spectra by comparing their m/z to a list of expected compounds coming from its internal database, after filtering the raw MS spectra with a Gaussian filter. In addition to lipid species, its database also contains data for unwanted fragments, like loss of water, enhancing the program's identification capabilities. Its isotope pattern calculations using the theoretical isotopic distribution of elements also complement its database. Batch processing was made available through command line, but an Excel interface presented as an add-on was also introduced. Another program, LipidXplorer[55], identifies peaks from mzXML MS and MS/MS spectra, by first merging them in a representative spectrum, aligning individual spectrum peaks together, while keeping each spectrum's abundance profile separate for quantification. Then, the program uses molecular fragmentation query language (MFQL) queries to identify peaks based on their precursor and fragmented ions' m/z. MFQL scripts for common lipid classes are provided with the distributed version of the program, which takes a step towards a more programming approach. LipID[56], also an Excel add-on, took the m/z query concept with a programming approach, steering away from the database approach. It allows users to obtain the possible lipid species candidates of experimental peak lists by comparing their m/z to calculated m/z of lipid species corresponding to user-selected available lipid classes, as presented in its interface, which also includes dehydration options. Taking lipid identification one step further, ALEX[51] streamlined the identification process to include quantification and exporting of the results. It first imports .raw formatted shotgun spectral data. It then identifies peaks by m/z matching using its internal database. After that, it quantifies them using their ion intensity information along with internal quantification standards' information through the auxiliary Orange software, and finally exports the results to the auxiliary Tableau software for visualisation.

Programs like LDA[57, 58] and LipidBlast[59] kept the database approach, but integrated the RT dimension to enhance identification and quantification of peaks. The LDA database is actually formed of rule sets, taking into account the structural components and intensity of peaks, which are then used to identify peaks in an experimental MS spectrum based on a user provided compound list. On the other hand, the database of LipidBlast is formed of MS/MS spectra, which are used as reference when assessing experimental spectra. The program then matches peaks based on a user provided compound list and returns them with a matching score ascending from 0 (not matched) to 999 (matched).

The scoring system concept was also exploited by LIQUID[60]. Imported LC-MS/MS spectra is processed by LIQUID, which then scores the match of observed peaks to a known identity using its database seeded from LIPID MAPS and complemented of some additional non-mammalian and newly discovered lipid species. The scoring is based on expected fragment intensity, and is thus the log likelihood of the fragments observed.

Mzmine2[61] also kept the database approach, and uses the RT dimension to better process raw data and align experimental spectra together, notably by using the RANSAC algorithm and by applying a loess smoothing on the model obtained. Peak detection is carried by the program, via various algorithms provided as plug-ins in order to adapt to variations in experimental spectra. Peak identification is achieved via online database query on the m/z dimension, or user customizable database, which can then contain RT information. Keeping up with the streamlining of data processing, the program also carries some data visualisation capabilities like 2D and 3D data visualisation, scatter plots and histograms.

Also streamlining the MS data analysis, XCMS Online[62] too offers peak detection, sample alignment, identification and data visualisation, in addition to some statistical analysis capabilities. Imported spectral data is processed according to user-defined parameters, set for different experimental setups, peaks are identified via database query, and user requested results are produced.

Specializing in identification of oxidized phospholipids, LPPtiger[63] uses an in silico generated fragment database of oxidized species. It then matches observed peaks according to the spectra matching approach using a reverse dot-product algorithm like the scoring system of LipidBlast does. It also applies

four additional scoring systems using the spectral data's ion intensity and signal-to-noise ratio as well as isotope distribution patterns to provide more information for species identification. The five scores are ultimately merged to represent the identification quality.

Taking from the proteomics world, Skyline was adapted for lipidomics[64], especially for MRM methods. Its building-block approach enables peaks to be identified and quantified before their data is carried through to statistical analysis. This approach does not query a database of known lipids, but rather uses an adaptable database to recognize each fragment observed in the LC-MS/MS spectra, as the building-block structure of each compound to identify and assess is required.

As seen in this section, many bioinformatics tools exist. Nevertheless, the challenges of identification, reproducibility and format independence have yet to be fully conquered in the case of LC-MS/MS MRM wiff formatted data. Indeed, the programs listed above, which are also listed in Figure 1.6, are not sufficient for the analysis of such data.

1.3.3.2 Identification

Assessment of structural information without extensive fragmentation information requires a pre-existing database of spectral data and validated species structures as mentioned previously. As comprehensive databases, universally applicable across lipidomic methodologies, have yet to be developed, search external databases of theoretically possible (*in silico)* or tissue-validated (empirical) lipid species can narrow down possible lipid species candidates along the m/z dimension.

As multiple scan modes allow users to target different lipid backbones and headgroups, the specific classes and subclasses of lipids can, in part, be predicted by a characteristic ion. The characteristic ion refers to the product ion, the fragment of a precursor ion after CID, or the neutral loss offset in m/z between precursor and product ions, which allows to narrow down the class and subclass of a lipid species. In MRM, it then refers to the second m/z of a transition, or the offset between the m/z of a transition. GPCs, for example, can be targeted in positive ion mode, using a product ion scan of m/z 184 corresponding to its phosphocholine fragment[43, 65]. Sphingolipids with a d18:1 sphingosine base or a t18:0

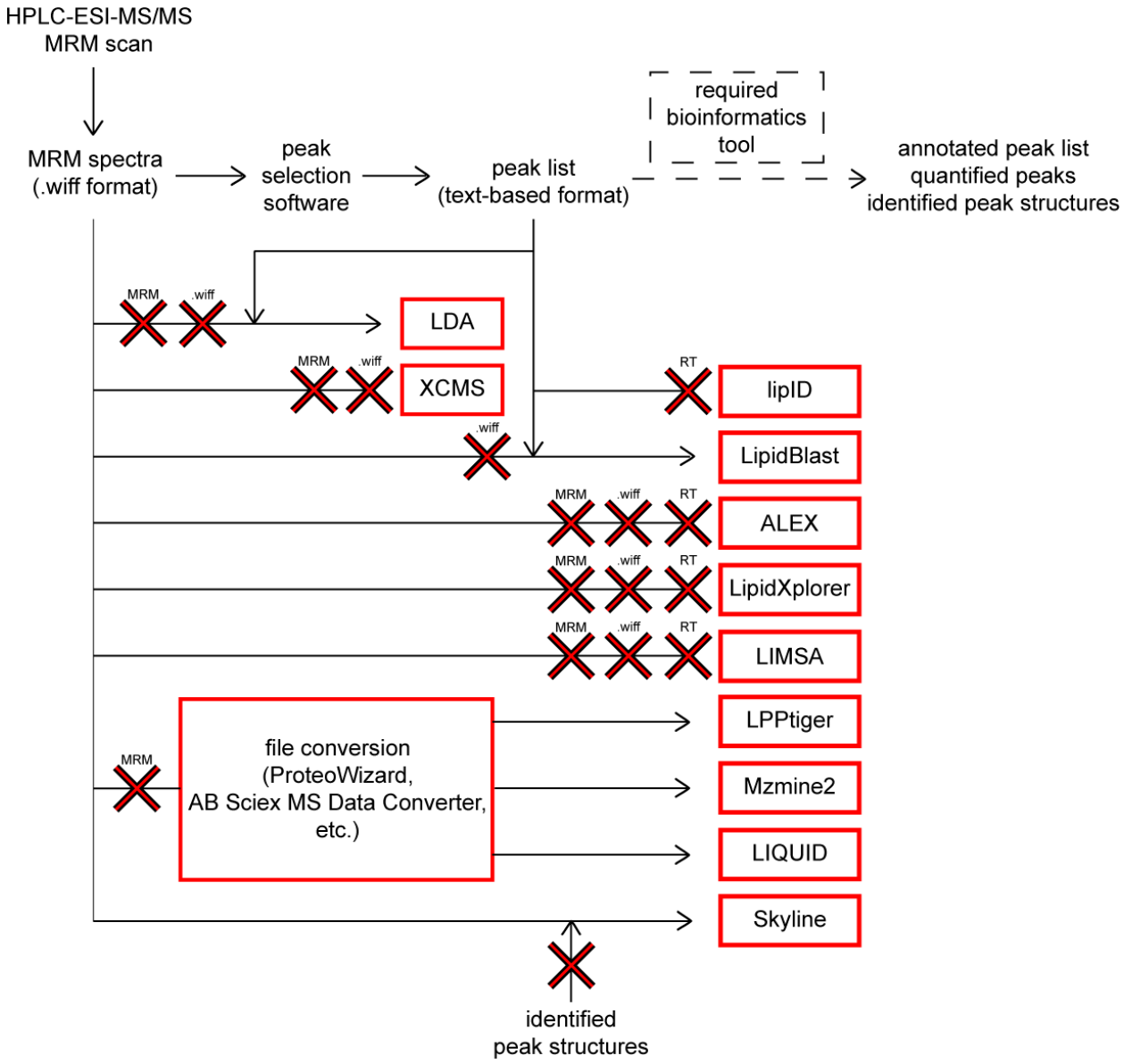

HPLC-ESI-MS/MS
MRM scan
MRM spectra
(.wiff format)
peak
selection
software
peak list
(text-based format)
required
bioinformatics
tool
annotated peak list
quantified peaks
identified peak structures
MRM
.wiff
LDA
XCMS
RT
lipID
LipidBlast
ALEX
LipidXplorer
LIMSA
LPPtiger
Mzmine2
LIQUID
Skyline
file conversion
(ProteoWizard,
AB Sciex MS Data Converter,
etc.)
identified
peak structures

Figure 1.6 HPLC-ESI-MS/MS MRM spectral data handling. Schematic depicting the lack of software available for targeted lipidomics approaches, as available software shown in the red boxes, cannot process the data produced (see section 1.3.3.2 and 1.3.3.4 for more details). Biological samples processed through HPLC-ESI-MS/MS MRM methods, resulting in MRM mass spectra of the .wiff format file extension, can be transformed into lists of detected peaks by peak selection via machine-specific or open-source software, although software to annotate these peaks that accept processed data in a text-based format are rare. Those that do exist require continuous MS spectra (Q1) whereas MRM scan discontinuous m/z targets. Moreover, most available software used to analyze and annotate MS spectra cannot process MRM data, .wiff formatted files, or do not consider front-end separation data (no retention time dimension is acknowledged), as shown by the red crosses in the flowchart (annotated MRM, .wiff, and RT respectively). File conversion of .wiff format to other file formats does not solve this issue as MRM data is not compatible with file converters. Annotation of peaks with Skyline requires full structure knowledge of the targeted peaks, and therefore cannot be used to annotate unknown or ambiguous structures. Thus, there is a need for a bioinformatics tool that annotates HPLC-ESI-MS/MS MRM peak lists.

phytosphingosine base can be targeted in positive ion mode using a product ion scan of m/z 264[66, 67]. GPEs can be detected in positive ion mode using a neutral loss scan with an offset of m/z 141[65, 68]. Table 1**.1** summarizes the characteristic ions and scan modes commonly used to distinguish lipid classes and subclasses in a targeted lipidomic approach.

Analyzing spectral data along the m/z dimension also requires the consideration of the ion adduct used for ionization. Indeed, the m/z of detected ions encompasses their ion adduct. For example, a GPC, as it contains a phosphocholine group with a positive charge on its nitrogen, is more prone to form a positive ion and is then profiled using a positive ion mode[43]. Thus, the ions detected after ESI will be protonated, and their mass will be defined as [M+H]+, as the ion adduct contributes to the detected ion m/z. Other ion adducts can also be used by using lithium or sodium, for example, which will modify the m/z detected, as the fragmented ion will be lithiated ([M+Li]+) or sodiated ([M+Na]+). By contrast, negative ion mode can be used if negatively charged fragments are expected. For example, GPE species can be analyzed in negative ion mode as their fragmentation yields a negatively charged dilyso phosphoethanolamine fragment ([M-H]-)[65].

Online or add-ons style search engines, like LIPID MAPS[69], LipidHome[52], the stand-alone lipid calculator of ALEX[51], lipID[56], and VaLID[70], specifically designed for lipidomics, all carry empirical and predicted spectral information to this regard in extensive databases[71]. These tools distinguish lipid species of different classes by name and m/z according to various mass tolerance and ion adducts settings. Thus, the m/z of detected peaks can be queried and possible candidates can be found, and vice-versa. Other databases, like the Human Metabolome Database[72] or the Metabolomics Workbench NIH Metabolomics Data Repository[73] are not designed specifically for lipids or relate to the data from lipidomics search engines as they are data depositories and not search engines, and thus will not be reviewed here.

The lipid species data stored in databases are most often recorded as monoisotopic, while they can also offer the option of displaying additional isotopes, as their mass is calculated using atoms that are all considered to be the most abundant isotope of an element. Their m/z will then only match the monoisotopic species unless another isotope is specifically queried. In practice, the natural distribution of

Lipid class/subclass	Ion mode	Scan mode	Characteristic ion	Reference
GPC	Positive	Precursor ion scan	Diagnostic ion of m/z 184	*43, 65*
SM	Positive	Precursor ion scan	Diagnostic ion of m/z 184	*65*
GPE	Positive	Neutral loss scan	Neutral loss offset of m/z 141	*65, 68*
GPE	Negative	Precursor ion scan	Diagnostic ion of m/z 196	*65*
GPS	Positive	Neutral loss scan	Neutral loss offset of m/z 87	*65*
GPS	Positive	Neutral loss scan	Neutral loss offset of m/z 185	*65*
SPH (sphingosine)	Positive	Precursor ion scan	Diagnostic ions of m/z 264 and 282	*66*
SPH (sphinganine)	Positive	Precursor ion scan	Diagnostic ions of m/z 266 and 284	*66*
SPH (phytosphingosine)	Positive	Precursor ion scan	Diagnostic ions of m/z 264 and 282	*67*

Table 1.1 List of commonly used characteristic ions for the GPC, GPS, GPE, and SPH lipid classes in positive or negative ion modes and precursor ion scan or neutral loss scan.

isotopes must be taken into account, as detected ions may not all be monoisotopic[40]. Isotopic peaks are only distinguishable along the m/z dimension by their m/z offset. Their biophysical properties remain similar as to produce near identical RTs, rendering isotopic peaks identical along the RT dimension. It is up to the investigator to decide whether all isotopic peaks or a single, monoisotopic peak are to be assessed and quantified. Recognition and correction for isotopic species is handled by software like LIMSA[54] and LDA[57], by either recognizing them based on a spectral database or by detecting them via their isotopic masses.

One challenge of identification is the identification of new species that are not present in a given spectral database, as algorithms used to identify species use the similarity of the experimental spectrum to spectra in their database. In addition to identifying lipid species peaks, identification has to encompass isotopic, dehydrated, deglycosylated, and oxidized peaks. Each tool has advantages. For example, lipID is able to recognize dehydrated lipid species based on their m/z[56]. LDA is able to recognize some unknown peaks given they are falling within certain rule sets extending its database[58]. However, there is a common caveat: information on oxidized and hydroxylated lipid species is lacking. Of the search engines listed above, only LIPID MAPS contains a few oxidized lipid species, of which none are computationally generated (26 oxidized glycerophospholipids out of 9629 catalogued glycerophospholipids, including 7 oxidized GPC and 16 oxidized GPE as of May 2018)[69]. LPPtiger[63], a software published in 2017, was designed to identify those oxidized species, but only spans five subclasses of oxidized glycerophospholipids. Moreover, it can only process spectral data in .raw input file format, as it was designed not as a search engine, but as a MS spectra processing tool.

By annotating spectral data along only the m/z dimension, identification of species of the same molecular mass, deemed isobars, remains ambiguous. Thus, the RT dimension available in HPLC-ESI-MS/MS lipidomics is used to further annotate the spectral data. This information is not processed by software designed for identifying species of direct infusion acquired data, such as lipID[56], LIMSA[54], LipidXplorer[55] and ALEX[51], as they do not consider this dimension (as shown in Figure 1.6).

Taking into account the RT dimension allows a higher confidence as to the accuracy of an annotation of molecular structure. However, the RT dimension cannot be handled like the m/z dimension. The m/z of detected ions is directly associated to its molecular mass, whereas its RT is not set in stone, and is prone to various experimental variations attributable to experimental conditions or differences between samples[50]. Thus, identification of experimental data with spectral datasets of m/z and RTs, in software like LipidBlast[59] and LIQUID[60], must account for experimental variability using a scoring system. Those scoring systems compute the similarity of the input data to the one in their database in order to find the most probable identity. In addition to the m/z and RT dimensions, the scoring system of LipidBlast also takes into account the instrument used for mass spectrometry[59], in order to compare experimental data to pertinent spectra, and reduce experimental variation.

Identification techniques can also rely on the fragmentation patterns and ion intensities, and thus identify lipid species by comparing the fragment ions and intensity, as well as the RT of experimental spectra, to their spectral database. LIQUID[60] is an example. Acquiring the fragmentation pattern of lipid species by MRM-IDA-EPI is one way to raise the accuracy of annotations by helping remove any ambiguity about the molecular structure of the detected ion. Although, using such a structural determination scan mode restrains the quantification that can be done. Furthermore, certain programs, like the adaptation of the proteomics tool Skyline to lipidomics, require users to know the structural identity of the observed lipid analytes prior to their annotation[64], which defeats the purpose of annotation prediction and renders impossible the identification of unknown or ambiguous structures. This is why establishing comprehensive databases capable of identifying detected ions based on their m/z and RT is critical. Such databases thereby enable the identification of isobars within and between matrices (i.e., lipid sample type, e.g., plasma, cerebrospinal fluid, different brain regions, etc.).

1.3.3.3 Reproducibility

As previously mentioned, the RT dimension is prone to experimental variation, which complicates the identification process. Software acquiring MS data thus need a correspondence step as

defined above in order to compare experimental spectra across matrices[50]. As many methods, instruments, and correction algorithms are used, identification by RT must be done in a relative fashion. One challenge of treating HPLC-ESI-MS/MS data is then to ensure the reproducibility of the data, across methods and samples.

LipidBlast tackles this challenge by using an in-silico approach. As a result, it claims its *in silico* generated spectra to be platform-independent, but is still limited by the differences in spectra produced by different instruments in finding an accurate threshold for its scoring identification[59]. LIQUID offers a trainable scoring system and a customizable database to address platform-dependent differences[60]. Skyline is target-oriented, and claims to be adaptable cross-platforms[64]. Reproducibility in HPLC-ESI-MS/MS data cross platforms and cross samples is then possible with a flexible or adaptable scoring system and/or database.

1.3.3.4 Format independence

Another bioinformatics challenge in handling HPLC-ESI-MS/MS MRM data is the mundane yet fundamental issue of file formatting, as shown in Figure 1.6. The acquisition of spectral data directly from mass spectrometers is handled by various data processing software, which are most commonly proprietarily associated with the machine itself. Such software allow the spectral data to be transformed into a list of detectable peaks with various features and in various formats. This end result is not, however, complete, as characterization of the detected peaks has yet to be done. Various programs are available for this purpose, some being part of the commercial data acquisition software, some being stand-alone open-source programs, some of which have been cited above. These programs can however only accept data in their specific file format requirements.

HPLC-ESI-MS/MS MRM spectral data cannot be analyzed as, or converted to, simple data acquired through un-targeted full scan methods (MS^1). Thus, programs like LDA[58], ALEX[51], LipidXplorer[55] and LIMSA[54], which require non-MRM data, cannot be used. Moreover, even if data acquired through MRM methods is supported by open-source programs, which annotate spectral data,

such as LipidBlast[59], LPPtiger[63], MzMine2[61], LIQUID[60] or Skyline[64], it is not necessarily in a compatible file format, limiting their use. Non-commercial programs exist for conversion, but they are not universally applicable to all proprietorial MS formats, and conversion of data without damaging it is an unspoken challenge in the field. For example, converters like ProteoWizard claim to convert .wiff formatted files[74], yet cannot convert MRM data of that format. XCMS[62] also claims to accept .wiff formatted files on their website, but the processing returns only error messages, as the MRM .wiff files are still incompatible with its services. Even AB SCIEX's own MS Data Converter User Guide states that it cannot convert MRM data correctly. Figure 1.6 summarizes this formatting challenge.

Spectral data after analysis can be exported as lists of peaks with their inherent features, in a text-based format. The information carried through would be reduced to peak features, and would be easy for programs to interpret. The .mgf file format LipidBlast uses[59] is a good example, as the Mascot Generic Format is widely used. However, again, no support of .wiff MRM data has been found.

Taken together, there are a number of bioinformatic challenges ranging from mundane issues of file format to more computationally complex issues of peak correspondence and identification that impede rapid data assessment in the field of lipidomics. My thesis will focus on obstacles that impact the use of MRM lipidomic data.

1.4 Objectives and hypotheses

The overarching goal of my thesis is to develop necessary bioinformatic tools addressing challenges in lipid identification *between* and *across* matrices, or sample type, in targeted MRM data analysis. Performing data annotation and analysis in lipidomics such that reproducibility is ensured has proven to be challenging. Causes may be different experimental methods, bioinformatic methods, or even inaccuracy of databases. It is of no surprise to see laboratories using varying programs and algorithms to acquire their data, as each team faces different sources of experimental variation due to mass spectrometry RT and m/z variations[50]. However, it remains crucial for the bioinformatics methods to

annotate and interpret these data without creating bias over such differences in data acquisition. For example, there was a report in 2014 of eight GPC species associated with Alzheimer's Disease using plasma samples[75], but it could not be reproduced by a different research team two years later using nearly identical experimental methods on serum samples of a larger cohort, including the same metabolomics assay kit used, yet different analysis methods to compare their experimental groups[76].

Specifically, I focus here on optimizing data processing and filtering in a format independent manner, as to enable reproducibility and comparisons of datasets across experiments, instruments, and laboratories. These important steps are emphasized in bold in Figure 1.3. If an equation system representing lipid structures can be based on their combinatorial structures, as depicted in Figure 1.1 and Figure 1.2, I hypothesize, by deduction, that it is possible to develop a robust *in silico* database of predicted lipid structures. Thus, a bioinformatic tool can be built to tackle the identification step in Figure 1.3. I further hypothesize that spectral data acquired by a targeted MRM method can be annotated and filtered after exportation to simple text format exploiting RT and m/z dimension to annotate species, identify corresponding lipids, and discriminate distinct species, by assessment of all data points simultaneously. In this way, the bioinformatics gap depicted by the dotted box in Figure 1.6, between MRM acquired peak lists and annotated data, will be addressed. These hypotheses will be tested through the aims below.

1.4.1 Aim 1

I aim to generate bioinformatic tools that will:

(a) compute the identities of possible lipids candidates with a given m/z and user-defined features and

(b) compute the m/z of a given lipid structure, to user specifications.

1.4.2 Aim 2

I aim to enhance the currently used lipid analysis pipeline in the Neural Regeneration Laboratory so as to optimize filtering of processed mass spectrometry spectral data acquired through a targeted method, providing accurate quantification and identification of lipid species, as well as graphical and statistical outputs.

Chapter 2: Lipid Identification Tool (LIT), a lipid search engine designed for predicting lipid structures according to user specifications*

2.1 Innovation

Lipid search engines, like LIPID MAPS[69] and others mentioned previously, store information in extensive online or offline databases. Every inquiry prompts a search through their database that can be tuned by different parameters such as mass tolerance or ion adduct. The information they can retrieve is fixed to the information contained by the database, predicted or empirical.

The innovation in LIT is that no databases of lipid structures are required. Structural lipid prediction is generated "on the fly" by a system of linear equations. Analogous to the rule sets system used by LipidBlast to extrapolate its database[59], LIT can compute the m/z or structure of previously undocumented lipid species according to user-specified constraints. These constraints allow for the identification by m/z, by name, or by modification (oxidation, hydroxylation, and dehydration) of lipid species of multiple lipid classes.

2.2 Method of computation

The m/z of a given lipid species is directly associated with its mass and its charge. The charge is defined by the experimental methods used, including the ion adduct, which is user-specified. The mass is defined by the linear combination of the masses of the structural components of the given lipid. Thus, by knowing the m/z of a lipid, it is possible to predict its structures by solving an equation of common features summing to its mass.

* This chapter is being prepared for submission to the journal Bioinformatics. The headings reflect the requirements of a contribution to the journal as an Application Note.

$$mass = atomsSub + atomsChain + atomsUnsat + atomsAdj + atomsMod + ion \quad (2.1)$$

Equation 2.1 Sum of all variables decomposing the mass of a lipid, where *mass* corresponds to the m/z of the lipid, and where *atomsSub*, *atomsChain*, *atomsUnsat*, *atomsAdj*, *atomsMod*, and *ion* correspond to the mass of the atoms in each "building-block" of the lipid: atoms in common for its lipid class, atoms in its carbon chains, atoms removed due to unsaturations on the carbon chains, atoms added or removed due to different linkages of the chain(s) to the backbone, different backbone, etc., atoms added or removed due to oxidation or hydroxylation of the carbon chains, and atoms added or removed during the ionization step in a mass spectrometry protocol respectively,

The equation used by LIT (Equation 2.1) resembles a building-block approach: each element of the right-hand side represents a defined group of atoms that can be interchanged depending on the lipid class, subclass, hydrocarbon chain length, etc. The total sum of these atoms contributes to the *mass* of the lipid. Lipid classes and subclasses have recurring atoms, like the glycerol backbone of glycerophospholipids or the phosphocholine of a GPC. Those atoms are accounted for in the *atomsSub* variable, standing for "atoms subclass". The number of carbons in the hydrocarbon chains and the number of unsaturation they possess are encompassed by the *atomsChain* and *atomsUnsat* variables, respectively. The type of linkage of those chains to the backbone of a lipid is taken into consideration by the *atomsAdj* variable ("atoms adjustment"), which adjusts the mass accordingly. Oxidation and hydroxylation of the lipid species is represented by the *atomsMod* variable ("atoms modification"). Finally, the *ion* variable allows any mass adjustment due to the experimental method used. Ergo, by using Equation 2.1 in its process displayed in Figure 2.7, LIT can identify plausible structures for a lipid of a given m/z, and the m/z of a given structure.

The mass of each atom of carbon, hydrogen, nitrogen, oxygen, and phosphate used in Equation 2.1 are the exact monoisotopic masses of the most abundant isotope of the listed elements. Thus, an atom of carbon accounts for 12.000000 atomic mass units (amu), an atom of hydrogen, for 1.007825 amu, an atom of nitrogen, for 14.003074 amu, an atom of oxygen, for 15.994915 amu, and an

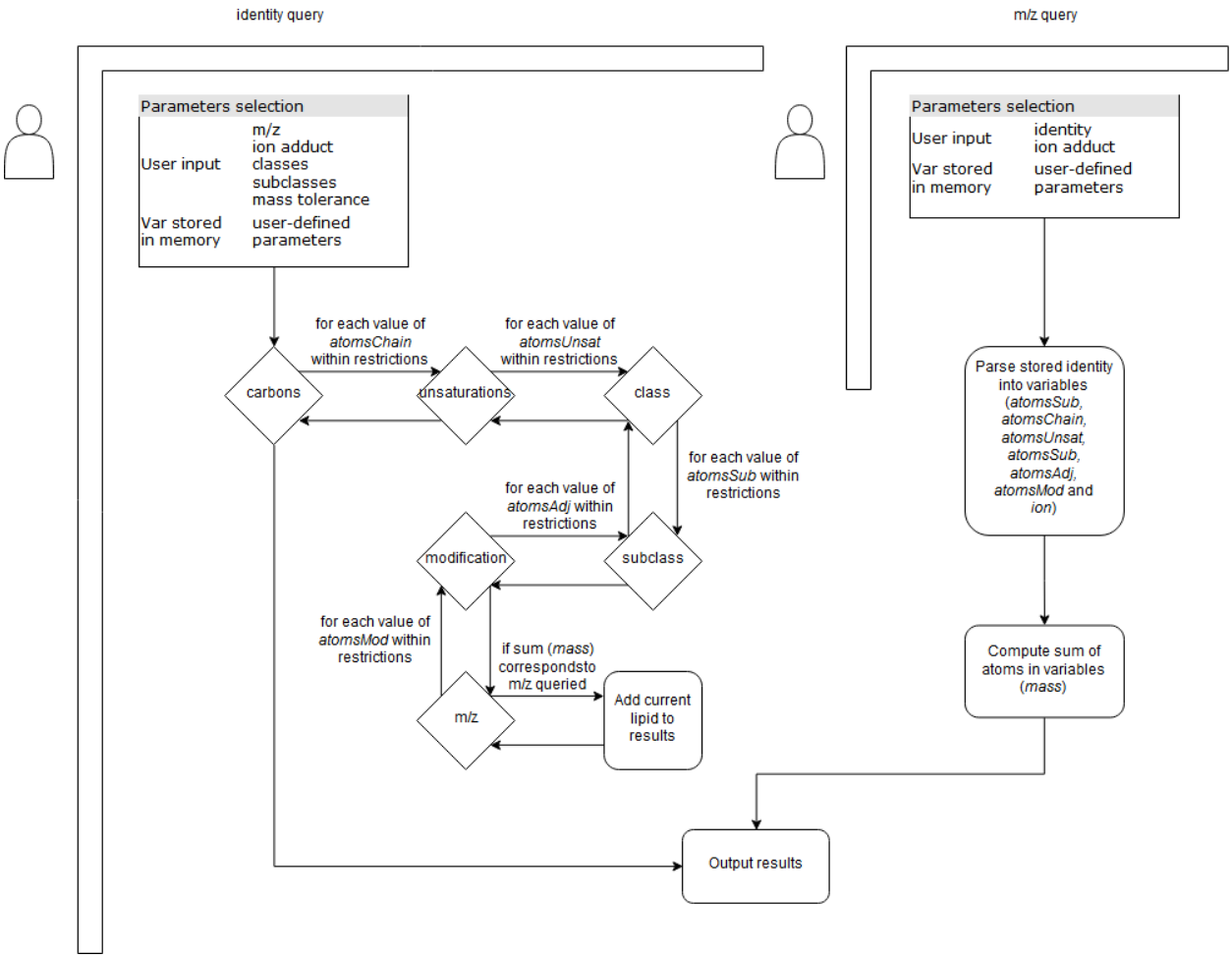

identity query
m/z query
Parameters selection
User input
m/z
ion adduct
classes
subclasses
mass tolerance
Var stored in memory
user-defined parameters
Parameters selection
User input
identity
ion adduct
Var stored in memory
user-defined parameters
for each value of atomsChain within restrictions
for each value of atomsUnsat within restrictions
carbons
unsaturations
class
for each value of atomsSub within restrictions
for each value of atomsAdj within restrictions
modification
subclass
for each value of atomsMod within restrictions
if sum (mass) correspondsto m/z queried
m/z
Add current lipid to results
Parse stored identity into variables (atomsSub, atomsChain, atomsUnsat, atomsSub, atomsAdj, atomsMod and ion)
Compute sum of atoms in variables (mass)
Output results

Figure 2.7 Flowchart of LIT. Lit allows the user to identify potential lipids based on m/z (Identity query) and to determine the m/z based on a lipid's identity (Mass-over-charge query). **Identity query**: LIT requires the user to enter the m/z of the lipid to identify, and to define the search restrictions for the class(es), subclass(es), ion adduct, and mass tolerance parameters (user-defined parameters, stored in the memory of the program). Then, LIT will cycle through all possible values for its five variables (*atomsChain*, *atomsUnsat*, *atomsSub*, *atomsAdj*, and *atomsMod*) within the set restrictions and output the identities for which the *mass* computed corresponds to the starting m/z. **Mass-over-charge query**: LIT requires the user to enter the identity of the lipid, as well as the desired ion adduct (user-defined parameters, stored in the memory of the program). Then, LIT will parse the input identity to its five variables (*atomsChain*, *atomsUnsat*, *atomsSub*, *atomsAdj*, and *atomsMod*), and compute the *mass* (m/z) of the lipid species.

atom of phosphate, for 30.973763 amu. The mass adjustments for the different ion adducts are computed using the same approach. The other atoms used in addition to carbon, hydrogen, nitrogen, oxygen and phosphate are sodium accounting for 22.989770 amu, potassium, for 38.963708 amu, lithium, for 7.016005 amu, and chlorine, for 34.968853 amu. LIT can also take in consideration deuterated lipids when they are queried by identity, and not by m/z. The mass of a deuterium atom replacing that of a hydrogen is an additional 2.014102 amu.

2.3 Interface

LIT was built in Python 3 (Python Software Foundation, Beaverton, Oregon) using the open-source wxPython cross-platform GUI library. It is compiled as a stand-alone executable file (or application). It is compatible and operational on both Windows and MAC operating systems.

The interface of LIT is shown in Figure 2.8. It is composed of two main panels: the inquiry panel on the left, and the results panel on the right. User-specified parameters and constraints are defined in the inquiry panel, whereas the results of inquiries are listed in the results panel. The user can search for plausible lipid species at a given m/z by entering it in the "Mass-over-charge ratio" field and selecting the desired ion adduct from the ion adduct drop-down menu below. The mass tolerance can be adjusted if needed via the mass tolerance drop-down menu over the "Mass-over-charge ratio" field. Different oxidation modifications can be toggled on or off by their respective checkboxes as required. The lower section of the inquiry panel lists the species to be interrogated. The species are organized in eight groups: GPS, GPE, GPC, sphingomyelins, monohydroxy SPH, dihydroxy SPH, trihydroxy SPH, and sterols (see Figure 2.8). Like the oxidation settings, each subclass can be toggled on or off by their respective checkboxes. When all parameters are set, the user can launch the search by clicking on the "Search!" button below the "Mass-over-charge ratio" field. LIT then computes the possible candidates and lists the results in the results panel. The results shown can be sorted by alphabetical (and numerical) order, and

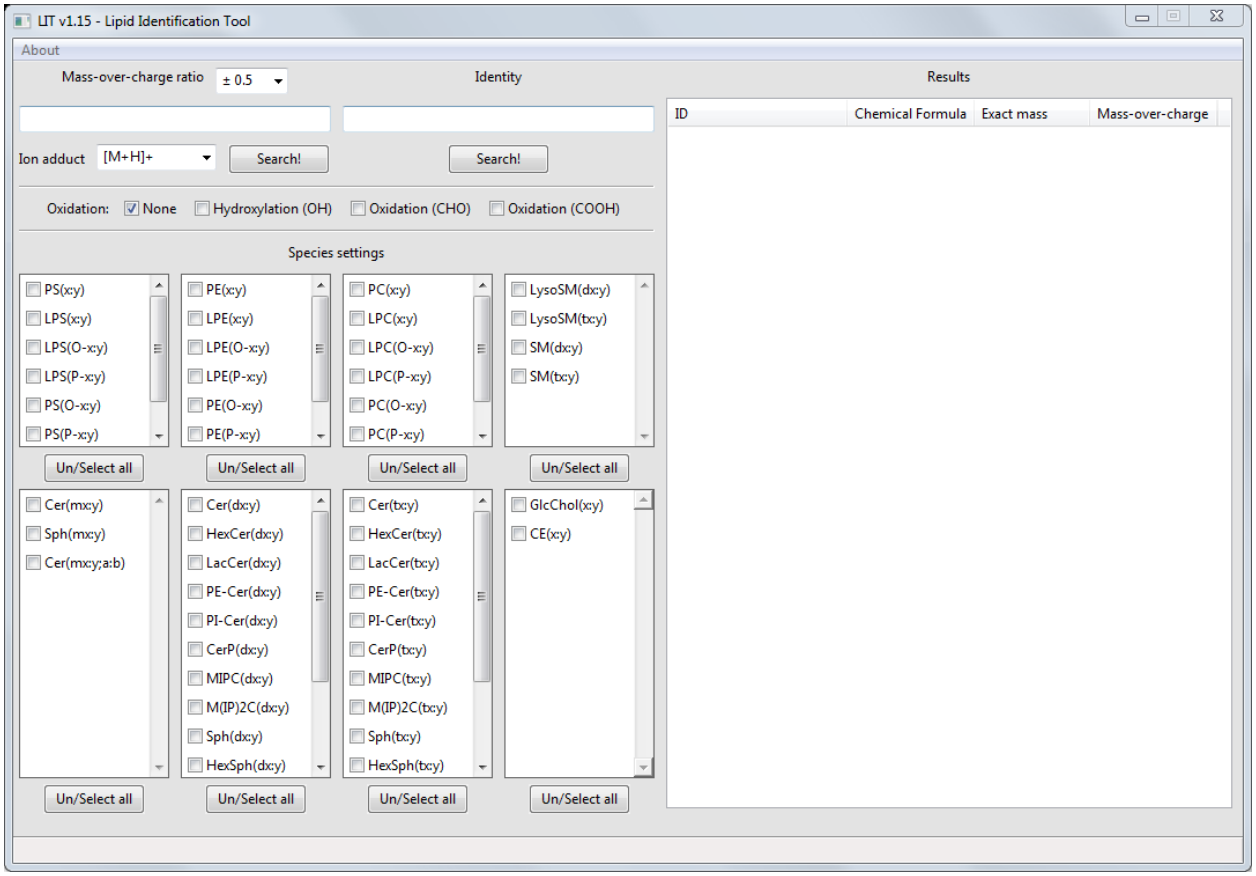

LIT v1.15 - Lipid Identification Tool
About
Mass-over-charge ratio
± 0.5
Identity
Results
ID
Chemical Formula
Exact mass
Mass-over-charge
Ion adduct
[M+H]+
Search!
Search!
Oxidation:
None
Hydroxylation (OH)
Oxidation (CHO)
Oxidation (COOH)
Species settings
PS(x:y)
LPS(x:y)
LPS(O-x:y)
LPS(P-x:y)
PS(O-x:y)
PS(P-x:y)
PE(x:y)
LPE(x:y)
LPE(O-x:y)
LPE(P-x:y)
PE(O-x:y)
PE(P-x:y)
PC(x:y)
LPC(x:y)
LPC(O-x:y)
LPC(P-x:y)
PC(O-x:y)
PC(P-x:y)
LysoSM(d:y)
LysoSM(t:y)
SM(d:y)
SM(t:y)
Un/Select all
Un/Select all
Un/Select all
Un/Select all
Cer(m:y)
Sph(m:y)
Cer(m:y;a:b)
Cer(d:y)
HexCer(d:y)
LacCer(d:y)
PE-Cer(d:y)
PI-Cer(d:y)
CerP(d:y)
MIPC(d:y)
M(IP)2C(d:y)
Sph(d:y)
HexSph(d:y)
Cer(t:y)
HexCer(t:y)
LacCer(t:y)
PE-Cer(t:y)
PI-Cer(t:y)
CerP(t:y)
MIPC(t:y)
M(IP)2C(t:y)
Sph(t:y)
HexSph(t:y)
GlcChol(x:y)
CE(x:y)
Un/Select all
Un/Select all
Un/Select all
Un/Select all

Figure 2.8 Interface of LIT version 1.15 on a Windows operating system.

exported as an .xlsx file via the drop-down menu accessible at the top-left corner of the window. For queries by lipid name, the name is entered in the "Identity" field and the ion adduct is selected from the ion adduct drop-down menu before clicking on the "Search!" button underneath the "Identity" field. The other parameters of the inquiry panel are ignored when searching for a lipid by its identity.

2.4 Discussion

LIT was built to address the need for a search engine with a GUI for modified lipid species, able to compute possible species for a given m/z and vice-versa to user specifications. Currently, the computations of LIT enable searches for GPS, GPE, GPC, sphingomyelins, mono-, di-, and trihydroxy SPH as well as glucosylcholesterols and cholesterol esters. It also enables searches for oxidized and hydroxylated species, as well as dehydrated, via its ion adducts selection. Lipid identities are recognized both in total carbon and molecular identities, while results are shown in their total carbon identities.

When compared to other lipid search engines, the species recognized and predicted by LIT are not available or of limited availability in other search engines. Table 2.2 summarizes the species LIT can predict and their availability in other search engines. LIT also allows the computation of the m/z of lipid species under 20 different adducts, as well as the neutral mass. This is consistent with LIPID MAPS, which allow use of the same adducts and neutral mass calculations[69]. ALEX counts 22 different ion adducts, but they are not available for each lipid class, and the neutral mass is not calculated[77]. As for VaLID, it counts only 5 ion adducts, but also computes the neutral mass[70]. LipidHome shows only the neutral mass and does not compute the m/z for any ion adduct[52].

In summary, LIT has been developed as a robust *in silico* database capable of predicting lipid structures based on Equation 2.1. This scheme converts the combinatorial structures of lipids shown in Figure 1.1 and Figure 1.2 into variables it can easily manipulate and use in order to build possible identities of possible lipids with a given m/z and user-defined features, and vice-versa. Its computation method, laid down in Figure 2.7, allows LIT to predict the structure of lipid species across 58 subclasses

(see Table 2.2), including 16 subclasses not listed in LIPID MAPS, 18 not in ALEX, 37 not in VaLID, and 43 not in LipidHome. As the structures are predicted, the number of carbons and unsaturations are not restricted by experimental data, and thus LIT includes a broader range of carbon chains and unsaturations than LIPID MAPS and ALEX, like VaLID and LipidHome. Last but not least, LIT is able to compute oxidized structures (OH, CHO, and COOH) for all subclasses it spans (when applicable), unlike its rivals. In brief, I developed a new bioinformatics tool to compute the possible identities of possible lipids with a given m/z and user-defined features and vice-versa, spanning lipid subclasses and species on which documentation is lacking in currently available lipid search engines.

LIT			LIPIDMAPS			ALEX			VaLID			Lipidhome		
subclass	carbons: unsaturations	oxidation	available	carbons: unsaturations	oxidation	available	carbons: unsaturations	oxidation	available	carbons: unsaturations	oxidation	available	carbons: unsaturations	oxidation
PS(x:y)	12-60:0-12	OH, CHO, COOH	yes	12-44:0-11	none	yes	26-46:0-12	none	yes	0-60:0-12	none	yes	4-60:0-18	none
LPS(x:y)	12-30:0-9	OH, CHO, COOH	yes	12--22:0-6	none	yes	10--26:0-6	none	yes	0-30:0-6	none	yes	2-30:0-9	none
LPS(O-x:y)	12-30:0-9	OH, CHO, COOH	yes	16-20:0	none	no	-	-	yes	0-30:0-6	none	yes	2-30:0-9	none
LPS(P-x:y)	12-30:0-9	OH, CHO, COOH	yes	16-20:0	none	no	-	-	yes	0-30:0-6	none	no	-	-
PS(O-x:y)	12-60:0-12	OH, CHO, COOH	yes	28-42:0-6	none	no	-	-	yes	0-60:0-12	none	yes	4-60:0-18	none
PS(P-x:y)	12-60:0-12	OH, CHO, COOH	yes	28-42:0-12	none	no	-	-	yes	0-60:0-12	none	no	-	-
dialkylPS(x:y)	12-60:0-12	OH, CHO, COOH	no	-		no	-	-	yes	0-60:0-12	none	yes	4-60:0-18	none
PE(x:y)	12-60:0-12	OH, CHO, COOH	yes	12-52:0-12	OH	yes	26-46:0-12	none	yes	0-60:0-12	none	yes	4-60:0-18	none
LPE(x:y)	12-30:0-9	OH, CHO, COOH	yes	12--24:0-6	none	yes	10--26:0-6	none	yes	0-30:0-6	none	yes	2-30:0-9	none
LPE(O-x:y)	12-30:0-9	OH, CHO, COOH	yes	16-20:0-1	none	yes	10--26:0-6	none	yes	0-30:0-6	none	yes	2-30:0-9	none
LPE(P-x:y)	12-30:0-9	OH, CHO, COOH	yes	16-20:0-1	none	no	-	-	yes	0-30:0-6	none	no	-	-
PE(O-x:y)	12-60:0-12	OH, CHO, COOH	yes	28-42:0-6	none	yes	26-42:0-12	none	yes	0-60:0-12	none	yes	4-60:0-18	none
PE(P-x:y)	12-60:0-12	OH, CHO, COOH	yes	28-42:0-10	OH	yes	26-42:0-12	none	yes	0-60:0-12	none	no	-	-
dialkylPE(x:y)	12-60:0-12	OH, CHO, COOH	no	-	-	no	-	-	yes	0-60:0-12	none	yes	4-60:0-18	none
PC(x:y)	4-60:0-12	OH, CHO, COOH	yes	4-52:0-12	OH, CHO, COOH	yes	26-46:0-12	none	yes	0-60:0-12	none	yes	4-60:0-18	none
LPC(x:y)	2-30:0-9	OH, CHO, COOH	yes	2--24:0-6	none	yes	10--26:0-6	none	yes	0-30:0-6	none	yes	2-30:0-9	none
LPC(O-x:y)	2-30:0-9	OH, CHO, COOH	yes	5--20:0-1	none	yes	10--26:0-6	none	yes	0-30:0-6	none	yes	2-30:0-9	none
LPC(P-x:y)	2-30:0-9	OH, CHO, COOH	yes	14-20:0-1	none	no	-	-	yes	0-30:0-6	none	no	-	-
PC(O-x:y)	3-60:0-12	OH, CHO, COOH	yes	3-44:0-6	none	yes	26-42:0-12	none	yes	0-60:0-12	none	yes	4-60:0-18	none
PC(P-x:y)	3-60:0-12	OH, CHO, COOH	yes	18-42:0-6	none	yes	26-42:0-12	none	yes	0-60:0-12	none	no	-	-
dialkylPC(x:y)	2-60:0-12	OH, CHO, COOH	no	-	-	no	-	-	yes	0-60:0-12	none	yes	4-60:0-18	none
LysoSM(dx:y)	12-30:0-3	OH	no	-	-	yes	12--22:0-2	OH	no	-	-	no	-	-
LysoSM(tx:y)	12-30:0-3	OH	no	-	-	yes	12--22:0-2	OH	no	-	-	no	-	-
SM(dx:y)	12-60:0-3	OH, CHO, COOH	yes	30-44:0-3	none	yes	30-48:0-2	OH	no	-	-	no	-	-
SM(tx:y)	12-60:0-3	OH, CHO, COOH	no	-	-	yes	30-48:0-2	OH	no	-	-	no	-	-
Cer(mx:y)	12-60:0-3	OH, CHO, COOH	yes	34-44:0-2	none	yes	30-48:0-2	OH	no	-	-	no	-	-
Sph(mx:y)	12-60:0-3	OH, CHO, COOH	no	-	-	yes	12--22:0-2	OH	no	-	-	no	-	-

LIT			LIPIDMAPS			ALEX			VaLID			Lipidhome		
subclass	carbons: unsaturations	oxidation	available	carbons: unsaturations	oxidation	available	carbons: unsaturations	oxidation	available	carbons: unsaturations	oxidation	available	carbons: unsaturations	oxidation
Cer(mx:y;a:b)	22-98:0-12	OH, CHO, COOH	no	-	-	yes	12--22:0-2	OH	no	-	-	no	-	-
Cer(dx:y)	12-60:0-3	OH, CHO, COOH	yes	20-46:0-3	OH	yes	30-48:0-2	OH	no	-	-	no	-	-
HexCer(dx:y)	12-60:0-3	OH, CHO, COOH	yes	34-44:0-3	OH	yes	30-48:0-2	OH	no	-	-	no	-	-
LacCer(dx:y)	12-60:0-3	OH, CHO, COOH	yes	30-44:0-2	none	no	-	-	no	-	-	no	-	-
PE-Cer(dx:y)	12-60:0-3	OH, CHO, COOH	yes	30-40:0-3	OH	yes	30-48:0-2	OH	no	-	-	no	-	-
PI-Cer(dx:y)	12-60:0-3	OH, CHO, COOH	yes	32-46:0-1	OH	yes	30-48:0-2	OH	no	-	-	no	-	-
CerP(dx:y)	12-60:0-3	OH, CHO, COOH	yes	20-44:0-2	none	yes	30-48:0-2	OH	no	-	-	no	-	-
MIPC(dx:y)	12-60:0-3	OH, CHO, COOH	yes	34-46:0	OH	yes	30-48:0-2	OH	no	-	-	no	-	-
M(IP)2C(dx:y)	12-60:0-3	OH, CHO, COOH	yes	34-46:0	OH	yes	30-48:0-2	OH	no	-	-	no	-	-
Sph(dx:y)	12-60:0-3	none	yes	14-19:1--3	OH	yes	12--22:0-2	OH	no	-	-	no	-	-
HexSph(dx:y)	12-60:0-3	none	yes	18:1	none	no	-	-	no	-	-	no	-	-
S1P(dx:y)	12-60:0-3	none	yes	16-19:0-1	none	yes	12--22:0-2	OH	no	-	-	no	-	-
Cer(a:b;dx:y)	22-98:0-12	OH, CHO, COOH	yes	48-66:0-1	none	yes	44-69:0-8	OH	no	-	-	no	-	-
Cer(dx:y;a:b)	22-98:0-12	OH, CHO, COOH	no	-	-	no	-	-	no	-	-	no	-	-
Gb3-Cer(dx:y)	12-60:0-3	OH, CHO, COOH	yes	34-44:1--2	none	no	-	-	no	-	-	no	-	-
Cer(tx:y)	12-60:0-3	OH, CHO, COOH	yes	34-46:0-1	OH	yes	30-48:0-2	OH	no	-	-	no	-	-
HexCer(tx:y)	12-60:0-3	OH, CHO, COOH	yes	34-44:0-1	OH	yes	30-48:0-2	OH	no	-	-	no	-	-
LacCer(tx:y)	12-60:0-3	OH, CHO, COOH	no	-	-	no	-	-	no	-	-	no	-	-
PE-Cer(tx:y)	12-60:0-3	OH, CHO, COOH	no	-	-	yes	30-48:0-2	OH	no	-	-	no	-	-
PI-Cer(tx:y)	12-60:0-3	OH, CHO, COOH	yes	34-46:0	OH	yes	30-48:0-2	OH	no	-	-	no	-	-
CerP(tx:y)	12-60:0-3	OH, CHO, COOH	no	-	-	yes	30-48:0-2	OH	no	-	-	no	-	-
MIPC(tx:y)	12-60:0-3	OH, CHO, COOH	yes	34-46:0	OH	yes	30-48:0-2	OH	no	-	-	no	-	-
M(IP)2C(tx:y)	12-60:0-3	OH, CHO, COOH	yes	34-46:0	OH	yes	30-48:0-2	OH	no	-	-	no	-	-
Sph(tx:y)	12-60:0-3	none	yes	16-18:0	OH	yes	12--22:0-2	OH	no	-	-	no	-	-
HexSph(tx:y)	12-60:0-3	none	no	-	-	no	-	-	no	-	-	no	-	-
S1P(tx:y)	12-60:0-3	none	yes	18:0	none	yes	12--22:0-2	OH	no	-	-	no	-	-
Cer(a:b;tx:y)	22-98:0-12	OH, CHO, COOH	no	-	-	yes	44-69:0-8	OH	no	-	-	no	-	-

LIT			LIPIDMAPS			ALEX			VaLID			Lipidhome		
subclass	carbons: unsaturations	oxidation	available	carbons: unsaturations	oxidation	available	carbons: unsaturations	oxidation	available	carbons: unsaturations	oxidation	available	carbons: unsaturations	oxidation
Cer(tx:y;a:b)	22-98:0-12	OH, CHO, COOH	no	-	-	no	-	-	no	-	-	no	-	-
Gb3-Cer(tx:y)	12-60:0-3	OH, CHO, COOH	no	-	-	no	-	-	no	-	-	no	-	-
GlcChol	0-60:0-12	none	yes	0-22:0-3	none	no	-	-	no	-	-	no	-	-
CE(x:y)	12-60:0-9	none	yes	12--24:0-6	none	yes	2--26:0-6	none	no	-	-	no	-	-

Table 2.2 List of lipid subclasses recognized by LIT and their availability in other lipid search engines. The lipid subclasses listed on the left-most column are as they appear in LIT's interface. The "G" in the GPC, GPS, and GPE abbreviations is dropped.

Chapter 3: LITL, a filtering data program designed for analysis of targeted lipidomics data

Addressing the need for annotating and filtering targeted lipidomics spectral data after peak identification in proprietary software and following exportation to simple text-based format led me to the creation of LITL (Lipid Identification for Targeted Lipidomics). I developed LITL to address the challenges of peak identification and filtering (see Figure 1.3) in four steps. First, LITL annotates and identifies lipid analytes by matching spectral information with structural information corresponding to retention times and m/z data derived from a library of multiple samples of different matrices. This library allows the RT covariance to be computed across species and matrices to ensure correspondence. Second, LITL identifies isotopes, dehydrations, and in-source deglycosylations based on RT and m/z relationships of diagnostic ions to their precursor molecules. Third, LITL computes the abundance of detected ions relative to internal quantification standards with comparable MS response. Finally, LITL outputs annotated data in a user-friendly format, allowing the production of statistical analyses and graphical representations. I developed this program using the MATLAB (2016a, 2016b, 2017a, MathWorks, Natick, Massachusetts) scripting language and compiled it as a stand-alone program using the compiling toolbox of MATLAB.

3.1 Principles

In order to build a program capable of such annotation features, the challenges they represent must be translated. Thus, to enable the computation of assignments, annotations, quantification and so on, sets of rules have to be established. Those rules are in turn based upon principles of biochemistry and mathematics, as well as assumptions, as explained in the next sections.

3.1.1 Biophysical rules

For LITL to assign proper identities to selected peaks in spectral data, it needs to consider biophysical rules that dictate the m/z and RTs of each analyte. In Chapter 2, the m/z was mentioned to be directly associated with the structure of the molecules. As for the RT, as mentioned in Chapter 1, it is the result of the hydrophobic interactions within the chromatography column during HPLC (see Figure 1.4B). Therefore, when peaks are identified using their m/z and RT, they have to abide the biophysical rules applying to their assigned identity.

When considering only m/z, it is possible to narrow down the possible identities of selected peaks, as demonstrated in Chapter 2. In mass spectrometry, the higher the recorded m/z of a selected peak is, the larger its mass. These systematic patterns can also be seen in the RT dimension, as a higher RT would then translate to a molecule with a higher affinity for the hydrophobic beads of a reversed-phase HPLC column. Thus, it is no surprise that spectral patterns based on chemical structures can be modeled, and used to predict the RT of certain lipid species[78, 79]. It is then possible to cluster certain species into series based on common structural properties, that, when viewed on the m/z versus RT plane, form regression lines parallel in orientation[80]. Misidentification of peaks would then be observable as a break in these patterns and as a lower correlation of these regression lines.

3.1.2 Multivariate normal distribution

The RTs of peaks in experimental data vary from sample to sample due to sample and experimental differences, as mentioned in section 1.3.3.2. However, the RTs of a spectrum as a whole can be conceptualized as a multivariate normal distribution, where each dimension represents the normal distribution of one species, and can be written as $N(\mu, \Sigma)$, with a vector μ of RT average corresponding to all the distinct lipid identities observed and Σ covariance matrix of all these distinct lipid identities observed. It is interesting to note that with this notation, RTs' dependence on one another appears, as the

covariance term Σ enables the relationship between observed peaks to be taken into account. These assumptions are consistent with the cluster model described in Daly et al., 2014[79], in which Gaussian distributions are used to model peak m/z, intensity, and RT, and in which the annotation of observed peaks also take into account other observed peaks.

3.1.3 Covariance and missing values

With the assumption of the RTs of spectra coming from a multivariate normal distribution as mentioned in the previous section, the model for assigning peak identities can be built. However, it first requires the definition of both the mean vector and the covariance matrix of the RTs. By using a database concatenating already seen lipid species in multiple samples of different matrices, these can be defined, and updated as the database grows.

Yet, for the covariance matrix to be computed, RTs needs to be defined across all samples and species in the database. In order to bypass this difficulty, as some lipids are not seen in all biological matrices, we can estimate the missing values by using an estimation-maximization algorithm described in Little & Rubin, 2014[81, 82], which makes use of a sweep operator described by Dempster in 1969[83]. When applied to the augmented covariance matrix, the sweep operator allows estimation of incomplete normal data: the missing RTs. Thus, the covariance can be computed while respecting the multivariate normal distribution assumption, and the model can be built.

The first step in this estimation-maximization is the initial parameters set-up. A straightforward solution is to use a mean imputation[82, 84] and compute the covariance of the resulting matrix in order to get an initial estimate. Then, the augmented covariance matrix is built (Equation 3.2) and swept (Equation 3.**3**). The resulting matrix is then swept again for each observed RT, in order to change them from being dependent variables to being independent, or predictor variables, which can then be used to estimate the missing values (Equation 3.4).

$$G = \begin{bmatrix} 1 & {\mu_A}^T \\ \mu_A & \Sigma_A + \mu_A {\mu_A}^T \end{bmatrix} \tag{3.2}$$

Equation 3.2 Augmented covariance matrix G of a given matrix A (that has had its missing values estimated) with μ vector of estimated means and Σ estimated covariance matrix.

$$\begin{aligned} H_{kk} &= -1/G_{kk} \\ H_{jk} &= H_{kj} = G_{jk}/G_{kk} \quad for\ j \neq k \\ H_{ji} &= G_{ji} - G_{jk}G_{ki}/G_{kk} \quad for\ j \neq k, i \neq k \end{aligned} \tag{3.3}$$

Equation 3.3 Sweep operator for a given matrix G, which is swept on a variable *k*, and results in the swept matrix H.

$$\widehat{A_{i,k}} = H_{1,i+1} + A_{j,k}H_{j+1,i+1} \tag{3.4}$$

Equation 3.4 Estimation of missing values in a given matrix A at its i^{th} rows and k^{th} columns, according to its swept augmented covariance matrix H. *i* denotes the rows with missing values in A, *j* denotes the rows without missing values in A, and *k* denotes the columns with missing values in A.

3.1.4 Log objective function

With the RTs of the reference database modeled as a multivariate normal distribution, it is now possible to define the assignment of correct identities to selected peaks as the maximization of the likelihood of those peaks being a part of the multivariate normal distribution of the reference database. Moreover, it is now possible to compute which assignment of identities to experimental data is the most probable. For this matter, the probability density function of the multivariate normal distribution is derived into a log objective function (see Equation 3.5). The value of this log objective function is relative to each sample, and will be maximum when the set of assignments computed have the highest likelihood.

$$P = \frac{1}{2\pi^{0.5n}|\Sigma|^{0.5}} * \exp\left(\frac{-(\mu-x)^T\Sigma^{-1}(\mu-x)}{2}\right) \tag{3.5}$$

$$logF_a = -\frac{n_a}{2}\log(2\pi) - \frac{1}{2}\log(|\Sigma_a|) + \frac{-(\mu_a-x_a)^T\Sigma_a^{-1}(\mu_a-x_a)}{2}$$

Equation 3.5 Log objective function derived from the probability density function of the multivariate normal distribution for a set of assignments *a*, where $logF_a$ is the log objective function value used to score a set of assignments, *n* the number of assignments, $|\Sigma_a|$ the determinant of the covariance submatrix of the reference database, μ_a the vector of reference database RTs, x_a the vector of experimental data RTs, and Σ_a^{-1} the inverse of the covariance submatrix of the reference database, all variables for the set of assignments *a*.

3.1.5 Isotopes, dehydrations, and deglycosylations

As mentioned in section 1.3.3.2, isotopic, dehydrated, and deglycosylated species can appear in experimental spectra, mimicking their homologue's peak's recorded signal in RT. They can be distinguished by their characteristic m/z offset and RT pattern, which can be described using three criteria. The first one defines the m/z offset (criterion 1 in Equation 3.6). If a peak detected is suspected of being an isotope, but the m/z offset is too large, the proposition is rejected. In the same way, if a peak is suspected of being the dehydrated form of another peak, their m/z offset must correspond with the molecular weight of water that would have been lost by the dehydration. The next criterion concerns the RT (criterion 2 in Equation 3.6). As the biophysical properties are mostly conserved or identical, (as an in-source dehydration would cause the recorded RT of a dehydrated species to be from its hydrated form) the RT of both species must be highly similar. Thus, if the RT of a suspected isotopic peak is not less than 0,1% different to its homologue, the proposition is rejected. The last of the criteria ensures the peaks are overlapping in the RT versus intensity plane (criterion 3 in Equation 3.6). Indeed, without their m/z offset, it would not be possible to analyze them separately. Thus, if the peaks are separated in the RT versus intensity plane, they are not considered isotopes, dehydrations, or deglycosylations.

$$criterion\ 1: m_a \begin{cases} isotope\ of\ m_b\ suspected\ if\ |m_a - m_b| \leq 4 \\ dehydration\ of\ m_b\ suspected\ if\ m_b - m_a = 18 \\ deglycosylation\ of\ m_b\ suspected\ if\ m_b - m_a = 162 \end{cases}$$

$$criterion\ 2: 0.999 \leq \frac{r_a}{r_b} \leq 1.001 \qquad (3.6)$$

$$criterion\ 3\text{: } 0.8 > \frac{|r_a - r_b|}{(h_a + h_b)/2}$$

Equation 3.6 The three criteria used to verify if a detected peak *a* is an isotope, dehydration, or deglycosylation of another detected peak *b*. Criterion 1 verifies the m/z offset, where *m* denotes the m/z of the peaks. An isotope is suspected when its mass differs by up to 4 amu, corresponding to a m/z offset of 4. A dehydration is suspected when the mass encompasses the loss of a molecule of water of 18 amu, corresponding to a m/z offset of 18. A deglycosylation is suspected when the mass encompasses the loss of a molecule of glucose of 168 amu, corresponding to a m/z offset of 162. Criterion 2 verifies the RT similarity, where *r* denotes the RT in minutes. Criterion 3 verifies if the peaks are overlapping (true) or not (false) in the RT versus intensity plane, where *r* denotes the RT and *h* the width of the peak at half of its maximal height. The three criteria have to be true in order for a peak to be annotated as a possible isotope, dehydration, or deglycosylation of another peak.

3.1.6 Quantification

As previously mentioned in section 1.3.2.5, the spectral data acquired from HPLC-ESI-MS/MS MRM methods only allows for relative quantification to be computed. This relative computation is based on an internal standard added during sample preparation. Using the following equation, the ion intensity of all detected peaks can be assessed in relation to the ion intensity of this internal standard:

$$pmol_p = \frac{\frac{area_p}{area_s} \times pmol_s}{N} \tag{3.7}$$

Equation 3.7 Relative quantification formula for the abundance of a detected peak *p* in equivalents of the internal standard *s* abundance, where *pmol* denotes the abundance in picomoles, *area* denotes the peak area recorded, and *N* denotes the quantity of biological matrix analysed to normalize the abundance per mg of tissue, per mg of protein, per mL of plasma, or per 1^{e}6 cells. Since the computation is relative, the resulting abundance is in picomoles equivalents of the internal standard used.

3.1.7 Normalization

3.1.7.1 SSRT

As the RT dimension can vary from sample to sample of a same biological matrix, from biological matrix to biological matrix, and from experimental conditions to experimental conditions[50], a means of normalization of RTs was needed. Thus, in order to conserve more spectral information, standardized and scaled RTs (SSRTs) were computed. This data transformation, relying on the presence of exogeneous standards, normalizes the RTs of detected peaks to the samples' exogeneous standards with the following formula:

$$r_{ssrt} = \frac{r - \bar{r}_s}{s_{rs}} \tag{3.8}$$

Equation 3.8 SSRT transformation that can be applied to raw spectral data RTs r to obtain standardized and scaled RTs r_{ssrt}, using the average RT of the exogeneous standards $\bar{r}_s$ and their standard deviation s_{rs}.

3.1.7.2 Loess smoothing

As certain libraries had less exogeneous standards, and thus did not have their standards span their RT spectrum, another mean to transform RTs was tested: loess smoothing. Loess smoothing uses locally weighted linear regression with a second degree polynomial model to smooth data according to set percentage of data. Thus, the data, from the library and from the experimental data, are smoothed based on peaks which identity is unambiguous. These exogeneous or endogenous peaks are unique in their m/z transition within a RT span equivalent of 4 standard deviations of the library data. They are used as anchors points, to which the remaining identities, or peaks, are smoothed with a loess smooth of 0.1 using the MATLAB's Curve Fitting Toolbox smooth function.

3.2 Methods

3.2.1 Library

In order for LITL to annotate and filter targeted lipidomics acquired spectral data, it requires a reference database containing verified lipid identities[33], as mentioned in Chapter 1. This reference database can be augmented with RTs to improve its annotation capabilities[85, 86]. This reference database will now be referred to as a "library".

As seen in Chapter 1, the HPLC-ESI-MS/MS MRM methods developed allows the filtration of ions through mass analyzers as to analyze spectra containing only one class or subclass of lipids with common molecular properties. Thus, the library encompasses multiple "sub" libraries, each corresponding to the mass analyzers set-up used to filter ions. They will now be referred to as the GPC&SM library (MRM method which filters ions based on their product ion having a m/z of 184 in positive ion mode, as to correspond with the phosphocholine component of GPC and sphingomyelins), the GPS library (MRM method which filters ions based on their neutral loss of 185 amu in positive ion mode, as to correspond with the phosphoserine polar head group of GPS), the GPE library (MRM method which filters ions based on their neutral loss of 141 amu in positive ion mode, as to correspond to the phosphoethanolamine polar head group of GPE), and the SPH library (MRM method which filters ions based on their product ion having a m/z of 264 in positive ion mode, as to correspond to the SPH species with d18:1 sphingosine and t18:0 phytosphingosine backbones).

In order to compile such libraries, lipid samples from various biological matrices were analyzed and their results were carefully concatenated. For each dataset added to a library, characterization of detected peaks was carried by a MRM-IDA-EPI method. As an additional mean of verification and in order to ensure that the assumption of multivariate normal distribution was not violated, the normality, kurtosis, and skewness of the concatenated RTs of each characterized peaks was assessed and used to further evaluate compiled results.

3.2.1.1 Datasets and sample collection

For the GPC&SM library, 758 samples from 17 different datasets and across 6 biological matrices (human plasma, human serum, human brain, macrophages, murine plasma, and murine brain) were analyzed and concatenated. For the GPS library, 172 samples from 11 different datasets and across 8 biological matrices (human vascular smooth muscle cells, human fibroblasts, human brain, murine dermal fibroblasts, murine macrophages, and murine brain) were analyzed and concatenated. For the GPE library, 362 samples from 10 different datasets and across 3 biological matrices (human brain, murine plasma, and murine brain) were analyzed and concatenated. For the SPH library, 722 samples from 10 different datasets and across 7 biological matrices (human plasma, human serum, human fibroblasts, human brain, human lymphoblasts, human neurons, and murine brain) were analyzed and concatenated. All samples were compiled from various studies all carried at the Neural Regeneration Laboratory.

Human plasma datasets consisted of samples from a population-based study of circulating GPC and SPH species, totalizing 319 individuals. Consent was obtained in strict accordance with the research ethics committee of the University Medical Centre of Goettingen. Human blood collected for plasma samples was drawn using Lavender Vacutainers tubes (cat. 367863, BD) and plasma was collected after centrifugation at 1500 x g for 10 minutes at 4°C, and stored at -80°C. Datasets of human serum used samples from 136 participants of the Saguenay Youth Study, a cross-sectional study of cardiometabolic and brain health[87]. Consent was obtained in strict accordance with the research ethics committees of the Chicoutimi Hospital (Chicoutimi, Quebec) and the Hospitals for Sick Children (Toronto, Ontario). Human blood collected for serum samples was drawn using silicone-coated gold Vacutainers (cat. 367986, BD), and serum was collected after clotting and separation by centrifugation at 1300 x g for 15 minutes at room temperature, and stored at -80°C. Human brain samples of hippocampus and entorhinal cortex, from 38 individuals, were obtained from the Douglas Hospital Research Centre Brain Bank (Montreal, Quebec, Canada) and the Queen Square Brain Bank (UCL Institute of Neurology, London, UK). Human induced pluripotent stem cell-derived smooth muscle cells and neurons were generated in Dr William Stanford's laboratory (Ottawa Hospital Research Institute, Ottawa, Ontario,

Canada) using modifications of the culture method described in a paper by Kinnear et al.[88]. Human fibroblast cell lines were provided by the Children's Hospital of Eastern Ontario (Ottawa, Ontario, Canada). Lymphoblasts were isolated in Dr David Dyment's laboratory at the Children's Hospital of Eastern Ontario as described in[89].

Datasets containing murine plasma, brain, macrophages, dermal fibroblasts, heart, and skin were obtained with C57BL/6 male and female mice. 112 of these mice were from Tg and NonTg cohorts of backcrossed C57BL/6 x C3H/HeJ TgCRND8 for 5 generations with a C57BL/6 lineage, separated in four cohorts according to their syringe-fed daily diet (48 fed only with normal chow (Harlan Laboratories, 2018 Teklad Global, 18% protein rodent diet), 25 which received an omega-3 free fatty acids 75 mg/kg body weight/day supplement (40/20 EE 1000 mg fish oil capsules, Ocean Nutrition, Nova Scotia), 27 which received an omega-6/omega-9 free fatty acids 75 mg/kg body weight/day supplement (1000 mg corn/soybean oil capsules, Ocean Nutrition), and 12 which received 5 mL/kg body weight/day of the supplement vehicle (25% sweetened condensed milk)). 30 male mice were N3 and N4 C57BL/6 x 129/SV mice. 87 mice were from three cohorts including wild type mice, $hSNCA^{A53T}$; mSNCA null mice, and $Gba1^{D409V/D409V}$;$hSNCA^{A53T}$;mSNCA null mice. Murine heart and skin were sampled from 13 C57BL/6 mice. Murine plasma was collected for 142 mice, while murine brain was collected for 230 mice. Mice were sacrificed by lethal injection of 0.15 mL of euthanyl (65 mg/ml, Bimeda-MTC Animal Health Ins., Cambridge, Ontario). Blood was collected using a 1 mL syringe with needle removed into heparin treated tubes, after incision of the right aorta. Plasma was collected after centrifugation, before being stored in a -80°C environment. For tissue samples, dissected regions included hippocampus and entorhinal cortex, which were concatenated under the matrix label of brain tissue, heart, and skin. Tissues were flash frozen in liquid nitrogen before being stored in a -80°C environment. Murine macrophage samples were collected as described in[90]. All mice procedures were approved by the University of Ottawa Animal Care Committee, and they were carried according to the strict ethical regulations and guidelines for animal experimentation of the Canadian Council on Animal Care.

3.2.1.2 Lipid extraction

All samples were extracted following a modified Bligh and Dyer method[91], previously described in articles by Ryan et al., Whitehead et al. and Xu et al.[92-94], using acid washed glassware and Fisherbrand borosilicate glass Pasteur pipets (Fisher Scientific, Hampton, New Hampshire). For plasma and serum samples, 50 µL (murine source) or 100 µL (human source) were placed in 10 mL Kimble glass threaded tubes (cat. 21020-640, VWR, Radnor, Pennsylvania) containing 3.2 mL of filtered 0.1 M sodium acetate (cat. S-2889, Sigma, St-Louis, Missouri), to which were added 4 mL of 2% acetic acid (cat. A35-500, Fisher Scientific, Ottawa, Ontario) in methanol (cat. BP1105-4, Fisher Scientific). For tissue samples, the tissues were placed in 10 mL Kimble glass threaded tubes containing 4 mL of 2% acetic acid in methanol before being homogenized. 3.2 mL of sodium acetate were added to the samples after addition of internal standards. For cells samples, pelleted cells were re-suspended in 3.2 mL of sodium acetate in 10 mL Kimble glass threaded tubes, to which were added 4 mL of 2% acetic acid in methanol. Cells could also be scraped off after washing their plate with PBS (PBS404.200, Bioshop, Burlington, Ontario) and adding 1 mL of 2% acetic acid in methanol to it (for a total of three 1 mL 2% acetic acid in methanol additions). Internal standards PC(13:0/0:0), PC(12:0/13:0), PS(12:0/13:0) and PE(12:0/13:0) (cat. LM-1600, LM-1000, LM-1300, and LM-1100, Avanti Polar Lipids, Alabaster, Alabama) were added to the GPC&SM, GPS, and GPE samples, and internal standards Cer(d18:1/16:0-d31), GalCer(d18:1/8:0), and GlcCer(d18:1/8:0) (cat. Avanti 868516, Avanti 860528, and Avanti 860540, Avanti Polar Lipids) were added to the SPH samples. For all samples, the organic phase was collected after addition of 3.8 mL of chloroform (cat. C298-500, Fisher Scientific) and centrifugation for 5 minutes at 600 x g and 4°C. The aqueous phase of the samples was back-extracted three times by addition of 2 mL of chloroform, centrifugation, and collection of the organic phase. Then, the chloroform of the collected organic phase was evaporated under nitrogen gas. The resulting dried lipid extracts were re-suspended in 300 µL of anhydrous ethanol (cat. P016EAAN, Commercial Alcohols, Brampton, Ontario) and incubated for 5 minutes in a 30°C water bath, centrifuged 1 minute at 600 x g at 4°C before being stored at -80°C by

50µL aliquots in amber glass vials (cat. C779100AW (vials) and C779200BB (caps), Chromatographic Specialties, Brockville, Ontario) flushed with nitrogen.

3.2.1.3 LC-ESI-MS/MS

Spectral data was acquired via reversed-phase HPLC-ESI-MS/MS on an Agilent 1290 system (Agilent Technologies, Santa Clara, California) coupled to an AB SCIEX QTRAP 5500 mass spectrometer (AB SCIEX, Concord, Ontario). GPC&SM and GPE sample preparations injected in the HPLC system were composed of 13.5 µL of solvent A1 (described below), 5µL of standard mixture (containing 1.25 ng of LPC(18:1-d7/0:0) (cat. Avanti 791643), 1.25 ng of PC(15:0/18:1-d7) (cat. Avanti 791637), 1.25 ng of LPE(18:1-d7/0:0) (cat. Avanti 791644), 1.25 ng of PE(15:0/18:1-d7) (cat. Avanti 701638), 1.25 ng of PS(15:0/18:1-d7) (cat. Avanti 791639), 1.25 ng of SM(d18:1/18:1-d9) (cat. Avanti 791649), 2.5 ng of PC(O-16:0/0:0-d4) (cat. Cayman 360906, Cayman Chemical, Ann Harbor, Michigan), 2.5 ng of PC(O-18:0/0:0-d4) (cat. Cayman 10010228), 1.25 ng of PC(O-16:0/2:0-d4) (cat. Cayman 360900), and 1.25 ng of PC(O-18:0/2:0-d4) (cat. Cayman 10010229)), and 5 µL of sample extract. GPS sample preparations injected in the HPLC system were composed of 13.5 µL of solvent A2 (described below), 5 µL of standard mixture (containing 0.524 µM of PS(17:0/14:1) (cat. Avanti LM-1304), 0.524 µM of PS(17:0/20:4) (cat. Avanti LM-1302), and 0.524 µM of PS(21:0/22:6) (cat. Avanti LM-1303)), and 5 µL of sample extract. SPH sample preparations injected in the HPLC system were composed of 16 µL of solvent A1 (described below), 5 µL of sample extract, and 2.5 µL of anhydrous ethanol. Sample preparations were loaded onto in house packed chromatography columns by an Agilent autosampler. The 100 mm by 250 µm columns were packed with ReproSil-Pur 200 C18 beads of 3 µm and pores of 200 Å (Dr. A. Maisch, Ammerbuch, Germany) for GPC&SM, GPE, and GPS samples, and with ReproSil-Pur 120 C8 beads of 3 µm and pores of 120 Å (Dr. A. Maisch, Ammerbuch, Germany) for SPH samples. The mobile phase consisted of a binary solvent: 0.1% formic acid (cat. 56302, Sigma-Aldrich) and 10 mM ammonium acetate (cat. 2145 OmniPur, Etobicoke, Ontario) in HPLC-graded water (cat. 9831-03, J.T. Baker Avantor, Center Valley, Pennsylvania) (solvent A1) for GPC&SM, GPE,

and SPH samples, or 0.2% formic acid, 10 mM ammonium acetate, and 5 µM HPLC-graded phosphoric acid (cat. 40779, Sigma) in a 5:1:4 solution of isopropanol (cat. A416-4, Fisher Scientific) /methanol/HPLC-graded water (solvent A2) for GPS samples, and 0.1% formic acid and 10 mM ammonium acetate in a 5:2 solution of acetonitrile (cat. 9829-03, J.T. Baker)/isopropanol (solvent B1) for GPC&SM, GPE, and SPH samples, or 0.2% formic acid, 10 mM ammonium acetate, and 5 µM HPLC-graded phosphoric acid in a 1:1 solution of acetonitrile/isopropanol (solvent B2) for GPS samples. For GPC&SM, and GPE, the binary gradient started with 30% solvent B and gradually increased to 100% solvent B within 8 minutes (0-8 minutes). It was maintained at 100% solvent B for 37 minutes (8-45 minutes), and decreased to 30% solvent B within a minute (45-46 minutes), after what it was maintained at 30% solvent B for 14 minutes (46-60 minutes). The flow rate was maintained constant at 10 µL/min (0-60 minutes). The binary gradient for GPS samples started with 20% solvent B, gradually increasing to 50% within 5 minutes (0-5 minutes), then to 80% within 15 minutes (5-20 minutes), then 100% within 1 minute (20-21 minutes). It was maintained at 100% solvent B for 19 minutes (21-40 minutes), before decreasing to 5% solvent B within 1 minute (40-41 minutes). It was then maintained at 5% solvent B for 4 minutes (41-45 minutes), before increasing to 20% solvent B within 1 minute (45-46 minutes), and being maintained at 20% solvent B for 14 minutes (46-60 minutes). The flow rate began at 5 µL/min and was maintained constant for 35 minutes (0-35 minutes). It was then increased to 10 µL/min within 1 minute (35-36 minutes), maintained for 9 minutes (36-45 minutes), and decreased back to 5 µL/min within 1 minute (45-46 minutes) and maintained for 14 minutes (46-60 minutes). The binary gradient for SPH samples started with 30% solvent B and gradually increased to 100% solvent B within 5 minutes (0-5 minutes). It was maintained at 100% solvent B for 30 minutes (5-35 minutes), and decreased to 30% solvent B within a minute (35-36 minutes), after what it was maintained at 30% solvent B for 14 minutes (36-50 minutes). The flow rate was maintained constant at 10 µL/min (0-50 minutes). The MRM methods used are as previously mentioned for each library. The Analyst software (AB SCIEX) was used for instrument control and data acquisition.

3.2.1.4 Data pre-processing

Spectral data were processed via AB SCIEX MultiQuant software, in which signals from detected peaks were assessed and saved in text-based files enumerating peaks properties such as RT, m/z, area, width, height, etc. In the GPC&SM library, 540 peaks were characterized, ranging from m/z 454 to m/z 909, while 178 peaks were characterized in the GPS library, ranging from m/z 428 to m/z 920, 290 peaks in the GPE library, ranging from m/z 452 to m/z 878, and 178 peaks in the SPH library, ranging from m/z 300 to m/z 1193.

3.2.2 Code

LITL was built in MATLAB using the Curve Fitting Toolbox, the Statistics Toolbox. It was compiled as a stand-alone executable file (or application) using the MATLAB Compiler. It is compatible and operational on Windows, Linux, and MAC operating systems, as long as they can install the MATLAB Runtime (available for free at https://www.mathworks.com/products/compiler/matlab-runtime.html). The interface of LITL, shown in Figure 3.9, and explained further in section 3.3.1, was built using the GUIDE user interface design environment built-in MATLAB.

The processes underlying LITL's execution are laid down in Figure 3.10, from user input, covariance estimation, quantification, identity assignment, to data output. LITL has two main modes, as seen in the top right corner of its interface: "Experimental data" and "Library testing". In the "Experimental data" mode, LITL will annotate experimental data using the library provided and according to the user defined settings. In the "Library testing" mode, LITL will split the provided library in half according to a user specified seed, and will assign one half to the other, in order to assess the accuracy of its identity assignments. It is also possible to input a library file into LITL in order to graph the library data onto the RT versus m/z plane. The regression lines formed by species of common structural properties would then help verify the library data does not violate biophysical rules, as mentioned in section 3.1.1.

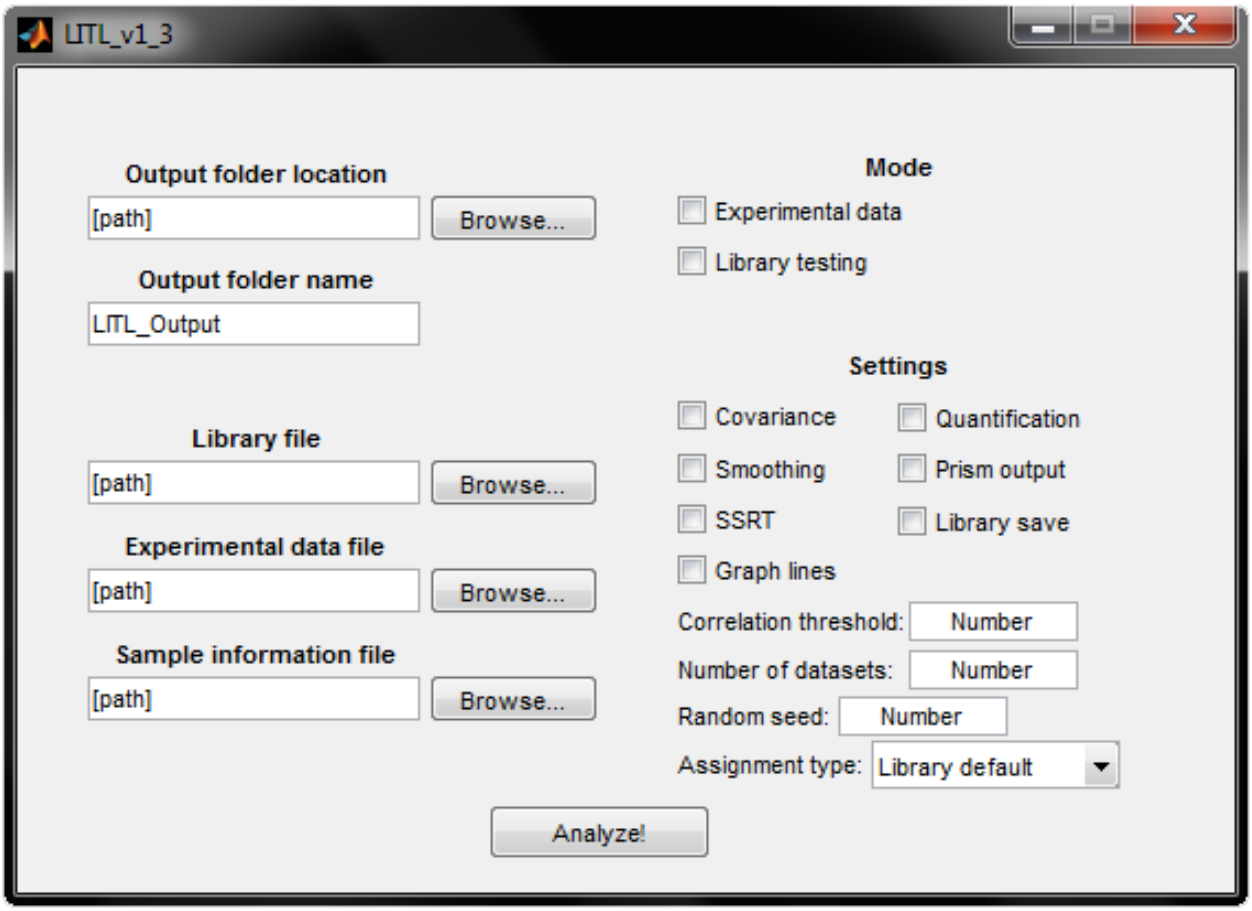

LITL_v1_3
Output folder location
[path]
Browse...
Output folder name
LITL_Output
Library file
[path]
Browse...
Experimental data file
[path]
Browse...
Sample information file
[path]
Browse...
Mode
Experimental data
Library testing
Settings
Covariance
Quantification
Smoothing
Prism output
SSRT
Library save
Graph lines
Correlation threshold:
Number
Number of datasets:
Number
Random seed:
Number
Assignment type:
Library default
Analyze!

Figure 3.9 Main window of the interface of LITL version 1.3 on a Windows operating system.

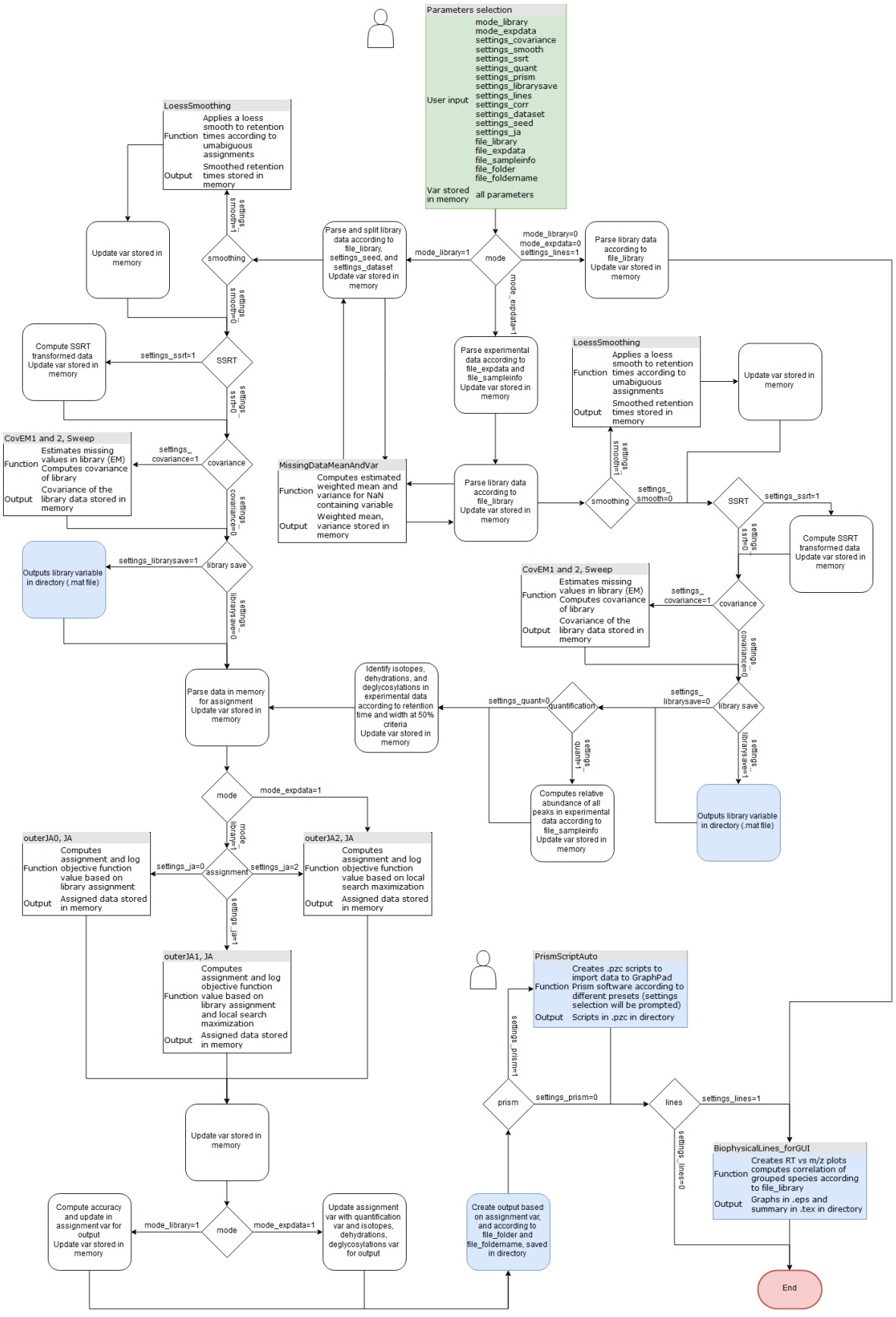

Parameters selection
User input
mode_library
mode_expdata
settings_covariance
settings_smooth
settings_ssrt
settings_quant
settings_prism
settings_librarysave
settings_lines
settings_corr
settings_dataset
settings_seed
settings_ja
file_library
file_expdata
file_sampleinfo
file_folder
file_foldername
Var stored in memory
all parameters
mode
mode_library=1
mode_expdata=1
mode_library=0 mode_expdata=0 settings_lines=1
Parse and split library data according to file_library, settings_seed, and settings_dataset Update var stored in memory
Parse library data according to file_library Update var stored in memory
Parse experimental data according to file_expdata and file_sampleinfo Update var stored in memory
LoessSmoothing
Function Applies a loess smooth to retention times according to umabiguous assignments
Output Smoothed retention times stored in memory
Update var stored in memory
smoothing
settings_smooth=1
settings_smooth=0
Update var stored in memory
Compute SSRT transformed data Update var stored in memory
settings_ssrt=1
SSRT
settings_ssrt=0
CovEM1 and 2, Sweep
Function Estimates missing values in library (EM) Computes covariance of library
Output Covariance of the library data stored in memory
settings_covariance=1
covariance
settings_covariance=0
MissingDataMeanAndVar
Function Computes estimated weighted mean and variance for NaN containing variable
Output Weighted mean, variance stored in memory
Outputs library variable in directory (.mat file)
settings_librarysave=1
library save
settings_librarysave=0
Parse data in memory for assignment Update var stored in memory
Identify isotopes, dehydrations, and deglycosylations in experimental data according to retention time and width at 50% criteria Update var stored in memory
settings_quant=0
quantification
settings_quant=1
Computes relative abundance of all peaks in experimental data according to file_sampleinfo Update var stored in memory
settings_librarysave=0
library save
settings_librarysave=1
Outputs library variable in directory (.mat file)
mode
mode_expdata=1
mode_library=1
outerJA0, JA
Function Computes assignment and log objective function value based on library assignment
Output Assigned data stored in memory
assignment
settings_ja=0
settings_ja=2
outerJA2, JA
Function Computes assignment and log objective function value based on local search maximization
Output Assigned data stored in memory
settings_ja=1
outerJA1, JA
Function Computes assignment and log objective function value based on library assignment and local search maximization
Output Assigned data stored in memory
Update var stored in memory
Compute accuracy and update in assignment var for output Update var stored in memory
mode_library=1
mode
mode_expdata=1
Update assignment var with quantification var and isotopes, dehydrations, deglycosylations var for output
Create output based on assignment var, and according to file_folder and file_foldername, saved in directory
PrismScriptAuto
Function Creates .pzc scripts to import data to GraphPad Prism software according to different presets (settings selection will be prompted)
Output Scripts in .pzc in directory
settings_prism=1
prism
settings_prism=0
lines
settings_lines=1
settings_lines=0
BiophysicalLines_forGUI
Function Creates RT vs m/z plots computes correlation of grouped species according to file_library
Output Graphs in .eps and summary in .tex in directory
End

Figure 3.10 Flowchart of LITL. User interaction is denoted by the "👤" symbol next to the corresponding process, which will pause the program until the user resumes its execution. The process with a green background shows where the program is initialized. Processes with a blue background produce an output in the output directory. The end of the flowchart is denoted by the red "End" oval. Diamond-shaped processes denote an option, which is enabled or not according to the corresponding parameters indicated in text.

In the two main modes ("Experimental data" and "Library testing"), the input data is first parsed and processed according to the user defined data handling settings (SSRT transformation, loess smoothing, covariance estimation, library saving, quantification of experimental data). Then, the peaks to be assigned are separated from the reference peaks in the provided library and organized by m/z. Each is assigned to a library identity, and the set of assignments is evaluated using the log objective function, as shown in section 3.1.4 and in Equation 3.5. The initial set of naive assignments is computed by matching the peaks to their closest library equivalent on the RT dimension. It is also possible to set the initial set of assignments to the library when in "Library testing" mode (set of assignments resulting in 100% accuracy, see section 3.3.1 for details). When a first set of assignments has been evaluated, LITL goes through each m/z and assesses the set of assignments using the log objective function after modifying each of the individual assignments, in an iterative local search pattern, and keeps in memory the set of assignments maximizing the log objective function value. At this time, the set of assignments has to encompass all peaks to be assigned. Indeed, no penalty variable has been implemented to enable a set of assignments to contain an unassigned peak.

The resulting set of assignments is then output in .xlsx format after being augmented with either additional annotations ("Experimental data" mode) or its accuracy ("Library testing" mode). In the "Experimental data" mode, three sheets are created. The first tab presents the assignments made for each sample (columns) and peak identity (rows). The RT of each peak is input in the corresponding cell. The first rows above the sample names present the log objective function values that were obtained for the set of assignments shown. The second sheet is the same as the first one, except for the RTs that are replaced by the relative abundance of each corresponding peak. In the third sheet, the RTs are replaced by annotations denoting if the detected peak is considered a standard, a saturated peak, a possible isotope of another peak, a possible dehydration of another peak, or a possible deglycosylation of another peak. In the "Library testing" mode, only two sheets are created. Similarly to the "Experimental data" mode, results are shown in a table opposing each sample (columns) to each peak identity (rows). The first sheet is the same as for the "Experimental data" mode, except that the 6^{th} row shows the number of detected peaks in

each sample. The second sheet still shows RTs, but each assignment that is not respecting the original assignment of the full library is replaced by the proper identity. The 6th row is replaced by the accuracy of the set of assignments for each sample, which corresponds to the number of correctly assigned peaks over the number of detected peaks.

In addition to the .xlsx output, the regression lines of species sharing structural properties can also be graphed in both modes, although they would only encompass the peaks that were assigned, not the library data, as to verify instead that the set of assignments does not violate biophysical rules, as mentioned in section 3.1.1. In "Experimental data" mode, it is also possible to create scripts to transfer the output data directly to the GraphPad Prism statistical software (GraphPad Software, San Diego, California).

3.3 Results

3.3.1 Interface

As previously mentioned in section 3.2.2, LITL's interface was built using the GUIDE design environment in MATLAB. Its main window is shown in Figure 3.9. During data processing a dialog box will indicate the progress of the analysis to the user. In the case of additional user prompts, dialog boxes will guide the user through (when creating scripts to transfer output data directly to the GraphPad Prism statistical software).

The main window can be divided into four sections, top-left, top-right, bottom-left, and bottom-right. Centered at the bottom, there is the "Analyze!" button, to be pressed when all fields and settings correspond to the user needs. The top-left section allows the specification of output options: where LITL will create the output folder, and how the output folder will be named. The top-right section allows the choice of the mode of analysis to execute: "Experimental data" or "Library testing". The bottom-left section lets the user input their data: the library in .mat or .xlsx format, the experimental data in .csv format, and the sample information in .csv format. Templates for these three files are available with the

installation package. The bottom-right section allows the specification of all analysis settings. The "Covariance" setting enables the estimation of the covariance matrix of the library provided. Leaving it unchecked will prompt LITL to replace the covariance matrix with a proper size identity matrix. The "Smoothing" and "SSRT" settings correspond to the data transformations mentioned in sections 3.1.7.1 and 3.1.7.2. The "Graph lines" setting corresponds to the creation of regression lines in the RT versus m/z plane as mentioned in sections 3.1.1 and 3.2.2. The "Quantification" setting is to be enabled to compute the relative abundance of peaks in experimental data. The "Prism output" setting allows the creation of scripts to export data directly to the GraphPad Prism statistical software. The "Library save" setting saves the input library after covariance estimation and before assignment (see library save decision point in Figure 3.9). The "Correlation threshold" setting refers to the correlation of the regression lines when the "Graph lines" setting is enabled. Regression lines with a correlation below that threshold will be shown in red instead of black. The "Number of datasets" setting is used to parse the library file when in "Library testing" mode. The "Random seed" setting specifies the seed determining the split of the data in "Library testing" mode. The "Assignment type" setting contains three options only available in "Library testing" mode: "Library default", "Library optimized", and "Optimized". "Library default" will skip the local search iteration during the assignment process and output the initial set of assignments corresponding to a 100% accuracy. "Library optimized" will proceed to the local search process with its initial set of assignments being the set of assignments resulting in 100% accuracy. "Optimized" will proceed to the local search process with its initial set of assignments computed by matching the peaks to their closest library equivalent on the RT dimension, as mentioned previously in section 3.2.2. The assignment type in "Experimental data" mode corresponds to the "Optimized" option.

Choosing one mode will remove the settings proper to the other mode only. Thus, when "Experimental data" mode is checked, the following fields disappear, as they are not used: "Number of datasets" setting, "Random seed" setting, and "Assignment type" setting. Similarly, when "Library testing" mode is checked, the following fields disappear: "Quantification" setting, "Prism output" setting, experimental data input field, and sample information input field.

3.3.2 Library testing method

In order to test the log objective function and its maximization by the implemented local search iterative process, the "Library testing" mode was used on the GPC&SM library of 758 samples counting 540 unique identified peaks. It was split in half by LITL: one half to be assigned to the library formed by the other half. The output assignments were evaluated as correct or not by referring back to the full GPC&SM library. The set of assignments was evaluated with and without data transformation and with and without the covariance estimation. In the case of no covariance, the covariance term in the log objective function would be replaced by an identity matrix, so that the covariance between peaks is ignored. This process was repeated with 10 different random seeds, as to obtain 10 replicates. To test the accuracy of each different analysis, as to assess if the covariance and data transformations are enhancing the accuracy of the assignments, the mean accuracy of each analysis across the 10 replicates was analyzed using a repeated measures ANOVA (Figure 3.11). The mean accuracy of each analysis not using the estimated covariance was significantly higher than their mean accuracy when using the estimated covariance. Additionally, transforming the data with a loess smoothing allowed LITL to perform significantly better than while using non-transformed data. Furthermore, to test the consistency of the accuracy of the assignments within replicates, the accuracy of each different analysis was assessed within each replicate with a repeated measures ANOVA test (Figure 3.12). For each analysis, the usage of the estimated covariance results in a larger spread of the accuracies, while not using the covariance term led to more consistent results.

For each analysis, regression lines of species sharing common structural properties were also graphed in the RT versus m/z plane (Figure 3.13). This was done to verify the accuracy of the assignments, as per the principle according to which these regression lines are parallel in orientation[80]. Therefore, a regression line breaking the exponential patterns of Figure 3.13 would hint at a misidentified peak. Indeed, the non perfect accuracy scores shown in Figure 3.11 and Figure 3.12 are verified by the

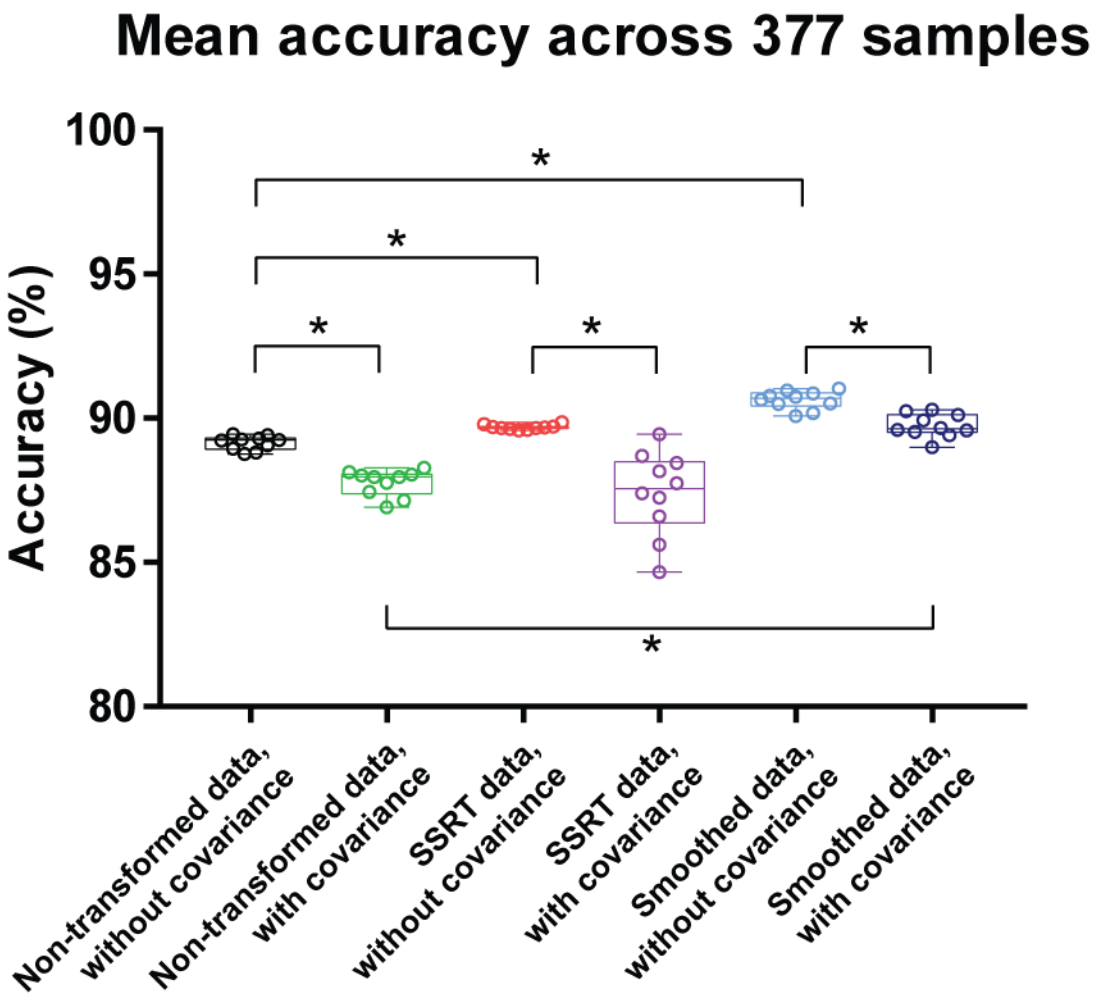

Mean accuracy across 377 samples
Accuracy (%)
100
95
90
85
80
*
*
*
*
*
*
*
Non-transformed data, without covariance
Non-transformed data, with covariance
SSRT data, without covariance
SSRT data, with covariance
Smoothed data, without covariance
Smoothed data, with covariance

Figure 3.11 Comparison of average accuracy between covariance usage and data transformations in the "Library testing" mode. The mean accuracy for each type of assignment corresponds to the percentage of correct assignments in a set of assignments, averaged across 377 samples. Each type of assignment was repeated to totalize 10 replicates of different random seed. They were analyzed with a one-way repeated-measures ANOVA (p-value < 0.05) and a Tukey's post-hoc where every type of assignment was compared to every other type of assignment. Only the comparisons between "without covariance" and "with covariance" within a type of assignment, as well as across types of assignments within "without covariance" or "with covariance" with a p-value < 0.05 are indicated by an asterisk (*).

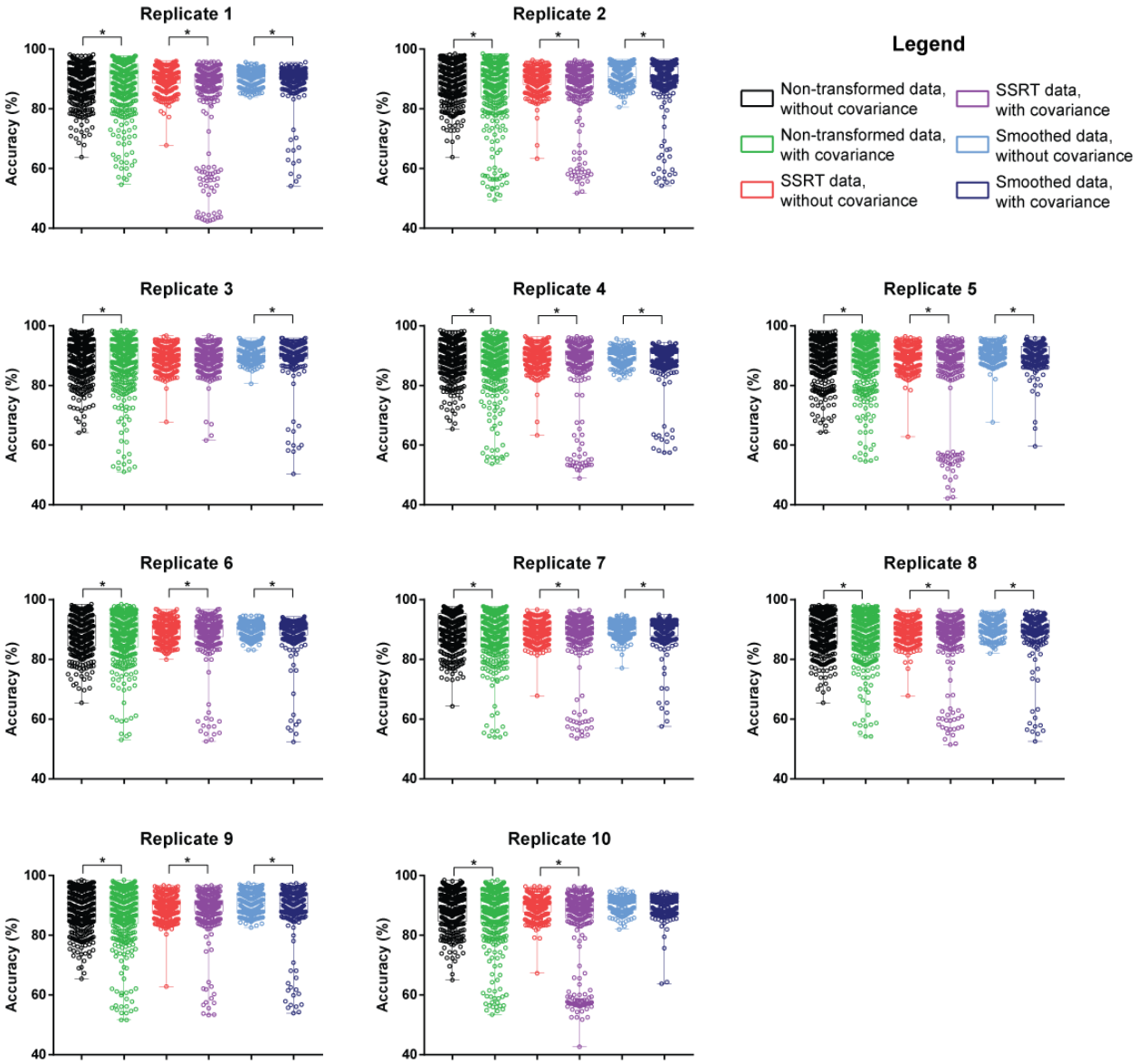

Replicate 1
Replicate 2
Legend
Non-transformed data, without covariance
Non-transformed data, with covariance
SSRT data, without covariance
SSRT data, with covariance
Smoothed data, without covariance
Smoothed data, with covariance
Replicate 3
Replicate 4
Replicate 5
Replicate 6
Replicate 7
Replicate 8
Replicate 9
Replicate 10
Accuracy (%)
100
80
60
40
*

Figure 3.12 Comparison of accuracy between covariance usage and data transformations in the "Library testing" mode. The accuracy for each type of assignment corresponds to the percentage of correct assignments in a set of assignments. Each type of assignment was repeated to totalize 10 replicates of different random seed, each replicate having 377 samples. They were analyzed with a one-way repeated-measures ANOVA (p-value < 0.05) and a Tukey's post-hoc where every type of assignment was compared to every other type of assignment. Only the comparisons between "without covariance" and "with covariance" within a type of assignment with a p-value < 0.05 are indicated by an asterisk (*).The accuracy per sample did not correlate with the number of detected peaks comprised in a sample's joint assignment to the library (not shown).

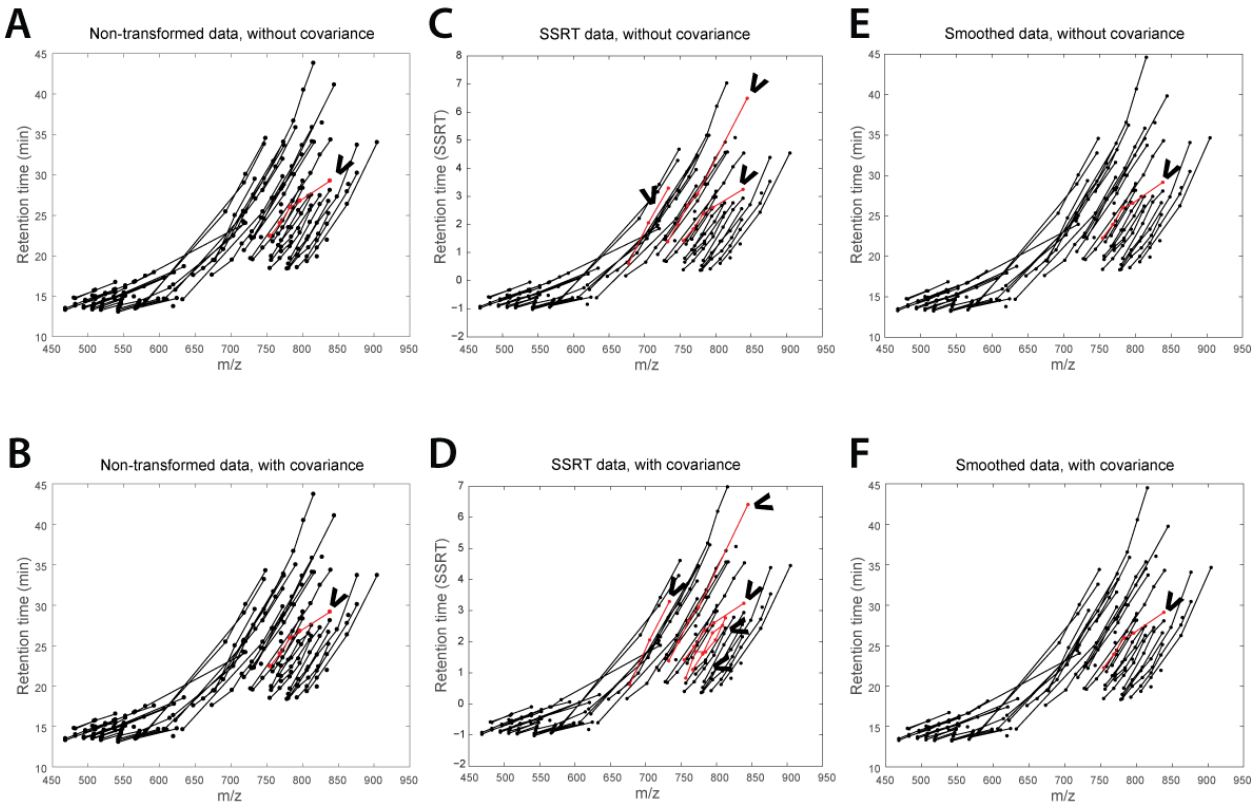

A
Non-transformed data, without covariance
Retention time (min)
m/z
C
SSRT data, without covariance
Retention time (SSRT)
m/z
E
Smoothed data, without covariance
Retention time (min)
m/z
B
Non-transformed data, with covariance
Retention time (min)
m/z
D
SSRT data, with covariance
Retention time (SSRT)
m/z
F
Smoothed data, with covariance
Retention time (min)
m/z

Figure 3.13 Regression lines grouping species identified as sharing common structural properties in the retention time versus m/z plane. The average retention times of the 377 samples of the same replicate but under different set of assignment parameters are shown. Regression lines with a coefficient of correlation below 0.98 are shown in red, and indicated with an arrow. The correlation of the log retention times is calculated. The data in (A) and (B) are non transformed, the data in (C) and (D) are transformed with the SSRT transformation described in Equation 3.8, and the data in (E) and (F) are smoothed with a loess function. The data in (A), (C), and (E) were assigned without the use of the covariance term, and the data in (B), (D), and (F) were assigned with the use of the covariance term.

presence of regression lines which correlation coefficient for their log retention times is below 0.98, corresponding to a violation of the biophysical rules mentioned in section 3.1.1.

3.3.3 Experimental data method

In order to test the annotation functions of LITL, one dataset of 12 samples of the GPC&SM library was used as experimental data. As to test the ability of LITL to identify isotopes in the data, the dataset used was its early version which was kept uncorrected, and thus contained several peak selection mistakes. As previously mentioned, all data in the experimental data is forced to be assigned, as penalties for unassignment have yet to be implemented. Therefore, no testing was done in the "Experimental data" mode regarding the accuracy of the assignments.

3.3.3.1 Quantification

With the "Quantification" setting enabled, all peaks detected in the experimental data were quantified using the data provided, as per the process shown in Figure 3.10, using Equation 3.7. The relative quantification, in picomoles equivalents of the internal quantification standard specified in the sample information file, appeared in the output sheet for quantification. All data matched the peak assignments of the first sheet (see section 3.2.2).

3.3.3.2 Isotope annotation

The dataset tested, as previously mentioned, contained user mistakes in regards to peak selection, spread across the m/z dimension, resulting in isotopes being selected instead of the intended species present in the library. As "Experimental data" mode was used, LITL automatically tested the data for possible isotopes, as per the process shown in Figure 3.10. The annotations were compiled into Table 3.3. The criterion 2 of Equation 3.6 comparing the difference in RT of two peaks successfully recognized the isotopes detected throughout the spectrum, as their RT offset was within its thresholds. The criterion 3

Sample	m_a	m_b	m/z criterion	r_a	r_b	Offset in seconds	Retention time criterion	h_a	h_b	Separation criterion	Conclusion
1	548.5	548.5	Passed	14.52069	14.50064	0.02005	Failed	0.09628649	0.09730280	Passed	-
2				14.52662	14.56090	0.03428	Failed	0.07860540	0.09651158	Passed	-
3				14.54559	14.56834	0.02275	Failed	0.08913632	0.10307039	Passed	-
4				14.57828	14.56337	0.01491	Failed	0.09984588	0.09871399	Passed	-
5				14.54429	14.54235	0.00194	Passed	0.12008610	0.10152234	Passed	Duplicated data
6				14.54708	14.54881	0.00173	Passed	0.11300243	0.11176227	Passed	Duplicated data
7				14.53440	14.53796	0.00356	Passed	0.10509639	0.10443733	Passed	Duplicated data
8				14.56272	14.56049	0.00223	Passed	0.09686896	0.11826887	Passed	Duplicated data
9				14.53804	14.53415	0.00389	Passed	0.10141231	0.10303164	Passed	Duplicated data
10				14.43505	14.43849	0.00344	Passed	0.12937515	0.12880826	Passed	Duplicated data
11				14.52826	14.53043	0.00217	Passed	0.11240303	0.11031757	Passed	Duplicated data
12				14.51926	14.51395	0.00531	Passed	0.10188634	0.09885886	Passed	Duplicated data
1	580.5	580.5	Passed	14.56948	14.55998	0.00950	Passed	0.12036787	0.12258178	Passed	Duplicated data
2				14.39132	14.39530	0.00398	Passed	0.11272791	0.14445522	Passed	Duplicated data
3				14.42052	14.59232	0.17180	Failed	0.14692246	0.12683094	Failed	-
4				14.42726	14.42064	0.00662	Passed	0.15220774	0.15302640	Passed	Duplicated data
5				14.41382	14.58637	0.17255	Failed	0.16899384	0.12524618	Failed	-
6				14.42057	14.41845	0.00212	Passed	0.16318445	0.16884099	Passed	Duplicated data
7				14.58469	14.58262	0.00207	Passed	0.11446525	0.13217607	Passed	Duplicated data
8				14.44021	14.43527	0.00494	Passed	0.16989859	0.17581826	Passed	Duplicated data
9				14.57985	14.58326	0.00341	Passed	0.14071263	0.12321864	Passed	Duplicated data
10				14.63750	14.63437	0.00313	Passed	0.13170953	0.11280023	Passed	Duplicated data
11				14.56213	14.56239	0.00026	Passed	0.16628415	0.18762503	Passed	Duplicated data
12				14.54763	14.54470	0.00293	Passed	0.17998173	0.18080466	Passed	Duplicated data
1	580.5	584.5	Passed	14.55998	14.56087	0.00089	Passed	0.12258178	0.08440381	Passed	Isotope
2				14.39530	14.59102	0.19572	Failed	0.14445522	0.07461512	Failed	Not an isotope
3				14.59232	14.57684	0.01548	Failed	0.12683094	0.08687658	Passed	Not an isotope
4				14.42064	14.41808	0.00256	Passed	0.15302640	0.10084895	Passed	Isotope
5				14.58637	14.57757	0.00880	Passed	0.12524618	0.09222664	Passed	Isotope
6				14.41845	14.59290	0.17445	Failed	0.16884099	0.08615098	Failed	Not an isotope
7				14.58262	14.57952	0.00310	Passed	0.13217607	0.09255696	Passed	Isotope
8				14.43527	14.40732	0.02795	Failed	0.17581826	0.08406488	Passed	Not an isotope

Sample	m_a	m_b	m/z criterion	r_a	r_b	Offset in seconds	Retention time criterion	h_a	h_b	Separation criterion	Conclusion
9	580.5	584.5	Passed	14.58326	14.57870	0.00456	Passed	0.12321864	0.08682700	Passed	Isotope
10				14.63437	14.66859	0.03422	Failed	0.11280023	0.12480837	Passed	Not an isotope
11				14.56239	14.57248	0.01009	Passed	0.18762503	0.09616385	Passed	Isotope
12				14.54470	14.56079	0.01609	Failed	0.18080466	0.08813234	Passed	Not an isotope
1	594.5	594.5	Passed	14.63482	14.62133	0.01349	Passed	0.12253562	0.13195033	Passed	Duplicated data
2				14.66567	14.66681	0.00114	Passed	0.13057142	0.09664340	Passed	Duplicated data
3				14.65376	14.66214	0.00838	Passed	0.12575694	0.11715623	Passed	Duplicated data
4				14.67439	14.67559	0.00120	Passed	0.10481142	0.11672081	Passed	Duplicated data
5				14.64691	14.64964	0.00273	Passed	0.12585781	0.11966120	Passed	Duplicated data
6				14.65705	14.65754	0.00049	Passed	0.11456014	0.11825297	Passed	Duplicated data
7				14.65197	14.65215	0.00018	Passed	0.11931997	0.11483573	Passed	Duplicated data
8				14.67476	14.67416	0.00060	Passed	0.13291117	0.11715400	Passed	Duplicated data
9				14.64514	14.64589	0.00075	Passed	0.12736443	0.13466633	Passed	Duplicated data
10				14.40071	14.40138	0.00067	Passed	0.13031992	0.12461453	Passed	Duplicated data
11				14.64063	14.63964	0.00099	Passed	0.12603079	0.12182091	Passed	Duplicated data
12				14.62934	14.63057	0.00123	Passed	0.12339561	0.11741215	Passed	Duplicated data
1	738.5	739.5	Passed	22.33506	22.33223	0.00283	Passed	0.57464383	0.55399800	Passed	Isotope
2				22.06300	22.08328	0.02028	Passed	0.56903107	0.59194625	Passed	Isotope
3				22.02464	22.03442	0.00978	Passed	0.52410320	0.53834170	Passed	Isotope
4				22.11864	22.12282	0.00418	Passed	0.53987746	0.55776854	Passed	Isotope
5				22.06718	22.26893	0.20175	Failed	0.60120786	0.96119098	Passed	Not an isotope
6				22.12649	22.31626	0.18977	Failed	0.60317925	0.93933752	Passed	Not an isotope
7				22.23745	22.40193	0.16448	Failed	0.61898752	1.08527619	Passed	Not an isotope
8				22.24971	22.38596	0.13625	Failed	0.61540762	1.04873197	Passed	Not an isotope
9				22.38416	22.67764	0.29348	Failed	0.61551546	1.36068041	Passed	Not an isotope
10				20.35294	20.45525	0.10231	Failed	0.41337195	0.56417518	Passed	Not an isotope
11				22.79039	23.08294	0.29255	Failed	0.72027657	0.91183887	Passed	Not an isotope
12				22.75557	23.04847	0.29290	Failed	0.68614117	0.90383365	Passed	Not an isotope
1	816.5	816.5	Passed	33.31458	33.31458	0	Passed	0.04811216	0.04811216	Passed	Duplicated data
2				33.47630	33.47630	0	Passed	0.08290166	0.08290166	Passed	Duplicated data
3				33.71427	33.71427	0	Passed	0.42519238	0.42519238	Passed	Duplicated data
4				33.87132	33.87132	0	Passed	0.38290644	0.38290644	Passed	Duplicated data

Sample	m_a	m_b	m/z criterion	r_a	r_b	Offset in seconds	Retention time criterion	h_a	h_b	Separation criterion	Conclusion
5	816.5	816.5	Passed	33.72541	33.72541	0	Passed	0.38676038	0.38676038	Passed	Duplicated data
6				33.75742	33.75742	0	Passed	0.70753834	0.70753834	Passed	Duplicated data
7				33.52524	33.52524	0	Passed	0.40965957	0.40965957	Passed	Duplicated data
8				33.59433	33.59433	0	Passed	0.11586667	0.11586667	Passed	Duplicated data
9				33.61082	33.61082	0	Passed	0.31610685	0.31610685	Passed	Duplicated data
10				33.65504	33.65504	0	Passed	0.25670208	0.25670208	Passed	Duplicated data
11				33.14372	33.14372	0	Passed	0.07375661	0.07375661	Passed	Duplicated data
12				33.41640	33.41640	0	Passed	0.06521079	0.06521079	Passed	Duplicated data
1	819.5	820.5	Passed	25.92509	25.91193	0.01316	Passed	0.26206135	0.31469453	Passed	Isotope
2				25.77316	25.77107	0.00209	Passed	0.32916457	0.30916357	Passed	Isotope
3				25.65188	25.62858	0.02330	Passed	0.31217415	0.41017446	Passed	Isotope
4				25.80857	25.79584	0.01273	Passed	0.28333518	0.37820158	Passed	Isotope
5				25.58939	25.57348	0.01591	Passed	0.31315321	0.37287976	Passed	Isotope
6				25.69690	25.67825	0.01865	Passed	0.28759355	0.34890982	Passed	Isotope
7				25.60233	25.57705	0.02528	Passed	0.32590167	0.34863031	Passed	Isotope
8				25.68557	25.67956	0.00601	Passed	0.31446437	0.35138694	Passed	Isotope
9				25.56813	25.55766	0.01047	Passed	0.39557070	0.44556804	Passed	Isotope
10				23.19500	23.18489	0.01011	Passed	0.60242176	0.65847582	Passed	Isotope
11				25.69417	25.69320	0.00097	Passed	0.47007518	0.56697229	Passed	Isotope
12				25.69480	25.67812	0.01668	Passed	0.44425819	0.57543863	Passed	Isotope

Table 3.3 Annotation of duplicated data and isotopes using the criteria of Equation 3.6. Three criteria are used to verify whether a detected peak *a* is an isotope of another detected peak *b* or not. Criterion 1 verifies the m/z *m* offset of the two peaks. Criterion 2 verifies the similarity between the retention times *r*. Criterion 3 verifies the separation of the peaks in the retention time versus ion intensity plane using their width at half maximal height *h*.

confirmed the presence of isotopes by testing their separation on the RT vs ion intensity plane. Moreover, by using the criteria of Equation 3.6, LITL was also able to filter through isobars, annotating species that were duplicated, or too close in the RT vs ion intensity plane to be considered as separate peaks.

3.3.3.3 Creation of script for statistical software

Taking into account the nature of the data output by LITL, the statistical tests that would most commonly be carried were made available with the current release. The three options available are able to format the output data for t-tests, one-way ANOVA, and two-way ANOVA, as to be recognized by GraphPad Prism, analyzed, and graphed, as per the template used. Templates for each test are available in the installation folder. Their customization is encouraged, as to select the proper analysis settings (correction factor, multiple comparison, number of subcolumns for two-way ANOVA, appearance of the graphs, etc.). The .pzc script will be automatically split into multiple parts as per the capabilities of the GraphPad Prism scripting functions. The script will also be split according to the number of GraphPad Prism files required, as one of its files can only carry information for 250 different tests. Thus, to import the data into the software, the .pzc scripts simply have to be executed in order.

3.4 Discussion

LITL was built in order to address the need for annotating and filtering targeted lipidomics acquired spectral data after their exportation to simple text-based format. Thus, it was built to annotate detected ions using their RT and m/z, based on a reference database, on their sample information, and on various criteria. The main features, that is, assignment of identity, annotation of isotopes, quantification of detected ions, and output to statistical software, are discussed below.

3.4.1 Assignments

It was hypothesized that taking the ensemble of observed peaks into account when assigning identities, as hinted in Daly et al., 2014[79], would improve the ability of the defined model to correctly assign identities to observed peaks. Thus, the covariance of the RTs in the library was factored into the scoring system of LITL (Equation 3.5). However, when tested, the set of assignments obtained when taking that term out of the equation were more accurate than the set of assignments obtained when using the covariance (Figure 3.11), whatever the data transformation applied. As the set of assignments without the covariance factored in are significantly more accurate, the hypothesis that covariance improves the assignment feature had to be rejected. Thus, the RT covariance is not sufficient to assign identities using the m/z and RT dimensions. These results might be explained by the low variance of the library data, and thus the tested data. As the covariance was thought to improve assignments by ignoring the variance of single RTs in favor of the whole spectrum of RTs, the low variance of single RTs in the library in the first place might counter-balance the benefits of the covariance. This could be further explored by collecting and processing more data from different sources and time points. It is also possible that the covariance estimation is not optimized because of a too few number of iterations of the EM algorithm. Although, the number of iterations of the EM algorithm was set to only allow for an average difference of 0.15 seconds between iterations on testing data. This could be explored by reassigning the test replicates with different libraries varying in the number of iterations set for their covariance matrix estimation.

Data transformation was also hypothesized to improve assignments, as it would lessen the effects of sample to sample variations, experimental conditions and methods variations, and so on[50]. Transforming data by converting RTs to SSRTs (Equation 3.8) allowed for a better accuracy than non-transformed data (Figure 3.11). Additionally, the parallel relationship between series in the RT versus m/z plane that can be observed when clustering RTs[80] seem to be conserved (Figure 3.13), even if less than the non-transformed or smoothed data. However, the SSRTs require a few internal standards in order to properly represent the span of RTs, thus their use is not optimal. Nevertheless, transforming the data by loess smoothing improved the accuracy of assignments compared to the non-transformed data as well

(Figure 3.11). This enhancement in accuracy did not disrupt the patterns observable in the RT versus m/z plane either (Figure 3.13). Therefore, data transformation may indeed improve identity assignments, by notably smoothing ambiguous data points to unambiguous data points, standards or not, as their span is more representative of a spectrum's data.

3.4.2 Isotopes

Annotation of isotopes was carried according to three criteria, representing the characteristic spectral patterns of isotopes in terms of m/z, RT, and width at half maximal height. Those three criteria (Equation 3.6) were tested on an early uncorrected dataset of 12 samples. The resulting annotations of 14 species, detecting 22 isotopes and 42 duplicated peaks, are shown in Table 3.3. The characteristic m/z offset used as criterion 1 distinguished isotopes from isobars, and allowed to filter through the experimental data. The dynamic RT offset used as criterion 2 was able to correctly identify isotopes and duplicated peaks from separate species. Those annotations were then confirmed by criterion 3, and the user mistakes were properly identified as isotopes and duplicated peaks.

As the RTs in minutes used for the criterion are interval variables, the threshold ratio between r_a and r_b will change in terms of seconds depending on the RTs themselves, since a 0.999 ratio at low RTs will correspond to a lower gap than at high RTs. This dynamic component increases the threshold by 0.06 second per minute of RT of a species. For example, to be within the isotope range according to criterion 2, a peak *a* has to be within a certain range from a peak *b*. This range can be derived from Equation 3.6, so that the range in seconds is equal to the RT of the peak *b* removed from its quotient with 0.999, times 60. Thus, this range varies from 1.20 seconds to 1.80 seconds for peaks having a RT from 20 minutes to 30 minutes.

Ergo, the isotope recognition by m/z, RT, and width at half maximal height can be implemented without using the theoretical distribution of isotopes like LDA[57]. This feature is fully functional without user tuning like in Mzmine2[61], as Equation 3.6 uses constants. Moreover, LITL applies the same criteria to isobars, enabling the detection of duplicated peaks.

3.4.3 Quantification

Quantification of peaks was implemented successfully, as each experimental detected peak had enough information to be relatively quantified using Equation 3.6. As the equation takes into account an internal quantification standard, the molar equivalents to this standard can be determined. However, this does not take into consideration the dynamic range of detection of the analytes, which limits the quantification process, and as stated in Chapter 1, this was not meant to be incorporated for the purpose of this thesis. Thus, the ratio of the largest to smallest detectable signal will be entered into the equation in order to determine whether or not a detected peak can be quantified before being quantified after submission, through the library data.

3.4.4 Output to statistical software

LITL was also built to easily transfer analyzed processed data to the GraphPad Prism statistical software, using its scripting language. This simple export feature is novel for HPLC-MS data analysis software, as they tend to not have any statistical analysis features[60, 63], offer mostly graphical representation[58], or encourage the manual export of the data for further statistical testing[61, 64]. Although, XCMS online has been implemented with statistical analyses[95], they are built in the online interface, therefore they are not available offline. Moreover, they do not offer the two-way ANOVA analysis GraphPad Prism does, which is an important statistical test in biological studies as it can assess whether or not there is an interaction between two independent variables on a dependent variable. The feature to export data directly to GraphPad Prism is thus enhancing the pipeline established in the Neural Regeneration Laboratory.

Chapter 4: General discussion

4.1 Summary of the programs developed

A challenge in the lipidomics field is the necessity of bioinformatic tools, notably for the analysis of mass spectrometry data from HPLC-ESI-MS/MS MRM methods. As emphasized in boldface in Figure 1.3, peak identification and peak annotation and filtering were the main focuses of this thesis. Peak identification was the challenge tackled by the development of LIT (Chapter 2), while peak annotation and filtering were addressed by the development of LITL (Chapter 3).

The search engine LIT was created as an offline stand-alone Python program, compatible with Windows and Mac operating systems. It takes advantage of the combinatorial structure of lipids (Figure 1.1 and Figure 1.2) to decompose lipids into variables defining Equation 2.1. This equation, at the core of LIT, provides a robust *in silico* database, formed by manipulating each variable, as shown in Figure 2.7. Thus, LIT allows the prediction of lipid structures, by computing the possible lipids identities with a given m/z and by computing the m/z of a given lipid structure.

The annotation tool LITL was created as an offline MATLAB program, compatible with any operating system able to support MATLAB or its MCR runtime application. LITL focuses on enabling reproducible annotation of detected peaks between and across matrices of pre-processed HPLC-ESI-MS/MS MRM data, as to address the bioinformatics gap shown in Figure 1.6 by the dotted box. To this matter, it takes spectral data acquired by a targeted MRM method in simple text format, and exploits their RT and m/z dimensions to annotate detected peaks, comparing them to a library of previously annotated spectra, counting more than 2,000 samples across 4 lipid families. The RTs of each spectrum are modeled after a multivariate normal distribution, enabling LITL to find the best set of assignments observed peak/lipid identity by using a local search algorithm to maximize Equation 3.4, which is derived from the probability density function of a multivariate normal distribution. This approach allows LITL to take into account all observed peaks instead of analyzing them one by one, as Equation 3.4 encompasses the covariance of the library, which is estimated for its missing values using a sweep operator on its

augmented form (Equation 3.2 , Equation 3.3, and Equation 3.4). While its accuracy is not improved by taking the covariance into the equation, as Figure 3.11, Figure 3.12, and Figure 3.13 show, LITL still manages to annotate peaks only using their RT and m/z dimensions. Additional spectral information are conserved by normalizing data, notably by smoothing the RT dimension with a loess smoothing function, as the accuracy of the set of assignments regarding smoothed data was significantly higher than the one of non-transformed data (Figure 3.11). Moreover, annotation of detected peaks is accompanied by the annotation of isotopes. By using the criteria listed in Equation 3.6, isotopic peaks can be identified, as well as duplicated isobars, as shown in Table 3.3. Furthermore, LITL enhances the analysis pipeline in the Neural Regeneration Laboratory, by quantifying each detected peak using Equation 3.7 and by providing a mean to export analyzed data directly to the GraphPad Prism statistical software according to user-defined templates, as depicted through its processes shown in Figure 3.10.

4.2 Innovation and importance

LIT has been developed to predict lipid structures in order to enhance the lipidomics pipeline at the level of peak identification, as previously mentioned. In this way, it fills the gap for predicting modified lipid species, as they are lacking from several online databases. Indeed, as laid down in Table 2.**2**, LIT includes oxidized structures for all applicable subclasses it encompasses, while only a few are present in the database of LIPID MAPS, which has the most extensive database in the search engines compared. In terms of versatility in ion adducts, LIT is consistent with what LIPID MAPS offers, which are 20 different adducts and the neutral mass. This tops ALEX, VaLID, and LipidHome, as ALEX encompasses 22 adducts, yet they are not available for each subclass, as VaLID encompasses 5 adducts and the neutral mass, and as LipidHome encompasses solely the neutral mass. While not spanning all lipid classes, LIT counts 58 of them, amongst which 16 are absent from LIPID MAPS, 18 from ALEX, 37 from VaLID, and 43 from LipidHome. Furthermore, LIT is innovative in the sense that its database is not stored and browsed by user searches like its rivals, but rather computed according to user preferences.

LITL has been developed to address a lack of bioinformatics tools for HPLC-ESI-MS/MS MRM data independently of data format, as shown in Figure 1.6. To this matter, LITL innovated by carrying peak annotation through two main dimensions, RT and m/z, through peak lists in text-based format, unlike most other software which demands the MS spectra file. Unfortunately, its accuracy was not up to par, as previously mentioned, and thus comparison with other software, if they could process the same spectral data, would be rather meaningless. In terms of detection of isotopes, LITL demonstrated isotope identification capabilities by using spectral patterns of species in m/z, RT, and width at half maximal height alone, without using theoretical isotope distribution nor user tuning, like LDA[57] and MzMine2[61]. In addition, LITL used the same criteria for isotope detection (Equation 3.6) to identify isobars which, as they could not be qualified as separated peaks, were considered duplicated peaks, making LITL an asset for data analysis. As for its quantification feature, it is up to date with the literature[32], as far as its dynamic range component, which will be implemented in the future. Moreover, the ability of LITL to export its processed data to the GraphPad Prism statistical software via the software scripting language is quite advantageous. Of all the software reviewed, only a few carry their own statistical analyses, while most will require the user to manually export and treat data after output.

4.3 Going forward

Going forward, LIT will keep being utilized as a stand-alone application, as it fulfilled its goal to predict lipid identities. Since its method of computation (Equation 2.1 and Figure 2.7) takes advantage of the combinatorial structure of lipids, it can be used for more lipid classes than what it currently spans. Thus, it will likely be updated with more lipid subclasses, like glycerophosphates, in order to span more of the lipidome, and allow enhancement of the Neural Regeneration Laboratory pipeline for studies encompassing lipids not in its database. An underlying database can also be built or referred to in LIT, as to document lipid species seen in biological samples.

On the other hand, LITL will need improvements before achieving its goal to annotate and filter peaks. Indeed, the hypothesis according to which the covariance of the RTs of its library would improve its assignment feature was rejected. However, this does not mean its underlying principle that taking the ensemble of observed peaks into account[79] is proven false. As mentioned previously, the Equation 3.5 used to maximize the likelihood of a set of assignments does not encompass the possibility of an observed peak not being assigned at all. Therefore, the addition of a penalty term in order to allow observed data to not be forced onto the library data may enhance the local search algorithm for the assignment feature of LITL. In addition, the dynamic range component of quantification will need to be implemented, as mentioned back in Chapter 1, after the submission of this thesis, as to not quantify species outside the dynamic range of the quantification standards. Nevertheless, LITL is on to a good start to tackle the challenges of peak annotation and filtering for HPLC-ESI-MS/MS MRM data, and its ineluctable updates will surely lead to an enhanced lipidomics pipeline.

References

1. Fahy, E., Subramaniam, S., Brown, H. A., Glass, C. K., Merrill, A. H., Murphy, R. C., Raetz, C. R. H., Russell, D. W., Seyama, Y., Shaw, W., Shimizu, T., Spener, F., van Meer, G., VanNieuwenhze, M. S., White, S. H., Witztum, J. L., and Dennis, E. A. (2005) A comprehensive classification system for lipids, *Journal of Lipid Research 46*, 839-862.
2. Shevchenko, A., and Simons, K. (2010) Lipidomics: coming to grips with lipid diversity, *Nature Reviews Molecular Cell Biology 11*, 593.
3. Singer, S. J., and Nicolson, G. L. (1972) The fluid mosaic model of the structure of cell membranes, *Science 175*, 720-731.
4. Wenk, M. R. (2005) The emerging field of lipidomics, *Nature Reviews Drug Discovery 4*, 594.
5. English, D., Cui, Y., and Siddiqui, R. A. (1996) Messenger functions of phosphatidic acid, *Chemistry and Physics of Lipids 80*, 117-132.
6. Levy, B. D., and Serhan, C. N. (2002) Polyisoprenyl phosphates: natural antiinflammatory lipid signals, *Cellular and Molecular Life Sciences CMLS 59*, 729-741.
7. Goñi, F. M. (2014) The basic structure and dynamics of cell membranes: An update of the Singer–Nicolson model, *Biochimica et Biophysica Acta (BBA) - Biomembranes 1838*, 1467-1476.
8. Hunte, C., and Richers, S. (2008) Lipids and membrane protein structures, *Current Opinion in Structural Biology 18*, 406-411.
9. Ostrom Rennolds, S., and Insel Paul, A. (2009) The evolving role of lipid rafts and caveolae in G protein-coupled receptor signaling: implications for molecular pharmacology, *British Journal of Pharmacology 143*, 235-245.
10. Phillips, R., Ursell, T., Wiggins, P., and Sens, P. (2009) Emerging roles for lipids in shaping membrane-protein function, *Nature 459*, 379-385.

11. Uhlenbrock, K., Gassenhuber, H., and Kostenis, E. (2002) Sphingosine 1-phosphate is a ligand of the human gpr3, gpr6 and gpr12 family of constitutively active G protein-coupled receptors, *Cellular Signalling 14*, 941-953.

12. Genheden, S., Essex, J. W., and Lee, A. G. (2017) G protein coupled receptor interactions with cholesterol deep in the membrane, *Biochimica et Biophysica Acta (BBA) - Biomembranes 1859*, 268-281.

13. Bell, R. M., Exton, J. H., and Prescott, S. M. (2013) Ceramide: A Novel Second Messenger and Lipid Mediator, In *Lipid Second Messengers*, pp 177-204, Springer US.

14. Ghosh, S., Strum, J., and Bell, R. M. (1997) Lipid biochemistry: functions of glycerolipids and sphingolipids in cellular signaling, *The FASEB Journal 11*, 45-50.

15. Smyth, M. J., Obeid, L. M., and Hannunf, Y. A. (1997) Ceramide: A Novel Lipid Mediator of Apoptosis, In *Advances in Pharmacology* (Kaufmann, S. H., Ed.), pp 133-154, Academic Press.

16. van Blitterswijk, W. J., van der Luit, A. H., Veldman, R. J., Verheij, M., and Borst, J. (2003) Ceramide: second messenger or modulator of membrane structure and dynamics?, *Biochemical Journal 369*, 199-211.

17. Spector, A. A., and Yorek, M. A. (1985) Membrane lipid composition and cellular function, *Journal of Lipid Research 26*, 1015-1035.

18. van Meer, G. (2011) Dynamic Transbilayer Lipid Asymmetry, *Cold Spring Harbor Perspectives in Biology*, a004671.

19. McMahon, H. T., and Gallop, J. L. (2005) Membrane curvature and mechanisms of dynamic cell membrane remodelling, *Nature 438*, 590.

20. Fahy, E., Subramaniam, S., Murphy, R. C., Nishijima, M., Raetz, C. R. H., Shimizu, T., Spener, F., van Meer, G., Wakelam, M. J. O., and Dennis, E. A. (2009) Update of the LIPID MAPS comprehensive classification system for lipids, *Journal of Lipid Research 50*, S9-S14.

21. Pruett, S. T., Bushnev, A., Hagedorn, K., Adiga, M., Haynes, C. A., Sullards, M. C., Liotta, D. C., and Merrill, A. H. (2008) Thematic Review Series: Sphingolipids. Biodiversity of sphingoid bases ("sphingosines") and related amino alcohols, *Journal of Lipid Research 49*, 1621-1639.

22. Spener, F., Lagarde, M., Géloên, A., and Record, M. (2003) Editorial: What is lipidomics?, *European Journal of Lipid Science and Technology 105*, 481-482.

23. Christie, W. W., and Han, X. (2012) Chapter 13 - Introduction to mass spectrometric analysis of lipids in lipidomics, In *Lipid Analysis (Fourth edition)*, pp 277-303, Woodhead Publishing.

24. Han, X. (2016) Lipids and Lipidomics, In *Lipidomics: Comprehensive Mass Spectrometry of Lipids*, pp 3-20, Wiley.

25. Han, X., and Gross, R. W. (2005) Shotgun lipidomics: electrospray ionization mass spectrometric analysis and quantitation of cellular lipidomes directly from crude extracts of biological samples, *Mass Spectrom Rev 24*, 367-412.

26. Bou Khalil, M., Hou, W., Zhou, H., Elisma, F., Swayne, L. A., Blanchard, A. P., Yao, Z., Bennett, S. A. L., and Figeys, D. (2010) Lipidomics era: accomplishments and challenges, *Mass Spectrom Rev 29*, 877-929.

27. Lam, S. M., and Shui, G. (2013) Lipidomics as a Principal Tool for Advancing Biomedical Research, *Journal of Genetics and Genomics 40*, 375-390.

28. Yang, K., and Han, X. (2016) Lipidomics: Techniques, Applications, and Outcomes Related to Biomedical Sciences, *Trends in Biochemical Sciences 41*, 954-969.

29. Sethi, S., and Brietzke, E. (2017) Recent advances in lipidomics: Analytical and clinical perspectives, *Prostaglandins & Other Lipid Mediators 128-129*, 8-16.

30. Christie, W. W., and Han, X. (2012) Chapter 3 - Lipid extraction, storage and sample handling, In *Lipid Analysis (Fourth edition)*, pp 55-66, Woodhead Publishing.

31. Wakelam, M. J. O., Pettitt, T. R., and Postle, A. D. (2007) Lipidomic Analysis of Signaling Pathways, In *Methods in Enzymology*, pp 233-246, Academic Press.

32. Han, X. (2016) Sample Preparation, In *Lipidomics: Comprehensive Mass Spectrometry of Lipids*, pp 283-304.

33. Han, X. (2016) Mass Spectrometry-Based Lipidomics Approaches, In *Lipidomics: Comprehensive Mass Spectrometry of Lipids*, pp 53-88.

34. Christie, W. W., and Han, X. (2012) Chapter 2 - Chromatographic analysis of lipids: general principles, In *Lipid Analysis (Fourth edition)*, pp 21-54, Woodhead Publishing.

35. Gritti, F., and Guiochon, G. (2013) Perspectives on the Evolution of the Column Efficiency in Liquid Chromatography, *Analytical Chemistry 85*, 3017-3035.

36. Roberts, L. D., McCombie, G., Titman, C. M., and Griffin, J. L. (2008) A matter of fat: An introduction to lipidomic profiling methods, *Journal of Chromatography B 871*, 174-181.

37. Clarke, W. (2017) Chapter 1 - Mass spectrometry in the clinical laboratory: determining the need and avoiding pitfalls, In *Mass Spectrometry for the Clinical Laboratory*, pp 1-15, Academic Press, San Diego.

38. Zaikin, V., and Halket, J. M. (2009) Introduction, In *A Handbook of Derivatives for Mass Spectrometry*, pp xvii-xxx, IM Publications.

39. Han, X. (2016) Mass Spectrometry for Lipidomics, In *Lipidomics: Comprehensive Mass Spectrometry of Lipids*, pp 21-52.

40. Han, X. (2016) Factors Affecting Accurate Quantification of Lipids, In *Lipidomics: Comprehensive Mass Spectrometry of Lipids*, pp 335-354.

41. Hu, Q., Noll, R. J., Li, H., Makarov, A., Hardman, M., and Graham Cooks, R. (2005) The Orbitrap: a new mass spectrometer, *J Mass Spectrom 40*, 430-443.

42. Schwudke, D., Schuhmann, K., Herzog, R., Bornstein, S. R., and Shevchenko, A. (2011) Shotgun Lipidomics on High Resolution Mass Spectrometers, *Cold Spring Harbor Perspectives in Biology*, a004614.

43. Hsu, F.-F., and Turk, J. (2003) Electrospray ionization/tandem quadrupole mass spectrometric studies on phosphatidylcholines: The fragmentation processes, *Journal of the American Society for Mass Spectrometry 14*, 352-363.

44. Hsu, F. F., and Turk, J. (2000) Characterization of phosphatidylethanolamine as a lithiated adduct by triple quadrupole tandem mass spectrometry with electrospray ionization, *Journal of Mass Spectrometry 35*, 595-606.

45. Han, X. (2016) Fragmentation Patterns of Glycerophospholipids, In *Lipidomics: Comprehensive Mass Spectrometry of Lipids*, pp 173-200.

46. Jarvis, M. (2011) Simultaneous Targeted and Unknown Screening using the 3200 QTRAP® LC/MS/MS System and Cliquid® Software, *AB SCIEX*, 1-9.

47. Gao, H., Materne, O. L., Howe, D. L., and Brummel, C. L. (2007) Method for rapid metabolite profiling of drug candidates in fresh hepatocytes using liquid chromatography coupled with a hybrid quadrupole linear ion trap, *Rapid Communications in Mass Spectrometry 21*, 3683-3693.

48. Huang, J., Bathena, S. P. R., and Alnouti, Y. (2010) Metabolite Profiling of Praziquantel and its Analogs During the Analysis of in vitro Metabolic Stability Using Information-Dependent Acquisition on a Hybrid Triple Quadrupole Linear Ion Trap Mass Spectrometer, *Drug Metabolism and Pharmacokinetics 25*, 487-499.

49. Lee, J.-Y., Lee, S. Y., Lee, K., Oh, S. J., and Kim, S. K. (2015) Determination of species-difference in microsomal metabolism of amitriptyline using a predictive MRM–IDA–EPI method, *Chemico-Biological Interactions 229*, 109-118.

50. Smith, R., Ventura, D., and Prince, J. T. (2015) LC-MS alignment in theory and practice: a comprehensive algorithmic review, *Briefings in Bioinformatics 16*, 104-117.

51. Husen, P., Tarasov, K., Katafiasz, M., Sokol, E., Vogt, J., Baumgart, J., Nitsch, R., Ekroos, K., and Ejsing, C. S. (2013) Analysis of Lipid Experiments (ALEX): A Software Framework for Analysis of High-Resolution Shotgun Lipidomics Data, *PLOS ONE 8*, e79736.

52. Foster, J. M., Moreno, P., Fabregat, A., Hermjakob, H., Steinbeck, C., Apweiler, R., Wakelam, M. J. O., and Vizcaíno, J. A. (2013) LipidHome: A Database of Theoretical Lipids Optimized for High Throughput Mass Spectrometry Lipidomics, *PLOS ONE 8*, e61951.

53. Blanchard, A. P., McDowell, G. S. V., Valenzuela, N., Xu, H., Gelbard, S., Bertrand, M., Slater, G. W., Figeys, D., Fai, S., and Bennett, S. A. L. (2013) Visualization and Phospholipid Identification (VaLID): online integrated search engine capable of identifying and visualizing glycerophospholipids with given mass, *Bioinformatics 29*, 284-285.

54. Haimi, P., Uphoff, A., Hermansson, M., and Somerharju, P. (2006) Software Tools for Analysis of Mass Spectrometric Lipidome Data, *Analytical Chemistry 78*, 8324-8331.

55. Herzog, R., Schuhmann, K., Schwudke, D., Sampaio, J. L., Bornstein, S. R., Schroeder, M., and Shevchenko, A. (2012) LipidXplorer: A Software for Consensual Cross-Platform Lipidomics, *PLOS ONE 7*, e29851.

56. Hübner, G., Crone, C., and Lindner, B. (2009) LipID - A software tool for automated assignment of lipids inmass spectra, *Journal of Mass Spectrometry 44*, 1676-1683.

57. Hartler, J., Triebl, A., Ziegl, A., Trötzmüller, M., Rechberger, G. N., Zeleznik, O. A., Zierler, K. A., Torta, F., Cazenave-Gassiot, A., Wenk, M. R., Fauland, A., Wheelock, C. E., Armando, A. M., Quehenberger, O., Zhang, Q., Wakelam, M. J. O., Haemmerle, G., Spener, F., Köfeler, H. C., and Thallinger, G. G. (2017) Deciphering lipid structures based on platform-independent decision rules, *Nature Methods 14*, 1171.

58. Hartler, J., Trötzmüller, M., Chitraju, C., Spener, F., Köfeler, H. C., and Thallinger, G. G. (2011) Lipid Data Analyzer: unattended identification and quantitation of lipids in LC-MS data, *Bioinformatics 27*, 572-577.

59. Kind, T., Liu, K.-H., Lee, D. Y., DeFelice, B., Meissen, J. K., and Fiehn, O. (2013) LipidBlast in silico tandem mass spectrometry database for lipid identification, *Nature Methods 10*, 755.

60. Kyle, J. E., Crowell, K. L., Casey, C. P., Fujimoto, G. M., Kim, S., Dautel, S. E., Smith, R. D., Payne, S. H., and Metz, T. O. (2017) LIQUID: an-open source software for identifying lipids in LC-MS/MS-based lipidomics data, *Bioinformatics 33*, 1744-1746.

61. Pluskal, T., Castillo, S., Villar-Briones, A., and Orešič, M. (2010) MZmine 2: Modular framework for processing, visualizing, and analyzing mass spectrometry-based molecular profile data, *BMC Bioinformatics 11*, 395.

62. Tautenhahn, R., Patti, G. J., Rinehart, D., and Siuzdak, G. (2012) XCMS Online: a web-based platform to process untargeted metabolomic data, *Analytical chemistry 84*, 5035-5039.

63. Ni, Z., Angelidou, G., Hoffmann, R., and Fedorova, M. (2017) LPPtiger software for lipidome-specific prediction and identification of oxidized phospholipids from LC-MS datasets, *Scientific Reports 7*, 15138.

64. Peng, B., and Ahrends, R. (2016) Adaptation of Skyline for Targeted Lipidomics, *Journal of Proteome Research 15*, 291-301.

65. Brugger, B., Erben, G., Sandhoff, R., Wieland, F. T., and Lehmann, W. D. (1997) Quantitative Analysis of Biological Membrane Lipids at the Low Picomole Level by Nano-Electrospray Ionization Tandem Mass Spectrometry, *Proceedings of the National Academy of Sciences of the United States of America 94*, 2339-2344.

66. Gu, M., Kerwin, J. L., Watts, J. D., and Aebersold, R. (1997) Ceramide Profiling of Complex Lipid Mixtures by Electrospray Ionization Mass Spectrometry, *Analytical Biochemistry 244*, 347-356.

67. Singh, A., and Del Poeta, M. (2016) Sphingolipidomics: An Important Mechanistic Tool for Studying Fungal Pathogens, *Frontiers in Microbiology 7*, 501.

68. Kayganich, K. A., and Murphy, R. C. (1992) Fast atom bombardment tandem mass spectrometric identification of diacyl, alkylacyl, and alk-1-enylacyl molecular species of glycerophosphoethanolamine in human polymorphonuclear leukocytes, *Analytical Chemistry 64*, 2965-2971.

69. Sud, M., Fahy, E., Cotter, D., Brown, A., Dennis, E. A., Glass, C. K., Merrill, J. A. H., Murphy, R. C., Raetz, C. R. H., Russell, D. W., and Subramaniam, S. (2007) LMSD: LIPID MAPS structure database, *Nucleic Acids Research 35*, D527-D532.

70. McDowell, G. S., Blanchard, A. P., Taylor, G. P., Figeys, D., Fai, S., and Bennett, S. A. (2014) Predicting glycerophosphoinositol identities in lipidomic datasets using VaLID (Visualization and Phospholipid Identification)--an online bioinformatic search engine, *BioMed research international*.

71. Han, X. (2016) Bioinformatics in Lipidomics, In *Lipidomics: Comprehensive Mass Spectrometry of Lipids*, pp 121-150.

72. Christie, W. W. (2012) Front matter A2 - Christie, William W, In *Lipid Analysis (Fourth edition)* (Han, X., Ed.), pp i-iii, Woodhead Publishing.

73. Sud, M., Fahy, E., Cotter, D., Azam, K., Vadivelu, I., Burant, C., Edison, A., Fiehn, O., Higashi, R., Nair, K. S., Sumner, S., and Subramaniam, S. (2016) Metabolomics Workbench: An international repository for metabolomics data and metadata, metabolite standards, protocols, tutorials and training, and analysis tools, *Nucleic Acids Research 44*, D463-D470.

74. Holman, J. D., Tabb, D. L., and Mallick, P. (2014) Employing ProteoWizard to Convert Raw Mass Spectrometry Data, *Current protocols in bioinformatics / editoral board, Andreas D. Baxevanis ... [et al.] 46*, 13.24.11-13.24.19.

75. Mapstone, M., Cheema, A. K., Fiandaca, M. S., Zhong, X., Mhyre, T. R., MacArthur, L. H., Hall, W. J., Fisher, S. G., Peterson, D. R., Haley, J. M., Nazar, M. D., Rich, S. A., Berlau, D. J., Peltz, C. B., Tan, M. T., Kawas, C. H., and Federoff, H. J. (2014) Plasma phospholipids identify antecedent memory impairment in older adults, *Nat Med 20*, 415-418.

76. Casanova, R., Varma, S., Simpson, B., Kim, M., An, Y., Saldana, S., Riveros, C., Moscato, P., Griswold, M., Sonntag, D., Wahrheit, J., Klavins, K., Jonsson, P. V., Eiriksdottir, G., Aspelund, T., Launer, L. J., Gudnason, V., Legido Quigley, C., and Thambisetty, M. (2016) Blood

metabolite markers of preclinical Alzheimer's disease in two longitudinally followed cohorts of older individuals, *Alzheimer's & Dementia 12*, 815-822.

77. Pauling, J. K., Hermansson, M., Hartler, J., Christiansen, K., Gallego, S. F., Peng, B., Ahrends, R., and Ejsing, C. S. (2017) Proposal for a common nomenclature for fragment ions in mass spectra of lipids, *PLOS ONE 12*, e0188394.

78. Broeckling, C. D., Ganna, A., Layer, M., Brown, K., Sutton, B., Ingelsson, E., Peers, G., and Prenni, J. E. (2016) Enabling efficient and confident annotation of LC− MS metabolomics data through MS1 spectrum and time prediction, *Analytical chemistry 88*, 9226-9234.

79. Daly, R., Rogers, S., Wandy, J., Jankevics, A., Burgess, K. E., and Breitling, R. (2014) MetAssign: probabilistic annotation of metabolites from LC–MS data using a Bayesian clustering approach, *Bioinformatics 30*, 2764-2771.

80. Loos, M., and Singer, H. (2017) Nontargeted homologue series extraction from hyphenated high resolution mass spectrometry data, *Journal of cheminformatics 9*, 12.

81. Little, R. J. A., and Rubin, D. B. (2014) Factored Likelihood Methods, Ignoring the Missing-Data Mechanism, *Statistical Analysis with Missing Data*, 133-163.

82. Little, R. J. A., and Rubin, D. B. (2014) Multivariate Normal Examples, Ignoring the Missing-Data Mechanism, *Statistical Analysis with Missing Data*, 221-252.

83. Dempster, A. P. (1969) *Elements of continuous multivariate analysis*, Addison-Wesley Pub. Co., 388 pp.

84. Little, R. J. A., and Rubin, D. B. (2014) Single Imputation Methods, *Statistical Analysis with Missing Data*, 59-74.

85. Aicheler, F., Li, J., Hoene, M., Lehmann, R., Xu, G., and Kohlbacher, O. (2015) Retention time prediction improves identification in nontargeted lipidomics approaches, *Analytical chemistry 87*, 7698-7704.

86. Stanstrup, J., Neumann, S., and Vrhovšek, U. k. (2015) PredRet: prediction of retention time by direct mapping between multiple chromatographic systems, *Analytical chemistry 87*, 9421-9428.

87. Paus, T., Pausova, Z., Abrahamowicz, M., Gaudet, D., Leonard, G., Pike, G. B., and Richer, L. (2015) Saguenay Youth Study: A multi-generational approach to studying virtual trajectories of the brain and cardio-metabolic health, *Developmental cognitive neuroscience 11*, 129-144.

88. Kinnear, C., Chang, W. Y., Khattak, S., Hinek, A., Thompson, T., de Carvalho Rodrigues, D., Kennedy, K., Mahmut, N., Pasceri, P., and Stanford, W. L. (2013) Modeling and rescue of the vascular phenotype of Williams-Beuren syndrome in patient induced pluripotent stem cells, *Stem cells translational medicine 2*, 2-15.

89. Wagner, J. D., Huang, L., Tetreault, M., Majewski, J., Boycott, K. M., Bulman, D. E., Dyment, D. A., and McMillan, H. J. (2015) Autosomal recessive axonal polyneuropathy in a sibling pair due to a novel homozygous mutation in IGHMBP2, *Neuromuscular Disorders 25*, 794-799.

90. Snider, S. A., Margison, K. D., Ghorbani, P., LeBlond, N. D., O'Dwyer, C., Nunes, J. R., Nguyen, T., Xu, H., Bennett, S. A., and Fullerton, M. D. (2018) Choline transport links macrophage phospholipid metabolism and inflammation, *Journal of Biological Chemistry*, jbc. RA118. 003180.

91. Bligh, E. G., and Dyer, W. J. (1959) A rapid method of total lipid extraction and purification, *Canadian journal of biochemistry and physiology 37*, 911-917.

92. Ryan, S. D., Whitehead, S. N., Swayne, L. A., Moffat, T. C., Hou, W., Ethier, M., Bourgeois, A. J. G., Rashidian, J., Blanchard, A. P., Fraser, P. E., Park, D. S., Figeys, D., and Bennett, S. A. L. (2009) Amyloid-β42 signals tau hyperphosphorylation and compromises neuronal viability by disrupting alkylacylglycerophosphocholine metabolism, *Proceedings of the National Academy of Sciences 106*, 20936.

93. Whitehead, S. N., Hou, W., Ethier, M., Smith, J. C., Bourgeois, A., Denis, R., Bennett, S. A. L., and Figeys, D. (2007) Identification and Quantitation of Changes in the Platelet Activating Factor Family of Glycerophospholipids over the Course of Neuronal Differentiation by High-Performance Liquid Chromatography Electrospray Ionization Tandem Mass Spectrometry, *Analytical Chemistry 79*, 8539-8548.

94. Xu, H., Valenzuela, N., Fai, S., Figeys, D., and Bennett, S. A. L. (2013) Targeted lipidomics – advances in profiling lysophosphocholine and platelet-activating factor second messengers, *The FEBS Journal 280*, 5652-5667.

95. Gowda, H., Ivanisevic, J., Johnson, C. H., Kurczy, M. E., Benton, H. P., Rinehart, D., Nguyen, T., Ray, J., Kuehl, J., Arevalo, B., Westenskow, P. D., Wang, J., Arkin, A. P., Deutschbauer, A. M., Patti, G. J., and Siuzdak, G. (2014) Interactive XCMS Online: Simplifying Advanced Metabolomic Data Processing and Subsequent Statistical Analyses, *Analytical Chemistry 86*, 6931-6939.

Appendix A: LITL v1.3 User Guide

In order to help the reader, the user guide, which includes input file formatting and step-by-step processes, has been attached to this thesis.

A1.1 General

LITL was developed to analyze pre-processed HPLC-ESI-MS/MS spectral data. It analyzes peak lists by m/z, retention time, and area (peak intensity). The results are exported in a .xlsx file and can contain a folder of graphs and a series of prism scripts to forward the data to the GraphPad Prism software. The stand-alone application was tested on both Windows and Mac operating systems. As LITL was developed in the MATLAB environment, it requires the MATLAB runtime, available for free at https://www.mathworks.com/products/compiler/matlab-runtime.html.
Please contact jfial097@uottawa.ca for troubleshooting.

A1.2 Modes and settings

A1.2.1 Library testing mode

In this mode, a library is cut in half and tested against itself. The library file has to contain an index column in its fourth sheet (see below in Library file section).

Required fields: Output folder location, Output folder name, Library file path

Available settings: Covariance, Smoothing, SSRT, Library save, Graph lines, Correlation threshold, Number of datasets, Random seed, Assignment type.

How-to (see Step-by-step section for an example):

1. Input where the output folder should be created using the **Output folder location** field (use the **Browse...**) in the upper left
2. Type in the name of the output folder using the **Output folder name** field in the upper left (LITL_Output by default)
3. Check the **Library testing** mode in the upper right
4. Input the library file (.xlsx) to test using the **Library file** field (use the **Browse...** button) on the left
5. Select the settings on the right:
    a. **Covariance**: when checked, the covariance will be computed and used in the log objective function
    b. **Smoothing**: when checked, the data entered will be smoothed by a loess function using the unambiguous assignments
    c. **SSRT**: when checked, the data will be standardized and scaled to its identified standards
    d. **Library save**: when checked, the library file will be exported as a .mat to the output folder
    e. **Graph lines**: when checked, the library retention times will be graphed according to their general ID
    f. **Correlation threshold**: when specified, LITL will use this as the correlation threshold when graphing (0.98 by default)
    g. **Number of datasets**: when specified, LITL will take the dataset averages and consider them as samples during the assignments (0 by default)
    h. **Random seed**: enter any number to define the library split
    i. **Assignment type**: select "**Library default**" to evaluate the log objective function value of the library assignment. Select "**Library optimized**" to start the log objective function value optimization with the library assignment. Select "**Optimized**" to start the log objective function value optimization with an estimated assignment (retention times within standard deviation span of library times)
6. Press the **Analyze!** button and wait for results (progress bar will inform you of analysis progression)

A1.2.2 Experimental data mode

In this mode, an experimental data file, described by a sample information file, is assigned to a library. The library file must not contain an index column in its fourth sheet (see below in Library file section).

Required fields: Output folder location, Output folder name, Library file path, Experimental data file path, Sample information file path

Available settings: Covariance, Smoothing, SSRT, Quantification, Prism output, Library save, Graph lines, Correlation threshold.

How-to (see Step-by-step section for an example):

1. Input where the output folder should be created using the **Output folder location** field (use the **Browse...**) in the upper left
2. Type in the name of the output folder using the **Output folder name** field in the upper left (LITL_Output by default)
3. Check the **Experimental data** mode in the upper right
4. Input the library file (.xlsx or .mat) to use using the **Library file** field (use the **Browse...** button) on the left
5. Input the experimental data file (.csv) to analyze using the **Experimental data file** field (use the **Browse...** button) on the left
6. Input the sample information file (.csv) of the corresponding experimental data using the **Sample information file** field (use the **Browse...** button) on the left
7. Select your settings on the right:
    a. **Covariance**: when checked, the covariance will be computed and used in the log objective function
    b. **Smoothing**: when checked, the data entered will be smoothed by a loess function using the unambiguous assignments
    c. **SSRT**: when checked, the data will be standardized and scaled to its identified standards
    d. **Quantification**: when checked, the data will be quantified according to the information provided in the sample information file and the peak area in the experimental data file
    e. **Prism output**: when checked, prompts for producing a prism script for statistical analysis (t-test, one-way and two-way anova) will appear at the end of the analysis. The quantification setting has to be on for the prompt to appear
    f. **Library save**: when checked, the library file will be exported as a .mat to the output folder
    g. **Graph lines**: when checked, the assigned experimental data will be graphed according to the general ID of its assigned barcodes
    h. **Correlation threshold**: when specified, LITL will use this as the correlation threshold when graphing (0.98 by default)
8. Press the **Analyze!** button and wait for results (progress bar will inform you of analysis progression)

A1.2.3 Creating graphs of the library

To access this mode, both checkboxes for library testing and experimental data modes must be unchecked. The data within a library file is graphed. The library file must be formatted like it would be for the Experimental data mode.

Required fields: Output folder location, Output folder name, Library file path, Graph lines, Correlation threshold

Available settings: Graph lines, Correlation threshold

How-to (see Step-by-step section for an example):

1. Input where the output folder should be created using the **Output folder location** field (use the **Browse...**) in the upper left
2. Type in the name of the output folder using the **Output folder name** field in the upper left (LITL_Output by default)

3. Leave the **Experimental data** and **Library** testing modes unchecked in the upper right
4. Input the library file (.xlsx or .mat) to use using the **Library file** field (use the **Browse...** button) on the left
5. Select your settings on the right:
    a. **Graph lines**: when checked, the assigned experimental data will be graphed according to the general ID of its assigned barcodes
    b. **Correlation threshold**: when specified, LITL will use this as the correlation threshold when graphing (0.98 by default)
6. Press the **Analyze!** button and wait for results (progress bar will inform you of analysis progression)

A1.3 LITL inputs

A1.3.1 Library file

In the following section, pay attention to the terms in **bold**, as they are important for data importation.

The **SECOND** and **FOURTH** sheets are imported by LITL.

Second sheet

Row 1 contains column headers. LITL does not use the headers, so typos in column headers will not cause any errors.

Rows 2+ contain information for all individual species (1 per row). **Make sure there are no additional rows after the data ends** (the cells underneath the last row of data should be clear) as this may cause LITL to end unexpectedly.

The following columns are required by LITL:
Column 10 (J) contains the **GeneralID** (string) of all species in the library.
C**olumn 11 (K)** contains the **family** (string) of all species in the library. Standard must be spelt "Standard", even if it contains other strings. For example, a PC standard can be noted "Standard-PC" as long as "Standard" is spelt correctly.
Column 12 (L) contains the **barcodes** (string) of all species in the library.
C**olumn 14 (N)** contains the **MasterID**, or full identifier (string) of all species in the library.
C**olumn 19 (S)** contains the **m/z** (number) of all species in the library. It must be the m/z of the transition and not the exact mass of the species. The species must be listed in **ascending order** of m/z. Within a same m/z transition, the species must also be listed in **ascending order** of retention time.

The other columns contain the following information, but they are not (yet) imported into LITL:
Column 1 (A) to 9 (I) contain information relative to the dynamic range;
Column 13 (M) contains the validation level of the species identification;
Column 15 (O) contains the LipidIDmz (MasterID without validation level nor barcode);
Column 16 (P) contains the molecular ID of the species;
Column 17 (Q) contains the total carbon ID of the species;
Column 18 (R) contains the exact mass of the species (computed from their total carbon ID, equal to the m/z when species are not identified);
Column 20 (T) contains the component name of the species;
Column 21 (U) contains the average average retention time of the species;
Column 22 (V) contains the median average retention time of the species;
Column 23 (W) contains the %cv of the average retention time of the species;
Column 24 (X) and so on contain the average retention time of the species in each dataset (columns).

Fourth sheet

A reference picture is available at the end of this section for the fourth sheet.

The first block of rows contains data about each standard (row) in each sample or dataset (columns), while the second block of rows contains data about all species INCLUDING standards. **Please note that if a standard is not present in a dataset, the dataset cannot be added to the library**. If the dataset needs to be in the library, but has one or more missing standards, the standards need to be deleted from the first block.

The first row contains column headers until the first occurrence of two consecutive blank cells (after the average of each dataset). The row separating the two blocks can be left blank **except for the "mz"** in the first (LIBRARY TESTING MODE) or second (EXPERIMENTAL DATA MODE) column.

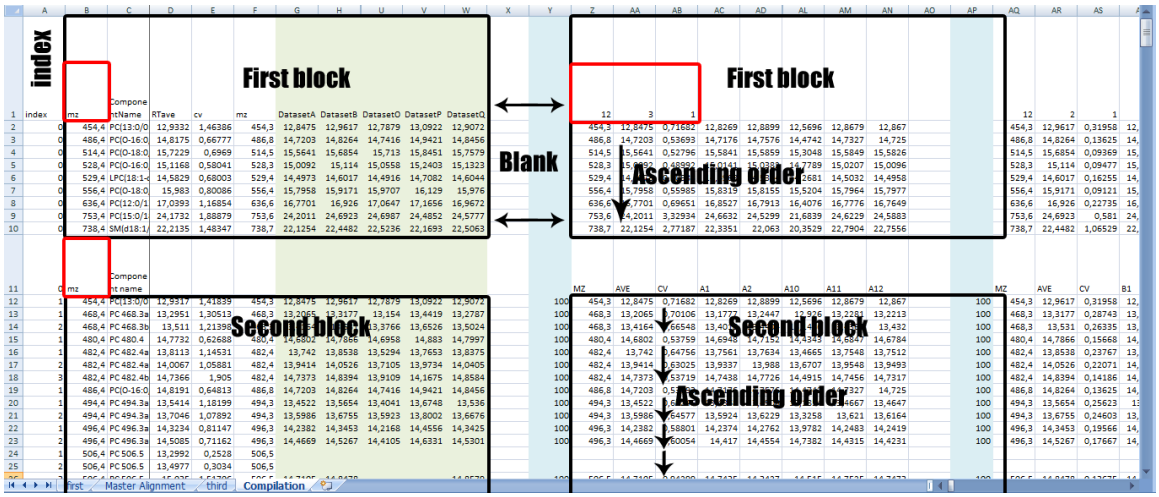
The **first column (A) (in LIBRARY TESTING MODE)** contains the **index** of each species within its transition. It is set to 0 for the first block of rows.
The **second column (B) (in LIBRARY TESTING MODE)** and the **first column (A) (in EXPERIMENTAL DATA MODE)** contains the m/z (number) of each species. Those m/z will have priority over the m/z specified for each dataset. **The first row must read "mz", as well as the row immediately after the first block of rows.**
The third column (C) (in LIBRARY TESTING MODE) or the second column (B) (in EXPERIMENTAL DATA MODE) contains the component name of the species.
The fourth column (D) (in LIBRARY TESTING MODE) or the third column (C) (in EXPERIMENTAL DATA MODE) contains the average retention time of the species.
The fifth column (E) (in LIBRARY TESTING MODE) or the fourth column (D) (in EXPERIMENTAL DATA MODE) contains the %cv of the species.
The sixth column (F) (in LIBRARY TESTING MODE) or the fifth column (E) (in EXPERIMENTAL DATA MODE) contains the m/z of the species.
The following columns contain the average retention times of each species (rows) in each dataset (columns).

Each dataset is separated by TWO CONSECUTIVE blank columns.
For each dataset:

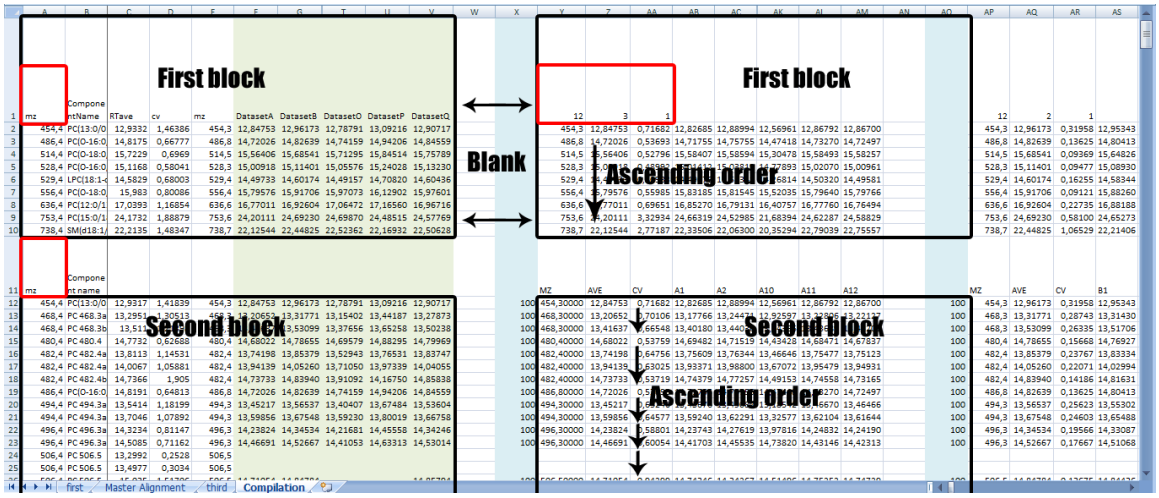
1st column:	**number of samples in dataset** (number) on the first row; m/z of each standard in the first block **in ascending order**; the header "MZ" on the row immediately after the first block; m/z of each species in the second block **in ascending order**.
2nd column:	**type[*] of the dataset** (number) on the first row; Average retention time of each standard in the first block **in ascending order** per transition; The header "AVE" on the row immediately after the first block; Average retention time of each species in the second block **in ascending order** per transition.
3rd column:	**1 if the dataset is to be taken into account**[†], on the first row; %CV of each standard in the first block; The header "CV" on the row immediately after the first block; %CV of each species in the second block.
Next columns:	clear cells on the first row; **Retention times of each standard** (rows) in each sample (columns) in the first block **in ascending order** per transition; **Sample names** (string) on the row immediately after the first block; **Retention times of each species** (rows) in each sample (columns) in the second block **in ascending order** per transition.

For LIBRARY TESTING mode:

[*] The type will be used to weight the samples appropriately. For example, if two datasets are of murine brain, their type will be noted with the same number.
[†] Ignoring a dataset in a library is achieved by changing this value to 0. PLEASE NOTE THAT THIS FEATURE IS NOT YET IMPLEMENTED AND IF THE VALUE IS NOT 1, IT WILL CAUSE LITL TO CRASH.



For EXPERIMENTAL DATA mode:



A1.3.2 Experimental data and sample information files

The experimental data file is created by exporting peak lists from MultiQuant and converting them into a comma delimited .csv file. The sample information file is made manually by using the template provided. It is also saved as a comma delimited .csv file.

The following protocol is focused for the Neural Regeneration Laboratory, hence the directory names. It however remains adaptable.

A. Opening & Exporting analyzed data from *.qsession* in MultiQuant

1. Log onto the BennettSyntax server, go to the directory **LITL/[your initials] and create the folder and subfolder [type of matrix]/[project name]/[lipid categories – *if applicable*]**
   - Note a: [Type of matrix] should indicate the tissue type and the subject from which the tissue was taken from. Example. Human plasma *or* Mouse brain.
   - Note b: [Project name] should indicate the year the project was done, the month & date the samples were run on the mass spec (if necessary; this will be very useful if samples are run, analyzed and then re-run for any reasons), the name of the project. Example: 2017Sept27 Death Study.

- Note c: [Lipid categories] subfolders may be created if necessary. For example, one study may be assessed for Sphingolipids and for Glyceroethanolamine. Hence, subfolders of Sphingolipids or GPE may be created in this directory.

2. Open MultiQuant software
3. Open your completed analysis file, the *.qsession* in MultiQuant.
4. Ensure the analysis for every transition in your *.qsession* is completed.
   - If no peaks are detected in a given transition, all of the values for that transition in the quantitation table should read "N/A".
   - If peaks are identified by the user as **saturated**, the **field "Used" must be unchecked prior to export**. This will automatically be set the value to "FALSE" following export.
   - **NO COMMAS** are used in any columns (comments for examples). Replace them by a semi-colon if needed.
5. Go to the File menu > Export > Results Table.... Ensure "all rows" and "all columns" are checked in the check box. Choose a place to save your file and ensure it saved as a text (**.txt**) file. It should be named after the original MultiQuant *.qsession* (ie. same name, different extension). MultiQuant can now be closed.
6. Open the MultiQuant (.txt) export just created in Microsoft Excel and save it as a **CSV UTF-8 (Comma delimited) (*.csv) file**. We will refer to this file as [MultiQuantExport.csv] from here till the rest of this document.
7. Transfer the [MultiQuantExport.csv] file to BennettSyntax/LITL/[your initials]/ [type of matrix]/[project name]/[lipid categories].

B. Preparing sample information for LITL

1. Go to BennettSyntax/LITL/Templates/. Find the file Sample Info.csv. **COPY** it to your computer. **DO NOT OVERWRITE** the original Sample Info.csv on the server as everyone will need this file in the future.
2. Rename this Sample Info.csv file into a new name which contains the name of the [MultiQuantExport.csv] file. Hence, Sample Info.csv will now be saved as "Sample Info_[MultiQuantExport].csv".
3. Open the Sample Info_[MultiQuantExport].csv file.
4. Row 1 contains the labels for each column field. **Do not modify anything in Row 1.** Row 2 contains examples for some of the column fields. Sample information will be added into this sheet from row 2 (**hence, replacing the examples in row 2**). Every row, from row 2, will contain information of each sample within the current dataset. Ensure that for every sample, **column A-O must be filled and none should be left empty**.
5. Start entering sample information for each sample:
   - Column A, **SampleName:** enter the sample name of each of your samples. These sample names **must be identical to the values in column "Sample Name" in your *.qsession***. Example: If your sample name in *.qsession* is **20170601_HyperFAM7999-180794 serum_PC MRM_HX** then you must input this value in the "SampleName" column as **20170601_HyperFAM7999-180794 serum_PC MRM_HX.**
   - Column B, **SampleID**: SampleID is an alternative identifier for the sample. It can be the same as SampleName or it can also be a more simplified version of the original sample. Example: If your SampleName cell is **20170601_HyperFAM7999-180794 serum_PC MRM_HX, you may wish to** shorten and modify it to **7999-180794 -Female** (eg. to include patient ID and sex)**.** Note: You can also set SampleID in your *.qsession*.
   - Column C, **IonAdduct**: Indicate the ion adduct that was generated in the mass spec method. Note: At present this value should be set to **$[M + H]^+$** as all of our current methods quantify this adduct.

- Column D, **MultiQuantVersion**: Indicate the version number of the MultiQuant software you used. The current version number is **3.0.2**. You can also find this in the Help > About… menu in the MultiQuant software.
- Column E, **AmountOfMatrix**: The matrix refers to the biological matrix your lipid samples were extracted from.
  - If the matrix was **plasma or serum**, please indicate the volume **in ml** that was used for the extraction.
  - If the matrix was **brain tissue** (ie. temporal cortex, hippocampus, etc), please indicate the mass **in mg** that was used for the extraction.
  - If the matrix was cultured **cells**, please indicate the number of cells used for extraction in 1E6 equivalents **(**eg. 1E6 cells would read as 1, 5E5 cells would read as 0.5).
- Column F, **ISComponentName**: Exogenous internal standards (IS) were added either during the lipid extraction procedure (to correct for loss of sample during the extraction) or at time of sample loading for mass spec analysis (to correct for loss of sample during mass spec run). The component name of an IS can be retrieved from the *.qsession* and **must match the ComponentName in the *.qsession***.
  - If a PC/SM mass spec method was run, the appropriate IS to normalize PC and/or SM is **PC(13:0/0:0)**. Input this component name if your dataset ran this mass spec method.
  - If a sphingolipid mass spec method was run, the appropriate IS to normalize sphingolipids is **C16-D31 Ceramide**. Input this component name if your dataset ran this mass spec method.
- Column G, **ISChemicalMass**: Based on the IS that was chosen in Column G, input the chemical mass of this IS into this cell. The chemical mass values of different IS are listed in a file called "StandardsforPmolCalculationsJuly92017.xlsx" found in folder BennettSyntax/LITL/Templates/.
  - Chemical mass of **PC(13:0/0:0)** is **453.55** g/mol. Input this chemical mass for PC(13:0/0:0).
  - Chemical mass of **C16-D31 Ceramide** is **569.092** g/mol. Input this chemical mass for C16-D31 Ceramide.
  - Chemical mass of **PS(12:0/13:0)** is **654.81** g/mol. Input this chemical mass for PS(12:0/13:0).
  - Chemical mass of **PE(12:0/13:0)** is **593.77** g/mol. Input this chemical mass for PE(12:0/13:0)/
- Column H, **ISng**: Indicate the **ng** of the IS that was chosen for normalization (in columns G and H). This **ng** is equivalent to the volume of the IS that was added to the sample during the lipid extraction procedure or at time of mass spec. The default values are below; the value of PC(13:0/0:0) belongs to the **50% protocol**, as listed in the StandardsforPmolCalculationsJuly92017.xlsx file. However, if you are analyzing samples you did not extract, and are unsure which extraction protocol your samples were done by, you must verify which protocol and use the corresponding ng. We have in the past added different ng of IS and if you analyzed and are identifying older samples you may have a different ng for some samples.
  - Amount of **ng** of **PC(13:0/0:0)** in 50% extraction protocol: **90.70** ng
  - Amount of **ng** of **C16-D31 Ceramide**: **298.5** ng
  - Amount of **ng** of **PS(12:0/13:0)**: **248.9** ng
  - Amount of **ng** of **PE(12:0/13:0)**: **99.60** ng
- Column I, **Units**: This is the column for "unit of the analyte" (lipid abundance per unit of extracted tissue). There are 3 unit examples that are listed in this column. Pick one that is appropriate for your tissue type and delete the others.
  - Use **pmol/1e6** cells for cell cultures,

- Use **pmol/mg** to normalize to tissue wet weight, and
- Use **pmol/ml** to normalize to volume of biological fluid.

- Column J, **StatTemplate:** Indicate the name of statistical template you wish to use as it appears in the file BennettSyntax/LITL/PROTOCOL & TEMPLATES/Graph and Stat Templates/**.** This folder contains examples of graphs and statistical methods that you may wish to use. Pick one of these examples for your analysis.
  - If the graph type and stat you wish to use is not yet available in the template, then discuss with Dr. Bennett the statistics and graph formats appropriate for your data. Follow the below step:
  - Open the file **"2018April11 PRISM Template for Templates (TN)".** Use the graphing format in this file; you can modify the Data Table if you wish to arrange your data differently from the template. You can also modify the statistical analysis, after getting approval by Dr. Bennett. If you change your statistical analysis from the template, you may wish to change your graph type.
  - MAKE SURE THAT YOU KEEP THE GRAPHING FORMAT THE SAME.

- Once you and Dr. Bennett are happy with your statistical analysis and graphs, this will be a new template.
- Add this new template into the folder BennettSyntax/LITL/PROTOCOL & TEMPLATES/Graph and Stat Templates/**.** Save the updated file as a new file with your initial and clearly indicate the type of statistical test used in the.
- COPY your template to your folder in BennettSyntax/LITL/[your initials]/ [type of matrix]/[project name]/[lipid categories]/.
- Once completed, add the name of this template for your PRISM output, to column J. The name must be exactly identical to the .pzfx file.
  - Columns K to O, **Feature1-5**: Features are the independent variables manipulated in your experiment. For each independent variable you are assessing, you must define that sample under each feature column.
    - If you only have one independent variable, fill in only Feature1.
    - If you have more, continue adding values into other Feature columns.
    - If you have **empty Features**, **make sure they are filled with "NA"**. Example: You may want to analyze your dataset first by sex differences. Indicate in Feature1 whether a sample was extracted from a MALE or a FEMALE subject. If you have a second feature, for example disease condition, then in Feature2, fill in the specific condition such as CTRL or DLB or AD etc. This is the name that will define each of your comparisons in Prism. These comparisons should match the labels of your Prism template chosen in Column J.
    - Note: For each Feature column, input values are case sensitive, meaning that, "MALE" and "MaLe" are two different groups. Hence, make sure that the values entered in this column are consistent.

6. Save this file.
7. Make a copy of this file and move the copied file to the folder BennettSyntax/ LITL/[your initials]/[type of matrix]/[project name]/[lipid categories]/.
   - Ensure that this folder now contains the [MultiQuantExport.csv], Sample Info_[MultiQuantExport].csv, and your PRISM .pzfx template.

A1.4 LITL outputs

A1.4.1 Excel files

A1.4.1.1 Library testing mode

The excel workbook created will contain two sheets.

The first sheet contains the assignments made for each species (rows) in each sample (columns). The retention time of each species is input in the corresponding cells. The first rows indicate the log objective function values obtained for each assignment. The sample names are indicated 2 rows above the retention times. The row immediately before the retention times indicates the number of species detected in each sample. The identity of each species is indicated in the first two columns.

The second sheet is in the same format as the first sheet, except that the assignments that are erroneous are replaced by their correct barcode. Also, the row immediately before the retention times is replaced to indicate the percentage of accuracy of the set of assignments (number of correct assignment divided by the number of assignments) in each sample.

A1.4.1.2 Experimental data mode

The excel workbook created will contain three sheets.

The first sheet contains the assignments made for each species (rows) in each sample (columns). The retention time of each species is input in the corresponding cells. The first rows indicate the log objective function values obtained for each assignment. The sample names are indicated above the retention times. The identity of each species is indicated in the first two columns.

The second sheet is in the same format as the first sheet, except that instead of the retention times, the pmol equivalents are indicated in the cells.

The third sheet is in the same format as the first sheet, except that instead of the retention times, the annotations (standard, saturation, isotope, dehydration, deglycosylation) are indicated in the cells.

A1.4.2 Graphs

The graphs are output in a separate "Graphs" folder inside the output folder. They are produced in .eps format. A .tex script is also output, and can be used to concatenate all the .eps graphs into a single .pdf file.

A1.4.3 Prism scripts

The .pzc scripts are output in the folders named "forprism-X-Y-Z" where X is the number of the analysis test, Y is the analysis test name (ttest, oneway, or twoway), and Z is the number of the prism file within the same analysis test X. Each folder may have more than one .pzc file, named "testscriptW", where W is the number of the scripts. The scripts **must be executed in that order**. Before executing the scripts, the template prism file must be **copied into the folder containing the scripts, and renamed "Template" (case-sensitive)**.

A1.5 Step-by-step

This tutorial will make use of the four examples files found in the installation folder: LIBRARY_forExpData.xlsx, LIBRARY_forLibraryTesting.xlsx, DATA_example.csv, and SAMPLEINFO_example.csv. All sample names were modified in these files for confidentiality of the data.

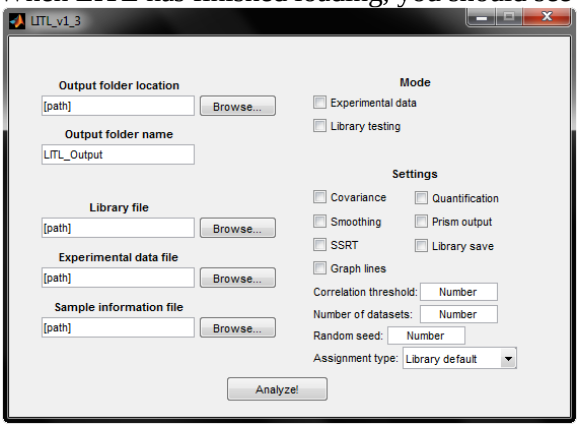
A1.5.1 Installation

If LITL is already installed on your device, move on to Step 1.
Open the installation folder and execute the "MyAppInstaller_mcr" file. This will install the MCR required to execute stand-alone MATLAB applications on your device. If you cannot find the file, or if you have any problems installing the MCR, you can also find it via https://www.mathworks.com/products/compiler/matlab-runtime.html.
You can now execute LITL.

A1.5.2 Start-up

Open LITL by executing the LITL_v1_x.exe or .app file (x being the version number). When LITL is loading, this window should appear.

When LITL has finished loading, you should see the following interface.

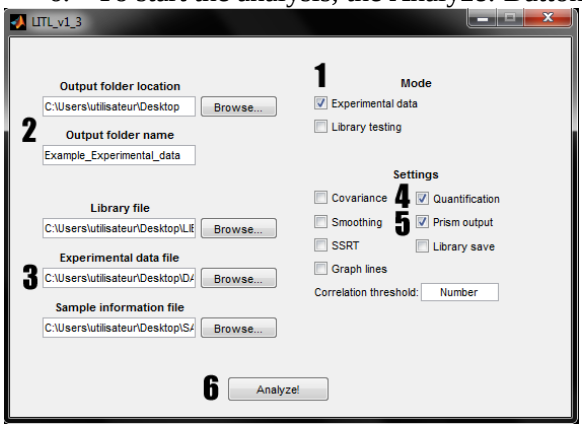


A1.5.3 Example 1: Experimental data analysis with prism script output

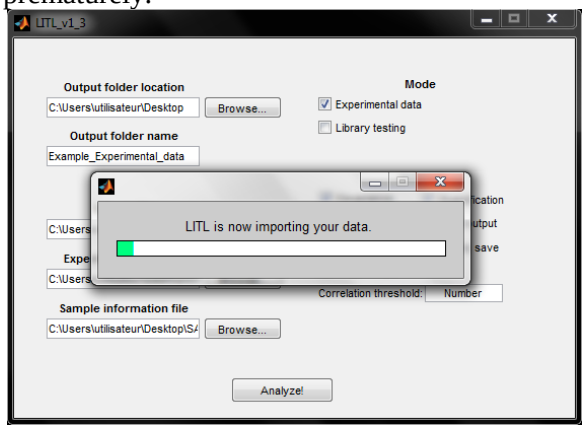
This will be an example to analyze the example DATA_example.csv.
The file contains 12 samples, divided in 3 groups of 4 samples. We want to identify the species in the samples, quantify them, and analyze their abundance to assess if the groups are statistically different.

1. For assigning experimental data to a library, the Experimental data mode has to be enabled.
2. The output folder location has to be specified, as well as the output folder name.
3. The files have to be input in the corresponding fields using the Browse.. button.

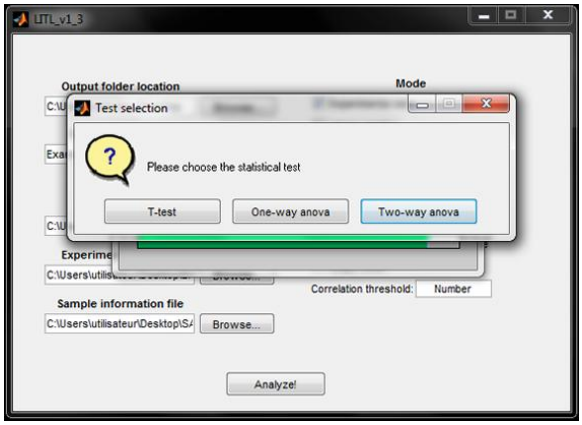
4. To quantify the data, so the Quantification setting must be toggled on.
5. To create scripts for statistical analysis, the Prism output setting must be toggled on.
6. To start the analysis, the Analyze! Button is pressed.



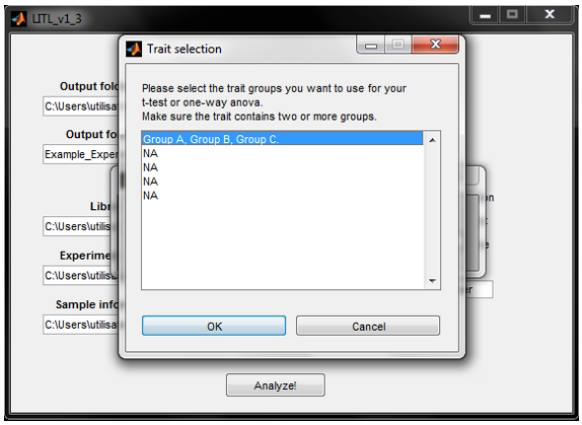
LITL will now import the data and analyze the samples. The progress of the analysis will be updated in the progress bar. Please note that closing the progress bar window will cause the analysis to end prematurely.



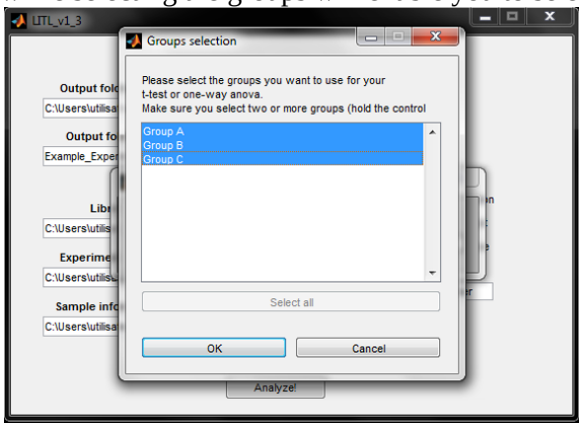
At the end of its analysis, LITL will prompt the user for the statistical analysis details. This example will create scripts for a one-way anova comparing 3 groups of 4 samples. Therefore, select the one-way anova test.



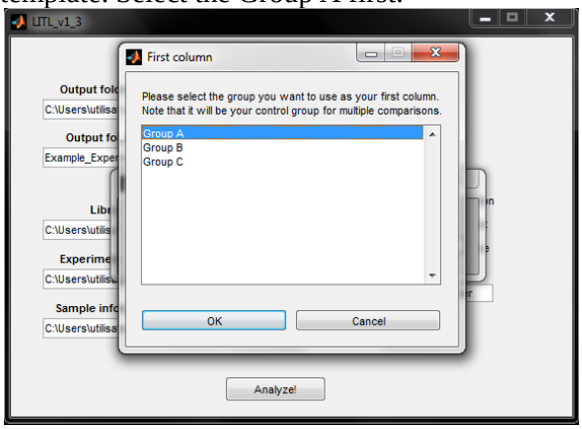
LITL will then ask which feature contains the groups to be compared. Select the first feature containing Group A, Group B, and Group C.



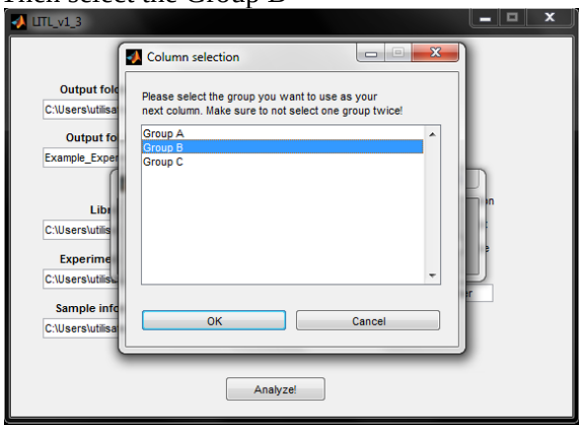
LITL will then ask within this feature, which groups are to be compared. Select the Group A, Group B, and Group C. Holding down the control key (Windows) or the command key (Mac) on the keyboard while selecting the groups will enable you to select multiple groups.



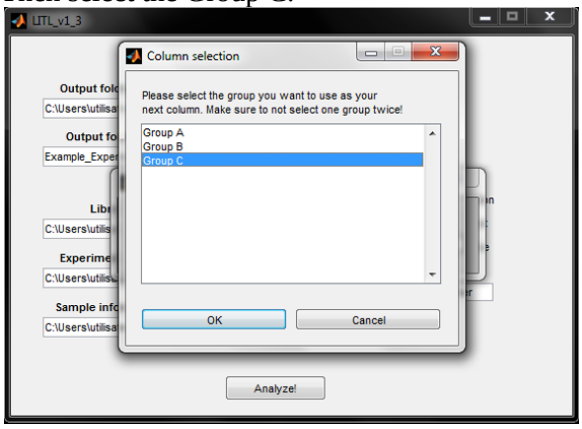
LITL will then ask you to select your groups one by one, in the order you want them in your prism template. Select the Group A first.



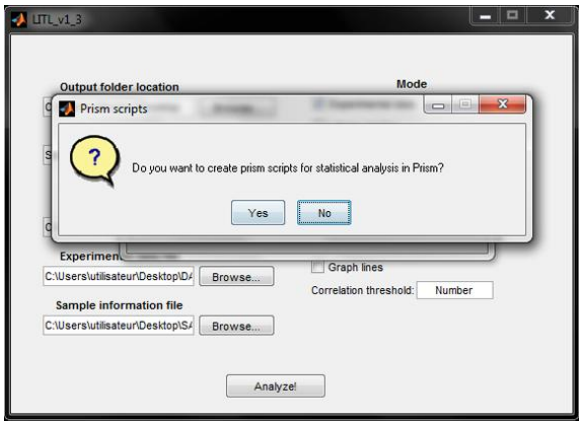
Then select the Group B



Then select the Group C.

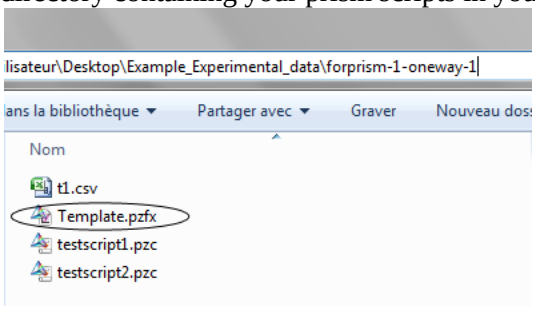


The scripts will now be output in the output folder. LITL will ask if another test requires scripts. Select no.

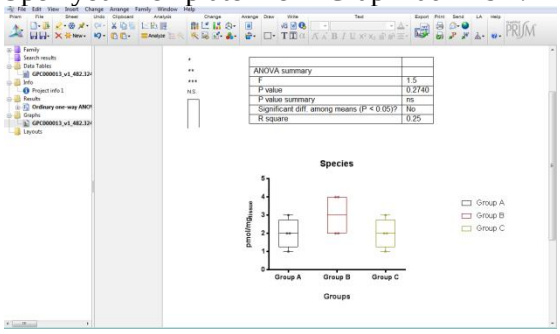


You can now exit LITL by pressing on the X at the top of the window. LITL will ask for your confirmation.

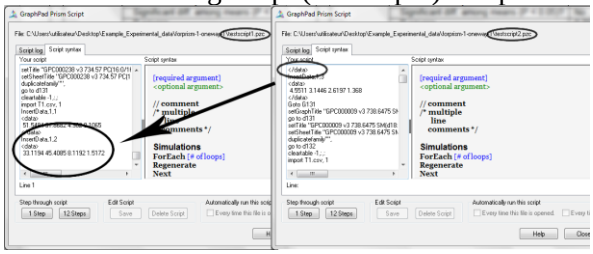
In order to produce the one-way anova analyses in GraphPad Prism, copy your prism template into the directory containing your prism scripts in your output folder, and rename it as Template.



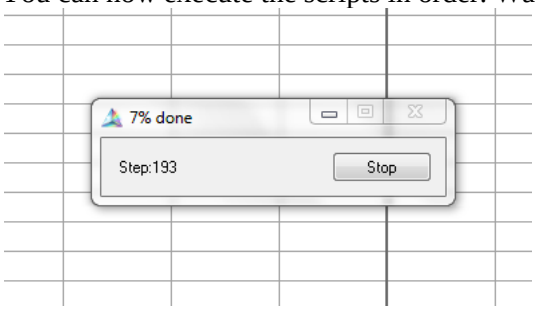
Open your Template file in GraphPad Prism.



Open the first script (testscript1) from the menu in GraphPad Prism (File>Run Script>Open Script File..). Make sure the scripts finishes without cuting a data block in half. If it does, cut the end of the data block from the following script (testscript2) and paste it at the end of the first one.



You can now execute the scripts in order. Wait until one script finishes until executing the next.

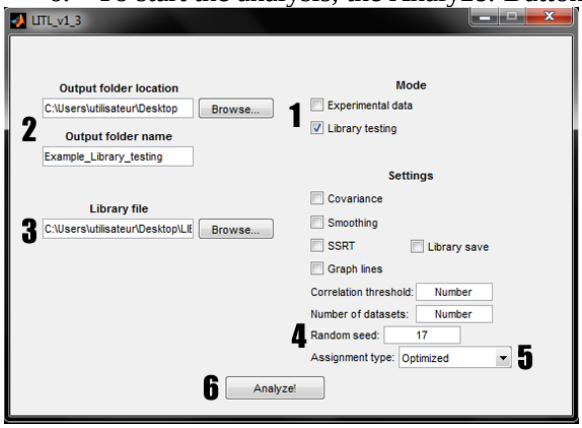


After the last script is done, you can save the prism file. The second half of the data can be analyzed in the same way, by copying the empty Template file to the second folder (forprism-1-oneway-2) and repeating the last steps.

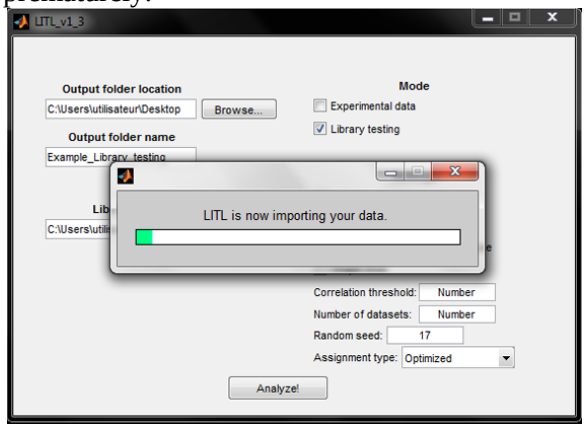
A1.5.4 Example 2: Testing a library in Library testing mode

This will be an example to analyze the example LIBRARY_forLibraryTesting.xlsx.
We want to test the quality of the library and of LITL's algorithm when using no covariance and no data transformation.

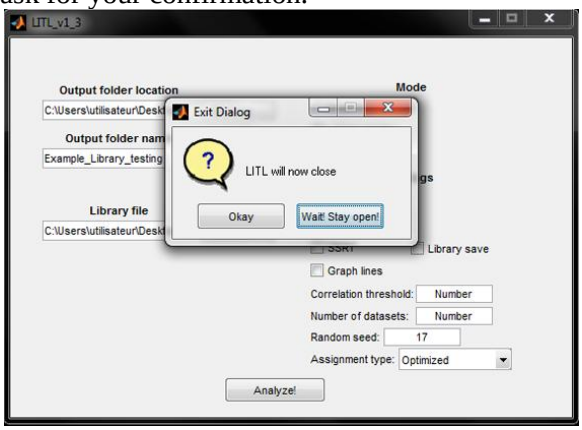
1. For testing a library, the Library testing mode has to be enabled.
2. The output folder location has to be specified, as well as the output folder name.
3. The files have to be input in the corresponding fields using the Browse.. button.
4. To be able to reproduce our data, a random seed is specified (17 in the example).
5. The "Optimized" Assignment type will be used, to test LITL's accuracy if the library was real data.
6. To start the analysis, the Analyze! Button is pressed.



LITL will now import the data and analyze the samples. The progress of the analysis will be updated in the progress bar. Please note that closing the progress bar window will cause the analysis to end prematurely.



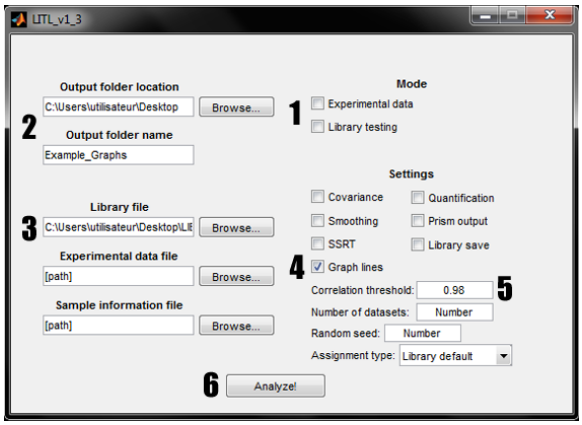
When the analysis is done, LITL can be closed by pressing on the X at the top of the window. LITL will ask for your confirmation.



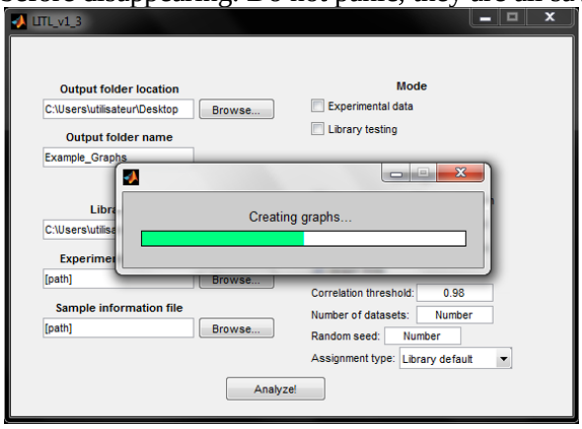
A1.5.5 Example 3: Producing graphs of the regression lines of lipid groups in the RT vs m/z plane

This will be an example to analyze the example library and produce graphs of each of its sample in the RT vs m/z plane. The file LIBRARY_forExpData.xlsx will be used.

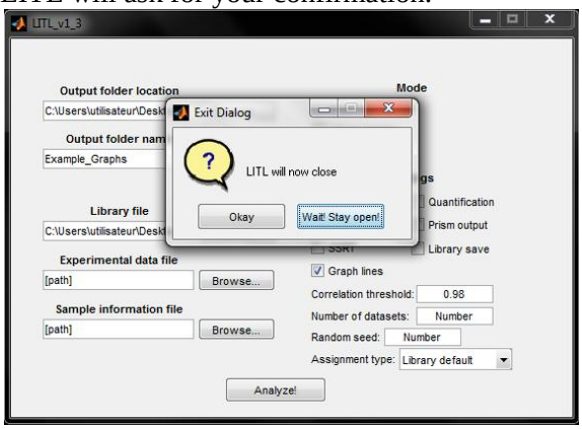
1. To only produce the graphs of a library, the Library testing mode and the Experimental data mode have to be toggled off.
2. The output folder location has to be specified, as well as the output folder name.
3. The library file has to be input in the corresponding field using the Browse.. button.
4. The Graph lines setting has to be toggled on.
5. The Correlation threshold has to be specified (here 0.98).
6. To start the analysis, the Analyze! Button is pressed.



LITL will now import the data and produce the graphs. Each graph will appear in a new window shortly before disappearing. Do not panic, they are all saved before automatically being closed.



When the graphs are all produced, LITL can be closed by pressing on the X at the top of the window. LITL will ask for your confirmation.



Appendix B: LIT source code

The following section is the source code of LITv1.16 (python code). For aesthetic purposes, the font of the appendix B has been changed to a smaller size Courrier New font, and the indentations were changed to 0.1” in order to respect the margins of this document.

```
## LIT v1.16 for Windows
import wx
import wx.xrc
import wx.lib.mixins.listctrl as listmix
import re
import numpy
import xlsxwriter
import datetime
import os, os.path
class notesFrame ( wx.Frame ):
 def __init__( self, parent ):
  wx.Frame.__init__ ( self, parent, id = wx.ID_ANY, title = "Release Notes", pos =
wx.DefaultPosition, size = wx.Size( 600,350 ), style =
wx.DEFAULT_FRAME_STYLE|wx.TAB_TRAVERSAL )
  self.SetSizeHints( wx.Size( 600,350 ), wx.DefaultSize )
  self.SetBackgroundColour( wx.SystemSettings.GetColour( wx.SYS_COLOUR_3DLIGHT ) )
  mainSizer = wx.BoxSizer( wx.VERTICAL )
  self.Intro_Text = wx.TextCtrl( self, wx.ID_ANY, u"LIT v1.16\nLast edited on May 29th
2018\n\nPlease report any bugs or concerns\n\n[18/05/28] v1.16\nDrop down menus now
read only\n\n[18/05/19] v1.15\nRevised list of constraints\n\n[18/05/07]
v1.14\nRevised list of species\n\n[18/04/17] v1.13\nNew GUI\n\n[18/03/14] v1.12\nFixed
minor GUI bugs\n\n[18/03/08] v1.11\nFixed GUI in macOS\n[18/03/08]\nDeuterated
standards are now recognized in identity searches\n[18/03/07]\nMass accuracy added as
a user-specified parameter\nMore ion adducts added\n\n[18/02/16] v1.10\nDehydration
ion adduct added [M+H-H2O]+\nMinor corrections\n\n[17/12/19] v1.9\nRecognition of lyso
species for O- and P- linkages.\nMinor bugs fixed for dialkyl, MIPC, M(IP)2C, and S1P
ID searches.\n\n[17/12/15] v1.8\nRecognition of lyso species when molecular carbon id
is entered.\nMinor PE-Cer bug fixed.\n\n[17/12/12] v1.7\nSphP identity changed to
S1P.\nGb3-Cer identities now available.\nMinor GlcChol bug fixed.\n\n[17/10/17]
v1.6\nLysoSM structures are now available.\n\n[17/10/10] v1.5\nEnabled molecular
identity recognition for identity searches of the form HEAD(a:b/c:d),
HEAD(a:b/c:d;e:f), HEAD(e:f;a:b/c:d), with or without oxidation.\nCorrected minor
input bugs.\n\n[17/10/06] v1.4\nWindows and MAC OS versions made available.\nGalCer
and GlcCer identities now recognized.\nDisparities between requested identity and
computed identity are now visible if any.\n[17/09/27]\nStatus bar and window titles
added.\nExporting window bug fixed.\nChemical formula now contains subscripted
numbers.\nResults are now sortable in alphabetical and ascending numerical order by
clicking on column headers of the results section.\nMinor edits to the 'List of
accepted identities' text box.\nHexSph, SphP, and SM link calculations corrected.
HexSph, Sph, and SphP species computed do not have a second fatty acyl
chain.\nAcylCeramides linked on the omega end (Cer(mx:y;a:b), Cer(dx:y;a:b), and
Cer(tx:y;a:b)), glucosyl cholestrol (GlcChol), and cholesteryl esters (CE(x:y))
identities made available.\nMinor species selection bugs fixed.\n[17/09/13]\nExport to
.xlsx menu option now available\n\n[17/09/13] v1.3\nNew GUI, families are now selected
with a checkbox system that allows cross-families searches, and ion adducts (Neutral,
[M+H]+, and [M+Na]+) are now available\n\n[17/09/08] v1.2\nAcylCeramides
(Cer(a:b;mx:y), Cer(a:b;dx:y), and Cer(a:b;tx:y)) identities made
available\n\n[17/08/28] v1.1\nM(IP)2C identities made available\n\n[17/07/11]
v1.0\nFirst GUI, with exclusion criterias made available\n\n[17/07/05] v0.7\nAdded
Sph, HexSph, and SphP species in the Ceramides, ajusted constraints\n\n[17/07/04]
v0.6\nAdded exact mass in [M+H]+ to the output", wx.DefaultPosition, wx.DefaultSize,
wx.TE_CHARWRAP|wx.TE_MULTILINE|wx.TE_READONLY|wx.NO_BORDER  )
  self.Intro_Text.SetBackgroundColour( wx.SystemSettings.GetColour(
wx.SYS_COLOUR_3DLIGHT ) )
  mainSizer.Add( self.Intro_Text, 1, wx.ALL|wx.ALIGN_CENTER_HORIZONTAL|wx.EXPAND, 5 )
  self.SetSizer( mainSizer )
  self.Layout()
  self.Centre( wx.BOTH )
 def __del__( self ):
  pass
class identitiesFrame ( wx.Frame ):
 title = "new Window"
 def __init__( self, parent ):
```

```
   wx.Frame.__init__ ( self, parent, id = wx.ID_ANY, title = "Constraints", pos =
wx.DefaultPosition, size = wx.Size( 300,350 ), style =
wx.DEFAULT_FRAME_STYLE|wx.TAB_TRAVERSAL )
   self.SetSizeHints( wx.Size( 300,350 ), wx.DefaultSize )
   self.SetBackgroundColour( wx.SystemSettings.GetColour( wx.SYS_COLOUR_3DLIGHT ) )
   mainSizer = wx.BoxSizer( wx.VERTICAL )
   fgSizer7 = wx.FlexGridSizer( 2, 3, 0, 0 )
   fgSizer7.SetFlexibleDirection( wx.BOTH )
   fgSizer7.SetNonFlexibleGrowMode( wx.FLEX_GROWMODE_SPECIFIED )
   self.m_staticText19 = wx.StaticText( self, wx.ID_ANY, u"Species",
wx.DefaultPosition, wx.DefaultSize, 0 )
   self.m_staticText19.Wrap( -1 )
   self.m_staticText19.SetFont( wx.Font( wx.NORMAL_FONT.GetPointSize(), 70, 90, 92,
False, wx.EmptyString ) )
   fgSizer7.Add( self.m_staticText19, 0, wx.ALL, 5 )
   self.m_staticText20 = wx.StaticText( self, wx.ID_ANY, u"Carbons:Unsaturations",
wx.DefaultPosition, wx.DefaultSize, 0 )
   self.m_staticText20.Wrap( -1 )
   self.m_staticText20.SetFont( wx.Font( wx.NORMAL_FONT.GetPointSize(), 70, 90, 92,
False, wx.EmptyString ) )
   fgSizer7.Add( self.m_staticText20, 0, wx.ALL, 5 )
   self.m_staticText21 = wx.StaticText( self, wx.ID_ANY, u"Oxidation",
wx.DefaultPosition, wx.DefaultSize, 0 )
   self.m_staticText21.Wrap( -1 )
   self.m_staticText21.SetFont( wx.Font( wx.NORMAL_FONT.GetPointSize(), 70, 90, 92,
False, wx.EmptyString ) )
   fgSizer7.Add( self.m_staticText21, 0, wx.ALL, 5 )
   mainSizer.Add( fgSizer7, 0, wx.ALL, 5 )
   self.m_scrolledWindow1 = wx.ScrolledWindow( self, wx.ID_ANY, wx.DefaultPosition,
wx.DefaultSize, wx.HSCROLL|wx.VSCROLL )
   self.m_scrolledWindow1.SetScrollRate( 5, 5 )
   fgSizer4 = wx.FlexGridSizer( 0, 3, 0, 0 )
   fgSizer4.SetFlexibleDirection( wx.BOTH )
   fgSizer4.SetNonFlexibleGrowMode( wx.FLEX_GROWMODE_SPECIFIED )
   self.m_staticText121 = wx.StaticText( self.m_scrolledWindow1, wx.ID_ANY,
u"PS(x:y)\nLPS(x:y)\nLPS(O-x:y)\nLPS(P-x:y)\nPS(O-x:y)\nPS(P-
x:y)\ndialkylPS(x:y)\n\nPE(x:y)\nLPE(x:y)\nLPE(O-x:y)\nLPE(P-x:y)\nPE(O-x:y)\nPE(P-
x:y)\ndialkylPE(x:y)\n\nPC(x:y)\nLPC(x:y)\nLPC(O-x:y)\nLPC(P-x:y)\nPC(O-x:y)\nPC(P-
x:y)\ndialkylPC(x:y)\n\nLysoSM(dx:y)\nLysoSM(tx:y)\nSM(dx:y)\nSM(tx:y)\n\nCer(mx:y)\nS
ph(mx:y)\nCer(mx:y;a:b)\n\nCer(dx:y)\nHexCer(dx:y)\nLacCer(dx:y)\nPE-Cer(dx:y)\nPI-
Cer(dx:y)\nCerP(dx:y)\nMIPC(dx:y)\nM(IP)2C(dx:y)\nSph(dx:y)\nHexSph(dx:y)\nS1P(dx:y)\n
Cer(a:b;dx:y)\nCer(dx:y;a:b)\nGb3-
Cer(dx:y)\n\nCer(tx:y)\nHexCer(tx:y)\nLacCer(tx:y)\nPE-Cer(tx:y)\nPI-
Cer(tx:y)\nCerP(tx:y)\nMIPC(tx:y)\nM(IP)2C(tx:y)\nSph(tx:y)\nHexSph(tx:y)\nS1P(tx:y)\n
Cer(a:b;tx:y)\nCer(tx:y;a:b)\nGb3-Cer(tx:y)\n\nGlcChol(x:y)\nCE(x:y)",
wx.DefaultPosition, wx.DefaultSize, 0 )
   self.m_staticText121.Wrap( -1 )
   fgSizer4.Add( self.m_staticText121, 0, wx.ALL, 5 )
   self.m_staticText13 = wx.StaticText( self.m_scrolledWindow1, wx.ID_ANY, u"12-60:0-
12\n12-30:0-9\n12-30:0-9\n12-30:0-9\n12-60:0-12\n12-60:0-12\n12-60:0-12\n\n12-60:0-
12\n12-30:0-9\n12-30:0-9\n12-30:0-9\n12-60:0-12\n12-60:0-12\n12-60:0-12\n\n4-60:0-
12\n2-30:0-9\n2-30:0-9\n2-30:0-9\n3-60:0-12\n3-60:0-12\n2-60:0-12\n\n12-30:0-3\n12-
30:0-3\n12-60:0-3\n12-60:0-3\n\n12-60:0-3\n12-60:0-3\n22-98:0-12\n\n12-60:0-3\n12-
60:0-3\n12-60:0-3\n12-60:0-3\n12-60:0-3\n12-60:0-3\n12-60:0-3\n12-60:0-3\n12-60:0-
3\n12-60:0-3\n12-60:0-3\n22-98:0-12\n22-98:0-12\n12-60:0-3\n\n12-60:0-3\n12-60:0-
3\n12-60:0-3\n12-60:0-3\n12-60:0-3\n12-60:0-3\n12-60:0-3\n12-60:0-3\n12-60:0-3\n12-
60:0-3\n12-60:0-3\n22-98:0-12\n22-98:0-12\n12-60:0-3\n\n0-60:0-12\n12-60:0-9",
wx.DefaultPosition, wx.DefaultSize, 0 )
   self.m_staticText13.Wrap( -1 )
   fgSizer4.Add( self.m_staticText13, 0, wx.ALL, 5 )
   self.m_staticText14 = wx.StaticText( self.m_scrolledWindow1, wx.ID_ANY, u"OH, CHO,
COOH\nOH, CHO, COOH\nOH, CHO, COOH\nOH, CHO, COOH\nOH, CHO, COOH\nOH, CHO, COOH\nOH,
CHO, COOH\n\nOH, CHO, COOH\nOH, CHO, COOH\nOH, CHO, COOH\nOH, CHO, COOH\nOH, CHO,
```

```
COOH\nOH, CHO, COOH\nOH, CHO, COOH\n\nOH, CHO, COOH\nOH, CHO, COOH\nOH, CHO, COOH\nOH,
CHO, COOH\nOH, CHO, COOH\nOH, CHO, COOH\nOH, CHO, COOH\n\nOH\nOH\nOH, CHO, COOH\nOH,
CHO, COOH\n\nOH, CHO, COOH\nOH, CHO, COOH\nOH, CHO, COOH\n\nOH, CHO, COOH\nOH, CHO,
COOH\nOH, CHO, COOH\nOH, CHO, COOH\nOH, CHO, COOH\nOH, CHO, COOH\nOH, CHO, COOH\nOH,
CHO, COOH\nnone\nnone\nnone\nOH, CHO, COOH\nOH, CHO, COOH\nOH, CHO, COOH\n\nOH, CHO,
COOH\nOH, CHO, COOH\nOH, CHO, COOH\nOH, CHO, COOH\nOH, CHO, COOH\nOH, CHO, COOH\nOH,
CHO, COOH\nOH, CHO, COOH\nnone\nnone\nnone\nOH, CHO, COOH\nOH, CHO, COOH\nOH, CHO,
COOH\n\nnone\nnone", wx.DefaultPosition, wx.DefaultSize, 0 )
   self.m_staticText14.Wrap( -1 )
   fgSizer4.Add( self.m_staticText14, 0, wx.ALL, 5 )
   self.m_scrolledWindow1.SetSizer( fgSizer4 )
   self.m_scrolledWindow1.Layout()
   fgSizer4.Fit( self.m_scrolledWindow1 )
   mainSizer.Add( self.m_scrolledWindow1, 1, wx.EXPAND |wx.ALL, 5 )
   self.SetSizer( mainSizer )
   self.Layout()
   self.Centre( wx.BOTH )
 def __del__( self ):
   pass
class mainFrame ( wx.Frame ):
 def __init__( self, parent ):
   wx.Frame.__init__ ( self, parent, id = wx.ID_ANY, title = "LIT v1.16 - Lipid
Identification Tool", pos = wx.DefaultPosition, size = wx.Size( 1080,750 ), style =
wx.SYSTEM_MENU|wx.MINIMIZE_BOX|wx.CAPTION|wx.CLOSE_BOX|wx.CLIP_CHILDREN|wx.TAB_TRAVERS
AL )
   # self.SetSizeHints( wx.Size( 1080,750 ), wx.Size( 1080,750 ) )
   self.SetSizeHints( wx.DefaultSize, wx.DefaultSize )
   self.SetBackgroundColour( wx.SystemSettings.GetColour( wx.SYS_COLOUR_3DLIGHT ) )
   # bg sizer
   mainSizer = wx.FlexGridSizer( 0, 0, 0, 0 )
   mainSizer.SetFlexibleDirection( wx.BOTH )
   mainSizer.SetNonFlexibleGrowMode( wx.FLEX_GROWMODE_SPECIFIED )
   leftSizer = wx.BoxSizer( wx.VERTICAL )
   # query block
   querySizer = wx.GridSizer( 0, 2, 0, 0 )
   subquerySizer = wx.FlexGridSizer( 0, 2, 0, 0 )
   subquerySizer.SetFlexibleDirection( wx.BOTH )
   subquerySizer.SetNonFlexibleGrowMode( wx.FLEX_GROWMODE_SPECIFIED )
   self.mz_search1 = wx.StaticText( self, wx.ID_ANY, u"Mass-over-charge ratio",
wx.DefaultPosition, wx.Size( -1,-1 ), wx.ALIGN_CENTRE )
   self.mz_search1.Wrap( -1 )
   subquerySizer.Add( self.mz_search1, 1, wx.ALL|wx.EXPAND|wx.ALIGN_CENTER_HORIZONTAL,
5 )
   accuracyChoices = [ u"± 0.01", u"± 0.05", u"± 0.1", u"± 0.5", u"± 1" ]
   self.accuracy = wx.ComboBox( self, wx.ID_ANY, u"Combo!", wx.DefaultPosition,
wx.DefaultSize, accuracyChoices, wx.CB_READONLY)
   subquerySizer.Add( self.accuracy, 0, wx.ALL, 5 )
   self.accuracy.SetSelection(3)
   querySizer.Add( subquerySizer, 0, wx.ALIGN_CENTER_HORIZONTAL, 5 )
   self.id_search1 = wx.StaticText( self, wx.ID_ANY, u"Identity", wx.DefaultPosition,
wx.Size( -1,-1 ), wx.ALIGN_CENTRE )
   self.id_search1.Wrap( -1 )
   querySizer.Add( self.id_search1, 1,
wx.ALL|wx.ALIGN_CENTER_VERTICAL|wx.ALIGN_CENTER_HORIZONTAL|wx.EXPAND, 5 )
   self.mz_search = wx.TextCtrl( self, wx.ID_ANY, wx.EmptyString, wx.DefaultPosition,
wx.DefaultSize, 0 )
   querySizer.Add( self.mz_search, 1, wx.ALL|wx.ALIGN_CENTER_HORIZONTAL|wx.EXPAND, 5 )
   self.id_search = wx.TextCtrl( self, wx.ID_ANY, wx.EmptyString, wx.DefaultPosition,
wx.DefaultSize, 0 )
   querySizer.Add( self.id_search, 1, wx.ALL|wx.EXPAND, 5 )
   leftSizer.Add( querySizer, 0, wx.EXPAND, 5 )
   # search block
   searchSizer = wx.GridSizer( 0, 2, 0, 0 )
```

```
   subsearchSizer = wx.FlexGridSizer( 0, 3, 0, 0 )
   subsearchSizer.SetFlexibleDirection( wx.BOTH )
   subsearchSizer.SetNonFlexibleGrowMode( wx.FLEX_GROWMODE_SPECIFIED )
   self.m_staticText10 = wx.StaticText( self, wx.ID_ANY, u"Ion adduct",
wx.DefaultPosition, wx.DefaultSize, 0 )
   self.m_staticText10.Wrap( -1 )
   subsearchSizer.Add( self.m_staticText10, 0, wx.ALL|wx.ALIGN_CENTER_VERTICAL, 5 )
   ionAdductChoices = [ u"Neutral", u"[M+H]+", u"[M+H-H2O]+", u"[M+2H]2+", u"[M+3H]3+",
u"[M+4H]4+", u"[M+K]+", u"[M+2K]2+", u"[M+2K-H]+", u"[M+Na]+", u"[M+2Na]2+", u"[M+2Na-
H]+", u"[M+Li]+", u"[M+2Li]2+", u"[M+NH4]+", u"[M-H]-", u"[M-2H]2-", u"[M-3H]3-",
u"[M-4H]4-", u"[M+Cl]-", u"[M+OAc]-" ]
   self.ionAdduct = wx.ComboBox( self, wx.ID_ANY, u"Combo!", wx.DefaultPosition,
wx.DefaultSize, ionAdductChoices, wx.CB_READONLY )
   subsearchSizer.Add( self.ionAdduct, 0, wx.ALL, 5 )
   self.ionAdduct.SetSelection(1)
   self.SearchButtonMZ = wx.Button( self, wx.ID_ANY, u"Search!", wx.DefaultPosition,
wx.DefaultSize, 0 )
   subsearchSizer.Add( self.SearchButtonMZ, 0, wx.ALL|wx.ALIGN_CENTER_HORIZONTAL, 5 )
   searchSizer.Add( subsearchSizer, 0, wx.ALIGN_CENTER_HORIZONTAL, 5 )
   self.SearchButtonID = wx.Button( self, wx.ID_ANY, u"Search!", wx.DefaultPosition,
wx.DefaultSize, 0 )
   searchSizer.Add( self.SearchButtonID, 0, wx.ALL|wx.ALIGN_CENTER_HORIZONTAL, 5 )
   leftSizer.Add( searchSizer, 0, wx.EXPAND, 5 )
   # oxi block
   self.SeparationLine1 = wx.StaticLine( self, wx.ID_ANY, wx.DefaultPosition,
wx.DefaultSize, wx.LI_HORIZONTAL )
   leftSizer.Add( self.SeparationLine1, 0, wx.EXPAND |wx.ALL, 5 )
   oxiSizer = wx.GridBagSizer( 0, 0 )
   oxiSizer.SetFlexibleDirection( wx.BOTH )
   oxiSizer.SetNonFlexibleGrowMode( wx.FLEX_GROWMODE_SPECIFIED )
   self.oxiText = wx.StaticText( self, wx.ID_ANY, u"Oxidation: ", wx.DefaultPosition,
wx.DefaultSize, 0 )
   self.oxiText.Wrap( -1 )
   oxiSizer.Add( self.oxiText, wx.GBPosition( 0, 0 ), wx.GBSpan( 1, 1 ), wx.ALL, 5 )
   self.NoOxy = wx.CheckBox( self, wx.ID_ANY, u"None", wx.DefaultPosition,
wx.DefaultSize, 0 )
   self.NoOxy.SetValue(True)
   oxiSizer.Add( self.NoOxy, wx.GBPosition( 0, 1 ), wx.GBSpan( 1, 1 ), wx.ALL, 5 )
   self.OH = wx.CheckBox( self, wx.ID_ANY, u"Hydroxylation (OH)", wx.DefaultPosition,
wx.DefaultSize, 0 )
   oxiSizer.Add( self.OH, wx.GBPosition( 0, 2 ), wx.GBSpan( 1, 1 ), wx.ALL, 5 )
   self.CHO = wx.CheckBox( self, wx.ID_ANY, u"Oxidation (CHO)", wx.DefaultPosition,
wx.DefaultSize, 0 )
   oxiSizer.Add( self.CHO, wx.GBPosition( 0, 3 ), wx.GBSpan( 1, 1 ), wx.ALL, 5 )
   self.COOH = wx.CheckBox( self, wx.ID_ANY, u"Oxidation (COOH)", wx.DefaultPosition,
wx.DefaultSize, 0 )
   oxiSizer.Add( self.COOH, wx.GBPosition( 0, 4 ), wx.GBSpan( 1, 1 ), wx.ALL, 5 )
   leftSizer.Add( oxiSizer, 0, wx.ALIGN_CENTER_HORIZONTAL, 5 )
   # species block
   self.SeparationLine2 = wx.StaticLine( self, wx.ID_ANY, wx.DefaultPosition,
wx.DefaultSize, wx.LI_HORIZONTAL )
   leftSizer.Add( self.SeparationLine2, 0, wx.EXPAND|wx.ALL, 5 )
   self.speciesText = wx.StaticText( self, wx.ID_ANY, u"Species settings",
wx.DefaultPosition, wx.DefaultSize, 0 )
   self.speciesText.Wrap( -1 )
   leftSizer.Add( self.speciesText, 0, wx.ALL|wx.ALIGN_CENTER_HORIZONTAL, 5 )
   speciesSizer = wx.FlexGridSizer( 2, 4, 0, 0 )
   speciesSizer.SetFlexibleDirection( wx.BOTH )
   speciesSizer.SetNonFlexibleGrowMode( wx.FLEX_GROWMODE_ALL )
   # PS
   psSizer = wx.BoxSizer( wx.VERTICAL )
   self.ps_scroll = wx.ScrolledWindow( self, wx.ID_ANY, wx.DefaultPosition,
wx.DefaultSize, wx.ALWAYS_SHOW_SB|wx.SIMPLE_BORDER|wx.VSCROLL )
```

```
   self.ps_scroll.SetScrollRate( 5, 5 )
   self.ps_scroll.SetMinSize( wx.Size( 130,150 ) )
   self.ps_scroll.SetBackgroundColour( wx.Colour( 255, 255, 255 ) )
   psInSizer = wx.BoxSizer( wx.VERTICAL )
   self.ps_1 = wx.CheckBox( self.ps_scroll, wx.ID_ANY, u"PS(x:y)", wx.DefaultPosition,
wx.DefaultSize, 0 )
   psInSizer.Add( self.ps_1, 0, wx.ALL, 5 )
   self.ps_2 = wx.CheckBox( self.ps_scroll, wx.ID_ANY, u"LPS(x:y)", wx.DefaultPosition,
wx.DefaultSize, 0 )
   psInSizer.Add( self.ps_2, 0, wx.ALL, 5 )
   self.ps_3 = wx.CheckBox( self.ps_scroll, wx.ID_ANY, u"LPS(O-x:y)",
wx.DefaultPosition, wx.DefaultSize, 0 )
   psInSizer.Add( self.ps_3, 0, wx.ALL, 5 )
   self.ps_4 = wx.CheckBox( self.ps_scroll, wx.ID_ANY, u"LPS(P-x:y)",
wx.DefaultPosition, wx.DefaultSize, 0 )
   psInSizer.Add( self.ps_4, 0, wx.ALL, 5 )
   self.ps_5 = wx.CheckBox( self.ps_scroll, wx.ID_ANY, u"PS(O-x:y)",
wx.DefaultPosition, wx.DefaultSize, 0 )
   psInSizer.Add( self.ps_5, 0, wx.ALL, 5 )
   self.ps_6 = wx.CheckBox( self.ps_scroll, wx.ID_ANY, u"PS(P-x:y)",
wx.DefaultPosition, wx.DefaultSize, 0 )
   psInSizer.Add( self.ps_6, 0, wx.ALL, 5 )
   self.ps_7 = wx.CheckBox( self.ps_scroll, wx.ID_ANY, u"dialkylPS(x:y)",
wx.DefaultPosition, wx.DefaultSize, 0 )
   psInSizer.Add( self.ps_7, 0, wx.ALL, 5 )
   self.ps_scroll.SetSizer( psInSizer )
   self.ps_scroll.Layout()
   psInSizer.Fit( self.ps_scroll )
   psSizer.Add( self.ps_scroll, 1, wx.ALL|wx.EXPAND, 5 )
   self.All_PS = wx.Button( self, wx.ID_ANY, u"Un/Select all", wx.DefaultPosition,
wx.DefaultSize, 0 )
   psSizer.Add( self.All_PS, 0, wx.ALIGN_CENTER_HORIZONTAL, 5 )
   speciesSizer.Add( psSizer, 1, wx.EXPAND, 5 )
   # PE
   peSizer = wx.BoxSizer( wx.VERTICAL )
   self.pe_scroll = wx.ScrolledWindow( self, wx.ID_ANY, wx.DefaultPosition,
wx.DefaultSize, wx.ALWAYS_SHOW_SB|wx.SIMPLE_BORDER|wx.VSCROLL )
   self.pe_scroll.SetScrollRate( 5, 5 )
   self.pe_scroll.SetMinSize( wx.Size( 130,150 ) )
   self.pe_scroll.SetBackgroundColour( wx.Colour( 255, 255, 255 ) )
   peInSizer = wx.BoxSizer( wx.VERTICAL )
   self.pe_1 = wx.CheckBox( self.pe_scroll, wx.ID_ANY, u"PE(x:y)", wx.DefaultPosition,
wx.DefaultSize, 0 )
   peInSizer.Add( self.pe_1, 0, wx.ALL, 5 )
   self.pe_2 = wx.CheckBox( self.pe_scroll, wx.ID_ANY, u"LPE(x:y)", wx.DefaultPosition,
wx.DefaultSize, 0 )
   peInSizer.Add( self.pe_2, 0, wx.ALL, 5 )
   self.pe_3 = wx.CheckBox( self.pe_scroll, wx.ID_ANY, u"LPE(O-x:y)",
wx.DefaultPosition, wx.DefaultSize, 0 )
   peInSizer.Add( self.pe_3, 0, wx.ALL, 5 )
   self.pe_4 = wx.CheckBox( self.pe_scroll, wx.ID_ANY, u"LPE(P-x:y)",
wx.DefaultPosition, wx.DefaultSize, 0 )
   peInSizer.Add( self.pe_4, 0, wx.ALL, 5 )
   self.pe_5 = wx.CheckBox( self.pe_scroll, wx.ID_ANY, u"PE(O-x:y)",
wx.DefaultPosition, wx.DefaultSize, 0 )
   peInSizer.Add( self.pe_5, 0, wx.ALL, 5 )
   self.pe_6 = wx.CheckBox( self.pe_scroll, wx.ID_ANY, u"PE(P-x:y)",
wx.DefaultPosition, wx.DefaultSize, 0 )
   peInSizer.Add( self.pe_6, 0, wx.ALL, 5 )
   self.pe_7 = wx.CheckBox( self.pe_scroll, wx.ID_ANY, u"dialkylPE(x:y)",
wx.DefaultPosition, wx.DefaultSize, 0 )
   peInSizer.Add( self.pe_7, 0, wx.ALL, 5 )
   self.pe_scroll.SetSizer( peInSizer )
```

```
   self.pe_scroll.Layout()
   peInSizer.Fit( self.pe_scroll )
   peSizer.Add( self.pe_scroll, 1, wx.ALL|wx.EXPAND, 5 )
   self.All_PE = wx.Button( self, wx.ID_ANY, u"Un/Select all", wx.DefaultPosition,
wx.DefaultSize, 0 )
   peSizer.Add( self.All_PE, 0, wx.ALIGN_CENTER_HORIZONTAL, 5 )
   speciesSizer.Add( peSizer, 1, wx.EXPAND, 5 )
   # PC
   pcSizer = wx.BoxSizer( wx.VERTICAL )
   self.pc_scroll = wx.ScrolledWindow( self, wx.ID_ANY, wx.DefaultPosition,
wx.DefaultSize, wx.ALWAYS_SHOW_SB|wx.SIMPLE_BORDER|wx.VSCROLL )
   self.pc_scroll.SetScrollRate( 5, 5 )
   self.pc_scroll.SetMinSize( wx.Size( 130,150 ) )
   self.pc_scroll.SetBackgroundColour( wx.Colour( 255, 255, 255 ) )
   pcInSizer = wx.BoxSizer( wx.VERTICAL )
   self.pc_1 = wx.CheckBox( self.pc_scroll, wx.ID_ANY, u"PC(x:y)", wx.DefaultPosition,
wx.DefaultSize, 0 )
   pcInSizer.Add( self.pc_1, 0, wx.ALL, 5 )
   self.pc_2 = wx.CheckBox( self.pc_scroll, wx.ID_ANY, u"LPC(x:y)", wx.DefaultPosition,
wx.DefaultSize, 0 )
   pcInSizer.Add( self.pc_2, 0, wx.ALL, 5 )
   self.pc_3 = wx.CheckBox( self.pc_scroll, wx.ID_ANY, u"LPC(O-x:y)",
wx.DefaultPosition, wx.DefaultSize, 0 )
   pcInSizer.Add( self.pc_3, 0, wx.ALL, 5 )
   self.pc_4 = wx.CheckBox( self.pc_scroll, wx.ID_ANY, u"LPC(P-x:y)",
wx.DefaultPosition, wx.DefaultSize, 0 )
   pcInSizer.Add( self.pc_4, 0, wx.ALL, 5 )
   self.pc_5 = wx.CheckBox( self.pc_scroll, wx.ID_ANY, u"PC(O-x:y)",
wx.DefaultPosition, wx.DefaultSize, 0 )
   pcInSizer.Add( self.pc_5, 0, wx.ALL, 5 )
   self.pc_6 = wx.CheckBox( self.pc_scroll, wx.ID_ANY, u"PC(P-x:y)",
wx.DefaultPosition, wx.DefaultSize, 0 )
   pcInSizer.Add( self.pc_6, 0, wx.ALL, 5 )
   self.pc_7 = wx.CheckBox( self.pc_scroll, wx.ID_ANY, u"dialkylPC(x:y)",
wx.DefaultPosition, wx.DefaultSize, 0 )
   pcInSizer.Add( self.pc_7, 0, wx.ALL, 5 )
   self.pc_scroll.SetSizer( pcInSizer )
   self.pc_scroll.Layout()
   pcInSizer.Fit( self.pc_scroll )
   pcSizer.Add( self.pc_scroll, 1, wx.ALL|wx.EXPAND, 5 )
   self.All_PC = wx.Button( self, wx.ID_ANY, u"Un/Select all", wx.DefaultPosition,
wx.DefaultSize, 0 )
   pcSizer.Add( self.All_PC, 0, wx.ALIGN_CENTER_HORIZONTAL, 5 )
   speciesSizer.Add( pcSizer, 1, wx.EXPAND, 5 )
   # SM
   smSizer = wx.BoxSizer( wx.VERTICAL )
   self.sm_scroll = wx.ScrolledWindow( self, wx.ID_ANY, wx.DefaultPosition,
wx.DefaultSize, wx.ALWAYS_SHOW_SB|wx.SIMPLE_BORDER|wx.VSCROLL )
   self.sm_scroll.SetScrollRate( 5, 5 )
   self.sm_scroll.SetMinSize( wx.Size( 130,150 ) )
   self.sm_scroll.SetBackgroundColour( wx.Colour( 255, 255, 255 ) )
   smInSizer = wx.BoxSizer( wx.VERTICAL )
   self.sm_1 = wx.CheckBox( self.sm_scroll, wx.ID_ANY, u"LysoSM(dx:y)",
wx.DefaultPosition, wx.DefaultSize, 0 )
   smInSizer.Add( self.sm_1, 0, wx.ALL, 5 )
   self.sm_2 = wx.CheckBox( self.sm_scroll, wx.ID_ANY, u"LysoSM(tx:y)",
wx.DefaultPosition, wx.DefaultSize, 0 )
   smInSizer.Add( self.sm_2, 0, wx.ALL, 5 )
   self.sm_3 = wx.CheckBox( self.sm_scroll, wx.ID_ANY, u"SM(dx:y)", wx.DefaultPosition,
wx.DefaultSize, 0 )
   smInSizer.Add( self.sm_3, 0, wx.ALL, 5 )
   self.sm_4 = wx.CheckBox( self.sm_scroll, wx.ID_ANY, u"SM(tx:y)", wx.DefaultPosition,
wx.DefaultSize, 0 )
```

```
   smInSizer.Add( self.sm_4, 0, wx.ALL, 5 )
   self.sm_scroll.SetSizer( smInSizer )
   self.sm_scroll.Layout()
   smInSizer.Fit( self.sm_scroll )
   smSizer.Add( self.sm_scroll, 1, wx.ALL|wx.EXPAND, 5 )
   self.All_SM = wx.Button( self, wx.ID_ANY, u"Un/Select all", wx.DefaultPosition,
wx.DefaultSize, 0 )
   smSizer.Add( self.All_SM, 0, wx.ALIGN_CENTER_HORIZONTAL, 5 )
   speciesSizer.Add( smSizer, 1, wx.EXPAND, 5 )
   # mCer
   cermSizer = wx.BoxSizer( wx.VERTICAL )
   self.cerm_scroll = wx.ScrolledWindow( self, wx.ID_ANY, wx.DefaultPosition,
wx.DefaultSize, wx.ALWAYS_SHOW_SB|wx.SIMPLE_BORDER|wx.VSCROLL )
   self.cerm_scroll.SetScrollRate( 5, 5 )
   self.cerm_scroll.SetMinSize( wx.Size( 130,250 ) )
   self.cerm_scroll.SetBackgroundColour( wx.Colour( 255, 255, 255 ) )
   cermInSizer = wx.BoxSizer( wx.VERTICAL )
   self.cerm_1 = wx.CheckBox( self.cerm_scroll, wx.ID_ANY, u"Cer(mx:y)",
wx.DefaultPosition, wx.DefaultSize, 0 )
   cermInSizer.Add( self.cerm_1, 0, wx.ALL, 5 )
   self.cerm_2 = wx.CheckBox( self.cerm_scroll, wx.ID_ANY, u"Sph(mx:y)",
wx.DefaultPosition, wx.DefaultSize, 0 )
   cermInSizer.Add( self.cerm_2, 0, wx.ALL, 5 )
   self.cerm_3 = wx.CheckBox( self.cerm_scroll, wx.ID_ANY, u"Cer(mx:y;a:b)",
wx.DefaultPosition, wx.DefaultSize, 0 )
   cermInSizer.Add( self.cerm_3, 0, wx.ALL, 5 )
   self.cerm_scroll.SetSizer( cermInSizer )
   self.cerm_scroll.Layout()
   cermInSizer.Fit( self.cerm_scroll )
   cermSizer.Add( self.cerm_scroll, 1, wx.ALL|wx.EXPAND, 5 )
   self.All_mCer = wx.Button( self, wx.ID_ANY, u"Un/Select all", wx.DefaultPosition,
wx.DefaultSize, 0 )
   cermSizer.Add( self.All_mCer, 0, wx.ALIGN_CENTER_HORIZONTAL, 5 )
   speciesSizer.Add( cermSizer, 1, wx.EXPAND, 5 )
   # dCer
   cerdSizer = wx.BoxSizer( wx.VERTICAL )
   self.cerd_scroll = wx.ScrolledWindow( self, wx.ID_ANY, wx.DefaultPosition,
wx.DefaultSize, wx.ALWAYS_SHOW_SB|wx.SIMPLE_BORDER|wx.VSCROLL )
   self.cerd_scroll.SetScrollRate( 5, 5 )
   self.cerd_scroll.SetMinSize( wx.Size( 130,250 ) )
   self.cerd_scroll.SetBackgroundColour( wx.Colour( 255, 255, 255 ) )
   cerdInSizer = wx.BoxSizer( wx.VERTICAL )
   self.cerd_1 = wx.CheckBox( self.cerd_scroll, wx.ID_ANY, u"Cer(dx:y)",
wx.DefaultPosition, wx.DefaultSize, 0 )
   cerdInSizer.Add( self.cerd_1, 0, wx.ALL, 5 )
   self.cerd_2 = wx.CheckBox( self.cerd_scroll, wx.ID_ANY, u"HexCer(dx:y)",
wx.DefaultPosition, wx.DefaultSize, 0 )
   cerdInSizer.Add( self.cerd_2, 0, wx.ALL, 5 )
   self.cerd_3 = wx.CheckBox( self.cerd_scroll, wx.ID_ANY, u"LacCer(dx:y)",
wx.DefaultPosition, wx.DefaultSize, 0 )
   cerdInSizer.Add( self.cerd_3, 0, wx.ALL, 5 )
   self.cerd_4 = wx.CheckBox( self.cerd_scroll, wx.ID_ANY, u"PE-Cer(dx:y)",
wx.DefaultPosition, wx.DefaultSize, 0 )
   cerdInSizer.Add( self.cerd_4, 0, wx.ALL, 5 )
   self.cerd_5 = wx.CheckBox( self.cerd_scroll, wx.ID_ANY, u"PI-Cer(dx:y)",
wx.DefaultPosition, wx.DefaultSize, 0 )
   cerdInSizer.Add( self.cerd_5, 0, wx.ALL, 5 )
   self.cerd_6 = wx.CheckBox( self.cerd_scroll, wx.ID_ANY, u"CerP(dx:y)",
wx.DefaultPosition, wx.DefaultSize, 0 )
   cerdInSizer.Add( self.cerd_6, 0, wx.ALL, 5 )
   self.cerd_7 = wx.CheckBox( self.cerd_scroll, wx.ID_ANY, u"MIPC(dx:y)",
wx.DefaultPosition, wx.DefaultSize, 0 )
   cerdInSizer.Add( self.cerd_7, 0, wx.ALL, 5 )
```

```
   self.cerd_8 = wx.CheckBox( self.cerd_scroll, wx.ID_ANY, u"M(IP)2C(dx:y)",
wx.DefaultPosition, wx.DefaultSize, 0 )
   cerdInSizer.Add( self.cerd_8, 0, wx.ALL, 5 )
   self.cerd_9 = wx.CheckBox( self.cerd_scroll, wx.ID_ANY, u"Sph(dx:y)",
wx.DefaultPosition, wx.DefaultSize, 0 )
   cerdInSizer.Add( self.cerd_9, 0, wx.ALL, 5 )
   self.cerd_10 = wx.CheckBox( self.cerd_scroll, wx.ID_ANY, u"HexSph(dx:y)",
wx.DefaultPosition, wx.DefaultSize, 0 )
   cerdInSizer.Add( self.cerd_10, 0, wx.ALL, 5 )
   self.cerd_11 = wx.CheckBox( self.cerd_scroll, wx.ID_ANY, u"S1P(dx:y)",
wx.DefaultPosition, wx.DefaultSize, 0 )
   cerdInSizer.Add( self.cerd_11, 0, wx.ALL, 5 )
   self.cerd_12 = wx.CheckBox( self.cerd_scroll, wx.ID_ANY, u"Cer(a:b;dx:y)",
wx.DefaultPosition, wx.DefaultSize, 0 )
   cerdInSizer.Add( self.cerd_12, 0, wx.ALL, 5 )
   self.cerd_13 = wx.CheckBox( self.cerd_scroll, wx.ID_ANY, u"Cer(dx:y;a:b)",
wx.DefaultPosition, wx.DefaultSize, 0 )
   cerdInSizer.Add( self.cerd_13, 0, wx.ALL, 5 )
   self.cerd_14 = wx.CheckBox( self.cerd_scroll, wx.ID_ANY, u"Gb3-Cer(dx:y)",
wx.DefaultPosition, wx.DefaultSize, 0 )
   cerdInSizer.Add( self.cerd_14, 0, wx.ALL, 5 )
   self.cerd_scroll.SetSizer( cerdInSizer )
   self.cerd_scroll.Layout()
   cerdInSizer.Fit( self.cerd_scroll )
   cerdSizer.Add( self.cerd_scroll, 1, wx.ALL|wx.EXPAND, 5 )
   self.All_dCer = wx.Button( self, wx.ID_ANY, u"Un/Select all", wx.DefaultPosition,
wx.DefaultSize, 0 )
   cerdSizer.Add( self.All_dCer, 0, wx.ALIGN_CENTER_HORIZONTAL, 5 )
   speciesSizer.Add( cerdSizer, 1, wx.EXPAND, 5 )
   # tCer
   certSizer = wx.BoxSizer( wx.VERTICAL )
   self.cert_scroll = wx.ScrolledWindow( self, wx.ID_ANY, wx.DefaultPosition,
wx.DefaultSize, wx.ALWAYS_SHOW_SB|wx.SIMPLE_BORDER|wx.VSCROLL )
   self.cert_scroll.SetScrollRate( 5, 5 )
   self.cert_scroll.SetMinSize( wx.Size( 130,250 ) )
   self.cert_scroll.SetBackgroundColour( wx.Colour( 255, 255, 255 ) )
   certInSizer = wx.BoxSizer( wx.VERTICAL )
   self.cert_1 = wx.CheckBox( self.cert_scroll, wx.ID_ANY, u"Cer(tx:y)",
wx.DefaultPosition, wx.DefaultSize, 0 )
   certInSizer.Add( self.cert_1, 0, wx.ALL, 5 )
   self.cert_2 = wx.CheckBox( self.cert_scroll, wx.ID_ANY, u"HexCer(tx:y)",
wx.DefaultPosition, wx.DefaultSize, 0 )
   certInSizer.Add( self.cert_2, 0, wx.ALL, 5 )
   self.cert_3 = wx.CheckBox( self.cert_scroll, wx.ID_ANY, u"LacCer(tx:y)",
wx.DefaultPosition, wx.DefaultSize, 0 )
   certInSizer.Add( self.cert_3, 0, wx.ALL, 5 )
   self.cert_4 = wx.CheckBox( self.cert_scroll, wx.ID_ANY, u"PE-Cer(tx:y)",
wx.DefaultPosition, wx.DefaultSize, 0 )
   certInSizer.Add( self.cert_4, 0, wx.ALL, 5 )
   self.cert_5 = wx.CheckBox( self.cert_scroll, wx.ID_ANY, u"PI-Cer(tx:y)",
wx.DefaultPosition, wx.DefaultSize, 0 )
   certInSizer.Add( self.cert_5, 0, wx.ALL, 5 )
   self.cert_6 = wx.CheckBox( self.cert_scroll, wx.ID_ANY, u"CerP(tx:y)",
wx.DefaultPosition, wx.DefaultSize, 0 )
   certInSizer.Add( self.cert_6, 0, wx.ALL, 5 )
   self.cert_7 = wx.CheckBox( self.cert_scroll, wx.ID_ANY, u"MIPC(tx:y)",
wx.DefaultPosition, wx.DefaultSize, 0 )
   certInSizer.Add( self.cert_7, 0, wx.ALL, 5 )
   self.cert_8 = wx.CheckBox( self.cert_scroll, wx.ID_ANY, u"M(IP)2C(tx:y)",
wx.DefaultPosition, wx.DefaultSize, 0 )
   certInSizer.Add( self.cert_8, 0, wx.ALL, 5 )
   self.cert_9 = wx.CheckBox( self.cert_scroll, wx.ID_ANY, u"Sph(tx:y)",
wx.DefaultPosition, wx.DefaultSize, 0 )
```

```
   certInSizer.Add( self.cert_9, 0, wx.ALL, 5 )
   self.cert_10 = wx.CheckBox( self.cert_scroll, wx.ID_ANY, u"HexSph(tx:y)",
wx.DefaultPosition, wx.DefaultSize, 0 )
   certInSizer.Add( self.cert_10, 0, wx.ALL, 5 )
   self.cert_11 = wx.CheckBox( self.cert_scroll, wx.ID_ANY, u"S1P(tx:y)",
wx.DefaultPosition, wx.DefaultSize, 0 )
   certInSizer.Add( self.cert_11, 0, wx.ALL, 5 )
   self.cert_12 = wx.CheckBox( self.cert_scroll, wx.ID_ANY, u"Cer(a:b;tx:y)",
wx.DefaultPosition, wx.DefaultSize, 0 )
   certInSizer.Add( self.cert_12, 0, wx.ALL, 5 )
   self.cert_13 = wx.CheckBox( self.cert_scroll, wx.ID_ANY, u"Cer(tx:y;a:b)",
wx.DefaultPosition, wx.DefaultSize, 0 )
   certInSizer.Add( self.cert_13, 0, wx.ALL, 5 )
   self.cert_14 = wx.CheckBox( self.cert_scroll, wx.ID_ANY, u"Gb3-Cer(tx:y)",
wx.DefaultPosition, wx.DefaultSize, 0 )
   certInSizer.Add( self.cert_14, 0, wx.ALL, 5 )
   self.cert_scroll.SetSizer( certInSizer )
   self.cert_scroll.Layout()
   certInSizer.Fit( self.cert_scroll )
   certSizer.Add( self.cert_scroll, 1, wx.ALL|wx.EXPAND, 5 )
   self.All_tCer = wx.Button( self, wx.ID_ANY, u"Un/Select all", wx.DefaultPosition,
wx.DefaultSize, 0 )
   certSizer.Add( self.All_tCer, 0, wx.ALIGN_CENTER_HORIZONTAL, 5 )
   speciesSizer.Add( certSizer, 1, wx.EXPAND, 5 )
   # Chol
   cholSizer = wx.BoxSizer( wx.VERTICAL )
   self.chol_scroll = wx.ScrolledWindow( self, wx.ID_ANY, wx.DefaultPosition,
wx.DefaultSize, wx.ALWAYS_SHOW_SB|wx.SIMPLE_BORDER|wx.VSCROLL )
   self.chol_scroll.SetScrollRate( 5, 5 )
   self.chol_scroll.SetMinSize( wx.Size( 130,250 ) )
   self.chol_scroll.SetBackgroundColour( wx.Colour( 255, 255, 255 ) )
   cholInSizer = wx.BoxSizer( wx.VERTICAL )
   self.chol_1 = wx.CheckBox( self.chol_scroll, wx.ID_ANY, u"GlcChol(x:y)",
wx.DefaultPosition, wx.DefaultSize, 0 )
   cholInSizer.Add( self.chol_1, 0, wx.ALL, 5 )
   self.chol_2 = wx.CheckBox( self.chol_scroll, wx.ID_ANY, u"CE(x:y)",
wx.DefaultPosition, wx.DefaultSize, 0 )
   cholInSizer.Add( self.chol_2, 0, wx.ALL, 5 )
   self.chol_scroll.SetSizer( cholInSizer )
   self.chol_scroll.Layout()
   cholInSizer.Fit( self.chol_scroll )
   cholSizer.Add( self.chol_scroll, 1, wx.ALL|wx.EXPAND, 5 )
   self.All_chol = wx.Button( self, wx.ID_ANY, u"Un/Select all", wx.DefaultPosition,
wx.DefaultSize, 0 )
   cholSizer.Add( self.All_chol, 0, wx.ALIGN_CENTER_HORIZONTAL, 5 )
   speciesSizer.Add( cholSizer, 1, wx.EXPAND, 5 )
   leftSizer.Add( speciesSizer, 1, wx.EXPAND, 5 )
   mainSizer.Add( leftSizer, 1, wx.EXPAND, 5 )
   # right block
   resultsSizer = wx.BoxSizer( wx.VERTICAL )
   self.ResultsText = wx.StaticText( self, wx.ID_ANY, u"Results", wx.DefaultPosition,
wx.DefaultSize, 0 )
   self.ResultsText.Wrap( -1 )
   resultsSizer.Add( self.ResultsText, 0, wx.ALL|wx.ALIGN_CENTER_HORIZONTAL, 5 )
   self.Results = wx.ListCtrl(self, wx.ID_ANY, wx.DefaultPosition, wx.Size( 490,-1 ),
wx.LC_REPORT )
   self.Results.InsertColumn(0,'ID', width=155)
   self.Results.InsertColumn(1,'Chemical Formula', width=110)
   self.Results.InsertColumn(2,'Exact mass', width=100)
   self.Results.InsertColumn(3,'Mass-over-charge', width=110)
   resultsSizer.Add( self.Results, 1, wx.ALL|wx.EXPAND, 5 )
   mainSizer.Add( resultsSizer, 1, wx.EXPAND, 5 )
   # menu bar
```

```
   self.SetSizer( mainSizer )
   self.Layout()
   self.menubar = wx.MenuBar( 0 )
   self.menu = wx.Menu()
   self.menu_ID = wx.MenuItem( self.menu, wx.ID_ANY, u"List of accepted identities",
wx.EmptyString, wx.ITEM_NORMAL )
   self.menu.Append( self.menu_ID )
   self.menu_export = wx.MenuItem( self.menu, wx.ID_ANY, u"Export to .xlsx",
wx.EmptyString, wx.ITEM_NORMAL )
   self.menu.Append( self.menu_export )
   self.menu.AppendSeparator()
   self.menu_notes = wx.MenuItem( self.menu, wx.ID_ANY, u"Release Notes",
wx.EmptyString, wx.ITEM_NORMAL )
   self.menu.Append( self.menu_notes )
   self.menu_quit = wx.MenuItem( self.menu, wx.ID_ANY, u"Quit", wx.EmptyString,
wx.ITEM_NORMAL )
   self.menu.Append( self.menu_quit )
   self.menubar.Append( self.menu, u"About" )
   self.SetMenuBar( self.menubar )
   # status bar
   self.statusBar = self.CreateStatusBar( 1, 0, wx.ID_ANY )
   # end of formatting
   self.Centre( wx.BOTH )
   # Connect Events
   self.SearchButtonMZ.Bind( wx.EVT_BUTTON, self.onSearchRequestMZ )
   self.SearchButtonID.Bind( wx.EVT_BUTTON, self.onSearchRequestID )
   self.All_PS.Bind( wx.EVT_BUTTON, self.onselect_PS )
   self.All_PE.Bind( wx.EVT_BUTTON, self.onselect_PE )
   self.All_PC.Bind( wx.EVT_BUTTON, self.onselect_PC )
   self.All_SM.Bind( wx.EVT_BUTTON, self.onselect_SM )
   self.All_mCer.Bind( wx.EVT_BUTTON, self.onselect_mCer )
   self.All_dCer.Bind( wx.EVT_BUTTON, self.onselect_dCer )
   self.All_tCer.Bind( wx.EVT_BUTTON, self.onselect_tCer )
   self.All_chol.Bind( wx.EVT_BUTTON, self.onselect_chol )
   self.Bind( wx.EVT_LIST_COL_CLICK, self.onColClick, self.Results)
   self.Bind( wx.EVT_MENU, self.onopenID, id = self.menu_ID.GetId() )
   self.Bind( wx.EVT_MENU, self.onopenExport, id = self.menu_export.GetId() )
   self.Bind( wx.EVT_MENU, self.onopenNotes, id = self.menu_notes.GetId() )
   self.Bind( wx.EVT_MENU, self.onquitButton, id = self.menu_quit.GetId() )
 def __del__( self ):
   pass
 # When Search MZ button clicked --> compute and output results in list
 def onSearchRequestMZ( self, event ):
   self.Results.DeleteAllItems() # Clear results box
   self.id_search.Clear() # Clear id search
   self.statusBar.SetStatusText('( / ¯^¯)/ Calculating...  (ง ^ᗜ^)ง')
   self.now=datetime.datetime.now() # Save time of request
   ################################################
   # status variable
   okInput=0
   # atomic mass of C H N O P
   massExact=[12.000000, 1.007825, 14.003074, 15.994915, 30.973763]
   # initialize head atoms
   atomsSub=[]
   countSub=-1
   # multiple entries for different sub categories [PS, PE, PC/SM, Cer, chol]
   namesSub=[-1,-1,-1,-1,-1]
   # atoms for each additional carbon (chain)
   atomsChain=[1,2,0,0,0]
   # atoms for each additional unsaturation (chain)
   atomsUnsat=[0,-2,0,0,0]
   # initialize oxidation
   atomsMod=[]
```

```
   countMod=-1
   # multiple entries for [no oxidation, OH, CHO, COOH]
   namesMod=[-1,-1,-1,-1]
   # initialize linkage
   atomsAdj=[]
   # multiple entries for different sub categories [PS, PE, PC/SM, Cer, chol]
   countAdj=[-1,-1,-1,-1,-1]
   # multiple entries for different species [PS, LPS, LPS-O, LPS-P, PS-O, PS-P,
dialkylPS]
   namesPSAdj=[-1,-1,-1,-1,-1,-1,-1]
   # multiple entries for different species [PE, LPE, LPE-O, LPE-P, PE-O, PE-P,
dialkylPS]
   namesPEAdj=[-1,-1,-1,-1,-1,-1,-1]
   # multiple entries for different species [PC, LPC, LPC-O, LPC-P, PC-O, PC-P,
dialkylPC, LysoSM(d, LysoSM(t, SM(d, SM(t]
   namesPCAdj=[-1,-1,-1,-1,-1,-1,-1,   -1,-1,-1,-1]
   # multiple entries for different species [m,d,t][Cer, HexCer, LacCer, PE-Cer, PI-
Cer, CerP, MIPC, M(IP)2C, Sph, HexSph, S1P, Gb3-Cer]
   namesCerAdj=[-1,-1,   -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,   -1,-1,-1,-1,-1,-1,-1,-
1,-1,-1,-1,-1]
   # multiple entries for different species [GlcChol, CE]
   namesCholAdj=[-1,-1]
   # initialiaze acyl linkage
   atomsAcylAdj=[]
   countAcylAdj=-1
   # multiple entries for acyls [acyl(d, acyl(t, acylEOS(m, acylEOS(d, acylEOS(t]
   namesAcylCerAdj=[-1,-1,-1,-1,-1]
   try: # getting the m/z (+head +links)
    mz=float(self.mz_search.GetValue()) # m/z retrieved
    # mz accuracy
    if (self.accuracy.GetStringSelection()=='± 0.01'):
     mzBound1=mz-0.01
     mzBound2=mz+0.01
    elif (self.accuracy.GetStringSelection()=='± 0.05'):
     mzBound1=mz-0.05
     mzBound2=mz+0.05
    elif (self.accuracy.GetStringSelection()=='± 0.1'):
     mzBound1=mz-0.1
     mzBound2=mz+0.1
    elif (self.accuracy.GetStringSelection()=='± 0.5'):
     mzBound1=mz-0.5
     mzBound2=mz+0.5
    elif (self.accuracy.GetStringSelection()=='± 1'):
     mzBound1=mz-1
     mzBound2=mz+1
    # Ion Adduct
    if (self.ionAdduct.GetStringSelection()=='Neutral'):
     ionAdd=0
     ionDiv=1
    elif (self.ionAdduct.GetStringSelection()=='[M+H]+'):
     ionAdd=1.007825
     ionDiv=1
     mz-=ionAdd
    elif (self.ionAdduct.GetStringSelection()=='[M+H-H2O]+'):
     ionAdd=-17.002740
     ionDiv=1
     mz-=ionAdd
    elif (self.ionAdduct.GetStringSelection()=='[M+2H]2+'):
     ionAdd=2.015650
     ionDiv=2
     mz-=ionAdd
    elif (self.ionAdduct.GetStringSelection()=='[M+3H]3+'):
     ionAdd=3.023475
```

```
  ionDiv=3
  mz-=ionAdd
elif (self.ionAdduct.GetStringSelection()=='[M+4H]4+'):
  ionAdd=4.031300
  ionDiv=4
  mz-=ionAdd
elif (self.ionAdduct.GetStringSelection()=='[M+K]+'):
  ionAdd=38.963708
  ionDiv=1
  mz-=ionAdd
elif (self.ionAdduct.GetStringSelection()=='[M+2K]2+'):
  ionAdd=77.927416
  ionDiv=2
  mz-=ionAdd
elif (self.ionAdduct.GetStringSelection()=='[M+2K-H]+'):
  ionAdd=76.919591
  ionDiv=1
  mz-=ionAdd
elif (self.ionAdduct.GetStringSelection()=='[M+Na]+'):
  ionAdd=22.989770
  ionDiv=1
  mz-=ionAdd
elif (self.ionAdduct.GetStringSelection()=='[M+2Na]2+'):
  ionAdd=45.979540
  ionDiv=2
  mz-=ionAdd
elif (self.ionAdduct.GetStringSelection()=='[M+2Na-H]+'):
  ionAdd=44.971715
  ionDiv=1
  mz-=ionAdd
elif (self.ionAdduct.GetStringSelection()=='[M+Li]+'):
  ionAdd=7.016005
  ionDiv=1
  mz-=ionAdd
elif (self.ionAdduct.GetStringSelection()=='[M+2Li]2+'):
  ionAdd=14.032010
  ionDiv=2
  mz-=ionAdd
elif (self.ionAdduct.GetStringSelection()=='[M+NH4]+'):
  ionAdd=18.034374
  ionDiv=1
  mz-=ionAdd
elif (self.ionAdduct.GetStringSelection()=='[M-H]-'):
  ionAdd=-1.007825
  ionDiv=1
  mz-=ionAdd
elif (self.ionAdduct.GetStringSelection()=='[M-2H]2-'):
  ionAdd=-2.015650
  ionDiv=2
  mz-=ionAdd
elif (self.ionAdduct.GetStringSelection()=='[M-3H]3-'):
  ionAdd=-3.023475
  ionDiv=3
  mz-=ionAdd
elif (self.ionAdduct.GetStringSelection()=='[M-4H]4-'):
  ionAdd=-4.031300
  ionDiv=4
  mz-=ionAdd
elif (self.ionAdduct.GetStringSelection()=='[M+Cl]-'):
  ionAdd=34.968853
  ionDiv=1
  mz-=ionAdd
elif (self.ionAdduct.GetStringSelection()=='[M+OAc]-'):
```

```
      ionAdd=59.013305
      ionDiv=1
      mz-=ionAdd
    # Getting the head and link selections
    ## sub and Link values
    # If at least one of the species are checked,
    # PS
    if self.ps_1.IsChecked() or self.ps_2.IsChecked() or self.ps_3.IsChecked() or
self.ps_4.IsChecked() or self.ps_5.IsChecked() or self.ps_6.IsChecked() or
self.ps_7.IsChecked():
      okInput=1 # status is ok
      atomsSub+=[6,10,1,10,1] # append the head atoms to atomsSub
      countSub+=1 # index the head
      namesSub[0]=countSub # which head atoms corresponds to the current head in
atomsSub
    # PE
    if self.pe_1.IsChecked() or self.pe_2.IsChecked() or self.pe_3.IsChecked() or
self.pe_4.IsChecked() or self.pe_5.IsChecked() or self.pe_6.IsChecked() or
self.pe_7.IsChecked():
      okInput=1
      atomsSub+=[5,10,1,8,1]
      countSub+=1
      namesSub[1]=countSub
    # PC, SM
    if self.pc_1.IsChecked() or self.pc_2.IsChecked() or self.pc_3.IsChecked() or
self.pc_4.IsChecked() or self.pc_5.IsChecked() or self.pc_6.IsChecked() or
self.pc_7.IsChecked() or self.sm_1.IsChecked() or self.sm_2.IsChecked() or
self.sm_3.IsChecked() or self.sm_4.IsChecked():
      okInput=1
      atomsSub+=[8,16,1,8,1]
      countSub+=1
      namesSub[2]=countSub
    # mCer, dCer, tCer
    if self.cerm_1.IsChecked() or self.cerm_2.IsChecked() or self.cerm_3.IsChecked()
or self.cerd_1.IsChecked() or self.cerd_2.IsChecked() or self.cerd_3.IsChecked() or
self.cerd_4.IsChecked() or self.cerd_5.IsChecked() or self.cerd_6.IsChecked() or
self.cerd_7.IsChecked() or self.cerd_8.IsChecked() or self.cerd_9.IsChecked() or
self.cerd_10.IsChecked() or self.cerd_11.IsChecked() or self.cerd_12.IsChecked() or
self.cerd_13.IsChecked() or self.cerd_14.IsChecked() or self.cert_1.IsChecked() or
self.cert_2.IsChecked() or self.cert_3.IsChecked() or self.cert_4.IsChecked() or
self.cert_5.IsChecked() or self.cert_6.IsChecked() or self.cert_7.IsChecked() or
self.cert_8.IsChecked() or self.cert_9.IsChecked() or self.cert_10.IsChecked() or
self.cert_11.IsChecked() or self.cert_12.IsChecked() or self.cert_13.IsChecked() or
self.cert_14.IsChecked():
      okInput=1
      atomsSub+=[0,1,1,2,0]
      countSub+=1
      namesSub[3]=countSub
    # Chol
    if self.chol_1.IsChecked() or self.chol_2.IsChecked():
      okInput=1
      atomsSub+=[27,46,0,1,0]
      countSub+=1
      namesSub[4]=countSub
    if okInput==0:
      self.statusBar.SetStatusText('Please select at least one species first!')
    else: # Oxi settings
      okInput=0
      if self.NoOxy.IsChecked(): # No oxidation
        okInput=1 # status is ok
        atomsMod+=[0,0,0,0,0] # append the oxidation atoms to atomsMod
        countMod+=1 # index the oxidation
        namesMod[0]=countMod # which atoms corresponds to the current oxidation
```

```
if self.OH.IsChecked(): # OH
 okInput=1
 atomsMod+=[0,0,0,1,0]
 countMod+=1
 namesMod[1]=countMod
if self.CHO.IsChecked(): # CHO
 okInput=1
 atomsMod+=[0,-2,0,1,0]
 countMod+=1
 namesMod[2]=countMod
if self.COOH.IsChecked(): # COOH
 okInput=1
 atomsMod+=[0,-2,0,2,0]
 countMod+=1
 namesMod[3]=countMod
# error on oxidation
if okInput==0:
 self.statusBar.SetStatusText('Please select at least one oxidation setting!')
else: # Linkages
# for each checkbox checked, append atoms to atomsAdj, index it like above
 # PS links
 if self.ps_1.IsChecked(): #PS checked
  atomsAdj+=[0,0,0,0,0]
  countAdj[0]+=1
  namesPSAdj[0]=countAdj[0]
 if self.ps_2.IsChecked(): # LPS
  atomsAdj+=[0,2,0,-1,0]
  countAdj[0]+=1
  namesPSAdj[1]=countAdj[0]
 if self.ps_3.IsChecked(): # LPS-O
  atomsAdj+=[0,4,0,-2,0]
  countAdj[0]+=1
  namesPSAdj[2]=countAdj[0]
 if self.ps_4.IsChecked(): # LPS-P
  atomsAdj+=[0,2,0,-2,0]
  countAdj[0]+=1
  namesPSAdj[3]=countAdj[0]
 if self.ps_5.IsChecked(): # PS-O
  atomsAdj+=[0,2,0,-1,0]
  countAdj[0]+=1
  namesPSAdj[4]=countAdj[0]
 if self.ps_6.IsChecked(): # PS-P
  atomsAdj+=[0,0,0,-1,0]
  countAdj[0]+=1
  namesPSAdj[5]=countAdj[0]
 if self.ps_7.IsChecked(): # dialkylPS
  atomsAdj+=[0,4,0,-2,0]
  countAdj[0]+=1
  namesPSAdj[6]=countAdj[0]
 ## PE links
 if self.pe_1.IsChecked(): #PE checked
  atomsAdj+=[0,0,0,0,0]
  countAdj[1]+=1
  namesPEAdj[0]=countAdj[1]
 if self.pe_2.IsChecked(): # LPE
  atomsAdj+=[0,2,0,-1,0]
  countAdj[1]+=1
  namesPEAdj[1]=countAdj[1]
 if self.pe_3.IsChecked(): # LPE-O
  atomsAdj+=[0,4,0,-2,0]
  countAdj[1]+=1
  namesPEAdj[2]=countAdj[1]
 if self.pe_4.IsChecked(): # LPE-P
```

```
 atomsAdj+=[0,2,0,-2,0]
 countAdj[1]+=1
 namesPEAdj[3]=countAdj[1]
if self.pe_5.IsChecked(): # PE-O
 atomsAdj+=[0,2,0,-1,0]
 countAdj[1]+=1
 namesPEAdj[4]=countAdj[1]
if self.pe_6.IsChecked(): # PE-P
 atomsAdj+=[0,0,0,-1,0]
 countAdj[1]+=1
 namesPEAdj[5]=countAdj[1]
if self.pe_7.IsChecked(): # dialkylPE
 atomsAdj+=[0,4,0,-2,0]
 countAdj[1]+=1
 namesPEAdj[6]=countAdj[1]
## PC/SM links
if self.pc_1.IsChecked(): #PC checked
 atomsAdj+=[0,0,0,0,0]
 countAdj[2]+=1
 namesPCAdj[0]=countAdj[2]
if self.pc_2.IsChecked(): # LPC
 atomsAdj+=[0,2,0,-1,0]
 countAdj[2]+=1
 namesPCAdj[1]=countAdj[2]
if self.pc_3.IsChecked(): # LPC-O
 atomsAdj+=[0,4,0,-2,0]
 countAdj[2]+=1
 namesPCAdj[2]=countAdj[2]
if self.pc_4.IsChecked(): # LPC-P
 atomsAdj+=[0,2,0,-2,0]
 countAdj[2]+=1
 namesPCAdj[3]=countAdj[2]
if self.pc_5.IsChecked(): # PC-O
 atomsAdj+=[0,2,0,-1,0]
 countAdj[2]+=1
 namesPCAdj[4]=countAdj[2]
if self.pc_6.IsChecked(): # PC-P
 atomsAdj+=[0,0,0,-1,0]
 countAdj[2]+=1
 namesPCAdj[5]=countAdj[2]
if self.pc_7.IsChecked(): # dialkylPC
 atomsAdj+=[0,4,0,-2,0]
 countAdj[2]+=1
 namesPCAdj[6]=countAdj[2]
if self.sm_1.IsChecked(): # LysoSM(d
 atomsAdj+=[-3,-1,1,-3,0]
 countAdj[2]+=1
 namesPCAdj[7]=countAdj[2]
if self.sm_2.IsChecked(): # LysoSM(t
 atomsAdj+=[-3,-1,1,-4,0]
 countAdj[2]+=1
 namesPCAdj[8]=countAdj[2]
if self.sm_3.IsChecked(): # SM(d
 atomsAdj+=[-3,-3,1,-2,0]
 countAdj[2]+=1
 namesPCAdj[9]=countAdj[2]
if self.sm_4.IsChecked(): # SM(t
 atomsAdj+=[-3,-3,1,-3,0]
 countAdj[2]+=1
 namesPCAdj[10]=countAdj[2]
## Cer links
## m
if self.cerm_1.IsChecked(): # Cer(m
```

```
 atomsAdj+=[0,0,0,0,0]
 countAdj[3]+=1
 namesCerAdj[0]=countAdj[3]
if self.cerm_2.IsChecked(): # Sph(m
 atomsAdj+=[0,2,0,-1,0]
 countAdj[3]+=1
 namesCerAdj[1]=countAdj[3]
if self.cerm_3.IsChecked(): # Acyl (EOS) Cer(m
 atomsAcylAdj+=[0,-2,0,2,0]
 countAcylAdj+=1
 namesAcylCerAdj[2]=countAcylAdj
## d
if self.cerd_1.IsChecked(): # Cer(d
 atomsAdj+=[0,0,0,1,0]
 countAdj[3]+=1
 namesCerAdj[2]=countAdj[3]
if self.cerd_2.IsChecked(): # HexCer(d
 atomsAdj+=[6,10,0,6,0]
 countAdj[3]+=1
 namesCerAdj[3]=countAdj[3]
if self.cerd_3.IsChecked(): # LacCer(d
 atomsAdj+=[12,20,0,11,0]
 countAdj[3]+=1
 namesCerAdj[4]=countAdj[3]
if self.cerd_4.IsChecked(): # PE-Cer(d
 atomsAdj+=[2,6,1,4,1]
 countAdj[3]+=1
 namesCerAdj[5]=countAdj[3]
if self.cerd_5.IsChecked(): # PI-Cer(d
 atomsAdj+=[6,11,0,9,1]
 countAdj[3]+=1
 namesCerAdj[6]=countAdj[3]
if self.cerd_6.IsChecked(): # CerP(d
 atomsAdj+=[0,1,0,4,1]
 countAdj[3]+=1
 namesCerAdj[7]=countAdj[3]
if self.cerd_7.IsChecked(): # MIPC(d
 atomsAdj+=[12,21,0,14,1]
 countAdj[3]+=1
 namesCerAdj[8]=countAdj[3]
if self.cerd_8.IsChecked(): # M(IP)2C(d
 atomsAdj+=[18,32,0,22,2]
 countAdj[3]+=1
 namesCerAdj[9]=countAdj[3]
if self.cerd_9.IsChecked(): # Sph(d
 atomsAdj+=[0,2,0,0,0]
 countAdj[3]+=1
 namesCerAdj[10]=countAdj[3]
if self.cerd_10.IsChecked(): # HexSph(d
 atomsAdj+=[6,12,0,5,0]
 countAdj[3]+=1
 namesCerAdj[11]=countAdj[3]
if self.cerd_11.IsChecked(): # S1P(d
 atomsAdj+=[0,3,0,3,1]
 countAdj[3]+=1
 namesCerAdj[12]=countAdj[3]
if self.cerd_12.IsChecked(): # Acyl Cer(d
 atomsAcylAdj+=[0,-2,0,2,0]
 countAcylAdj+=1
 namesAcylCerAdj[0]=countAcylAdj
if self.cerd_13.IsChecked(): # Acyl (EOS) Cer(d
 atomsAcylAdj+=[0,-2,0,3,0]
 countAcylAdj+=1
```

```
 namesAcylCerAdj[3]=countAcylAdj
if self.cerd_14.IsChecked(): # Gb3-Cer(d
 atomsAdj+=[18,30,0,16,0]
 countAdj[3]+=1
 namesCerAdj[13]=countAdj[3]
## t
if self.cert_1.IsChecked(): # Cer(t
 atomsAdj+=[0,0,0,2,0]
 countAdj[3]+=1
 namesCerAdj[14]=countAdj[3]
if self.cert_2.IsChecked(): # HexCer(t
 atomsAdj+=[6,10,0,7,0]
 countAdj[3]+=1
 namesCerAdj[15]=countAdj[3]
if self.cert_3.IsChecked(): # LacCer(t
 atomsAdj+=[12,20,0,12,0]
 countAdj[3]+=1
 namesCerAdj[16]=countAdj[3]
if self.cert_4.IsChecked(): # PE-Cer(t
 atomsAdj+=[2,6,1,5,1]
 countAdj[3]+=1
 namesCerAdj[17]=countAdj[3]
if self.cert_5.IsChecked(): # PI-Cer(t
 atomsAdj+=[6,11,0,10,1]
 countAdj[3]+=1
 namesCerAdj[18]=countAdj[3]
if self.cert_6.IsChecked(): # CerP(t
 atomsAdj+=[0,1,0,5,1]
 countAdj[3]+=1
 namesCerAdj[19]=countAdj[3]
if self.cert_7.IsChecked(): # MIPC(t
 atomsAdj+=[12,21,0,15,1]
 countAdj[3]+=1
 namesCerAdj[20]=countAdj[3]
if self.cert_8.IsChecked(): # M(IP)2C(t
 atomsAdj+=[18,32,0,23,2]
 countAdj[3]+=1
 namesCerAdj[21]=countAdj[3]
if self.cert_9.IsChecked(): # Sph(t
 atomsAdj+=[0,2,0,1,0]
 countAdj[3]+=1
 namesCerAdj[22]=countAdj[3]
if self.cert_10.IsChecked(): # HexSph(t
 atomsAdj+=[6,12,0,6,0]
 countAdj[3]+=1
 namesCerAdj[23]=countAdj[3]
if self.cert_11.IsChecked(): # S1P(t
 atomsAdj+=[0,3,0,4,1]
 countAdj[3]+=1
 namesCerAdj[24]=countAdj[3]
if self.cert_12.IsChecked(): # Acyl Cer(t
 atomsAcylAdj+=[0,-2,0,3,0]
 countAcylAdj+=1
 namesAcylCerAdj[1]=countAcylAdj
if self.cert_13.IsChecked(): # Acyl (EOS) Cer(t
 atomsAcylAdj+=[0,-2,0,4,0]
 countAcylAdj+=1
 namesAcylCerAdj[4]=countAcylAdj
if self.cert_14.IsChecked(): # Gb3-Cer(t
 atomsAdj+=[18,30,0,17,0]
 countAdj[3]+=1
 namesCerAdj[25]=countAdj[3]
# chol links
```

```
      if self.chol_1.IsChecked(): #GlcChol checked
       atomsAdj+=[6,10,0,6,0]
       countAdj[4]+=1
       namesCholAdj[0]=countAdj[4]
      if self.chol_2.IsChecked(): # CE
       atomsAdj+=[0,-2,0,1,0]
       countAdj[4]+=1
       namesCholAdj[1]=countAdj[4]
  # error could not retrieve m/z
  except ValueError:
   okInput=0
   self.statusBar.SetStatusText('m/z should be a number, and a period should mark the
decimal place if needed.')
 ################################################
  # settings retrieved
  if okInput==1:
   # initialize results lists
   listOfResultsID=[]
   listOfResultsATOMS=[]
   listOfResultsMASS=[]
   listOfResultsMASS1=[]
   GlcChol=0
   # cycle through all the possible identities with above settings
   # 12 to 60 carbon chains (start at 2 for PC)
   for C in range(0,61): # 2 for PC # 12 for the rest
    # 0 to 12 unsaturations on chains
    for U in range(0,13):
     # for each head in settings
     for H in range(0,countSub+1):
      if namesSub[0]==H:
       currentHead=0
       cumuL=0
      elif namesSub[1]==H:
       currentHead=1
      elif namesSub[2]==H:
       currentHead=2
      elif namesSub[3]==H:
       currentHead=3
      elif namesSub[4]==H:
       currentHead=4
      # for each link in settings
      for L in range(0,countAdj[currentHead]+1):
       cumuL=L # index appropriately depending on the other heads
       if currentHead==1 and countAdj[0]!=-1:
        cumuL+=countAdj[0]+1
       elif currentHead==2:
        if countAdj[0]!=-1:
         cumuL+=countAdj[0]+1
        if countAdj[1]!=-1:
         cumuL+=countAdj[1]+1
       elif currentHead==3:
        if countAdj[0]!=-1:
         cumuL+=countAdj[0]+1
        if countAdj[1]!=-1:
         cumuL+=countAdj[1]+1
        if countAdj[2]!=-1:
         cumuL+=countAdj[2]+1
       elif currentHead==4:
        if countAdj[0]!=-1:
         cumuL+=countAdj[0]+1
        if countAdj[1]!=-1:
         cumuL+=countAdj[1]+1
        if countAdj[2]!=-1:
```

```
            cumuL+=countAdj[2]+1
           if countAdj[3]!=-1:
            cumuL+=countAdj[3]+1
          # for each oxidation in settings
          for O in range(0,countMod+1):
           atomsCurrent=[sum(i) for i in zip(atomsSub[(0+5*H):(5+5*H)],[i*C for i in
atomsChain],[i*U for i in
atomsUnsat],atomsMod[(0+5*O):(5+5*O)],atomsAdj[(0+5*cumuL):(5+5*cumuL)])]
           mzCurrent=(sum([a*b for a,b in zip(massExact,atomsCurrent)])+ionAdd)/ionDiv
           if mzBound1<=mzCurrent and mzCurrent<=mzBound2 :
            alpha='' # prefix of identity
            lyso=0 # binary on lyso
            sph=0 # binary on sphingoid
            chol=0 # binary on cholesterol identity
            if namesSub[0]==H and C>=12: # PS head
             if namesPSAdj[0]==L:
              alpha='PS('
             elif namesPSAdj[1]==L and U<=9 and C<=30:
              lyso=1
              alpha='LPS('
             elif namesPSAdj[2]==L and U<=9 and C<=30:
              lyso=1
              alpha='LPS(O-'
             elif namesPSAdj[3]==L and U<=9 and C<=30:
              lyso=1
              alpha='LPS(P-'
             elif namesPSAdj[4]==L:
              alpha='PS(O-'
             elif namesPSAdj[5]==L:
              alpha='PS(P-'
             elif namesPSAdj[6]==L:
              alpha='dialkyl PS('
            elif namesSub[1]==H and C>=12: # PE head
             if namesPEAdj[0]==L:
              alpha='PE('
             elif namesPEAdj[1]==L and U<=9 and C<=30:
              lyso=1
              alpha='LPE('
             elif namesPEAdj[2]==L and U<=9 and C<=30:
              lyso=1
              alpha='LPE(O-'
             elif namesPEAdj[3]==L and U<=9 and C<=30:
              lyso=1
              alpha='LPE(P-'
             elif namesPEAdj[4]==L:
              alpha='PE(O-'
             elif namesPEAdj[5]==L:
              alpha='PE(P-'
             elif namesPEAdj[6]==L:
              alpha='dialkyl PE('
            elif namesSub[2]==H and C>=2: # PC or SM head
             if namesPCAdj[0]==L and C>=4:
              alpha='PC('
             elif namesPCAdj[1]==L and U<=9 and C<=30:
              lyso=1
              alpha='LPC('
             elif namesPCAdj[2]==L and U<=9 and C<=30:
              lyso=1
              alpha='LPC(O-'
             elif namesPCAdj[3]==L and U<=9 and C<=30:
              lyso=1
              alpha='LPC(P-'
             elif namesPCAdj[4]==L and C>=3:
```

```
    alpha='PC(O-'
  elif namesPCAdj[5]==L and C>=3:
    alpha='PC(P-'
  elif namesPCAdj[6]==L:
    alpha='dialkyl PC('
  elif namesPCAdj[7]==L and U<=3 and C>=12 and C<=30:
    alpha='LysoSM(d'
  elif namesPCAdj[8]==L and U<=3 and C>=12 and C<=30:
    alpha='LysoSM(t'
  elif namesPCAdj[9]==L and U<=3 and C>=12:
    alpha='SM(d'
  elif namesPCAdj[10]==L and U<=3 and C>=12:
    alpha='SM(t'
elif namesSub[3]==H and U<=3 and C>=12: # Cer head
  if namesCerAdj[0]==L:
    alpha='Cer(m'
  elif namesCerAdj[2]==L:
    alpha='Cer(d'
  elif namesCerAdj[14]==L:
    alpha='Cer(t'
  elif namesCerAdj[3]==L:
    alpha='HexCer(d'
  elif namesCerAdj[15]==L:
    alpha='HexCer(t'
  elif namesCerAdj[4]==L:
    alpha='LacCer(d'
  elif namesCerAdj[16]==L:
    alpha='LacCer(t'
  elif namesCerAdj[5]==L:
    alpha='PE-Cer(d'
  elif namesCerAdj[17]==L:
    alpha='PE-Cer(t'
  elif namesCerAdj[6]==L:
    alpha='PI-Cer(d'
  elif namesCerAdj[18]==L:
    alpha='PI-Cer(t'
  elif namesCerAdj[7]==L:
    alpha='CerP(d'
  elif namesCerAdj[19]==L:
    alpha='CerP(t'
  elif namesCerAdj[8]==L:
    alpha='MIPC(d'
  elif namesCerAdj[20]==L:
    alpha='MIPC(t'
  elif namesCerAdj[9]==L:
    alpha='M(IP)2C(d'
  elif namesCerAdj[21]==L:
    alpha='M(IP)2C(t'
  elif namesCerAdj[1]==L:
    sph=1
    alpha='Sph(m'
  elif namesCerAdj[10]==L:
    sph=1
    alpha='Sph(d'
  elif namesCerAdj[22]==L:
    sph=1
    alpha='Sph(t'
  elif namesCerAdj[11]==L:
    sph=1
    alpha='HexSph(d'
  elif namesCerAdj[23]==L:
    sph=1
    alpha='HexSph(t'
```

```
            elif namesCerAdj[12]==L:
              sph=1
              alpha='S1P(d'
            elif namesCerAdj[24]==L:
              sph=1
              alpha='S1P(t'
            elif namesCerAdj[13]==L:
              alpha='Gb3-Cer(d'
            elif namesCerAdj[25]==L:
              alpha='Gb3-Cer(t'
           elif namesSub[4]==H: # chol head
            if namesCholAdj[0]==L and GlcChol==0:
              chol=1
              alpha='GlcChol('
            elif namesCholAdj[1]==L and U<=9 and C>=12:
              chol=2
              alpha='CE('
           if namesMod[0]==O:
            beta=')'
           elif namesMod[1]==O and sph==0 and chol==0:
            beta='(OH))'
           elif namesMod[2]==O and lyso==0 and U!=0 and sph==0 and chol==0:
            beta='(CHO))'
           elif namesMod[3]==O and lyso==0 and U!=0 and sph==0 and chol==0:
            beta='(COOH))'
           else:
            alpha=''
           if alpha=='':
            continue
           else:
            mzExact=round(sum([a*b for a,b in zip(massExact,atomsCurrent)]),4)
            mzIon=round((sum([a*b for a,b in
zip(massExact,atomsCurrent)])+ionAdd)/ionDiv,4)
            if chol==1 and C==0: # GlcChol
             atomsCurrent[1]-=1
            listOfResultsID.append('%s%i:%i%s'%(alpha,C,U,beta))
            atoms='C'
            for i in range(0,5):
             for j in range(0,len(str(atomsCurrent[i]))):
              if str(atomsCurrent[i])[j]=='0':
               atoms+=str("\u2080")
              elif str(atomsCurrent[i])[j]=='1':
               atoms+=str("\u2081")
              elif str(atomsCurrent[i])[j]=='2':
               atoms+=str("\u2082")
              elif str(atomsCurrent[i])[j]=='3':
               atoms+=str("\u2083")
              elif str(atomsCurrent[i])[j]=='4':
               atoms+=str("\u2084")
              elif str(atomsCurrent[i])[j]=='5':
               atoms+=str("\u2085")
              elif str(atomsCurrent[i])[j]=='6':
               atoms+=str("\u2086")
              elif str(atomsCurrent[i])[j]=='7':
               atoms+=str("\u2087")
              elif str(atomsCurrent[i])[j]=='8':
               atoms+=str("\u2088")
              elif str(atomsCurrent[i])[j]=='9':
               atoms+=str("\u2089")
             if 'P' in atoms:
              continue
             elif 'O' in atoms:
              atoms+='P'
```

```
                elif 'N' in atoms:
                 atoms+='O'
                elif 'H' in atoms:
                 atoms+='N'
                elif 'C' in atoms:
                 atoms+='H'
               listOfResultsATOMS.append(atoms)
               listOfResultsMASS.append(mzExact)
               listOfResultsMASS1.append(mzIon)
    # acyls
    if namesSub[3]!=-1 and countAcylAdj!=-1: # if Cer head and acyl selected
     for AC in range(10,39):
      for AU in range(0,10):
       for C in range(12,61):
        for U in range(0,4):
         for L in range(0,countAcylAdj+1):
          for O in range(0,countMod+1):
           atomsCurrent=[sum(i) for i in zip([0,1,1,2,0],[i*(C+AC) for i in
atomsChain],[i*(U+AU) for i in
atomsUnsat],atomsMod[(0+5*O):(5+5*O)],atomsAcylAdj[(0+5*L):(5+5*L)])]
            mzCurrent=(sum([a*b for a,b in
zip(massExact,atomsCurrent)])+ionAdd)/ionDiv
            if mzBound1<=mzCurrent and mzCurrent<=mzBound2 :
             alpha=''
             if namesAcylCerAdj[0]==L:
              alpha='Cer('
              beta='d'
             elif namesAcylCerAdj[1]==L:
              alpha='Cer('
              beta='t'
             elif namesAcylCerAdj[2]==L:
              alpha='Cer(m'
              beta=''
             elif namesAcylCerAdj[3]==L:
              alpha='Cer(d'
              beta=''
             elif namesAcylCerAdj[4]==L:
              alpha='Cer(t'
              beta=''
             if namesMod[0]==O:
              charlie=')'
             elif namesMod[1]==O:
              charlie='(OH))'
             elif namesMod[2]==O and U!=0:
              charlie='(CHO))'
             elif namesMod[3]==O and U!=0:
              charlie='(COOH))'
             else:
              alpha=''
             if alpha=='':
              continue
             else:
              mzExact=round(sum([a*b for a,b in zip(massExact,atomsCurrent)]),4)
              mzIon=round((sum([a*b for a,b in
zip(massExact,atomsCurrent)])+ionAdd)/ionDiv,4)
              if namesAcylCerAdj[0]==L or namesAcylCerAdj[1]==L or
namesAcylCerAdj[2]==L: # N-terminal

  listOfResultsID.append('%s%i:%i;%s%i:%i%s'%(alpha,AC,AU,beta,C,U,charlie))
              else: # Omega end
               listOfResultsID.append('%s%i:%i;%i:%i%s'%(alpha,C,U,AC,AU,charlie))
              atoms='C'
              for i in range(0,5):
```

```
                for j in range(0,len(str(atomsCurrent[i]))):
                 if str(atomsCurrent[i])[j]=='0':
                  atoms+=str("\u2080")
                 elif str(atomsCurrent[i])[j]=='1':
                  atoms+=str("\u2081")
                 elif str(atomsCurrent[i])[j]=='2':
                  atoms+=str("\u2082")
                 elif str(atomsCurrent[i])[j]=='3':
                  atoms+=str("\u2083")
                 elif str(atomsCurrent[i])[j]=='4':
                  atoms+=str("\u2084")
                 elif str(atomsCurrent[i])[j]=='5':
                  atoms+=str("\u2085")
                 elif str(atomsCurrent[i])[j]=='6':
                  atoms+=str("\u2086")
                 elif str(atomsCurrent[i])[j]=='7':
                  atoms+=str("\u2087")
                 elif str(atomsCurrent[i])[j]=='8':
                  atoms+=str("\u2088")
                 elif str(atomsCurrent[i])[j]=='9':
                  atoms+=str("\u2089")
                if 'P' in atoms:
                 continue
                elif 'O' in atoms:
                 atoms+='P'
                elif 'N' in atoms:
                 atoms+='O'
                elif 'H' in atoms:
                 atoms+='N'
                elif 'C' in atoms:
                 atoms+='H'
               listOfResultsATOMS.append(atoms)
               listOfResultsMASS.append(mzExact)
               listOfResultsMASS1.append(mzIon)
   for i in range(len(listOfResultsID)):
    pos=self.Results.InsertItem(0+i,listOfResultsID[i])
    self.Results.SetItem(pos,1,str(listOfResultsATOMS[i]))
    self.Results.SetItem(pos,2,str(listOfResultsMASS[i]))
    self.Results.SetItem(pos,3,str(listOfResultsMASS1[i]))
    self.now=datetime.datetime.now()
   self.statusBar.SetStatusText('')
# When Search ID button clicked --> compute and output result in list
def onSearchRequestID( self, event ):
 self.Results.DeleteAllItems()
 self.mz_search.Clear()
 self.statusBar.SetStatusText('( / ¯^¯)/ Calculating... (ง ^Ꮻ^)ง')
 request=self.id_search.GetValue()
 massExact=[12.000000, 1.007825, 14.003074, 15.994915, 30.973763]
 namesSub=[-1,-1,-1,-1,-1]
 atomsChain=[1,2,0,0,0]
 atomsUnsat=[0,-2,0,0,0]
 atomsDeuter=0
 listOfResultsID=[]
 listOfResultsATOMS=[]
 listOfResultsMASS=[]
 listOfResultsMASS1=[]
 if (self.ionAdduct.GetStringSelection()=='Neutral'):
  ionAdd=0
  ionDiv=1
 elif (self.ionAdduct.GetStringSelection()=='[M+H]+'):
  ionAdd=1.007825
  ionDiv=1
 elif (self.ionAdduct.GetStringSelection()=='[M+H-H2O]+'):
```

```
    ionAdd=-17.002740
    ionDiv=1
   elif (self.ionAdduct.GetStringSelection()=='[M+2H]2+'):
    ionAdd=2.015650
    ionDiv=2
   elif (self.ionAdduct.GetStringSelection()=='[M+3H]3+'):
    ionAdd=3.023475
    ionDiv=3
   elif (self.ionAdduct.GetStringSelection()=='[M+4H]4+'):
    ionAdd=4.031300
    ionDiv=4
   elif (self.ionAdduct.GetStringSelection()=='[M+K]+'):
    ionAdd=38.963708
    ionDiv=1
   elif (self.ionAdduct.GetStringSelection()=='[M+2K]2+'):
    ionAdd=77.927416
    ionDiv=2
   elif (self.ionAdduct.GetStringSelection()=='[M+2K-H]+'):
    ionAdd=76.919591
    ionDiv=1
   elif (self.ionAdduct.GetStringSelection()=='[M+Na]+'):
    ionAdd=22.989770
    ionDiv=1
   elif (self.ionAdduct.GetStringSelection()=='[M+2Na]2+'):
    ionAdd=45.979540
    ionDiv=2
   elif (self.ionAdduct.GetStringSelection()=='[M+2Na-H]+'):
    ionAdd=44.971715
    ionDiv=1
   elif (self.ionAdduct.GetStringSelection()=='[M+Li]+'):
    ionAdd=7.016005
    ionDiv=1
   elif (self.ionAdduct.GetStringSelection()=='[M+2Li]2+'):
    ionAdd=14.032010
    ionDiv=2
   elif (self.ionAdduct.GetStringSelection()=='[M+NH4]+'):
    ionAdd=18.034374
    ionDiv=1
   elif (self.ionAdduct.GetStringSelection()=='[M-H]-'):
    ionAdd=-1.007825
    ionDiv=1
   elif (self.ionAdduct.GetStringSelection()=='[M-2H]2-'):
    ionAdd=-2.015650
    ionDiv=2
   elif (self.ionAdduct.GetStringSelection()=='[M-3H]3-'):
    ionAdd=-3.023475
    ionDiv=3
   elif (self.ionAdduct.GetStringSelection()=='[M-4H]4-'):
    ionAdd=-4.031300
    ionDiv=4
   elif (self.ionAdduct.GetStringSelection()=='[M+Cl]-'):
    ionAdd=34.968853
    ionDiv=1
   elif (self.ionAdduct.GetStringSelection()=='[M+OAc]-'):
    ionAdd=59.013305
    ionDiv=1
   try:
    ## Deuterated standards
    if '-d' in request or '-D' in request:
     requestSplit=re.split('-d|-D',request)
     atomsDeuter=re.split(r'(\d+)',requestSplit[1])[1] #splits on the grouped digits
\d+
     requestSplit[1]=re.sub(atomsDeuter,'',requestSplit[1],count=1)
```

```
     atomsDeuter=int(re.sub('[^0-9]','',atomsDeuter))
     request=requestSplit[0]+requestSplit[1]
    ## Molecular ID .. add up first
    if '/' in request:
     ## AcylCeramide Molecular ID
     if ';' in request:
      requestSplit=re.split('[(:;/)]',request)
      if 'm' in requestSplit[1] or 'd' in requestSplit[1] or 't' in requestSplit[1]: #
omega end
       temp=int(re.sub('[^0-9]','',requestSplit[1]))
       requestSplit[1]=requestSplit[1][0]
       requestSplit[1]+= str(temp + int(requestSplit[3]))
       requestSplit[2]= str(int(requestSplit[2])+int(requestSplit[4]))
       request=requestSplit[0] + '(' + requestSplit[1] + ':' + requestSplit[2] + ';'
+ requestSplit[5] + ':' + requestSplit[6] + '(' + requestSplit[7] + '))'
      elif 'm' in requestSplit[3] or 'd' in requestSplit[3] or 't' in requestSplit[3]:
# n-terminal
       temp=int(re.sub('[^0-9]','',requestSplit[3]))
       requestSplit[3]=requestSplit[3][0]
       requestSplit[3]+= str(temp + int(requestSplit[5]))
       requestSplit[4]= str(int(requestSplit[4])+int(requestSplit[6]))
       request=requestSplit[0] + '(' + requestSplit[1] + ':' + requestSplit[2] + ';'
+ requestSplit[3] + ':' + requestSplit[4] + '(' + requestSplit[7] + '))'
     # Molecular ID
     else:
      requestSplit=re.split('[(:/)]',request) # split ( : / )
      tmp=''
      if 'IP' in requestSplit[1]: # M(IP)2C
       requestSplit[0]='M(IP)2C'
       requestSplit[1]=requestSplit[3]
       requestSplit[2]=requestSplit[4]
       requestSplit[3]=requestSplit[5]
       requestSplit[4]=requestSplit[6]
       requestSplit[5]=requestSplit[7]
      if '-' in requestSplit[1]: # O- P-
       if 'O' in requestSplit[1]:
        tmp='O-'
        requestSplit[1]=re.split('-',requestSplit[1])[1] # split -, keep carbons
       elif 'P' in requestSplit[1]:
        tmp='P-'
        requestSplit[1]=re.split('-',requestSplit[1])[1] # split -, keep carbons
      if int(requestSplit[3])==0 and int(requestSplit[4])==0: # lyso!
       if 'LP' not in requestSplit[0] and 'Lyso' not in requestSplit[0]: # not a lyso
already
        if 'SM' in requestSplit[0]:
         requestSplit[0]='Lyso'+requestSplit[0] # lyso status added
        elif 'PS' in requestSplit[0] or 'PE' in requestSplit[0] or 'PC' in
requestSplit[0]:
         requestSplit[0]='L'+requestSplit[0] # lyso status added
      if 'm' in requestSplit[1] or 'd' in requestSplit[1] or 't' in requestSplit[1]: #
sphingoid base
       temp=int(re.sub('[^0-9]','',requestSplit[1]))
       requestSplit[1]=requestSplit[1][0]
       requestSplit[1]+= str(temp + int(requestSplit[3]))
       requestSplit[2]= str(int(requestSplit[2])+int(requestSplit[4]))
       request=requestSplit[0] + '(' + requestSplit[1] + ':' + requestSplit[2] + '('
+ requestSplit[5] + '))'
      else:
       requestSplit[1]= str(int(requestSplit[1]) + int(requestSplit[3]))
       requestSplit[2]= str(int(requestSplit[2])+int(requestSplit[4]))
       request=requestSplit[0] + '(' + tmp + requestSplit[1] + ':' + requestSplit[2]
+ '(' + requestSplit[5] + '))'
    ## AcylCeramide
```

```
    if ';' in request:
     requestSplit=re.split('[(:;)]',request)
     if 'Cer' in requestSplit[0]:
      okInput=1
      atomsSub=[0,1,1,2,0]
     else:
      okInput=0
      self.statusBar.SetStatusText('I do not recognize this identity!')
     if okInput==1:
      if 'd' in requestSplit[3]:
       nterm=1
       atomsAcylAdj=[0,-2,0,2,0]
       alpha='Cer('
       beta='d'
      elif 't' in requestSplit[3]:
       nterm=1
       atomsAcylAdj=[0,-2,0,3,0]
       alpha='Cer('
       beta='t'
      elif 'm' in requestSplit[1]:
       nterm=0
       atomsAcylAdj=[0,-2,0,2,0]
       alpha='Cer(m'
       beta=''
      elif 'd' in requestSplit[1]:
       nterm=0
       atomsAcylAdj=[0,-2,0,3,0]
       alpha='Cer(d'
       beta=''
      elif 't' in requestSplit[1]:
       nterm=0
       atomsAcylAdj=[0,-2,0,4,0]
       alpha='Cer(t'
       beta=''
      else:
       okInput=0
       self.statusBar.SetStatusText('Ceramide identities require a [m] [d] or [t] in
their name. For example, Sph(d18:1) is okay, but LIT will not recognize Sph(18:1).')
      ## Carbons and Unsaturations
      if okInput==1:
       AC=int(re.sub('[^0-9]','',requestSplit[1]))
       C=int(re.sub('[^0-9]','',requestSplit[3]))
       U=int(requestSplit[4])
       AU=int(requestSplit[2])
       ## Oxidation and Hydroxylation
       if 'COOH' in requestSplit:
        if U!=0:
         atomsMod=[0,-2,0,2,0]
         charlie='(COOH))'
        else:
         okInput=0
         self.statusBar.SetStatusText('A saturated, lyso, or Sph lipid species cannot
be oxydized.')
       elif 'CHO' in requestSplit:
        if U!=0:
         atomsMod=[0,-2,0,1,0]
         charlie='(CHO))'
        else:
         okInput=0
         self.statusBar.SetStatusText('A saturated, lyso, or Sph lipid species cannot
be oxydized.')
       elif 'OH' in requestSplit:
        atomsMod=[0,0,0,1,0]
```

```
          charlie='(OH))'
         else:
          atomsMod=[0,0,0,0,0]
          charlie=')'
        ## Expected mz
        if okInput==1:
         atomsCurrent=[sum(i) for i in zip(atomsSub,[i*(C+AC) for i in
atomsChain],[i*(U+AU) for i in atomsUnsat],atomsMod,atomsAcylAdj)]
         mzExact=round(sum([a*b for a,b in
zip(massExact,atomsCurrent)])+1.006277*atomsDeuter,4)
         mzIon=round((sum([a*b for a,b in
zip(massExact,atomsCurrent)])+1.006277*atomsDeuter+ionAdd)/ionDiv,4)
         if nterm==1: # N-terminal
          listOfResultsID.append('%s%i:%i;%s%i:%i%s'%(alpha,AC,AU,beta,C,U,charlie))
         else: # Omega end
          listOfResultsID.append('%s%i:%i;%i:%i%s'%(alpha,AC,AU,C,U,charlie))
         atoms='C'
         for i in range(0,5):
          for j in range(0,len(str(atomsCurrent[i]))):
           if str(atomsCurrent[i])[j]=='0':
            atoms+=str("\u2080")
           elif str(atomsCurrent[i])[j]=='1':
            atoms+=str("\u2081")
           elif str(atomsCurrent[i])[j]=='2':
            atoms+=str("\u2082")
           elif str(atomsCurrent[i])[j]=='3':
            atoms+=str("\u2083")
           elif str(atomsCurrent[i])[j]=='4':
            atoms+=str("\u2084")
           elif str(atomsCurrent[i])[j]=='5':
            atoms+=str("\u2085")
           elif str(atomsCurrent[i])[j]=='6':
            atoms+=str("\u2086")
           elif str(atomsCurrent[i])[j]=='7':
            atoms+=str("\u2087")
           elif str(atomsCurrent[i])[j]=='8':
            atoms+=str("\u2088")
           elif str(atomsCurrent[i])[j]=='9':
            atoms+=str("\u2089")
          if 'P' in atoms:
           continue
          elif 'O' in atoms:
           atoms+='P'
          elif 'N' in atoms:
           atoms+='O'
          elif 'H' in atoms:
           atoms+='N'
          elif 'C' in atoms:
           atoms+='H'
         listOfResultsATOMS.append(atoms)
         listOfResultsMASS.append(mzExact)
         listOfResultsMASS1.append(mzIon)
    # Not AcylCeramide
    else:
     requestSplit=re.split('[(:)]',request) # split ( : and )
     #sub and adj values
     if 'PS' in requestSplit[0]: ## PS
      okInput=1
      atomsSub=[6,10,1,10,1]
      sub='PS'
     elif ('PC' in requestSplit[0] or 'SM' in requestSplit[0]) and not 'MIPC' in
requestSplit: ## PC/SM
       okInput=1
```

```
       if 'SM' in requestSplit[0]:
        if 'd' not in requestSplit[1] and 't' not in requestSplit[1]:
         okInput=0
         self.statusBar.SetStatusText('SM identities require a [d] or [t] in their
name. For example, SM(d46:2) is okay, but LIT will not recognize SM(46:2).')
       atomsSub=[8,16,1,8,1]
       sub='PC/SM'
      elif 'GlcChol' in requestSplit[0]: # GlcChol
       okInput=1
       atomsSub=[27,46,0,1,0]
       sub='GlcChol'
      elif 'CE' in requestSplit[0]: # CE
       okInput=1
       atomsSub=[27,46,0,1,0]
       sub='CE'
      elif 'Cer' in requestSplit[0] or 'MIPC' in requestSplit[0] or 'Sph' in
requestSplit[0] or 'S1P' in requestSplit[0] or 'IP' in requestSplit[1]: ## Cer
       if 'IP' in requestSplit[1]: ## M(IP)2C
        requestSplit=requestSplit[2:len(requestSplit)]
        requestSplit[0]='M(IP)2C'
       if 'm' in requestSplit[1] or 'd' in requestSplit[1] or 't' in requestSplit[1]:
        okInput=1
       else:
        okInput=0
        self.statusBar.SetStatusText('Ceramide identities require a [m] [d] or [t] in
their name. For example, Sph(d18:1) is okay, but LIT will not recognize Sph(18:1).')
       atomsSub=[0,1,1,2,0]
       sub='Cer'
      elif 'PE' in requestSplit[0]: ## PE
       okInput=1
       atomsSub=[5,10,1,8,1]
       sub='PE'
      else:
       okInput=0
       self.statusBar.SetStatusText('I do not recognize this identity.')
      if okInput==1:
       ## Linkage
       lyso=0
       sph=0
       chol=0
       if sub=='PC/SM' or sub=='PS' or sub=='PE':
        if 'DIALKYL' in requestSplit[0].upper(): ## dialkyl, link 7 in PC/SM
         atomsAdj=[0,4,0,-2,0]
         if sub=='PC/SM':
          alpha='dialkyl PC('
         elif sub=='PS':
          alpha='dialkyl PS('
         elif sub=='PE':
          alpha='dialkyl PE('
        elif requestSplit[0][0]=='L':
         lyso=1
         if 'O-' in requestSplit[1]:
          atomsAdj=[0,4,0,-2,0]
          if sub=='PC/SM':
           alpha='LPC(O-'
          elif sub=='PS':
           alpha='LPS(O-'
          elif sub=='PE':
           alpha='LPE(O-'
         elif 'P-' in requestSplit[1]:
          atomsAdj=[0,2,0,-2,0]
          if sub=='PC/SM':
           alpha='LPC(P-'
```

```
          elif sub=='PS':
           alpha='LPS(P-'
          elif sub=='PE':
           alpha='LPE(P-'
         elif 'Lyso' in requestSplit[0]:
          if sub=='PC/SM' and 'd' in requestSplit[1]:
           atomsAdj=[-3,-1,1,-3,0]
           alpha='LysoSM(d'
          elif sub=='PC/SM' and 't' in requestSplit[1]:
           atomsAdj=[-3,-1,1,-4,0]
           alpha='LysoSM(t'
         else:
          atomsAdj=[0,2,0,-1,0]
          if sub=='PC/SM':
           alpha='LPC('
          elif sub=='PS':
           alpha='LPS('
          elif sub=='PE':
           alpha='LPE('
        elif 'O-' in requestSplit[1]:
         atomsAdj=[0,2,0,-1,0]
         if sub=='PC/SM':
          alpha='PC(O-'
         elif sub=='PS':
          alpha='PS(O-'
         elif sub=='PE':
          alpha='PE(O-'
        elif 'P-' in requestSplit[1]:
         atomsAdj=[0,0,0,-1,0]
         if sub=='PC/SM':
          alpha='PC(P-'
         elif sub=='PS':
          alpha='PS(P-'
         elif sub=='PE':
          alpha='PE(P-'
        elif 'SM' in requestSplit[0] and 'd' in requestSplit[1]:
         atomsAdj=[-3,-3,1,-2,0]
         alpha='SM(d'
        elif 'SM' in requestSplit[0] and 't' in requestSplit[1]:
         atomsAdj=[-3,-3,1,-3,0]
         alpha='SM(t'
        else:
         atomsAdj=[0,0,0,0,0]
         if sub=='PC/SM':
          alpha='PC('
         elif sub=='PS':
          alpha='PS('
         elif sub=='PE':
          alpha='PE('
       elif sub=='Cer':
        if requestSplit[0]=='HexCer' or requestSplit[0]=='GalCer' or
requestSplit[0]=='GlcCer':
         if 'm' in requestSplit[1]:
          okInput=0
          self.statusBar.SetStatusText('This ceramide cannot have a monohydroxylated
[m] backbone.')
         elif 'd' in requestSplit[1]:
          atomsAdj=[6,10,0,6,0]
          alpha='HexCer(d'
         elif 't' in requestSplit[1]:
          atomsAdj=[6,10,0,7,0]
          alpha='HexCer(t'
        elif requestSplit[0]=='LacCer':
```

```
          if 'm' in requestSplit[1]:
           okInput=0
           self.statusBar.SetStatusText('This ceramide cannot have a monohydroxylated
[m] backbone.')
          elif 'd' in requestSplit[1]:
           atomsAdj=[12,20,0,11,0]
           alpha='LacCer(d'
          elif 't' in requestSplit[1]:
           atomsAdj=[12,20,0,12,0]
           alpha='LacCer(t'
         elif requestSplit[0]=='PE-Cer':
          if 'm' in requestSplit[1]:
           okInput=0
           self.statusBar.SetStatusText('This ceramide cannot have a monohydroxylated
[m] backbone.')
          elif 'd' in requestSplit[1]:
           atomsAdj=[2,6,1,4,1]
           alpha='PE-Cer(d'
          elif 't' in requestSplit[1]:
           atomsAdj=[2,6,1,5,1]
           alpha='PE-Cer(t'
         elif requestSplit[0]=='PI-Cer':
          if 'm' in requestSplit[1]:
           okInput=0
           self.statusBar.SetStatusText('This ceramide cannot have a monohydroxylated
[m] backbone.')
          elif 'd' in requestSplit[1]:
           atomsAdj=[6,11,0,9,1]
           alpha='PI-Cer(d'
          elif 't' in requestSplit[1]:
           atomsAdj=[6,11,0,10,1]
           alpha='PI-Cer(t'
         elif requestSplit[0]=='CerP':
          if 'm' in requestSplit[1]:
           okInput=0
           self.statusBar.SetStatusText('This ceramide cannot have a monohydroxylated
[m] backbone.')
          elif 'd' in requestSplit[1]:
           atomsAdj=[0,1,0,4,1]
           alpha='CerP(d'
          elif 't' in requestSplit[1]:
           atomsAdj=[0,1,0,5,1]
           alpha='CerP(t'
         elif requestSplit[0]=='MIPC':
          if 'm' in requestSplit[1]:
           okInput=0
           self.statusBar.SetStatusText('This ceramide cannot have a monohydroxylated
[m] backbone.')
          elif 'd' in requestSplit[1]:
           atomsAdj=[12,21,0,14,1]
           alpha='MIPC(d'
          elif 't' in requestSplit[1]:
           atomsAdj=[12,21,0,15,1]
           alpha='MIPC(t'
         elif requestSplit[0]=='M(IP)2C':
          if 'm' in requestSplit[1]:
           okInput=0
           self.statusBar.SetStatusText('This ceramide cannot have a monohydroxylated
[m] backbone.')
          elif 'd' in requestSplit[1]:
           atomsAdj=[18,32,0,22,2]
           alpha='M(IP)2C(d'
          elif 't' in requestSplit[1]:
```

```
          atomsAdj=[18,32,0,23,2]
          alpha='M(IP)2C(t'
        elif requestSplit[0]=='Sph':
         sph=1
         if 'm' in requestSplit[1]:
          atomsAdj=[0,2,0,-1,0]
          alpha='Sph(m'
         elif 'd' in requestSplit[1]:
          atomsAdj=[0,2,0,0,0]
          alpha='Sph(d'
         elif 't' in requestSplit[1]:
          atomsAdj=[0,2,0,1,0]
          alpha='Sph(t'
        elif requestSplit[0]=='HexSph':
         sph=1
         if 'm' in requestSplit[1]:
          okInput=0
          self.statusBar.SetStatusText('This ceramide cannot have a monohydroxylated
[m] backbone.')
         elif 'd' in requestSplit[1]:
          atomsAdj=[6,12,0,5,0]
          alpha='HexSph(d'
         elif 't' in requestSplit[1]:
          atomsAdj=[6,12,0,6,0]
          alpha='HexSph(t'
        elif requestSplit[0]=='S1P':
         sph=1
         if 'm' in requestSplit[1]:
          okInput=0
          self.statusBar.SetStatusText('This ceramide cannot have a monohydroxylated
[m] backbone.')
         elif 'd' in requestSplit[1]:
          atomsAdj=[0,3,0,3,1]
          alpha='S1P(d'
         elif 't' in requestSplit[1]:
          atomsAdj=[0,3,0,4,1]
          alpha='S1P(t'
        elif requestSplit[0]=='Gb3-Cer':
         if 'm' in requestSplit[1]:
          okInput=0
          self.statusBar.SetStatusText('This ceramide cannot have a monohydroxylated
[m] backbone.')
         elif 'd' in requestSplit[1]:
          atomsAdj=[18,30,0,16,0]
          alpha='Gb3-Cer(d'
         elif 't' in requestSplit[1]:
          atomsAdj=[18,30,0,17,0]
          alpha='Gb3-Cer(t'
        else:
         if 'm' in requestSplit[1]:
          atomsAdj=[0,0,0,0,0]
          alpha='Cer(m'
         elif 'd' in requestSplit[1]:
          atomsAdj=[0,0,0,1,0]
          alpha='Cer(d'
         elif 't' in requestSplit[1]:
          atomsAdj=[0,0,0,2,0]
          alpha='Cer(t'
       elif sub=='GlcChol':
        chol=1
        atomsAdj=[6,10,0,6,0]
        alpha='GlcChol('
       elif sub=='CE':
```

```
        chol=2
        atomsAdj=[0,-2,0,1,0]
        alpha='CE('
       ## Carbons and Unsaturations
       C=int(re.sub('[^0-9]','',requestSplit[1]))
       U=int(requestSplit[2])
       ## Oxidation and Hydroxylation
       if 'COOH' in requestSplit:
        if U!=0 and lyso==0 and sph==0 and chol==0:
         atomsMod=[0,-2,0,2,0]
         beta='(COOH))'
        else:
         okInput=0
         self.statusBar.SetStatusText('A saturated, lyso, Sph, or Chol lipid species
cannot be oxydized.')
       elif 'CHO' in requestSplit:
        if U!=0 and lyso==0 and sph==0 and chol==0:
         atomsMod=[0,-2,0,1,0]
         beta='(COOH))'
        else:
         okInput=0
         self.statusBar.SetStatusText('A saturated, lyso, Sph, or Chol lipid species
cannot be oxydized.')
       elif 'OH' in requestSplit:
        if sph==0 and chol==0:
         atomsMod=[0,0,0,1,0]
         beta='(OH))'
        else:
         okInput=0
         self.statusBar.SetStatusText('A Sph or Chol lipid species cannot be
hydroxylated')
       else:
        atomsMod=[0,0,0,0,0]
        beta=')'
       ## Expected mz
       if okInput==1:
        atomsCurrent=[sum(i) for i in zip(atomsSub,[i*(C) for i in atomsChain],[i*(U)
for i in atomsUnsat],atomsMod,atomsAdj)]
        mzExact=round(sum([a*b for a,b in
zip(massExact,atomsCurrent)])+1.006277*atomsDeuter,4)
        mzIon=round((sum([a*b for a,b in
zip(massExact,atomsCurrent)])+1.006277*atomsDeuter+ionAdd)/ionDiv,4)
        if chol==1 and C==0:
         atomsCurrent[1]-=1
        if atomsDeuter!=0: ## add -d after U
         listOfResultsID.append('%s%i:%i-d%i%s'%(alpha,C,U,atomsDeuter,beta))
        else:
         listOfResultsID.append('%s%i:%i%s'%(alpha,C,U,beta))
        atoms='C'
        atomsCurrent[1]-=atomsDeuter
        for i in range(0,5):
         for j in range(0,len(str(atomsCurrent[i]))):
          if str(atomsCurrent[i])[j]=='0':
           atoms+=str("\u2080")
          elif str(atomsCurrent[i])[j]=='1':
           atoms+=str("\u2081")
          elif str(atomsCurrent[i])[j]=='2':
           atoms+=str("\u2082")
          elif str(atomsCurrent[i])[j]=='3':
           atoms+=str("\u2083")
          elif str(atomsCurrent[i])[j]=='4':
           atoms+=str("\u2084")
          elif str(atomsCurrent[i])[j]=='5':
```

```
            atoms+=str("\u2085")
           elif str(atomsCurrent[i])[j]=='6':
            atoms+=str("\u2086")
           elif str(atomsCurrent[i])[j]=='7':
            atoms+=str("\u2087")
           elif str(atomsCurrent[i])[j]=='8':
            atoms+=str("\u2088")
           elif str(atomsCurrent[i])[j]=='9':
            atoms+=str("\u2089")
          if 'P' in atoms:
           if atomsDeuter!=0:
            atoms+='D'
            for j in range(0,len(str(atomsDeuter))):
             if str(atomsDeuter)[j]=='0':
              atoms+=str("\u2080")
             elif str(atomsDeuter)[j]=='1':
              atoms+=str("\u2081")
             elif str(atomsDeuter)[j]=='2':
              atoms+=str("\u2082")
             elif str(atomsDeuter)[j]=='3':
              atoms+=str("\u2083")
             elif str(atomsDeuter)[j]=='4':
              atoms+=str("\u2084")
             elif str(atomsDeuter)[j]=='5':
              atoms+=str("\u2085")
             elif str(atomsDeuter)[j]=='6':
              atoms+=str("\u2086")
             elif str(atomsDeuter)[j]=='7':
              atoms+=str("\u2087")
             elif str(atomsDeuter)[j]=='8':
              atoms+=str("\u2088")
             elif str(atomsDeuter)[j]=='9':
              atoms+=str("\u2089")
           else:
            continue
          elif 'O' in atoms:
           atoms+='P'
          elif 'N' in atoms:
           atoms+='O'
          elif 'H' in atoms:
           atoms+='N'
          elif 'C' in atoms:
           atoms+='H'
         listOfResultsATOMS.append(atoms)
         listOfResultsMASS.append(mzExact)
         listOfResultsMASS1.append(mzIon)
     for i in range(len(listOfResultsID)):
      pos=self.Results.InsertItem(0+i,listOfResultsID[i])
      self.Results.SetItem(pos,1,str(listOfResultsATOMS[i]))
      self.Results.SetItem(pos,2,str(listOfResultsMASS[i]))
      self.Results.SetItem(pos,3,str(listOfResultsMASS1[i]))
      self.now=datetime.datetime.now()
     if okInput==1:
      self.statusBar.SetStatusText('')
   except (IndexError, ValueError):
    self.statusBar.SetStatusText('I do not recognize this identity!')
   ################################################
   event.Skip()
  #(de)select all PS
  def onselect_PS( self, event ):
   if self.ps_1.IsChecked() or self.ps_2.IsChecked() or self.ps_3.IsChecked() or
self.ps_4.IsChecked() or self.ps_5.IsChecked() or self.ps_6.IsChecked() or
self.ps_7.IsChecked():
```

```
    self.ps_1.SetValue(False)
    self.ps_2.SetValue(False)
    self.ps_3.SetValue(False)
    self.ps_4.SetValue(False)
    self.ps_5.SetValue(False)
    self.ps_6.SetValue(False)
    self.ps_7.SetValue(False)
   else:
    self.ps_1.SetValue(True)
    self.ps_2.SetValue(True)
    self.ps_3.SetValue(True)
    self.ps_4.SetValue(True)
    self.ps_5.SetValue(True)
    self.ps_6.SetValue(True)
    self.ps_7.SetValue(True)
   event.Skip()
  #(de)select all PE
  def onselect_PE( self, event ):
   if self.pe_1.IsChecked() or self.pe_2.IsChecked() or self.pe_3.IsChecked() or
self.pe_4.IsChecked() or self.pe_5.IsChecked() or self.pe_6.IsChecked() or
self.pe_7.IsChecked():
    self.pe_1.SetValue(False)
    self.pe_2.SetValue(False)
    self.pe_3.SetValue(False)
    self.pe_4.SetValue(False)
    self.pe_5.SetValue(False)
    self.pe_6.SetValue(False)
    self.pe_7.SetValue(False)
   else:
    self.pe_1.SetValue(True)
    self.pe_2.SetValue(True)
    self.pe_3.SetValue(True)
    self.pe_4.SetValue(True)
    self.pe_5.SetValue(True)
    self.pe_6.SetValue(True)
    self.pe_7.SetValue(True)
   event.Skip()
  #(de)select all PC
  def onselect_PC( self, event ):
   if self.pc_1.IsChecked() or self.pc_2.IsChecked() or self.pc_3.IsChecked() or
self.pc_4.IsChecked() or self.pc_5.IsChecked() or self.pc_6.IsChecked() or
self.pc_7.IsChecked():
    self.pc_1.SetValue(False)
    self.pc_2.SetValue(False)
    self.pc_3.SetValue(False)
    self.pc_4.SetValue(False)
    self.pc_5.SetValue(False)
    self.pc_6.SetValue(False)
    self.pc_7.SetValue(False)
   else:
    self.pc_1.SetValue(True)
    self.pc_2.SetValue(True)
    self.pc_3.SetValue(True)
    self.pc_4.SetValue(True)
    self.pc_5.SetValue(True)
    self.pc_6.SetValue(True)
    self.pc_7.SetValue(True)
   event.Skip()
  #(de)select all SM
  def onselect_SM( self, event ):
   if self.sm_1.IsChecked() or self.sm_2.IsChecked() or self.sm_3.IsChecked() or
self.sm_4.IsChecked():
    self.sm_1.SetValue(False)
```

```
    self.sm_2.SetValue(False)
    self.sm_3.SetValue(False)
    self.sm_4.SetValue(False)
   else:
    self.sm_1.SetValue(True)
    self.sm_2.SetValue(True)
    self.sm_3.SetValue(True)
    self.sm_4.SetValue(True)
   event.Skip()
  #(de)select all mCer
  def onselect_mCer( self, event ):
   if self.cerm_1.IsChecked() or self.cerm_2.IsChecked() or self.cerm_3.IsChecked() or
self.cerm_4.IsChecked():
    self.cerm_1.SetValue(False)
    self.cerm_2.SetValue(False)
    self.cerm_3.SetValue(False)
   else:
    self.cerm_1.SetValue(True)
    self.cerm_2.SetValue(True)
    self.cerm_3.SetValue(True)
   event.Skip()
  #(de)select all dCer
  def onselect_dCer( self, event ):
   if self.cerd_1.IsChecked() or self.cerd_2.IsChecked() or self.cerd_3.IsChecked() or
self.cerd_4.IsChecked() or self.cerd_5.IsChecked() or self.cerd_6.IsChecked() or
self.cerd_7.IsChecked() or self.cerd_8.IsChecked() or self.cerd_9.IsChecked() or
self.cerd_10.IsChecked() or self.cerd_11.IsChecked() or self.cerd_12.IsChecked() or
self.cerd_13.IsChecked() or self.cerd_14.IsChecked():
    self.cerd_1.SetValue(False)
    self.cerd_2.SetValue(False)
    self.cerd_3.SetValue(False)
    self.cerd_4.SetValue(False)
    self.cerd_5.SetValue(False)
    self.cerd_6.SetValue(False)
    self.cerd_7.SetValue(False)
    self.cerd_8.SetValue(False)
    self.cerd_9.SetValue(False)
    self.cerd_10.SetValue(False)
    self.cerd_11.SetValue(False)
    self.cerd_12.SetValue(False)
    self.cerd_13.SetValue(False)
    self.cerd_14.SetValue(False)
   else:
    self.cerd_1.SetValue(True)
    self.cerd_2.SetValue(True)
    self.cerd_3.SetValue(True)
    self.cerd_4.SetValue(True)
    self.cerd_5.SetValue(True)
    self.cerd_6.SetValue(True)
    self.cerd_7.SetValue(True)
    self.cerd_8.SetValue(True)
    self.cerd_9.SetValue(True)
    self.cerd_10.SetValue(True)
    self.cerd_11.SetValue(True)
    self.cerd_12.SetValue(True)
    self.cerd_13.SetValue(True)
    self.cerd_14.SetValue(True)
   event.Skip()
  #(de)select all tCer
  def onselect_tCer( self, event ):
   if self.cert_1.IsChecked() or self.cert_2.IsChecked() or self.cert_3.IsChecked() or
self.cert_4.IsChecked() or self.cert_5.IsChecked() or self.cert_6.IsChecked() or
self.cert_7.IsChecked() or self.cert_8.IsChecked() or self.cert_9.IsChecked() or
```

```
self.cert_10.IsChecked() or self.cert_11.IsChecked() or self.cert_12.IsChecked() or
self.cert_13.IsChecked() or self.cert_14.IsChecked():
    self.cert_1.SetValue(False)
    self.cert_2.SetValue(False)
    self.cert_3.SetValue(False)
    self.cert_4.SetValue(False)
    self.cert_5.SetValue(False)
    self.cert_6.SetValue(False)
    self.cert_7.SetValue(False)
    self.cert_8.SetValue(False)
    self.cert_9.SetValue(False)
    self.cert_10.SetValue(False)
    self.cert_11.SetValue(False)
    self.cert_12.SetValue(False)
    self.cert_13.SetValue(False)
    self.cert_14.SetValue(False)
  else:
    self.cert_1.SetValue(True)
    self.cert_2.SetValue(True)
    self.cert_3.SetValue(True)
    self.cert_4.SetValue(True)
    self.cert_5.SetValue(True)
    self.cert_6.SetValue(True)
    self.cert_7.SetValue(True)
    self.cert_8.SetValue(True)
    self.cert_9.SetValue(True)
    self.cert_10.SetValue(True)
    self.cert_11.SetValue(True)
    self.cert_12.SetValue(True)
    self.cert_13.SetValue(True)
    self.cert_14.SetValue(True)
  event.Skip()
 #(de)select all chol
 def onselect_chol( self, event ):
  if self.chol_1.IsChecked() or self.chol_2.IsChecked():
    self.chol_1.SetValue(False)
    self.chol_2.SetValue(False)
  else:
    self.chol_1.SetValue(True)
    self.chol_2.SetValue(True)
  event.Skip()
 # Results sorting
 def onColClick( self, event ):
  colIndex=event.GetColumn()
  self.statusBar.SetStatusText('Sorting...')
  alpha=[]
  beta=[]
  for i in range(0,self.Results.GetItemCount()):
    for j in range(i+1,self.Results.GetItemCount()):
      alpha0=self.Results.GetItemText(i,0)
      alpha1=self.Results.GetItemText(i,1)
      alpha2=self.Results.GetItemText(i,2)
      alpha3=self.Results.GetItemText(i,3)
      beta0=self.Results.GetItemText(j,0)
      beta1=self.Results.GetItemText(j,1)
      beta2=self.Results.GetItemText(j,2)
      beta3=self.Results.GetItemText(j,3)
      alpha=[alpha0,alpha1,alpha2,alpha3]
      beta=[beta0,beta1,beta2,beta3]
      if alpha[colIndex]>beta[colIndex]:
        self.Results.SetItem(i,0,beta[0])
        self.Results.SetItem(j,0,alpha[0])
        self.Results.SetItem(i,1,beta[1])
```

```
        self.Results.SetItem(j,1,alpha[1])
        self.Results.SetItem(i,2,beta[2])
        self.Results.SetItem(j,2,alpha[2])
        self.Results.SetItem(i,3,beta[3])
        self.Results.SetItem(j,3,alpha[3])
    self.statusBar.SetStatusText('')
    event.Skip()
  # Opens ID window
  def onopenID( self, event ):
    id = identitiesFrame(parent=self)
    id.Show()
  # Opens export dialog
  def onopenExport(self, event):
    self.statusBar.SetStatusText('Exporting... \(*¤* )⌋')
    saveFileDialog = wx.FileDialog(self, "Save As", "", "","Excel workbook
(*.xlsx)|*.xlsx",wx.FD_SAVE | wx.FD_OVERWRITE_PROMPT)
    saveFileDialog.ShowModal()
    try:
     path=str(saveFileDialog.GetPath())
     workbook = xlsxwriter.Workbook(path)
     worksheet = workbook.add_worksheet()
     bold=workbook.add_format({'bold':True})
     num=workbook.add_format({'num_format': '##0'})
     worksheet.set_column('A:A', 25)
     worksheet.set_column('B:B', 20)
     worksheet.set_column('C:C', 15)
     worksheet.set_column('D:D', 15)
     col = 0
     worksheet.write(0, 0, 'Results produced by LITv1.13 Windows OS',    bold)
     worksheet.write(1, 0,  self.now.strftime("%Y/%m/%d %H:%M"))
     if self.mz_search.IsEmpty(): # if mz empty, id search was conducted, no need to
output settings:
      worksheet.write(3, 0, 'requested id', bold)
      worksheet.write(3, 1, str(self.id_search.GetValue()))
      row=6
     else:
      worksheet.write(3, 0, 'Settings', bold)
      oxidation=''
      if self.NoOxy.IsChecked(): # No oxidation
       if oxidation!='':
        oxidation+=', '
       oxidation+='No oxidation'
      if self.OH.IsChecked(): # OH
       if oxidation!='':
        oxidation+=', '
       oxidation+='OH hydroxylation'
      if self.CHO.IsChecked(): # CHO
       if oxidation!='':
        oxidation+=', '
       oxidation+='CHO hydroxylation'
      if self.COOH.IsChecked(): # COOH
       if oxidation!='':
        oxidation+=', '
       oxidation+='COOH hydroxylation'
      inPS=''
      if self.ps_1.IsChecked():
       if inPS!='':
        inPS+=', '
       inPS+='PS(x:y)'
      if self.ps_2.IsChecked():
       if inPS!='':
        inPS+=', '
       inPS+='LPS(x:y)'
```

```
if self.ps_3.IsChecked():
 if inPS!='':
  inPS+=', '
 inPS+='LPS(O-x:y)'
if self.ps_4.IsChecked():
 if inPS!='':
  inPS+=', '
 inPS+='LPS(P-x:y)'
if self.ps_5.IsChecked():
 if inPS!='':
  inPS+=', '
 inPS+='PS(O-x:y)'
if self.ps_6.IsChecked():
 if inPS!='':
  inPS+=', '
 inPS+='PS(P-x:y)'
if self.ps_7.IsChecked():
 if inPS!='':
  inPS+=', '
 inPS+='dialkylPS(x:y)'
inPE=''
if self.pe_1.IsChecked():
 if inPE!='':
  inPE+=', '
 inPE+='PE(x:y)'
if self.pe_2.IsChecked():
 if inPE!='':
  inPE+=', '
 inPE+='LPE(x:y)'
if self.pe_3.IsChecked():
 if inPE!='':
  inPE+=', '
 inPE+='LPE(O-x:y)'
if self.pe_4.IsChecked():
 if inPE!='':
  inPE+=', '
 inPE+='LPE(P-x:y)'
if self.pe_5.IsChecked():
 if inPE!='':
  inPE+=', '
 inPE+='PE(O-x:y)'
if self.pe_6.IsChecked():
 if inPE!='':
  inPE+=', '
 inPE+='PE(P-x:y)'
if self.pe_7.IsChecked():
 if inPE!='':
  inPE+=', '
 inPE+='dialkylPE(x:y)'
inPC=''
if self.pc_1.IsChecked():
 if inPC!='':
  inPC+=', '
 inPC+='PC(x:y)'
if self.pc_2.IsChecked():
 if inPC!='':
  inPC+=', '
 inPC+='LPC(x:y)'
if self.pc_3.IsChecked():
 if inPC!='':
  inPC+=', '
 inPC+='LPC(O-x:y)'
if self.pc_4.IsChecked():
```

```
 if inPC!='':
  inPC+=', '
 inPC+='LPC(P-x:y)'
if self.pc_5.IsChecked():
 if inPC!='':
  inPC+=', '
 inPC+='PC(O-x:y)'
if self.pc_6.IsChecked():
 if inPC!='':
  inPC+=', '
 inPC+='PC(P-x:y)'
if self.pc_7.IsChecked():
 if inPC!='':
  inPC+=', '
 inPC+='dialkylPC(x:y)'
inSM=''
if self.sm_1.IsChecked():
 if inSM!='':
  inSM+=', '
 inSM+='LysoSM(dx:y)'
if self.sm_2.IsChecked():
 if inSM!='':
  inSM+=', '
 inSM+='LysoSM(tx:y)'
if self.sm_3.IsChecked():
 if inSM!='':
  inSM+=', '
 inSM+='SM(dx:y)'
if self.sm_4.IsChecked():
 if inSM!='':
  inSM+=', '
 inSM+='SM(tx:y)'
inCerm=''
if self.cerm_1.IsChecked():
 if inCerm!='':
  inCerm+=', '
 inCerm+='Cer(mx:y)'
if self.cerm_2.IsChecked():
 if inCerm!='':
  inCerm+=', '
 inCerm+='Sph(mx:y)'
if self.cerm_3.IsChecked():
 if inCerm!='':
  inCerm+=', '
 inCerm+='Cer(mx:y;a:b)'
inCerd=''
if self.cerm_1.IsChecked():
 if inCerd!='':
  inCerd+=', '
 inCerd+='Cer(dx:y)'
if self.cerd_2.IsChecked():
 if inCerd!='':
  inCerd+=', '
 inCerd+='HexCer(dx:y)'
if self.cerd_3.IsChecked():
 if inCerd!='':
  inCerd+=', '
 inCerd+='LacCer(dx:y)'
if self.cerd_4.IsChecked():
 if inCerd!='':
  inCerd+=', '
 inCerd+='PE-Cer(dx:y)'
if self.cerd_5.IsChecked():
```

```
  if inCerd!='':
   inCerd+=', '
  inCerd+='PI-Cer(dx:y)'
 if self.cerd_6.IsChecked():
  if inCerd!='':
   inCerd+=', '
  inCerd+='CerP(dx:y)'
 if self.cerd_7.IsChecked():
  if inCerd!='':
   inCerd+=', '
  inCerd+='MIPC(dx:y)'
 if self.cerd_8.IsChecked():
  if inCerd!='':
   inCerd+=', '
  inCerd+='M(IP)2C(dx:y)'
 if self.cerd_9.IsChecked():
  if inCerd!='':
   inCerd+=', '
  inCerd+='Sph(dx:y)'
 if self.cerd_10.IsChecked():
  if inCerd!='':
   inCerd+=', '
  inCerd+='HexSph(dx:y)'
 if self.cerd_11.IsChecked():
  if inCerd!='':
   inCerd+=', '
  inCerd+='S1P(dx:y)'
 if self.cerd_12.IsChecked():
  if inCerd!='':
   inCerd+=', '
  inCerd+='Cer(a:b;dx:y)'
 if self.cerd_13.IsChecked():
  if inCerd!='':
   inCerd+=', '
  inCerd+='Cer(dx:y;a:b)'
 if self.cerd_14.IsChecked():
  if inCerd!='':
   inCerd+=', '
  inCerd+='Gb3-Cer(dx:y)'
 inCert=''
 if self.cert_1.IsChecked():
  if inCert!='':
   inCert+=', '
  inCert+='Cer(tx:y)'
 if self.cert_2.IsChecked():
  if inCert!='':
   inCert+=', '
  inCert+='HexCer(tx:y)'
 if self.cert_3.IsChecked():
  if inCert!='':
   inCert+=', '
  inCert+='LacCer(tx:y)'
 if self.cert_4.IsChecked():
  if inCert!='':
   inCert+=', '
  inCert+='PE-Cer(tx:y)'
 if self.cert_5.IsChecked():
  if inCert!='':
   inCert+=', '
  inCert+='PI-Cer(tx:y)'
 if self.cert_6.IsChecked():
  if inCert!='':
   inCert+=', '
```

```
 inCert+='CerP(tx:y)'
if self.cert_7.IsChecked():
 if inCert!='':
  inCert+=', '
 inCert+='MIPC(tx:y)'
if self.cert_8.IsChecked():
 if inCert!='':
  inCert+=', '
 inCert+='M(IP)2C(tx:y)'
if self.cert_9.IsChecked():
 if inCert!='':
  inCert+=', '
 inCert+='Sph(tx:y)'
if self.cert_10.IsChecked():
 if inCert!='':
  inCert+=', '
 inCert+='HexSph(tx:y)'
if self.cert_11.IsChecked():
 if inCert!='':
  inCert+=', '
 inCert+='S1P(tx:y)'
if self.cert_12.IsChecked():
 if inCert!='':
  inCert+=', '
 inCert+='Cer(a:b;tx:y)'
if self.cert_13.IsChecked():
 if inCert!='':
  inCert+=', '
 inCert+='Cer(tx:y;a:b)'
if self.cert_14.IsChecked():
 if inCert!='':
  inCert+=', '
 inCert+='Gb3-Cer(tx:y)'
inChol=''
if self.chol_1.IsChecked():
 if inChol!='':
  inChol+=', '
 inChol+='GlcChol(x:y)'
if self.chol_2.IsChecked():
 if inChol!='':
  inChol+=', '
 inChol+='CE(x:y)'
worksheet.write(4, 0, 'PS species included',bold)
worksheet.write(4, 1, inPS)
worksheet.write(5, 0, 'PE species included',bold)
worksheet.write(5, 1, inPE)
worksheet.write(6, 0, 'PC species included',bold)
worksheet.write(6, 1, inPC)
worksheet.write(7, 0,   'SM species included', bold)
worksheet.write(7, 1, inSM)
worksheet.write(8, 0,   'Cer(m) species included',    bold)
worksheet.write(8, 1, inCerm)
worksheet.write(9, 0,   'Cer(d) species included',    bold)
worksheet.write(9, 1, inCerd)
worksheet.write(10, 0,'Cer(t) species included',     bold)
worksheet.write(10, 1,inCert)
worksheet.write(11, 0,  'chol species included',     bold)
worksheet.write(11, 1,inChol)
worksheet.write(12, 0,'Oxidation and hydroxylation', bold)
worksheet.write(12, 1,oxidation)
worksheet.write(13, 0,'requested m/z', bold)
worksheet.write(13, 1,self.mz_search.GetValue())
if (self.accuracy.GetStringSelection()=='± 0.01'):
```

```
   worksheet.write(13, 2,     '± 0.01')
  elif (self.accuracy.GetStringSelection()=='± 0.05'):
   worksheet.write(13, 2,     '± 0.05')
  elif (self.accuracy.GetStringSelection()=='± 0.1'):
   worksheet.write(13, 2,     '± 0.1')
  elif (self.accuracy.GetStringSelection()=='± 0.5'):
   worksheet.write(13, 2,     '± 0.5')
  elif (self.accuracy.GetStringSelection()=='± 1'):
   worksheet.write(13, 2,     '± 1')
  row=17
worksheet.write(row-2, 0, 'Ion adduct',bold)
if (self.ionAdduct.GetStringSelection()=='Neutral'):
  worksheet.write(row-2, 1,     'Neutral')
elif (self.ionAdduct.GetStringSelection()=='[M+H]+'):
  worksheet.write(row-2, 1,     '[M+H]+')
elif (self.ionAdduct.GetStringSelection()=='[M+H-H2O]+'):
  worksheet.write(row-2, 1,     '[M+H-H2O]+')
elif (self.ionAdduct.GetStringSelection()=='[M+2H]2+'):
  worksheet.write(row-2, 1,     '[M+2H]2+')
elif (self.ionAdduct.GetStringSelection()=='[M+3H]3+'):
  worksheet.write(row-2, 1,     '[M+3H]3+')
elif (self.ionAdduct.GetStringSelection()=='[M+4H]4+'):
  worksheet.write(row-2, 1,     '[M+4H]4+')
elif (self.ionAdduct.GetStringSelection()=='[M+K]+'):
  worksheet.write(row-2, 1,     '[M+K]+')
elif (self.ionAdduct.GetStringSelection()=='[M+2K]2+'):
  worksheet.write(row-2, 1,     '[M+2K]2+')
elif (self.ionAdduct.GetStringSelection()=='[M+2K-H]+'):
  worksheet.write(row-2, 1,     '[M+2K-H]+')
elif (self.ionAdduct.GetStringSelection()=='[M+Na]+'):
  worksheet.write(row-2, 1,     '[M+Na]+')
elif (self.ionAdduct.GetStringSelection()=='[M+2Na]2+'):
  worksheet.write(row-2, 1,     '[M+2Na]2+')
elif (self.ionAdduct.GetStringSelection()=='[M+2Na-H]+'):
  worksheet.write(row-2, 1,     '[M+2Na-H]+')
elif (self.ionAdduct.GetStringSelection()=='[M+Li]+'):
  worksheet.write(row-2, 1,     '[M+Li]+')
elif (self.ionAdduct.GetStringSelection()=='[M+2Li]2+'):
  worksheet.write(row-2, 1,     '[M+2Li]2+')
elif (self.ionAdduct.GetStringSelection()=='[M+NH4]+'):
  worksheet.write(row-2, 1,     '[M+NH4]+')
elif (self.ionAdduct.GetStringSelection()=='[M-H]-'):
  worksheet.write(row-2, 1,     '[M-H]-')
elif (self.ionAdduct.GetStringSelection()=='[M-2H]2-'):
  worksheet.write(row-2, 1,     '[M-2H]2-')
elif (self.ionAdduct.GetStringSelection()=='[M-3H]3-'):
  worksheet.write(row-2, 1,     '[M-3H]3-')
elif (self.ionAdduct.GetStringSelection()=='[M-4H]4-'):
  worksheet.write(row-2, 1,     '[M-4H]4-')
elif (self.ionAdduct.GetStringSelection()=='[M+Cl]-'):
  worksheet.write(row-2, 1,     '[M+Cl]-')
elif (self.ionAdduct.GetStringSelection()=='[M+OAc]-'):
  worksheet.write(row-2, 1,     '[M+OAc]-')
worksheet.write(row, 0,     'ID', bold)
worksheet.write(row, 1,     'Chemical Formula',bold)
worksheet.write(row, 2,     'Exact Mass', bold)
worksheet.write(row, 3,     'Mass-over-charge',bold)
for i in range(self.Results.GetItemCount()):
  row+=1
  worksheet.write(row, col,     self.Results.GetItemText(i,0))
  worksheet.write(row, col + 1, self.Results.GetItemText(i,1))
  worksheet.write(row, col + 2, self.Results.GetItemText(i,2),num)
  worksheet.write(row, col + 3, self.Results.GetItemText(i,3),num)
```

```
    workbook.close()
    self.statusBar.SetStatusText('')
   except:
    self.statusBar.SetStatusText('')
   saveFileDialog.Destroy()
   event.Skip()
  # Opens notes window
  def onopenNotes( self, event ):
   notes = notesFrame(parent=self)
   notes.Show()
  # Quits
  def onquitButton( self, event ):
   self.Destroy()
app = wx.App(False)
frame = mainFrame(None)
frame.Show(True)
#start the applications
app.MainLoop()
```

Appendix C: LITL source code

The following section is the source code of LITLv1.3 (matlab language). For aesthetic purposes, the font of the appendix B has been changed to a smaller size Courrier New font, and the indentations were changed to 0.1" in order to respect the margins of this document. Required underlying functions written for this thesis are included at the end of the script.

```
function varargout = LITL_v1_3(varargin)
% LITL_v1_3 MATLAB code for LITL_v1_3.fig
% Last updated August 2018
% Developped by Julie Fiala
% Inquiries: jfial097@uottawa.ca
%
%      LITL_v1_3, by itself, creates a new LITL window instance (LITL_v1_3.fig)
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
    'gui_Singleton',  gui_Singleton, ...
    'gui_OpeningFcn', @LITL_v1_3_OpeningFcn, ...
    'gui_OutputFcn',  @LITL_v1_3_OutputFcn, ...
    'gui_LayoutFcn',  [] , ...
    'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
% Executes just before LITL_v1_3 is made visible
function LITL_v1_3_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to LITL_v1_3 (see VARARGIN)
% Default handles and settings
handles.output = hObject;
handles.mode_library=0;
handles.mode_expdata=0;
handles.settings_covariance=0;
handles.settings_smooth=0;
handles.settings_ssrt=0;
handles.settings_quant=0;
handles.settings_prism=0;
handles.settings_librarysave=0;
handles.settings_lines=0;
handles.settings_corr=0.98;
handles.settings_dataset=0;
handles.settings_seed = 42;
handles.settings_ja=0;
handles.file_library='';
handles.file_expdata='';
handles.file_sampleinfo='';
handles.file_folder='';
handles.file_foldername='LITL_Output';
% Update handles structure
guidata(hObject, handles);
% Outputs returning to the command line
function varargout = LITL_v1_3_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Get default command line output from handles structure
varargout{1} = handles.output;
function path_library_Callback(hObject, eventdata, handles)
% Executes during object creation, after setting all properties
```

```
function path_library_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
% Executes on button press (browse_library)
function browse_library_Callback(hObject, eventdata, handles)
[FileName,Path] = uigetfile({'*.mat';'*.xlsx'},'Please select the library to use.');
set(handles.path_library,'String',strcat(Path,FileName));
handles.file_library=strcat(Path,FileName);
guidata(hObject,handles);
function path_expdata_Callback(hObject, eventdata, handles)
% Executes during object creation, after setting all properties
function path_expdata_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
% Executes on button press (browse_expdata)
function browse_expdata_Callback(hObject, eventdata, handles)
[FileName,Path] = uigetfile('*.csv','Please select your MultiQuant export.');
set(handles.path_expdata,'String',strcat(Path,FileName))
handles.file_expdata=strcat(Path,FileName);
guidata(hObject,handles);
function path_sampleinfo_Callback(hObject, eventdata, handles)
% Executes during object creation, after setting all properties
function path_sampleinfo_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
% Executes on button press (browse_sampleinfo)
function browse_sampleinfo_Callback(hObject, eventdata, handles)
[FileName,Path] = uigetfile('*.csv','Please select your sample information file.');
set(handles.path_sampleinfo,'String',strcat(Path,FileName))
handles.file_sampleinfo=strcat(Path,FileName);
guidata(hObject,handles);
function path_outputfolder_Callback(hObject, eventdata, handles)
% Executes during object creation, after setting all properties
function path_outputfolder_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
% Executes on button press (browse_folderpath)
function browse_folderpath_Callback(hObject, eventdata, handles)
Path = uigetdir('','Where should I create the output folder?');
set(handles.path_outputfolder,'String',Path)
handles.file_folder=strcat(Path);
guidata(hObject,handles);
% Executes on text editing (edit_outputfoldername)
function edit_outputfoldername_Callback(hObject, eventdata, handles)
handles.file_foldername=get(hObject,'String');
guidata(hObject,handles);
% Executes during object creation, after setting all properties
function edit_outputfoldername_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
% Executes on button press (checkbox_covariance)
function checkbox_covariance_Callback(hObject, eventdata, handles)
handles.settings_covariance = get(hObject,'Value');
```

```
guidata(hObject, handles)
% Executes on button press (checkbox_expdata)
function checkbox_expdata_Callback(hObject, eventdata, handles)
handles.mode_expdata = get(hObject,'Value');
if handles.mode_library==1
    set(handles.checkbox_library, 'Value', abs(handles.mode_expdata-1))
    handles.mode_library=abs(handles.mode_expdata-1);
end
if handles.mode_expdata==1 % hide library only settings
    set(handles.checkbox_quantification,'Visible','on')
    set(handles.checkbox_prism,'Visible','on')
    set(handles.path_expdata,'Visible','on')
    set(handles.browse_expdata,'Visible','on')
    set(handles.path_sampleinfo,'Visible','on')
    set(handles.browse_sampleinfo,'Visible','on')
    set(handles.text_expdata,'Visible','on')
    set(handles.text_sampleinfo,'Visible','on')
    set(handles.text_datasets,'Visible','off')
    set(handles.edit_dataset,'Visible','off')
    set(handles.text_seed,'Visible','off')
    set(handles.edit_seed,'Visible','off')
    set(handles.edit_dataset,'Visible','off')
    set(handles.text_assignment,'Visible','off')
    set(handles.popupmenu_assignment,'Visible','off')
elseif handles.mode_expdata==0 % show all settings
%     set(handles.checkbox_quantification,'Visible','off')
%     set(handles.checkbox_prism,'Visible','off')
%     set(handles.path_expdata,'Visible','off')
%     set(handles.browse_expdata,'Visible','off')
%     set(handles.path_sampleinfo,'Visible','off')
%     set(handles.browse_sampleinfo,'Visible','off')
%     set(handles.text_expdata,'Visible','off')
%     set(handles.text_sampleinfo,'Visible','off')
    set(handles.text_datasets,'Visible','on')
    set(handles.edit_dataset,'Visible','on')
    set(handles.text_seed,'Visible','on')
    set(handles.edit_seed,'Visible','on')
    set(handles.edit_dataset,'Visible','on')
    set(handles.text_assignment,'Visible','on')
    set(handles.popupmenu_assignment,'Visible','on')
end
guidata(hObject, handles)
% Executes on button press (checkbox_library)
function checkbox_library_Callback(hObject, eventdata, handles)
handles.mode_library = get(hObject,'Value');
if handles.mode_expdata==1
    set(handles.checkbox_expdata, 'Value', abs(handles.mode_library-1))
    handles.mode_expdata=abs(handles.mode_library-1);
end
if handles.mode_library==1 % hide expdata only settings
    set(handles.checkbox_quantification,'Visible','off')
    set(handles.checkbox_prism,'Visible','off')
    set(handles.path_expdata,'Visible','off')
    set(handles.browse_expdata,'Visible','off')
    set(handles.path_sampleinfo,'Visible','off')
    set(handles.browse_sampleinfo,'Visible','off')
    set(handles.text_expdata,'Visible','off')
    set(handles.text_sampleinfo,'Visible','off')
    set(handles.text_datasets,'Visible','on')
    set(handles.edit_dataset,'Visible','on')
    set(handles.text_seed,'Visible','on')
    set(handles.edit_seed,'Visible','on')
    set(handles.edit_dataset,'Visible','on')
```

```
    set(handles.text_assignment,'Visible','on')
    set(handles.popupmenu_assignment,'Visible','on')
elseif handles.mode_library==0 % show all settings
    set(handles.checkbox_quantification,'Visible','on')
    set(handles.checkbox_prism,'Visible','on')
    set(handles.path_expdata,'Visible','on')
    set(handles.browse_expdata,'Visible','on')
    set(handles.path_sampleinfo,'Visible','on')
    set(handles.browse_sampleinfo,'Visible','on')
    set(handles.text_expdata,'Visible','on')
    set(handles.text_sampleinfo,'Visible','on')
%     set(handles.text_datasets,'Visible','off')
%     set(handles.edit_dataset,'Visible','off')
%     set(handles.text_seed,'Visible','off')
%     set(handles.edit_seed,'Visible','off')
%     set(handles.edit_dataset,'Visible','off')
%     set(handles.text_assignment,'Visible','off')
%     set(handles.popupmenu_assignment,'Visible','off')
end
guidata(hObject, handles)
% Executes on button press (checkbox_smoothing)
function checkbox_smoothing_Callback(hObject, eventdata, handles)
handles.settings_smooth = get(hObject,'Value');
guidata(hObject, handles)
% Executes on button press (checkbox_ssrt)
function checkbox_ssrt_Callback(hObject, eventdata, handles)
handles.settings_ssrt = get(hObject,'Value');
guidata(hObject, handles)
% Executes on button press (checkbox_quantification)
function checkbox_quantification_Callback(hObject, eventdata, handles)
handles.settings_quant = get(hObject,'Value');
if handles.settings_quant==0;
    set(handles.checkbox_prism, 'Value', 0)
    handles.settings_prism=0;
end
guidata(hObject, handles)
% Executes on button press (checkbox_prism)
function checkbox_prism_Callback(hObject, eventdata, handles)
handles.settings_prism = get(hObject,'Value');
if handles.settings_prism==1;
    set(handles.checkbox_quantification, 'Value', 1)
    handles.settings_quant=1;
end
guidata(hObject, handles)
% Executes on button press (checkbox_librarysave)
function checkbox_librarysave_Callback(hObject, eventdata, handles)
handles.settings_librarysave = get(hObject,'Value');
guidata(hObject, handles)
% Executes on button press (checkbox_lines)
function checkbox_lines_Callback(hObject, eventdata, handles)
handles.settings_lines = get(hObject,'Value');
guidata(hObject, handles)
% Executes on text editing (edit_corr)
function edit_corr_Callback(hObject, eventdata, handles)
corr = str2double(get(hObject,'String'));
if isnan(corr) || corr>1 || corr<0
    corr = 0.98;
    set(hObject,'String',corr);
    errordlg('The correlation threshold must be between 0 and 1.', 'Error')
end
handles.settings_corr = corr;
guidata(hObject,handles)
% Executes during object creation, after setting all properties
```

```
function edit_corr_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
% Executes on text editing (edit_dataset)
function edit_dataset_Callback(hObject, eventdata, handles)
dataset = str2double(get(hObject,'String'));
if isnan(dataset) || dataset<0 || mod(dataset,1) ~= 0
    dataset = 0;
    set(hObject,'String',dataset);
    errordlg('The number of datasets must be a positive integer.', 'Error')
end
handles.settings_dataset = dataset;
guidata(hObject,handles)
% Executes during object creation, after setting all properties
function edit_dataset_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
% Executes on text editing (edit_seed)
function edit_seed_Callback(hObject, eventdata, handles)
seed = str2double(get(hObject,'String'));
if isnan(seed)
    seed = 42;
    set(hObject,'String',seed);
    errordlg('The random seed must be a number', 'Error')
end
handles.settings_seed = seed;
guidata(hObject,handles)
% Executes during object creation, after setting all properties
function edit_seed_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
% Executes on selection change (popupmenu_assignment)
function popupmenu_assignment_Callback(hObject, eventdata, handles)
switch get(handles.popupmenu_assignment,'Value')
    case 1
        handles.settings_ja=0;
    case 2
        handles.settings_ja=1;
    case 3
        handles.settings_ja=2;
    otherwise
end
guidata(hObject, handles);
% Executes during object creation, after setting all properties
function popupmenu_assignment_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
% Executes on button press (pushbutton_analyze)
function pushbutton_analyze_Callback(hObject, eventdata, handles)
timestart=clock;
% dialog_box=msgbox('LITL is now importing your data.','Status','replace');
progress=waitbar(0,'LITL is now importing your data.');
progress_prop=findobj(progress,'Type','Patch');
set(progress_prop,'EdgeColor',[0 1 1],'FaceColor',[0 1 0.5]) % changes the color to
green
```

```
try
    if handles.mode_library==1
        % Upload .xlsx library file to split and test
        % split: library (half) + expData (half)
        % **PLEASE MAKE SURE**
        % - "Master sheet" is the second sheet starting from the left
        % - "Compilation sheet" is the fourth sheet starting from the left
        % - For the "split run" with a "best tweak" setting, the A column of
        % the "Compilation sheet" needs to contain the proper correct
        % assignment index
        % - the term "mz" is present TWICE in the B column of the "Compilation
        % sheet", and that it is located in the row IMMEDIATELY BEFORE the data
        % starts
        % - There are no empty transition (species with no recorded retention
        % time across all samples)
        [~,~,ext] = fileparts(handles.file_library);
        if strcmp(ext, '.xlsx')~=1
            delete(progress);
            error('This library is not compatible with LITL.');
        end
        try
            [~,~,library.table.compilation]=xlsread(handles.file_library,4);
            [~,~,library.table.master]=xlsread(handles.file_library,2);
            ind=find(strcmp(library.table.compilation(:,2),'mz'));
            ind=ind(2); % index of row delimitation for standards in library
            waitbar(0.05,progress);
            % Variable initialization
            doubleblanks=0;
            expData={};
            sampleDataset=1;
            sampleName={};
            sampleLib={};
            testSample=2+handles.settings_dataset;
            trainSample=0;
            testLib=0;
            dataset=2;
            library.rt=[];
            library.type=[];
            library.st.rt=[];
            library.st.mean=[];
            library.st.sd=[];
            % going through every column of compilation sheet and splitting it
            % tmp: index of the columns in the compilation sheet
            for tmp=1:size(library.table.compilation,2)
                if doubleblanks==2 && ~isnan(library.table.compilation{1,tmp})
                    n=library.table.compilation{1,tmp}; % number of samples
                    sampleDataset=[sampleDataset sampleDataset(end)]; % per dataset
                    rng(handles.settings_seed);
                    ntest{tmp}=tmp+2+randperm(n,floor(n/2));
                    for sample=tmp+3:tmp+2+n % all samples in current dataset
                        if sum(ntest{tmp} == sample) == 1 % sample in test
                            testSample=testSample+1;
                            testTransition=0;

sampleName{testSample,1}=library.table.compilation{ind,sample};

expDataST{testSample}=horzcat(library.table.compilation{2:ind-1,sample});
                            for transition=ind+1:size(library.table.compilation,1) %
transitions
                                if
isempty(library.table.compilation{transition,sample})==0
                                    if
isnan(library.table.compilation{transition,sample})==0
```

```
                                    testTransition=testTransition+1;

expData{testSample,1}(testTransition,1)=library.table.compilation{transition,1}; % X

expData{testSample,1}(testTransition,2)=library.table.compilation{transition,2}; % m/z

expData{testSample,1}(testTransition,3)=library.table.compilation{transition,sample};
% rt

expData{testSample,2}{testTransition,1}=library.table.master{transition,12}; % barcode
                                end
                            end
                        end
                    else % sample in train
                        sampleDataset(end)=sampleDataset(end)+1;
                        testLib=testLib+1;

sampleLib{testLib,1}=library.table.compilation{ind,sample};

library.rt=horzcat(library.rt,[library.table.compilation(ind+1:end,sample)]);

library.type=horzcat(library.type,library.table.compilation{1,tmp+1});

library.st.rt=horzcat(library.st.rt,[library.table.compilation(2:ind-1,sample)]);
                    end
                end
                doubleblanks=0;
            end
            if isnan(library.table.compilation{2,tmp})
                doubleblanks=doubleblanks+1;
            else
                doubleblanks=0;
            end
        end
        % library formatting
        % standards
        emptyIndex = cellfun(@isempty,library.st.rt); %indices of empty cells
        library.st.rt(emptyIndex) = {NaN};
        library.st.index=cellfun(@(c) ischar(c) && ~isempty(strfind(c,
'Standard')), library.table.master(:,11));
        library.st.index=find(library.st.index==1);
        library.st.index=library.st.index(library.st.index>ind)-ind;
        % tmp: index of the different standards
        for tmp=1:size(library.st.rt,2)
            library.st.mean(1,tmp)=mean([library.st.rt{:,tmp}]);
            library.st.sd(1,tmp)=std([library.st.rt{:,tmp}]);
        end
        % rt, barcode, mz
        emptyIndex = cellfun(@isempty,library.rt); %indices of empty cells
        library.rt(emptyIndex) = {NaN};
        library.barcode=[];
        library.genID=[];
        library.mz=[];
        % tmp: index of the species
        for tmp=ind+1:size(library.table.compilation,1)
            library.barcode=vertcat(library.barcode,library.table.master(tmp,12));
            library.genID=vertcat(library.genID,library.table.master(tmp,10));

library.mz=vertcat(library.mz,floor(library.table.master{tmp,19})+0.5);
        end
        % tmp: index of the rows in library.rt
        % tmp2: index of the columns in library.rt
        for tmp=size(library.rt,1):-1:1
```

```
                for tmp2=size(library.rt,2):-1:1
                    if isempty(library.rt{tmp,tmp2})
                        library.rt{tmp,tmp2}=NaN;
                    end
                end
                if all(isnan([library.rt{tmp,:}]))
                    library.rt(tmp,:)=[];
                    library.barcode(tmp,:)=[];
                    library.genID(tmp,:)=[];
                    library.mz(tmp,:)=[];
                end
            end
            % expdata formatting
            % averages non weighted
            sampleName{2,1}='library average non weighted';
            expDataST{1,2}=transpose(nanmean(cell2mat(library.st.rt),2));
            expData{2,1}(:,3)=nanmean(cell2mat(library.rt),2); % rt

expData{2,1}(:,1)=cell2mat(library.table.compilation(ind+1:size(library.table.compilat
ion,1),1)); % X

expData{2,1}(:,2)=cell2mat(library.table.compilation(ind+1:size(library.table.compilat
ion,1),2)); % m/z

expData{2,2}=transpose({library.table.master{ind+1:size(library.table.compilation,1),1
2}}); % barcode
            if handles.settings_dataset~=0
                % standards, rt, mz, assignment index, barcode
                % tmp: index of the different datasets
                for tmp=3:size(sampleDataset,2)+1

expDataST{1,tmp}=transpose(nanmean(cell2mat(library.st.rt(:,sampleDataset(tmp-
2):sampleDataset(tmp-1)-1)),2));

expData{tmp,1}(:,3)=nanmean(cell2mat(library.rt(:,sampleDataset(tmp-
2):sampleDataset(tmp-1)-1)),2); % rt

expData{tmp,1}(:,1)=cell2mat(library.table.compilation(ind+1:size(library.table.compil
ation,1),1)); % X

expData{tmp,1}(:,2)=cell2mat(library.table.compilation(ind+1:size(library.table.compil
ation,1),2)); % m/z

expData{tmp,2}=transpose({library.table.master{ind+1:size(library.table.compilation,1)
,12}}); % barcode
                    % clear out the blanks
                    for tmp2=size(expData{tmp,1},1):-1:1
                        if isnan(expData{tmp,1}(tmp2,3)) %blank
                            expData{tmp,1}(tmp2,:)=[];
                            expData{tmp,2}(tmp2,:)=[];
                        end
                    end
                end
            end
            library.rt=cell2mat(library.rt);
            % Weighted library averages
            % tmp: temporary variable for expDataST

[library.mean,library.diag]=MissingDataMeanAndVar(library.type,library.rt);
            [tmp,~]=MissingDataMeanAndVar(library.type,cell2mat(library.st.rt));
            library.va=diag(library.diag);
            % weighted average in expData
            expData{1,1}=expData{2,1};
```

```
            expData{1,2}=expData{2,2};
            testTransition=0;
            sampleName{1,1}='library average weighted';
            expDataST{1,1}=transpose(tmp);
            for transition=1:size(library.mean,1) % transitions
                if isempty(library.mean(transition,1))==0
                    if isnan(library.mean(transition,1))==0
                        testTransition=testTransition+1;
                        expData{1,1}(testTransition,3)=library.mean(transition,1); %
rt
                    end
                end
            end
            % Loess smoothing
            if handles.settings_smooth==1
                waitbar(0.05,progress,'Smoothing data...'); %
dialog_box=msgbox('Smoothing data...','Status','replace');
                [rawExpData,library,expData]=LoessSmoothing(library,expData,0.1);
                % library.st fix
                library.st.rt=library.rt(library.st.index,:);
                % tmp: index of the different standards
                for tmp=1:size(library.st.rt,2)
                    library.st.mean(1,tmp)=mean([library.st.rt(:,tmp)]);
                    library.st.sd(1,tmp)=std([library.st.rt(:,tmp)]);
                end
                % library.mean fix

[library.mean,library.diag]=MissingDataMeanAndVar(library.type,library.rt);
                library.va=diag(library.diag);
                % weighted average in expData fix
                % tmp: temporary variable for expDataST
                [tmp,~]=MissingDataMeanAndVar(library.type,library.st.rt);
                expData{1,1}=expData{2,1};
                expData{1,2}=expData{2,2};
                testTransition=0;
                sampleName{1,1}='library average weighted';
                expDataST{1,1}=transpose(tmp);
                for transition=1:size(library.mean,1) % transitions
                    if isempty(library.mean(transition,1))==0
                        if isnan(library.mean(transition,1))==0
                            testTransition=testTransition+1;
                            expData{1,1}(testTransition,3)=library.mean(transition,1);
% rt
                        end
                    end
                end
                % averages non weighted
                expDataST{1,2}=transpose(nanmean(library.st.rt,2));
                expData{2,1}(:,3)=nanmean(library.rt,2); % rt

expData{2,1}(:,1)=cell2mat(library.table.compilation(ind+1:size(library.table.compilat
ion,1),1)); % X

expData{2,1}(:,2)=cell2mat(library.table.compilation(ind+1:size(library.table.compilat
ion,1),2)); % m/z

expData{2,2}=transpose({library.table.master{ind+1:size(library.table.compilation,1),1
2}}); % barcode
                if handles.settings_dataset~=0
                    % tmp: index of the different datasets
                    for tmp=3:size(sampleDataset,2)+1
                        expData{tmp,1}=[];
                        expData{tmp,2}=[];
```

```
expDataST{1,tmp}=transpose(nanmean(library.st.rt(:,sampleDataset(tmp-
2):sampleDataset(tmp-1)-1),2));
                        expData{tmp,1}(:,3)=nanmean(library.rt(:,sampleDataset(tmp-
2):sampleDataset(tmp-1)-1),2); % rt

expData{tmp,1}(:,1)=cell2mat(library.table.compilation(ind+1:size(library.table.compil
ation,1),1)); % X

expData{tmp,1}(:,2)=cell2mat(library.table.compilation(ind+1:size(library.table.compil
ation,1),2)); % m/z

expData{tmp,2}=transpose({library.table.master{ind+1:size(library.table.compilation,1)
,12}}); % barcode
                        % clear out the blanks
                        for tmp2=size(expData{tmp,1},1):-1:1
                            if isnan(expData{tmp,1}(tmp2,3)) %blank
                                expData{tmp,1}(tmp2,:)=[];
                                expData{tmp,2}(tmp2,:)=[];
                            end
                        end
                    end
                end
            end
            % SSRT
            if handles.settings_ssrt==1 % ssrt
                waitbar(0.1,progress,'Standardizing and scaling data...'); %
dialog_box=msgbox('Standardizing and scaling data...','Status','replace');
                % standardized and scaled retention times
                % SSRT = (RT - mean(standard RTs)) / sd(standard RTs)
                library.ssrt=library.rt;
                for tmpx=1:size(library.mz,1)
                    for tmpy=1:size(library.rt,2)
                        if isnan(library.rt(tmpx,tmpy))
                            library.ssrt(tmpx,tmpy)=NaN;
                        else
                            library.ssrt(tmpx,tmpy)=( library.rt(tmpx,tmpy) -
library.st.mean(1,tmpy) ) / library.st.sd(1,tmpy);
                        end
                    end
                end
                % tmps: index of the sample in expData
                for tmps=1:size(expData,1)
                    standardMean=mean(expDataST{tmps});
                    standardSD=std(expDataST{tmps});
                    % tmp: index of the different species in a sample tmps
                    for tmp=1:size(expData{tmps,1},1)
                        % SSRT = (RT - mean(standard rts)) / std(standard rts)
                        expData{tmps,1}(tmp,4)= (expData{tmps,1}(tmp,3) - standardMean
) / standardSD;
                    end
                end

[library.ssmean,library.ssdiag]=MissingDataMeanAndVar(library.type,library.ssrt);
                library.ssva=diag(library.ssdiag);
            end
            % Covariance of train library
            waitbar(0.15,progress,'Computing the library covariance...'); %
dialog_box=msgbox('Computing the library covariance...','Status','replace');
            if handles.settings_covariance==0
                library.cov=eye(size(library.rt,1));
                if handles.settings_ssrt==1 % ssrt to be computed
                    library.sscov=eye(size(library.rt,1));
```

```
            end
        elseif handles.settings_covariance==1 % covariance
            iterations=0.001;
            % RT
            [library.cov,iterations,library.est]=CovEM1(library.rt,iterations);
            if handles.settings_ssrt==1 % ssrt to be computed
                % SSRT
                [library.sscov,library.ssest]=CovEM2(library.ssrt,iterations);
            end
        end
        if handles.settings_librarysave==1
            waitbar(0.175,progress,'Saving library...'); %
dialog_box=msgbox('Saving library...','Status','replace');
            tmp2=fix(clock);
            if ispc==1
                tmp3=handles.file_library(find(handles.file_library == '/', 1,
'last')+1:find(handles.file_library == '.', 1, 'last')-1);
            else
                tmp3=handles.file_library(find(handles.file_library == '/', 1,
'last')+1:find(handles.file_library == '.', 1, 'last')-1);
            end
            save(['LITL_Lib_' num2str(tmp2(1)) '-' num2str(tmp2(2)) '-'
num2str(tmp2(3)) '_' tmp3{1}],'library')
        end
    catch e
        delete(progress);
        error('The Library file input is not formatted properly for the settings
selected.');
    end
elseif handles.mode_expdata==1
    % Upload .csv data export from MultiQuant
    % **PLEASE MAKE SURE**
    % - Delimiter of csv file is A COMMA
    % - Export contains the columns "Index", "
    raw.import=readtable(handles.file_expdata,'Delimiter',',');
    raw.import.Properties.VariableNames{1} = 'Index';
    raw.info=dir(handles.file_expdata);
    waitbar(0.05,progress);
    % Upload .csv data information about the data export from MultiQuant
    % **PLEASE MAKE SURE**
    % - Delimiter of csv file is A COMMA
    % - follow the template
    sampleInfo=readtable(handles.file_sampleinfo,'Delimiter',',');
    sampleInfo.Properties.VariableNames{1} = 'SampleName';
    % Extracts data from .csv import
    expData={};
    counterSample=1; % index of samples, maxed at size(sampleInfo,1)
    counterTransition=0; % index of transitions in each sample
    for counterRow=1:size(raw.import,1) % index of the rows in the raw.import
        % same sample or new sample
        if
strcmp(raw.import.SampleName(counterRow),sampleInfo.SampleName(counterSample))==0
            % new sample to input, change a index, reset c

counterSample=strmatch(raw.import.SampleName(counterRow),sampleInfo.SampleName);
            counterTransition=1;
        elseif
strcmp(raw.import.SampleName(counterRow),sampleInfo.SampleName(counterSample))==1
            % same sample, new transition (c+1)
            counterTransition=counterTransition+1;
        end
        if isempty(counterSample)
            delete(progress);
```

```
                error(['The sample ', raw.import.SampleName{counterRow}, ' is not
found in the Sample Information file provided.']);
            elseif strcmp(raw.import.RetentionTime(counterRow),'N/A')==1
                %         counterTransition=counterTransition-1; %discard N/A
transition
            else
                % storing the index of raw.import
                expData{counterSample,1}(counterTransition,1)=counterRow;
                % extract mz from MassInfo
                mz_all=cell2mat(raw.import.MassInfo(counterRow));
                mz_str='';
                temp=1;
                % temp: number of characters in the MassInfo cells
                while temp~=0 && temp<=size(mz_all,2)
                    if mz_all(temp)=='0' || mz_all(temp)=='1' || mz_all(temp)=='2' ||
mz_all(temp)=='3' || mz_all(temp)=='4' || mz_all(temp)=='5' || mz_all(temp)=='6' ||
mz_all(temp)=='7' || mz_all(temp)=='8' || mz_all(temp)=='9' || mz_all(temp)=='.'
                        mz_str=[mz_str mz_all(temp)];
                        temp=temp+1;
                    else
                        temp=0;
                    end
                end
                clear temp
                expData{counterSample,1}(counterTransition,2)=str2double(mz_str);
                % extract retention time and other data if non NaN

expData{counterSample,1}(counterTransition,3)=str2double(cell2mat(raw.import.Retention
Time(counterRow))); % retention time
                expData{counterSample,2}=sampleInfo.ISComponentName(counterSample); %
name of standard (string) to normalize to in each sample, to find in
MULTIQUANT_EXPORT.ComponentName

expData{counterSample,1}(counterTransition,4)=str2double(cell2mat(raw.import.Area(coun
terRow))); % peak area

expData{counterSample,1}(counterTransition,7)=str2double(cell2mat(raw.import.WidthAt50
_(counterRow))); % width at 50%
                if strcmp(raw.import.IS{counterRow},'TRUE')==1
                    expData{counterSample,7}{counterTransition,1}='Standard ';
                else
                    expData{counterSample,7}{counterTransition,1}='';
                end
                if strcmp(raw.import.Used{counterRow},'TRUE')==0
                    expData{counterSample,7}{counterTransition,2}='Saturated ';
     else
                    expData{counterSample,7}{counterTransition,2}='';
                end
                expData{counterSample,7}{counterTransition,3}='';
            end
        end
        % finding the data of the corresponding standard
        for counterSample=1:size(expData,1)
            actualStart=1;
            actualEnd=size(expData{counterSample,1},1);
            while expData{counterSample,1}(actualStart,1)==0
                actualStart=actualStart+1;
            end
            while expData{counterSample,1}(actualEnd,1)==0
                actualEnd=actualEnd-1;
            end
```

```
temp=find((strcmp(raw.import.ComponentName(expData{counterSample,1}(actualStart,1):exp
Data{counterSample,1}(actualEnd,1)),expData{counterSample,2})),1)+actualStart-1;
            expData{counterSample,3}=expData{counterSample,1}(temp,4);%
str2double(cell2mat( raw.import.Area( temp ) ));
            expData{counterSample,4}=sampleInfo.ISng(counterSample); % amount of
standard in sample (ng)
            expData{counterSample,5}=sampleInfo.ISChemicalMass(counterSample); %
molecular mass of standard to normalize to (microgram/mol)
            expData{counterSample,6}=sampleInfo.AmountOfMatrix(counterSample); %
sample weight (mg)
        end
        % Upload .xlsx library file OR .mat "already formatted" library file
        % **PLEASE MAKE SURE** (if .xlsx input)
        % - "Master sheet" is the second sheet starting from the left
        % - "Compilation sheet" is the fourth sheet starting from the left
        % - the term "mz" is present TWICE in the A column of the "Compilation
        % sheet", and that it is located in the row IMMEDIATELY BEFORE the data
        % starts
        waitbar(0.1,progress);
        [~,~,ext] = fileparts(handles.file_library);
        if strcmp(ext, '.mat') % load matrix
            load(handles.file_library);
        elseif strcmp(ext, '.xlsx') % load .xlsx library + estimate covariance
            try
                [~,~,library.table.compilation]=xlsread(handles.file_library,4);
                [~,~,library.table.master]=xlsread(handles.file_library,2);
                ind=find(strcmp(library.table.compilation(:,1),'mz'));
                ind=ind(2); % index of row delimitation for standards in library
                doubleblanks=0;
                library.rt=[];
                library.type=[];
                library.st.rt=[];
                library.st.mean=[];
                library.st.sd=[];
                % going through every column of compilation
                % tmp: index of columns in the compilation sheet
                for tmp=1:size(library.table.compilation,2)
                    if doubleblanks==2 && ~isnan(library.table.compilation{1,tmp})
                        if library.table.compilation{1,tmp+2}==1 % dataset to be
included

library.rt=horzcat(library.rt,[library.table.compilation(ind+1:end,tmp+3:tmp+(library.
table.compilation{1,tmp})+2)]);

library.type=horzcat(library.type,library.table.compilation{1,tmp+1}*ones(1,library.ta
ble.compilation{1,tmp}));

library.st.rt=horzcat(library.st.rt,[library.table.compilation(2:ind-
1,tmp+3:tmp+(library.table.compilation{1,tmp})+2)]);
                        end
                        tmp=tmp+(library.table.compilation{1,tmp})+2;
                        doubleblanks=0;
                    end
                    if isnan(library.table.compilation{2,tmp})
                        doubleblanks=doubleblanks+1;
                    else
                        doubleblanks=0;
                    end
                end
                library.st.index=cellfun(@(c) ischar(c) && ~isempty(strfind(c,
'Standard')), library.table.master(:,11));
                library.st.index=find(library.st.index==1);
```

```
                library.st.index=library.st.index(library.st.index>ind)-ind;
                for tmp=1:size(library.st.rt,2)
                    library.st.mean(1,tmp)=mean([library.st.rt{:,tmp}]);
                    library.st.sd(1,tmp)=std([library.st.rt{:,tmp}]);
                end
                library.barcode=[];
                library.genID=[];
                library.mz=[];
                for tmp=ind+1:size(library.table.compilation,1)

library.barcode=vertcat(library.barcode,library.table.master(tmp,12));
                    library.genID=vertcat(library.genID,library.table.master(tmp,10));

library.mz=vertcat(library.mz,floor(library.table.master{tmp,19})+0.5);
                end
                for tmp=size(library.rt,1):-1:1
                    for tmp2=size(library.rt,2):-1:1
                        if isempty(library.rt{tmp,tmp2})
                            library.rt{tmp,tmp2}=NaN;
                        end
                    end
                    if all(isnan([library.rt{tmp,:}]))
                        library.rt(tmp,:)=[];
                        library.barcode(tmp,:)=[];
                        library.mz(tmp,:)=[];
                    end
                end
                library.rt=cell2mat(library.rt);

[library.mean,library.diag]=MissingDataMeanAndVar(library.type,library.rt);
                library.va=diag(library.diag);
                % Loess smoothing
                if handles.settings_smooth==1
                    waitbar(0.11,progress,'Smoothing data...'); %
dialog_box=msgbox('Smoothing...','Status','replace');
                    [rawExpData,library,expData]=LoessSmoothing(library,expData,0.1);
                    % library.st fix
                    library.st.rt=library.rt(library.st.index,:);
                    for tmp=1:size(library.st.rt,2)
                        library.st.mean(1,tmp)=mean([library.st.rt(:,tmp)]);
                        library.st.sd(1,tmp)=std([library.st.rt(:,tmp)]);
                    end
                    % library.mean fix

[library.mean,library.diag]=MissingDataMeanAndVar(library.type,library.rt);
                    library.va=diag(library.diag);
                end
                if handles.settings_ssrt==1
                    waitbar(0.12,progress,'Standardardizing and scaling data...'); %
dialog_box=msgbox('Standardizing and scaling data...','Status','replace');
                    % SSRT
                    % computes the SSRT transformation for the expData
                    % standardized and scaled retention times
                    % SSRT = (RT - mean(standard RTs)) / sd(standard RTs)
                    library.ssrt=library.rt;
                    for tmpx=1:size(library.mz,1)
                        for tmpy=1:size(library.rt,2)
                            if isnan(library.rt(tmpx,tmpy))
                                library.ssrt(tmpx,tmpy)=NaN;
                            else
                                library.ssrt(tmpx,tmpy)=( library.rt(tmpx,tmpy) -
library.st.mean(1,tmpy) ) / library.st.sd(1,tmpy);
                            end
```

```
                        end
                    end

[library.ssmean,library.ssdiag]=MissingDataMeanAndVar(library.type,library.ssrt);
                    library.ssva=diag(library.ssdiag);
                    % tmps: index of the sample in expData
                    for tmps=1:size(expData,1)
                        actualStart=1;
                        actualEnd=size(expData{tmps,1},1);
                        while expData{tmps,1}(actualStart,1)==0
                            actualStart=actualStart+1;
                        end
                        while expData{tmps,1}(actualEnd,1)==0
                            actualEnd=actualEnd-1;
                        end

standardInd=find(strcmp(raw.import.IS(expData{tmps,1}(actualStart,1):expData{tmps,1}(a
ctualEnd,1)),'TRUE'))+actualStart-1;
                        standardMean=mean(expData{tmps,1}(standardInd,3));
                        standardSD=std(expData{tmps,1}(standardInd,3));
                        % tmp: index of the different species in a sample tmps
                        for tmp=1:size(expData{tmps,1},1)
                            % SSRT = (RT - mean(standard rts)) / std(standard rts)
                            expData{tmps,1}(tmp,6)= (expData{tmps,1}(tmp,3) -
standardMean ) / standardSD;
                        end
                    end
                end
                % Clearing the empty transitions
                % tmps: index of the rows in expData (samples)
                % tmp: index of the rows in a sample (species)
                for tmps=1:size(expData,1)
                    for tmp=size(expData{tmps,1},1):-1:1
                        if expData{tmps,1}(tmp,1)==0 %empty transition
                            expData{tmps,1}(tmp,:)=[];
                            expData{tmps,7}(tmp,:)=[];
                        end
                    end
                end
                % Covariance
                waitbar(0.13,progress,'Computing the library covariance...'); %
dialog_box=msgbox('Computing the library covariance...','Status','replace');
                if handles.settings_covariance==0
                    library.cov=eye(size(library.rt,1));
                    if handles.settings_ssrt==1 % ssrt to be computed
                        library.sscov=eye(size(library.rt,1));
                    end
                elseif handles.settings_covariance==1
                    iterations=0.001;
                    % RT

[library.cov,iterations,library.est]=CovEM1(library.rt,iterations);
                    if handles.settings_ssrt==1 % ssrt to be computed
                        % SSRT
                        [library.sscov,library.ssest]=CovEM2(library.ssrt,iterations);
                    end
                end
                if handles.settings_librarysave==1
                    waitbar(0.17,progress,'Saving library...'); %
dialog_box=msgbox('Saving library...','Status','replace');
                    tmp2=fix(clock);
                    tmp3=handles.file_library(find(handles.file_library == '/', 1,
'last')+1:find(handles.file_library == '.', 1, 'last')-1);
```

```
                    save(['LITL_Lib_' num2str(tmp2(1)) '-' num2str(tmp2(2)) '-'
num2str(tmp2(3)) '_' tmp3{1}],'library')
                end
            catch
                delete(progress);
                error('The data file input is not formatted properly for the settings
selected.');
            end
        else % error
            delete(progress);
            error('This library is not compatible with LITL.');
        end
        if handles.settings_quant==1
            waitbar(0.18,progress,'Quantifying data...'); %
dialog_box=msgbox('Quantifying data...','Status','replace');
            % Quantification
            % pmol/mg standard/mg
            % (PA/PAs)*( ng / g/mol *1000 )/tissue amount
            % peak areas are in expData{sample,1}(species,4)
            % peak areas of quantification standards are in expData{sample,3}
            % ng of the standards are in expData{sample,4}
            % mg/mol of the standards are in expData{sample,5}, they will be
multiplied
            % by a factor of 1000 to get g/mol
            % sample amounts are in expData{sample,6} (mg, 1e6 cells, mL)
            for counterSample=1:size(expData,1)
                for counterTransition=1:size(expData{counterSample,1},1)
                    expData{counterSample,1}(counterTransition,5) = (
expData{counterSample,1}(counterTransition,4) / expData{counterSample,3} ) * (
expData{counterSample,4} / expData{counterSample,5} * 1000 ) /
expData{counterSample,6};
                end
            end
        end
        % Isotopes/dehydrations/deglycosylations
        waitbar(0.19,progress,'Looking for isotopes, dehydrations, and/or
deglycosylations...'); % dialog_box=msgbox('Looking for isotopes, dehydrations and/or
deglycosylations...','Status','replace');
        % If a species rt and width at 50% point to it being a duplication of a
previous
        % species, the mention "Possible isotope/dehydration/deglycosylation of ..."
is added to it in
        % expData{sample,7}{species,3} (third sheet in the .xlsx output)
        % rt correlation
        % 0.999 < rt/rt < 1.001
        % width at 50% separation
        % 0.8 > abs(rt-rt) / ((width+width)/2)
        for sample=1:size(expData,1)
            for x=1:size(expData{sample,1},1)
                for y=x+1:size(expData{sample,1},1)
                    % too close (mz - 0)
                    if expData{sample,1}(y,2)-expData{sample,1}(x,2) < 1
                        if expData{sample,1}(y,3)/expData{sample,1}(x,3) > 0.999 &&
expData{sample,1}(y,3)/expData{sample,1}(x,3) < 1.001
                            if 0.8 > abs( expData{sample,1}(y,3) -
expData{sample,1}(x,3) ) / (( expData{sample,1}(y,7) + expData{sample,1}(x,7) ) /2)
                                expData{sample,7}{y,3}=[expData{sample,7}{y,3} '
Possible duplicate of ' num2str(expData{sample,1}(x,2))];
                            end
                        end
                        % isotope (mz - 1,2,3,4)
                    elseif expData{sample,1}(y,2)-expData{sample,1}(x,2) <= 4 &&
expData{sample,1}(y,2)~=expData{sample,1}(x,2)
```

```
                        if expData{sample,1}(y,3)/expData{sample,1}(x,3) > 0.999 &&
expData{sample,1}(y,3)/expData{sample,1}(x,3) < 1.001
                            if 0.8 > abs( expData{sample,1}(y,3) -
expData{sample,1}(x,3) ) / (( expData{sample,1}(y,7) + expData{sample,1}(x,7) ) /2)
                                expData{sample,7}{y,3}=[expData{sample,7}{y,3} '
Possible isotope of ' num2str(expData{sample,1}(x,2))];
                            end
                        end
                        % dehydration (mz - 18)
                    elseif expData{sample,1}(y,2)-expData{sample,1}(x,2) >= 17 &&
expData{sample,1}(y,2)-expData{sample,1}(x,2) <= 19 &&
expData{sample,1}(y,2)~=expData{sample,1}(x,2)
                        if expData{sample,1}(y,3)/expData{sample,1}(x,3) > 0.999 &&
expData{sample,1}(y,3)/expData{sample,1}(x,3) < 1.001
                            if 0.8 > abs( expData{sample,1}(y,3) -
expData{sample,1}(x,3) ) / (( expData{sample,1}(y,7) + expData{sample,1}(x,7) ) /2)
                                expData{sample,7}{x,3}=[expData{sample,7}{x,3} '
Possible dehydration of ' num2str(expData{sample,1}(y,2))];
                            end
                        end
                        % deglycosylation (mz - 168)
                    elseif expData{sample,1}(y,2)-expData{sample,1}(x,2) >= 167 &&
expData{sample,1}(y,2)-expData{sample,1}(x,2) <= 169 &&
expData{sample,1}(y,2)~=expData{sample,1}(x,2)
                        if expData{sample,1}(y,3)/expData{sample,1}(x,3) > 0.999 &&
expData{sample,1}(y,3)/expData{sample,1}(x,3) < 1.001
                            if 0.8 > abs( expData{sample,1}(y,3) -
expData{sample,1}(x,3) ) / (( expData{sample,1}(y,7) + expData{sample,1}(x,7) ) /2)
                                expData{sample,7}{x,3}=[expData{sample,7}{x,3} '
Possible deglycosylation of ' num2str(expData{sample,1}(y,2))];
                            end
                        end
                    end
                end
            end
        end
    elseif handles.settings_lines==1
        % outputs the biophysical lines for selected library file
        % Upload .xlsx library file OR .mat "already formatted" library file
        % **PLEASE MAKE SURE** (if .xlsx input)
        % - "Master sheet" is the second sheet starting from the left
        % - "Compilation sheet" is the fourth sheet starting from the left
        % - the term "mz" is present TWICE in the A column of the "Compilation
        % sheet", and that it is located in the row IMMEDIATELY BEFORE the data
        % starts
        [~,~,ext] = fileparts(handles.file_library);
        if strcmp(ext, '.mat') % load matrix
            load(handles.file_library);
        elseif strcmp(ext, '.xlsx') % load .xlsx library + estimate covariance
            try
                [~,~,library.table.compilation]=xlsread(handles.file_library,4);
                [~,~,library.table.master]=xlsread(handles.file_library,2);
                ind=find(strcmp(library.table.compilation(:,1),'mz'));
                ind=ind(2); % index of row delimitation for standards in library
                doubleblanks=0;
                library.rt=[];
                library.type=[];
                library.st.rt=[];
                library.st.mean=[];
                library.st.sd=[];
                tmp2=0;
                % going through every column of compilation
                % tmp: index of columns in the compilation sheet
```

```
                for tmp=1:size(library.table.compilation,2)
                    if doubleblanks==2 && ~isnan(library.table.compilation{1,tmp})
                        if library.table.compilation{1,tmp+2}==1 % dataset to be
included
                            n=library.table.compilation{1,tmp}; % number of samples
                            for sample=tmp+3:tmp+2+n % all samples in current dataset
                                tmp2=tmp2+1;

sampleName{tmp2,1}=library.table.compilation{ind,sample};

library.rt=horzcat(library.rt,[library.table.compilation(ind+1:end,sample)]);

library.type=horzcat(library.type,library.table.compilation{1,tmp+1});

library.st.rt=horzcat(library.st.rt,[library.table.compilation(2:ind-1,sample)]);
                            end
                        end
                        tmp=tmp+(library.table.compilation{1,tmp})+2;
                        doubleblanks=0;
                    end
                    if isnan(library.table.compilation{2,tmp})
                        doubleblanks=doubleblanks+1;
                    else
                        doubleblanks=0;
                    end
                end
                library.st.index=cellfun(@(c) ischar(c) && ~isempty(strfind(c,
'Standard')), library.table.master(:,11));
                library.st.index=find(library.st.index==1);
                library.st.index=library.st.index(library.st.index>ind)-ind;
                for tmp=1:size(library.st.rt,2)
                    library.st.mean(1,tmp)=mean([library.st.rt{:,tmp}]);
                    library.st.sd(1,tmp)=std([library.st.rt{:,tmp}]);
                end
                library.barcode=[];
                library.genID=[];
                library.mz=[];
                for tmp=ind+1:size(library.table.compilation,1)

library.barcode=vertcat(library.barcode,library.table.master(tmp,12));
                    library.genID=vertcat(library.genID,library.table.master(tmp,10));

library.mz=vertcat(library.mz,floor(library.table.master{tmp,19})+0.5);
                end
                for tmp=size(library.rt,1):-1:1
                    for tmp2=size(library.rt,2):-1:1
                        if isempty(library.rt{tmp,tmp2})
                            library.rt{tmp,tmp2}=NaN;
                        end
                    end
                    if all(isnan([library.rt{tmp,:}]))
                        library.rt(tmp,:)=[];
                        library.barcode(tmp,:)=[];
                        library.mz(tmp,:)=[];
                    end
                end
                library.rt=cell2mat(library.rt);
            catch
                delete(progress);
                error('The data file input is not formatted properly for the settings
selected.');
            end
        else % error
```

```
            delete(progress);
            error('This library is not compatible with LITL.');
        end
        % formats library file for BiophysicalLines_forGUI function
        dataRT={};
        for tmpx=1:size(library.barcode,1)
            dataRT{6+tmpx,1}=library.barcode{tmpx,1};
        end
        for tmpy=1:size(library.rt,2)
            dataRT{6,tmpy+1}=sampleName{tmpy,1};
        end
        for tmpx=1:size(library.rt,1)
            for tmpy=1:size(library.rt,2)
                dataRT{tmpx+6,tmpy+1}=library.rt(tmpx,tmpy);
            end
        end
        waitbar(0.5,progress,'Creating graphs...'); % dialog_box=msgbox('Creating
graphs...','Status','replace');
        sample_adjust=2;
        [Corrs,RTE] =
BiophysicalLines_forGUI(6,strcat(handles.file_folder,'/',handles.file_foldername),data
RT,sample_adjust,library,handles.settings_corr);
    else
        delete(progress);
        error('Please select an analysis mode.');
    end
 if handles.mode_library==1 || handles.mode_expdata==1
  %% Joint assignment
  % expI unique mz-by-sample cell
  % {-,counterSample} for each unique mz, containing a double row vector
  % first column: mz
  % subsequent columns: rts (top row) and their indeces ind (bot row), refering back
to expData{-,1}(ind)
  % lib unique mz-by-1 cell
  % {-,1} for each unique mz, containing a double row vector
  % first column: mz
  % subsequent columns: rts (top row) and their indeces ind (bot row), refering back
to library.ind(ind)
  % I sample-by-unique mz cell
  % each I{sample,mz}{1,1} cell contains the information from expI
  % each I{sample,mz}{2,1} cell contains the information from lib
  I={};
  if handles.settings_ssrt==1
   library.va=library.ssva;
   library.diag=library.ssdiag;
   library.rt=library.ssrt;
   library.mean=library.ssmean;
   library.cov=library.sscov;
   if handles.settings_covariance==1
    library.est=library.ssest;
   end
  end
  for counterSample=1:size(expData,1)
            if handles.settings_lines==1
                waitbar(0.2+(0.5/size(expData,1))*counterSample,progress,['Analyzing
sample ', num2str(counterSample), '...']);
            else
                waitbar(0.2+(0.6/size(expData,1))*counterSample,progress,['Analyzing
sample ', num2str(counterSample), '...']); % dialog_box=msgbox(['Analyzing sample
',num2str(counterSample),'...'],'Status','replace');
            end
            % tmp2: index of unique mz
    % tmp: index of species (rows)
```

```
    tmp2=1;
    for tmp=1:size(expData{counterSample,1},1)
     if tmp==1 % first iteration
      expI{tmp2,counterSample}=[expData{counterSample,1}(tmp,2)
expData{counterSample,1}(tmp,3+handles.settings_ssrt);
       expData{counterSample,1}(tmp,2) tmp];
     elseif expData{counterSample,1}(tmp-1,2)==expData{counterSample,1}(tmp,2) % same
mz as previous, concatenate current line

 expI{tmp2,counterSample}=horzcat(expI{tmp2,counterSample},[expData{counterSample,1}(t
mp,3+handles.settings_ssrt);tmp]);
     else % different mz as previous, new line
      tmp2=tmp2+1;
      expI{tmp2,counterSample}=[expData{counterSample,1}(tmp,2)
expData{counterSample,1}(tmp,3+handles.settings_ssrt);
       expData{counterSample,1}(tmp,2) tmp];
     end
    end
    expIsize=tmp2;
    tmp2=1;
    for tmp=1:size(library.rt,1)
     if tmp==1 % first iteration
      lib{tmp2,1}=[library.mz(tmp,1) library.mean(tmp,1);
       library.mz(tmp,1) tmp];
     elseif library.mz(tmp-1,1)==library.mz(tmp,1) % same mz as previous, concatenate
current line
      lib{tmp2,1}=horzcat(lib{tmp2,1},[library.mean(tmp,1);tmp]);
     else % different mz as previous, new line
      tmp2=tmp2+1;
      lib{tmp2,1}=[library.mz(tmp,1) library.mean(tmp,1);
       library.mz(tmp,1) tmp];
     end
    end
    % Matching
    % tmp: index of unique mz in experimental data (rows in expI)
    % tmp2: index of unique mz in library (rows in lib)
    % tmp3: index of unique mz, put in common (columns in I)
    tmp=1;
    tmp2=1;
    tmp3=1;
    while tmp<=expIsize %size(expI(:,counterSample),1)
     if (floor(expI{tmp,counterSample}(1,1))+0.5)==lib{tmp2,1}(1,1) % if same mz
      I{counterSample,tmp3}{1,1}=expI{tmp,counterSample}(1:2,2:end);
      I{counterSample,tmp3}{2,1}=lib{tmp2,1}(1:2,2:end);
      tmp=tmp+1;
      tmp2=tmp2+1;
      tmp3=tmp3+1;
     elseif (floor(expI{tmp,counterSample}(1,1))+0.5)>lib{tmp2,1}(1,1) % library has
extra, pass
      tmp2=tmp2+1;
     elseif (floor(expI{tmp,counterSample}(1,1))+0.5)<lib{tmp2,1}(1,1) % library
doesn't have it, pass
      tmp=tmp+1;
     end
    end
    if handles.mode_library==1
     if handles.settings_ja==0 % library assignment - no tweaking

 Assignment{counterSample}=outerJA0(I(counterSample,:),library,expData{counterSample,1
}(:,1:2));
     elseif handles.settings_ja==1 % library assignment start (with X)
```

```
 Assignment{counterSample}=outerJA1(I(counterSample,:),library,expData{counterSample,1
}(:,1:2));
     elseif handles.settings_ja==2 % possible assignment start (lower RT difference)
      Assignment{counterSample}=outerJA2(I(counterSample,:),library);
     end
    elseif handles.mode_expdata==1
     Assignment{counterSample}=outerJA2(I(counterSample,:),library);
    end
   end
   % tmp3: index of samples (columns in Assignment)
   % tmp: counter index for species (rows in Assignment sub variables)
   % tmp1: index of unique mz (columns in I, rows in sub variable X)
   % tmp2: index of species per unique mz (columns in I cells and sub variable X cells)
   for tmp3=1:size(Assignment,2) % samples
    tmp=0;
    for tmp1=1:size(Assignment{1,tmp3}.X,1) % unique mz
     for tmp2=1:size(Assignment{1,tmp3}.X{tmp1,1},2) % species assigned per mz
      if Assignment{1,tmp3}.X{tmp1,1}(tmp2)~=0
       tmp=tmp+1;

 Assignment{1,tmp3}.barcode{tmp,1}=library.barcode{I{tmp3,tmp1}{2,1}(2,Assignment{1,tm
p3}.X{tmp1,1}(tmp2)),1};

 Assignment{1,tmp3}.mz(tmp,1)=library.mz(I{tmp3,tmp1}{2,1}(2,Assignment{1,tmp3}.X{tmp1
,1}(tmp2)),1);

 Assignment{1,tmp3}.id{tmp,1}=library.table.master{ind+I{tmp3,tmp1}{2,1}(2,Assignment{
1,tmp3}.X{tmp1,1}(tmp2)),14};
        if handles.mode_expdata==1
         Assignment{1,tmp3}.pmol(tmp,1)=expData{tmp3,1}(I{tmp3,tmp1}{1,1}(2,tmp2),5);
         Assignment{1,tmp3}.notes{tmp,1}=expData{tmp3,7}{I{tmp3,tmp1}{1,1}(2,tmp2),1};
         Assignment{1,tmp3}.notes{tmp,2}=expData{tmp3,7}{I{tmp3,tmp1}{1,1}(2,tmp2),2};
         Assignment{1,tmp3}.notes{tmp,3}=expData{tmp3,7}{I{tmp3,tmp1}{1,1}(2,tmp2),3};
        end
       end
      end
     end
   end
   % OUTPUT
        if handles.settings_lines==1
            waitbar(0.8,progress,'Creating output...');
        else
            waitbar(0.9,progress,'Creating output...'); % dialog_box=msgbox('Creating
output...','Status','replace');
        end
   mkdir(handles.file_folder,handles.file_foldername)
   if handles.mode_library==1 % accuracy output
    OUT.rt={};
    for species=1:size(library.barcode,1)
     OUT.rt{6+species,1}=library.barcode{species,1};
    end
    for sample=1:size(expData,1)
     counter=1;
     rtcount=0;
     OUT.rt{1,2}='Logobj';
     OUT.rt{2,2}='Logobj-last';
     OUT.rt{6,2}='n';
     OUT.rt{1,2+sample}=Assignment{1,sample}.logobjF(1);
     OUT.rt{2,2+sample}=Assignment{1,sample}.logobjF(2);
     OUT.rt{5,2+sample}=sampleName{sample,1};
     for species=1:size(library.barcode,1)
      if counter<=size(Assignment{1,sample}.barcode,1)
```

```
        if OUT.rt{6+species,1}==Assignment{1,sample}.barcode{counter,1}
         OUT.rt{6+species,2+sample}=Assignment{1,sample}.RT(counter,1);
         rtcount=rtcount+1;
         if isempty(OUT.rt{6+species,2})
          OUT.rt{6+species,2}=Assignment{1,sample}.id{counter,1};
         end
         counter=counter+1;
        end
       end
      end
      OUT.rt{6,2+sample}=rtcount;
    end
    OUT.accuracy=OUT.rt;
    for sample=1:size(expData,1)
     counter=1;
     miscount=0;
     OUT.accuracy{6,2}='accuracy';
     for species=1:size(library.barcode,1)
      if counter<=size(Assignment{1,sample}.barcode,1)
       if OUT.rt{6+species,1}==Assignment{1,sample}.barcode{counter,1}
        if Assignment{1,sample}.barcode{counter,1}==expData{sample,2}{counter,1} %
correct
         else % incorrect
          OUT.accuracy{6+species,2+sample}=expData{sample,2}{counter,1};
          miscount=miscount+1;
         end
         counter=counter+1;
        end
       end
      end
      OUT.accuracy{6,2+sample}=(OUT.rt{6,2+sample}-miscount)/OUT.rt{6,2+sample}*100;
    end
    for species=size(OUT.rt,1):-1:7
     if isempty(OUT.rt{species,2})
      OUT.rt(species,:)=[];
      OUT.accuracy(species,:)=[];
     end
           end
           if ispc==1
               tmp=handles.file_library(find(handles.file_library == '\', 1,
'last')+1:find(handles.file_library == '.', 1, 'last')-1);
           else
               tmp=handles.file_library(find(handles.file_library == '/', 1,
'last')+1:find(handles.file_library == '.', 1, 'last')-1);
           end
           tmp=strcat('LITL_',tmp,'_Split analysis_',num2str(handles.settings_ja));
    if handles.settings_covariance==0
     tmp=strcat(tmp,'_nocov');
    elseif handles.settings_covariance==1
     tmp=strcat(tmp,'_cov');
    end
    if handles.settings_ssrt==0
     tmp=strcat(tmp,'_RT');
    elseif handles.settings_ssrt==1
     tmp=strcat(tmp,'_SSRT');
    end
    if handles.settings_smooth==0
     tmp=strcat(tmp,'_Raw');
    elseif handles.settings_smooth==1
     tmp=strcat(tmp,'_Smooth');
    end
    tmp=strcat(tmp,'_',num2str(handles.settings_seed));
    if ismac==0
```

```
xlswrite(char(strcat(handles.file_folder,'\',handles.file_foldername,'\',tmp,'.xlsx'))
,OUT.rt,1)

xlswrite(char(strcat(handles.file_folder,'\',handles.file_foldername,'\',tmp,'.xlsx'))
,OUT.accuracy,2)
            else

xlwrite(char(strcat(handles.file_folder,'/',handles.file_foldername,'/',tmp,'.xlsx')),
OUT.rt,1)

xlwrite(char(strcat(handles.file_folder,'/',handles.file_foldername,'/',tmp,'.xlsx')),
OUT.accuracy,2)
            end
            if handles.settings_lines==1
     waitbar(0.9,progress,'Creating graphs...'); % dialog_box=msgbox('Creating
graphs...','Status','replace');
     sample_adjust=3+2+handles.settings_dataset;
     [Corrs,RTE] =
BiophysicalLines_forGUI(5,strcat(handles.file_folder,'/',handles.file_foldername),OUT.
rt,sample_adjust,library,handles.settings_corr);
    end
   elseif handles.mode_expdata==1 % pmol output
    OUT.pmol={};
    for species=1:size(library.barcode,1)
     OUT.pmol{6+species,1}=library.barcode{species,1};
    end
    for sample=1:size(sampleInfo,1)
     counter=1;
     OUT.pmol{1,2}='Feature1';
     OUT.pmol{2,2}='Feature2';
     OUT.pmol{3,2}='Feature3';
     OUT.pmol{4,2}='Feature4';
     OUT.pmol{5,2}='Feature5';
     OUT.pmol{6,2}='SampleName';
     OUT.pmol{1,2+sample}=sampleInfo.Feature1{sample};
     OUT.pmol{2,2+sample}=sampleInfo.Feature2{sample};
     OUT.pmol{3,2+sample}=sampleInfo.Feature3{sample};
     OUT.pmol{4,2+sample}=sampleInfo.Feature4{sample};
     OUT.pmol{5,2+sample}=sampleInfo.Feature5{sample};
     OUT.pmol{6,2+sample}=sampleInfo.SampleName{sample};
     for species=1:size(library.barcode,1)
      if counter<=size(Assignment{1,sample}.barcode,1)
       if OUT.pmol{6+species,1}==Assignment{1,sample}.barcode{counter,1}
        OUT.pmol{6+species,2+sample}=Assignment{1,sample}.pmol(counter,1);
        if isempty(OUT.pmol{6+species,2})
         OUT.pmol{6+species,2}=Assignment{1,sample}.id{counter,1};
        end
        counter=counter+1;
       end
      end
     end
    end
    OUT.rt=OUT.pmol;
    OUT.notes=OUT.pmol;
    for sample=1:size(sampleInfo,1)
     counter=1;
     for species=1:size(library.barcode,1)
      if counter<=size(Assignment{1,sample}.barcode,1)
       if OUT.pmol{6+species,1}==Assignment{1,sample}.barcode{counter,1}
        OUT.rt{6+species,2+sample}=Assignment{1,sample}.RT(counter,1);
        OUT.notes{6+species,2+sample}=[Assignment{1,sample}.notes{counter,:}];
        counter=counter+1;
```

```
        end
       end
      end
     end
     for species=size(OUT.pmol,1):-1:7
      if isempty(OUT.pmol{species,2})
       OUT.pmol(species,:)=[];
       OUT.rt(species,:)=[];
       OUT.notes(species,:)=[];
      end
            end
            if ispc==1
                tmp=handles.file_expdata(find(handles.file_expdata == '\', 1,
'last')+1:find(handles.file_expdata == '.', 1, 'last')-1);
            else
                tmp=handles.file_expdata(find(handles.file_expdata == '/', 1,
'last')+1:find(handles.file_expdata == '.', 1, 'last')-1);
            end
            tmp=strcat('LITL_analysis',tmp);
     if handles.settings_covariance==0
      tmp=strcat(tmp,'_nocov');
     elseif handles.settings_covariance==1
      tmp=strcat(tmp,'_cov');
     end
     if handles.settings_ssrt==0
      tmp=strcat(tmp,'_RT');
     elseif handles.settings_ssrt==1
      tmp=strcat(tmp,'_SSRT');
     end
     if handles.settings_smooth==0
      tmp=strcat(tmp,'_Raw');
     elseif handles.settings_smooth==1
      tmp=strcat(tmp,'_Smooth');
            end
            if ismac==0

xlswrite(char(strcat(handles.file_folder,'\',handles.file_foldername,'\',tmp,'.xlsx'))
,OUT.rt,1)

xlswrite(char(strcat(handles.file_folder,'\',handles.file_foldername,'\',tmp,'.xlsx'))
,OUT.pmol,2)

xlswrite(char(strcat(handles.file_folder,'\',handles.file_foldername,'\',tmp,'.xlsx'))
,OUT.notes,3)
            else

xlwrite(char(strcat(handles.file_folder,'/',handles.file_foldername,'/',tmp,'.xlsx')),
OUT.rt,1)

xlwrite(char(strcat(handles.file_folder,'/',handles.file_foldername,'/',tmp,'.xlsx')),
OUT.pmol,2)

xlwrite(char(strcat(handles.file_folder,'/',handles.file_foldername,'/',tmp,'.xlsx')),
OUT.notes,3)
            end
            if handles.settings_prism==1
      PrismScriptAuto(OUT.pmol,handles.file_folder,handles.file_foldername)
     end
     if handles.settings_lines==1
      waitbar(0.9,progress,'Creating graphs...'); % dialog_box=msgbox('Creating
graphs...','Status','replace');
      sample_adjust=3;
```

```
     [Corrs,RTE] =
BiophysicalLines_forGUI(6,strcat(handles.file_folder,'/',handles.file_foldername),OUT.
rt,sample_adjust,library,handles.settings_corr);
    end
   end
  end
    timeend=clock;
    timetaken=timeend-timestart;
    if timetaken(6)<0
        timetaken(6)=timetaken(6)+60;
        timetaken(5)=timetaken(5)-1;
    end
    if timetaken(5)<0
        timetaken(5)=timetaken(5)+60;
        timetaken(4)=timetaken(4)-1;
    end
    timetaken(4)=timetaken(4)+24*timetaken(3);
    waitbar(1,progress,['LITL finished the analysis in ' num2str(timetaken(4)) '
hour(s) ' num2str(timetaken(5)) ' minute(s) and ' num2str(timetaken(6)) '
second(s).']); % dialog_box=msgbox(['LITL finished the analysis in '
num2str(timetaken(4)) ' hour(s) ' num2str(timetaken(5)) ' minute(s) and '
num2str(timetaken(6)) ' second(s).'],'Status','replace');
catch e %e is an MException struct
    if
strcmp(e.identifier,'MATLAB:hg:udd_interface:CannotDelete')||~isempty(strfind(e.identi
fier,'waitbar'))
        errordlg('Analysis cancelled by the user.','Error');
    else
        errordlg([e.identifier e.message],'Error');
    end
end
% --- Executes when user attempts to close figure1.
function figure1_CloseRequestFcn(hObject, eventdata, handles)
button = questdlg('LITL will now close','Exit Dialog','Okay','Wait! Stay open!','Wait!
Stay open!');
switch button
    case 'Okay',
        delete(hObject)
    case 'Wait! Stay open!',
        quit cancel;
end

function [Mean,Var] = MissingDataMeanAndVar(G,T)
% function [Mean,Var] = MissingDataMeanAndVar(G,T) computes an estimated
% weighted mean and variance, with assumption of zero covariance
% for the real m-by-n matrix T.
% The columns of T correspond to samples, or "observations".
% T may have NaN entries.
% The 1-by-n vector of numbers G assign a group number to each sample.
% The mean and covariances estimates give equal weight to each
% distinct group.
% First, check for no rows of all nans.
IsNAN = isnan(T);
AllNAN = min(IsNAN')';
if any(AllNAN)
    I = find(AllNAN);
    disp(['Error, entity ',num2str(I(1)),', and possibly others, are all nan. Please
remove from matrix and try again.']);
    Mean = [];
    Var = [];
    return;
end
UniqG = unique(G);
```

```
NumGroups = length(UniqG);
[NumEntities,NumSamples] = size(T);
% Count appearances of entities in groups
N_entity_group = zeros(NumEntities,NumGroups);
for i=1:NumEntities
 for j=1:NumSamples
  if ~isnan(T(i,j))
   g = find(UniqG==G(j));
   N_entity_group(i,g) = N_entity_group(i,g)+1;
  end
 end
end
N_entity = sum(N_entity_group>0,2);
% Create weights
W_entity_sample = zeros(NumEntities,NumSamples);
for i=1:NumEntities
 for j=1:NumSamples
  if ~isnan(T(i,j))
   g = find(UniqG==G(j));
   W_entity_sample(i,j) = (1.0/N_entity(i))*(1.0/N_entity_group(i,g));
  end
 end
end
% Mean estimates
Mean = zeros(NumEntities,1);
for i=1:NumEntities
 for j=1:NumSamples
  if ~isnan(T(i,j))
   Mean(i) = Mean(i) + W_entity_sample(i,j)*T(i,j);
  end
 end
end
% Variance estimates
Var = zeros(NumEntities);
for i=1:NumEntities
 for j=1:NumSamples
  if ~isnan(T(i,j))
   Var(i,i) = Var(i,i) + W_entity_sample(i,j)*(T(i,j)-Mean(i))^2;
  end
 end
end
return;

function [COVARIANCE,iteration,ESTIMATION]=CovEM1(RT,iterationmax)
% In order to use the QP approach, the covariance of the Library data and
% its inverse is required. However, it is not rare to see lipid species not
% detected in certain matrices. Therefore, LITL estimates the missing
% values in the Library by using an EM algorithm described in
% [1] (Little, R. J. A., et al. (2002). Multivariate Normal Examples,
% Ignoring the Missing-Data Mechanism.  Statistical Analysis with Missing Data,
% John Wiley & Sons, Inc.: 221-252.)
% iteration : iteration counter
% cova : structure containing the below described matrices
% cova.raw : transitions-by-samples matrix of retention times (with NaN values)
% cova.hat : transitions-by-samples matrix of retention times (NaN values estimated)
% cova.mu : transitions-by-1 vector of mean retention times (Non weighted)
% cova.sigma : transitions-by-transition estimated covariance matrix
% cova.tmp : temporarory transitions-by-1 column vector storing covariance
%            estimates for one sample
% cova.delta : 1-by-iterations vector of the max delta between iterations
% cova.loglikelihood : 1-by-iterations vector of the loglikelihood of the
%                      covariance matrix after each iteration
% --------------------------------------------------------------------
```

```
cova.raw=RT;
iteration=1;
cova.hat(:,:,iteration)=cova.raw; %for first estimation
% first rough estimation of the nan values (average)
for tmp=1:size(cova.raw,1)
    tmpAve=nanmean(cova.raw(tmp,:));
    cova.mu(tmp,1)=tmpAve;
    for tmp2=1:size(cova.raw,2)
        if isnan(cova.raw(tmp,tmp2))==1
            cova.hat(tmp,tmp2,iteration)=tmpAve;
        end
    end
end
clear tmp tmp2 tmpAve
% first rough estimation of the covariance
cova.sigma=cov(cova.hat(:,:,iteration)',1);
iteration=1;
cova.delta(1)=9;
difference=999;
while abs(difference)>iterationmax % continue iterating until threshold
    %----
    if iteration~=1
        % Update mu and sigma estimated values
        % mu
        for tmp=1:size(cova.raw,1)
            tmpAve=nanmean(cova.hat(tmp,:,iteration));
            cova.mu(tmp,1)=tmpAve;
        end
        clear tmp tmp2 tmpAve
        % sigma
        cova.sigma=cov(cova.hat(:,:,iteration)',1);
        cova.sigma2=cov(cova.hat(:,:,iteration)',1)+10^-10*eye(size(cova.sigma,1));
        % For diagnostic : loglikelihood
        tmp=mvnpdf(cova.hat(:,:,iteration)',cova.mu',cova.sigma2);
        tmp=log(tmp);
        cova.loglikelihood(iteration-1)=sum(tmp);
        clear tmp
    end
    %----
    % E-step Yt
    for samples=1:size(cova.raw,2)
        Species=length(cova.mu);
        G=[1 cova.mu'; cova.mu cova.sigma+cova.mu*cova.mu'];
        G=Sweep(G,1);
        % looping around the species (rows) and sweeping if not NaN
        for i=1:Species
            if ~isnan(cova.raw(i,samples))
                G=Sweep(G,i+1);
            end
        end
        cova.tmp=cova.raw(:,samples);
        for i=1:Species
            if isnan(cova.raw(i,samples))
                cova.tmp(i)=G(1,i+1);
                for j=1:Species
                    if ~isnan(cova.raw(j,samples))
                        cova.tmp(i)=cova.tmp(i)+cova.raw(j,samples)*G(j+1,i+1);
                    end
                end
            end
        end
        cova.hat(:,samples,iteration+1)=cova.tmp; % assign the estimated sample
    end
```

```
    delta=reshape(abs(cova.hat(:,:,iteration+1)-cova.hat(:,:,iteration)),[],1);
    delta(delta==0)=NaN;
    cova.delta(iteration+1)=nanmean(delta);
    difference=cova.delta(iteration+1)-cova.delta(iteration);
    iteration=iteration+1; % update iteration counter
end
% output variables
iteration=iteration-1; % adjust iteration counter
COVARIANCE=cov(cova.hat(:,:,iteration+1)');
ESTIMATION=cova.hat(:,:,iteration+1);

function [COVARIANCE,ESTIMATION]=CovEM2(RT,iterationmax)
% In order to use the QP approach, the covariance of the Library data and
% its inverse is required. However, it is not rare to see lipid species not
% detected in certain matrices. Therefore, LITL estimates the missing
% values in the Library by using an EM algorithm described in
% [1] (Little, R. J. A., et al. (2002). Multivariate Normal Examples,
% Ignoring the Missing-Data Mechanism. Statistical Analysis with Missing Data,
% John Wiley & Sons, Inc.: 221-252.)
% iteration : iteration counter
% cova : structure containing the below described matrices
% cova.raw : transitions-by-samples matrix of retention times (with NaN values)
% cova.hat : transitions-by-samples matrix of retention times (NaN values estimated)
% cova.mu : transitions-by-1 vector of mean retention times (Non weighted)
% cova.sigma : transitions-by-transition estimated covariance matrix
% cova.tmp : temporarory transitions-by-1 column vector storing covariance
%            estimates for one sample
% cova.delta : 1-by-iterations vector of the max delta between iterations
% cova.loglikelihood : 1-by-iterations vector of the loglikelihood of the
%                      covariance matrix after each iteration
% ------------------------------------------------------------------------
cova.raw=RT;
iteration=1;
cova.hat(:,:,iteration)=cova.raw; %for first estimation
% first rough estimation of the nan values (average)
for tmp=1:size(cova.raw,1)
    tmpAve=nanmean(cova.raw(tmp,:));
    cova.mu(tmp,1)=tmpAve;
    for tmp2=1:size(cova.raw,2)
        if isnan(cova.raw(tmp,tmp2))==1
            cova.hat(tmp,tmp2,iteration)=tmpAve;
        end
    end
end
clear tmp tmp2 tmpAve
% first rough estimation of the covariance
cova.sigma=cov(cova.hat(:,:,iteration)',1);
for iteration=1:iterationmax
    %----
 if iteration~=1
 % Update mu and sigma estimated values
    % mu
    for tmp=1:size(cova.raw,1)
        tmpAve=nanmean(cova.hat(tmp,:,iteration));
        cova.mu(tmp,1)=tmpAve;
    end
    clear tmp tmp2 tmpAve
    % sigma
    cova.sigma=cov(cova.hat(:,:,iteration)',1);
    cova.sigma2=cov(cova.hat(:,:,iteration)',1)+10^-10*eye(size(cova.sigma,1));
    % For diagnostic : loglikelihood
    tmp=mvnpdf(cova.hat(:,:,iteration)',cova.mu',cova.sigma2);
    tmp=log(tmp);
```

```
    cova.loglikelihood(iteration-1)=sum(tmp);
    clear tmp
 end
 %----
    % E-step Yt
    for samples=1:size(cova.raw,2)
        Species=length(cova.mu);
        G=[1 cova.mu'; cova.mu cova.sigma+cova.mu*cova.mu'];
        G=Sweep(G,1);
        % looping around the species (rows) and sweeping if not NaN
        for i=1:Species
            if ~isnan(cova.raw(i,samples))
                G=Sweep(G,i+1);
            end
        end
        cova.tmp=cova.raw(:,samples);
        for i=1:Species
            if isnan(cova.raw(i,samples))
                cova.tmp(i)=G(1,i+1);
                for j=1:Species
                    if ~isnan(cova.raw(j,samples))
                        cova.tmp(i)=cova.tmp(i)+cova.raw(j,samples)*G(j+1,i+1);
                    end
                end
            end
        end
        cova.hat(:,samples,iteration+1)=cova.tmp; % assign the estimated sample
    end
end
% output variables
COVARIANCE=cov(cova.hat(:,:,iteration+1)');
ESTIMATION=cova.hat(:,:,iteration+1);

function S=Sweep(M, k)
% sweep operator as described in section 7.4.3 in the book Statistical Analysis with
Missing Data Second Edition
% For a symmetric matrix j-by-l "M", and an integer sweep argument "k", the sweep
% operator will apply the following transformation to the matrix M:
% if j=k=l
%   Sweeped(j,l)= -1/M(j,l)
% if j~=k XOR l~=k
%   Sweeped(j,l)= M(j,k)/M(k,k) (if j~=k)
%   Sweeped(j,l)= M(l,k)/M(k,k) (if l~=k)
% if j~=k AND l~=k
%   Sweeped(j,l)= M(j,l) - M(j,k)*M(k,l)/M(k,k)
% Creating Sweeped matrix
S=zeros(size(M,1),size(M,1));
% Sweeping
for j=1:size(M,1)
    for l=1:j
        if j==k && l==k
            S(j,l) = -1/M(j,l);
        elseif j~=k && l~=k
            S(j,l) = M(j,l) - M(j,k)*M(k,l)/M(k,k);
            S(l,j) = S(j,l);
        elseif j~=k && l==k
            S(j,l) = M(j,k)/M(k,k);
            S(l,j) = S(j,l);
        elseif l~=k && j==k
            S(j,l) = M(l,k)/M(k,k);
            S(l,j) = S(j,l);
        end
    end
```

```
end

function [rawExpData,library,expData]=LoessSmoothing(library,expData,X)
% Smoothes retention times based on "anchor" points set to unambiguous assignments
(standards and points that are unique within a 4 sd span)
%% Data needed
% library.mz
% library.rt
% expData{:,1}(:,2)
% library.mean
% library.va --> for sd
% X (smoothing setting)
sdMAT=[];
for tmp=1:size(library.mean,1) % every point
    sdMAT(tmp,1)=library.mz(tmp,1); % mz
    sdMAT(tmp,2)=library.mean(tmp,1); % raw mean
    sdMAT(tmp,3)=library.mean(tmp,1) - 4*sqrt(library.va(tmp,1)); % -4sd
    sdMAT(tmp,4)=library.mean(tmp,1) + 4*sqrt(library.va(tmp,1)); % +4sd
end
currMZ=0;
for tmp=1:size(library.mean,1) % every point
    if currMZ==sdMAT(tmp,1) % same mz as before, check sd span
        if sdMAT(tmp-1,4) > sdMAT(tmp,3) % sd span overlap
            sdMAT(tmp-1,5)=0;
            sdMAT(tmp,5)=0;
        end
    else
        sdMAT(tmp,5)=1;
    end
    currMZ=sdMAT(tmp,1);
end
forLoessLib=[];
for tmp=1:size(library.rt,2) %every library sample
    for tmp2=1:size(library.rt,1) %every data point in sample
        if sdMAT(tmp2,5)==0 % can't use
            forLoessLib(tmp2,tmp)=NaN;
        elseif sdMAT(tmp2,5)==1 % can use
            if isnan(library.rt(tmp2,tmp))
                forLoessLib(tmp2,tmp)=NaN;
            elseif library.rt(tmp2,tmp) >= sdMAT(tmp2,3) && library.rt(tmp2,tmp) <=
sdMAT(tmp2,4)
                forLoessLib(tmp2,tmp)=library.mean(tmp2,1);
            end
        end
    end
end
forLoessExp=[];
forLoessExpLib=[];
for tmp=1:size(expData,1) %every expData sample
    tmpSD=1;
    tmp2=1;
    while tmp2<=size(expData{tmp,1},1) %every data point in sample
        currMZ=expData{tmp,1}(tmp2,2);
        if floor(currMZ)+0.5 <= sdMAT(tmpSD,1) && floor(currMZ)+0.5 >= sdMAT(tmpSD,1)
% mz match
            if expData{tmp,1}(tmp2,3) >= sdMAT(tmpSD,3) && expData{tmp,1}(tmp2,3) <=
sdMAT(tmpSD,4)
                if sdMAT(tmpSD,5)==0 % can't use
                    forLoessExp(tmp2,tmp)=NaN;
                elseif sdMAT(tmpSD,5)==1 % can use
                    forLoessExp(tmp2,tmp)=sdMAT(tmpSD,2);
                end
            else
```

```
                forLoessExp(tmp2,tmp)=NaN;
            end
            tmp2=tmp2+1;
        elseif currMZ < sdMAT(tmpSD,1)
            forLoessExp(tmp2,tmp)=NaN;
            tmp2=tmp2+1;
        elseif currMZ > sdMAT(tmpSD,1)
            forLoessExp(tmp2,tmp)=NaN;
            tmpSD=tmpSD+1;
        elseif expData{tmp,1}(tmp2,3) < sdMAT(tmpSD,3) % RT too small
            forLoessExp(tmp2,tmp)=NaN;
            tmp2=tmp2+1;
        elseif expData{tmp,1}(tmp2,3) > sdMAT(tmpSD,4) % RT too big
            forLoessExp(tmp2,tmp)=NaN;
            tmpSD=tmpSD+1;
        end
    end
end
forLoessExp(forLoessExp==0)=NaN;
%% For loess
rawExpData=expData;
for tmp=1:size(expData,1) % all samples
    [expRTSort,I] = sort(rawExpData{tmp,1}(:,3),'ascend');
    libRTSort = forLoessExp(I,tmp);
    newRTSort = smooth(expRTSort,libRTSort,X,'loess');
    for tmp2=1:size(newRTSort,1)
        expData{tmp,1}(tmp2,3)=newRTSort(find(I==tmp2),1);
    end
    % h = plot(expRTSort,expData{tmp,1}(:,3),'k-','linewidth',1);
end
% figure
% hold on
% plot(expData{37,1}(:,3),'go')
% plot(rawExpData{37,1}(:,3),'ro')
library.raw=library.rt;
for tmp=1:size(library.rt,2) % all samples
    [RTSort,I] = sort(library.raw(:,tmp),'ascend');
    libRTSort = forLoessLib(I,tmp);
    newRTSort = smooth(RTSort,libRTSort,X,'loess');
    for tmp2=1:size(newRTSort,1)
        library.rt(tmp2,tmp)=newRTSort(find(I==tmp2),1);
    end
end

function [Assignment]=outerJA0(I,library,V)
% computes assignment based on the library, computes log objective function
Assignment.logobjF = [- 100000000, -10000000000];
Z=1;
X={}; % contains 1 col current assignment, other subsequent columns all permutations
Y=[]; % number of permutations per m/z
W=[];
I=I(~cellfun('isempty',I));
for tmp=1:size(I,2)
    tmp2=1;
    if size(I{1,tmp}{1,1},2)==1
        for tmp4=1:size(I{1,tmp}{2,1},2)
            X{tmp,tmp4+1}=tmp4;
        end
        Y(tmp,1)=1;
        Y(tmp,2)=size(I{1,tmp}{2,1},2);
    else
        nbre=min(size(I{1,tmp}{1,1},2),size(I{1,tmp}{2,1},2));
        permutations=nchoosek([1:size(I{1,tmp}{2,1},2)],nbre);
```

```
        for tmp4=1:size(permutations,1)
            X{tmp,tmp4+1}=permutations(tmp4,:);
        end
        Y(tmp,1)=1;
        Y(tmp,2)=size(permutations,1);
    end
end
% default
for tmp=1:size(Y,1)
    X{tmp,1}=1:min(size(I{1,tmp}{1,1},2),size(I{1,tmp}{2,1},2));
end
W=NaN(size(X,1),size(X,2));
for tmp=1:size(Y,1) % for each unique mz
    for tmp2=1:Y(tmp,2) % for each possibility
        for tmp3=1:size(X{tmp,1+tmp2},2) % for each matched
            if isnan(W(tmp,tmp2+1))
                W(tmp,tmp2+1)=abs( I{1,tmp}{1}(1,tmp3) -
I{1,tmp}{2}(1,X{tmp,1+tmp2}(tmp3)) );
            else
                W(tmp,tmp2+1)=W(tmp,tmp2+1) + abs( I{1,tmp}{1}(1,tmp3) -
I{1,tmp}{2}(1,X{tmp,1+tmp2}(tmp3)) );
            end
        end
    end
    [~,X{tmp,1}]=min(W(tmp,:));
    X{tmp,1}=X{tmp,X{tmp,1}};
end
% library assignment (X)
tmp2=1;
for tmp=1:size(V,1)
   if tmp~=1
       if V(tmp,2)==V(tmp-1,2) % same m/z, add to assignment
           X{tmp2,1}=[X{tmp2,1} V(tmp,1)];
       else
           tmp2=tmp2+1;
           X{tmp2,1}=V(tmp,1);
       end
   else
       X{tmp2,1}=V(tmp,1);
   end
end
% Default JA
RT=[];
LIB=[];
tmp3=1;
indCOV=[];
for tmp=1:size(I,2)
    tmp2=1;
    [RT,LIB,indCOV,tmp2,tmp3]=JA(I,RT,LIB,indCOV,tmp,tmp2,tmp3,X{tmp,1});
end
% Computing COV and iCOV of default JA
COV=library.cov(indCOV,indCOV);
iCOV=inv(COV);
% Computing log objective function value of default JA
logobjF = [];
logobjF (1) = ( -(size(RT,1)/2)*log(2*pi) - 0.5*sum(log(eig(COV))) + (-(LIB -
RT)'*(iCOV)*(LIB - RT)/2) );
logobjF (2) = -(size(RT,1)/2)*log(2*pi) - 0.5*sum(log(eig(COV)));
% default JA
Assignment.RT=RT;
Assignment.LIB=LIB;
Assignment.indCOV=indCOV;
Assignment.X=X(:,1);
```

```
Assignment.logobjF=logobjF;
end

function [Assignment]=outerJA1(I,library,V)
% computes JA by local search tweaks, 1 tweak at a time,
% ascending on m/z dimension -- computes log objective function value
Assignment.logobjF = [- 100000000, -10000000000];
Z=1;
X={}; % contains 1 col current assignment, other subsequent columns all permutations
Y=[]; % number of permutations per m/z
W=[];
I=I(~cellfun('isempty',I));
for tmp=1:size(I,2)
    tmp2=1;
    if size(I{1,tmp}{1,1},2)==1
        for tmp4=1:size(I{1,tmp}{2,1},2)
            X{tmp,tmp4+1}=tmp4;
        end
        Y(tmp,1)=1;
        Y(tmp,2)=size(I{1,tmp}{2,1},2);
    else
        nbre=min(size(I{1,tmp}{1,1},2),size(I{1,tmp}{2,1},2));
        permutations=nchoosek([1:size(I{1,tmp}{2,1},2)],nbre);
        for tmp4=1:size(permutations,1)
            X{tmp,tmp4+1}=permutations(tmp4,:);
        end
        Y(tmp,1)=1;
        Y(tmp,2)=size(permutations,1);
    end
end
% default
for tmp=1:size(Y,1)
    X{tmp,1}=1:min(size(I{1,tmp}{1,1},2),size(I{1,tmp}{2,1},2));
end
W=NaN(size(X,1),size(X,2));
for tmp=1:size(Y,1) % for each unique mz
    for tmp2=1:Y(tmp,2) % for each possibility
        for tmp3=1:size(X{tmp,1+tmp2},2) % for each matched
            if isnan(W(tmp,tmp2+1))
                W(tmp,tmp2+1)=abs( I{1,tmp}{1}(1,tmp3) -
I{1,tmp}{2}(1,X{tmp,1+tmp2}(tmp3)) );
            else
                W(tmp,tmp2+1)=W(tmp,tmp2+1) + abs( I{1,tmp}{1}(1,tmp3) -
I{1,tmp}{2}(1,X{tmp,1+tmp2}(tmp3)) );
            end
        end
    end
    [~,X{tmp,1}]=min(W(tmp,:));
    X{tmp,1}=X{tmp,X{tmp,1}};
end
% library assignment (X)
tmp2=1;
for tmp=1:size(V,1)
   if tmp~=1
       if V(tmp,2)==V(tmp-1,2) % same m/z, add to assignment
           X{tmp2,1}=[X{tmp2,1} V(tmp,1)];
       else
           tmp2=tmp2+1;
           X{tmp2,1}=V(tmp,1);
       end
   else
       X{tmp2,1}=V(tmp,1);
   end
```

```
end
% Default JA
RT=[];
LIB=[];
tmp3=1;
indCOV=[];
for tmp=1:size(I,2)
    tmp2=1;
    [RT,LIB,indCOV,tmp2,tmp3]=JA(I,RT,LIB,indCOV,tmp,tmp2,tmp3,X{tmp,1});
end
% Computing COV and iCOV of default JA
COV=library.cov(indCOV,indCOV);
iCOV=inv(COV);
% Computing log objective function value of default JA
logobjF = [];
logobjF (1) = ( -(size(RT,1)/2)*log(2*pi) - 0.5*sum(log(eig(COV))) + (-(LIB -
RT)'*(iCOV)*(LIB - RT)/2) );
logobjF (2) = -(size(RT,1)/2)*log(2*pi) - 0.5*sum(log(eig(COV)));
% default JA
Assignment.RT=RT;
Assignment.LIB=LIB;
Assignment.indCOV=indCOV;
Assignment.X=X(:,1);
Assignment.logobjF=logobjF;
changed=1; % change
while changed==1 % until nothing changes (changed=1 when new tweak recorded)
    Assignment.logobjF(1)
    changed=0;
    for counter1=1:size(Y,1) % each row
        for counter2=1:Y(counter1,2) % each possibility
            NewY=Y;
            NewY(counter1,1)=counter2;
            % Applying current JA
            for tmp=1:size(Y,1)
                X{tmp,1}=X{tmp,1+NewY(tmp,1)};
            end
            % Computing RT, LIB, and indCOV of current JA
            RT=[];
            LIB=[];
            tmp3=1;
            indCOV=[];
            for tmp=1:size(I,2)
                tmp2=1;
                [RT,LIB,indCOV,tmp2,tmp3]=JA(I,RT,LIB,indCOV,tmp,tmp2,tmp3,X{tmp,1});
            end
            % Computing COV and iCOV of current JA
            COV=library.cov(indCOV,indCOV);
            iCOV=inv(COV);
            % Computing log objective function value of current JA
            logobjF = [];
            logobjF (1) = ( -(size(RT,1)/2)*log(2*pi) - 0.5*sum(log(eig(COV))) + (-
(LIB - RT)'*(iCOV)*(LIB - RT)/2) );
            logobjF (2) = -(size(RT,1)/2)*log(2*pi) - 0.5*sum(log(eig(COV)));
            % Comparing to best JA
            if logobjF(1)>Assignment.logobjF(1)
                Assignment.RT=RT;
                Assignment.LIB=LIB;
                Assignment.indCOV=indCOV;
                Assignment.X=X(:,1);
                Assignment.logobjF=logobjF;
                Y=NewY;
                changed=1;
            end
```

```
        end
    end
end
RT=Assignment.RT;
LIB=Assignment.LIB;
indCOV=Assignment.indCOV;
end

function [Assignment]=outerJA2(I,library)
% computes JA based on possible JA (minimum RT difference), then local search tweaks,
1 tweak at a time,
% ascending on m/z dimension -- computes log objective function value
Assignment.logobjF = [- 100000000, -10000000000];
Z=1;
X={}; % contains 1 col current assignment, other subsequent columns all permutations
Y=[]; % number of permutations per m/z
W=[];
I=I(~cellfun('isempty',I));
for tmp=1:size(I,2)
    tmp2=1;
    if size(I{1,tmp}{1,1},2)==1
        for tmp4=1:size(I{1,tmp}{2,1},2)
            X{tmp,tmp4+1}=tmp4;
        end
        Y(tmp,1)=1;
        Y(tmp,2)=size(I{1,tmp}{2,1},2);
    else
        nbre=min(size(I{1,tmp}{1,1},2),size(I{1,tmp}{2,1},2));
        permutations=nchoosek([1:size(I{1,tmp}{2,1},2)],nbre);
        for tmp4=1:size(permutations,1)
            X{tmp,tmp4+1}=permutations(tmp4,:);
        end
        Y(tmp,1)=1;
        Y(tmp,2)=size(permutations,1);
    end
end
% default
for tmp=1:size(Y,1)
    X{tmp,1}=1:min(size(I{1,tmp}{1,1},2),size(I{1,tmp}{2,1},2));
end
W=NaN(size(X,1),size(X,2));
for tmp=1:size(Y,1) % for each unique mz
    for tmp2=1:Y(tmp,2) % for each possibility
        for tmp3=1:size(X{tmp,1+tmp2},2) % for each matched
            if isnan(W(tmp,tmp2+1))
                W(tmp,tmp2+1)=abs( I{1,tmp}{1}(1,tmp3) -
I{1,tmp}{2}(1,X{tmp,1+tmp2}(tmp3)) );
            else
                W(tmp,tmp2+1)=W(tmp,tmp2+1) + abs( I{1,tmp}{1}(1,tmp3) -
I{1,tmp}{2}(1,X{tmp,1+tmp2}(tmp3)) );
            end
        end
    end
    [~,X{tmp,1}]=min(W(tmp,:));
    X{tmp,1}=X{tmp,X{tmp,1}};
end
% Default JA
RT=[];
LIB=[];
tmp3=1;
indCOV=[];
for tmp=1:size(I,2)
    tmp2=1;
```

```
    [RT,LIB,indCOV,tmp2,tmp3]=JA(I,RT,LIB,indCOV,tmp,tmp2,tmp3,X{tmp,1});
end
% Computing COV and iCOV of default JA
COV=library.cov(indCOV,indCOV);
iCOV=inv(COV);
% Computing log objective function value of default JA
logobjF = [];
logobjF (1) = ( -(size(RT,1)/2)*log(2*pi) - 0.5*sum(log(eig(COV))) + (-(LIB -
RT)'*(iCOV)*(LIB - RT)/2) );
logobjF (2) = -(size(RT,1)/2)*log(2*pi) - 0.5*sum(log(eig(COV)));
% default JA
Assignment.RT=RT;
Assignment.LIB=LIB;
Assignment.indCOV=indCOV;
Assignment.X=X(:,1);
Assignment.logobjF=logobjF;
changed=1; % change
while changed==1 % until nothing changes (changed=1 when new tweak recorded)
    Assignment.logobjF(1)
    changed=0;
    for counter1=1:size(Y,1) % each row
        for counter2=1:Y(counter1,2) % each possibility
            NewY=Y;
            NewY(counter1,1)=counter2;
            % Applying current JA
            for tmp=1:size(Y,1)
                X{tmp,1}=X{tmp,1+NewY(tmp,1)};
            end
            % Computing RT, LIB, and indCOV of current JA
            RT=[];
            LIB=[];
            tmp3=1;
            indCOV=[];
            for tmp=1:size(I,2)
                tmp2=1;
                [RT,LIB,indCOV,tmp2,tmp3]=JA(I,RT,LIB,indCOV,tmp,tmp2,tmp3,X{tmp,1});
            end
            % Computing COV and iCOV of current JA
            COV=library.cov(indCOV,indCOV);
            iCOV=inv(COV);
            % Computing log objective function value of current JA
            logobjF = [];
            logobjF (1) = ( -(size(RT,1)/2)*log(2*pi) - 0.5*sum(log(eig(COV))) + (-
(LIB - RT)'*(iCOV)*(LIB - RT)/2) );
            logobjF (2) = -(size(RT,1)/2)*log(2*pi) - 0.5*sum(log(eig(COV)));
            % Comparing to best JA
            if logobjF(1)>Assignment.logobjF(1) % better assignment
                Assignment.RT=RT;
                Assignment.LIB=LIB;
                Assignment.indCOV=indCOV;
                Assignment.X=X(:,1);
                Assignment.logobjF=logobjF;
                Y=NewY;
                changed=1;
            end
        end
    end
end
RT=Assignment.RT;
LIB=Assignment.LIB;
indCOV=Assignment.indCOV;
end
% tmp : 1:size(possibilities,1), current lipid in RT
```

```
% tmp2 : 1:min(size(I{1,tmp}{1,1},2),size(I{1,tmp}{2,1},2)), minimum in either RT or
LIB at this mz
% tmp3 : index in RT and LIB
% X : which LIB for current tmp RT
% IF TMP
% IF TMP2

function [RT,LIB,indCOV,tmp2,tmp3]=JA(I,RT,LIB,indCOV,tmp,tmp2,tmp3,X)
for X_count=1:size(X,2)
    RT(tmp3,1)=I{1,tmp}{1,1}(1,tmp2); % retrieves current RT
    LIB(tmp3,1)=I{1,tmp}{2,1}(1,X(X_count)); % retrieves assigned LIB
    indCOV=horzcat(indCOV,I{1,tmp}{2,1}(2,X(X_count))); % retrieves index of assigned
LIB
    tmp2=tmp2+1;
    tmp3=tmp3+1;
end
end

function [Corrs,RTE] =
BiophysicalLines_forGUI(a,location,input_file,sample_adjust,library,CorrelationThresho
ld)
% Extract file stem, create output directory, if needed. Empty it, if needed.
FileStem = 'Graphs'; %XlsFileName(1:(I(end)-1));
mkdir(location,FileStem);
% First, we need to get the data in
Data.Barcode = input_file(7:end,1);
for tmp=1:size(Data.Barcode,1)
    index=find(not(cellfun('isempty', strfind(library.barcode,Data.Barcode{tmp,1}))));
    Data.MZ(tmp,1)=library.mz(index);
    Data.GeneralID(tmp,1)=library.genID(index);
end
Data.Sample = input_file(a,sample_adjust:end);
Data.RT = input_file(7:end,sample_adjust:end);
emptyIndex = cellfun(@isempty,Data.RT);
Data.RT(emptyIndex)={NaN};
Data.RT=cell2mat(Data.RT);
% Identify lipid groups
GroupNames = GetGroupNames(Data.GeneralID);
% Open latex file output
fid = fopen([location,'/',FileStem,'/BiophysicalCurves.tex'],'w');
fprintf(fid,'\\documentclass{article}\n');
fprintf(fid,'\\evensidemargin=0in\n');
fprintf(fid,'\\oddsidemargin=0in\n');
fprintf(fid,'\\textwidth=6.5in\n');
fprintf(fid,'\\usepackage{epsfig}\n');
fprintf(fid,'\\begin{document}\n');
% Analyze mean behaviour
figure()
Corrs = GroupedLogRTvsMZ(GroupNames,Data.GeneralID,Data.MZ,nanmean(Data.RT,2));
RTE = RTErrors(GroupNames,Data.GeneralID,Data.MZ,nanmean(Data.RT,2));
ShowGroupedMZRT(Data.Barcode,Data.GeneralID,Data.MZ,nanmean(Data.RT,2),GroupNames,Corr
s,RTE,CorrelationThreshold);
title('Mean across all samples');
print('-f1','-depsc2',[location,'/',FileStem,'/MeanCurves.eps']);
fprintf(fid,'\\epsfig{file=MeanCurves.eps,width=\\textwidth}\n');
close()
% Analyzing individual samples
for i=1:length(Data.Sample)
figure()
    Corrs = GroupedLogRTvsMZ(GroupNames,Data.GeneralID,Data.MZ,Data.RT(:,i));
    RTE = RTErrors(GroupNames,Data.GeneralID,Data.MZ,Data.RT(:,i));
```

```
ShowGroupedMZRT(Data.Barcode,Data.GeneralID,Data.MZ,Data.RT(:,i),GroupNames,Corrs,RTE,
CorrelationThreshold);
    title(Data.Sample{i},'interpreter','none');
    SampStr = num2str(10000+i);
    SampStr = SampStr(2:end);
    print('-f1','-depsc2',[location,'/',FileStem,'/Sample',SampStr,'Curves.eps']);
    fprintf(fid,['\\epsfig{file=Sample',SampStr,'Curves.eps,width=\\textwidth}\n']);
    close()
end
fprintf(fid,'\\end{document}\n');
fclose(fid);
end
function GroupNames = GetGroupNames(GeneralID)
GroupNames = {};
for i=1:length(GeneralID)
    if any(GeneralID{i}=='X')
        GroupNames{end+1,1} = GeneralID{i};
    end
end
GroupNames = unique(GroupNames);
end
% Corrs are the correlations of log10(RTs)to MZs for groups of lipids
function Corrs = GroupedLogRTvsMZ(GroupNames,GeneralID,MZ,RT)
Corrs = ones(length(GroupNames),1);
for i=1:length(GroupNames)
    % Find lipids in the group
    G = GroupNames{i};
    I = find(strcmp(G,GeneralID) & ~isnan(RT));
    if length(I)>=2
        % Find their MZ and mean RTs
        GroupMZ = MZ(I);
        GroupRT = RT(I);
        % Compute correlations
        R = corrcoef(GroupMZ,log(GroupRT));
        Corrs(i) = R(1,2);
    end
end
end
% RT errors are computed by looking at each group individually. Within the
% group, each point is left out in turn, and a regression is constructed
% using the other points. The error is the difference in observed RT versus
% predicted by the line.
function RTE = RTErrors(GroupNames,GeneralID,MZ,RT)
RTE = zeros(length(MZ),1);
CorrWithout = ones(length(MZ),1);
for i=1:length(GroupNames)
    % Find lipids in the group
    G = GroupNames{i};
    I = find(strcmp(G,GeneralID) & ~isnan(RT));
    % Proceed only if have at least 3 points to work with
    if length(I)>=3
        % Loop through each point
        for j=1:length(I)
            % Find the indeces of points not being excluded
            Iuse = setdiff(I,I(j));
            % Find their MZ and mean RTs
            GroupMZ = MZ(Iuse);
            GroupRT = RT(Iuse);
            % Do the regression
            B = regress(log10(GroupRT),[GroupMZ GroupMZ*0+1]);
            % Make the prediction
            RTpred = 10.^(B(2)+B(1)*MZ(I(j)));
```

```
            % Compute error
            RTE(I(j)) = RT(I(j))-RTpred;
        end
    end
end
end
function [] =
ShowGroupedMZRT(Barcode,GeneralID,MZ,RT,GroupNames,Corrs,RTE,CorrelationThreshold)
% Plots RT-vs-MZ with lines connecting groups of lipids having the same
% GeneralID.
clf;
% Show the groups with their curves
for GoodBad=1:2
    for i=1:length(GroupNames)
        % Find lipids in the group
        G = GroupNames{i};
        I = find(strcmp(G,GeneralID) & ~isnan(RT));
        if ~isempty(I)
            % Find their MZ and mean RTs
            GroupMZ = MZ(I);
            GroupRT = RT(I);
            % Sorting
            [GroupMZ,J] = sort(GroupMZ);
            GroupRT = GroupRT(J);
            % Plotting -- funny trick here to get the red curves plotted on
            % top
            hold on;
            if GoodBad==1 & Corrs(i)>=CorrelationThreshold
                h = plot(GroupMZ,GroupRT,'ko-
','markerfacecolor','k','markersize',3,'linewidth',1);
            end
            if GoodBad==2 & Corrs(i)<CorrelationThreshold
                h = plot(GroupMZ,GroupRT,'ro-
','markerfacecolor','r','markersize',3,'linewidth',1);
                [MaxErr,MaxErri] = max(abs(RTE(I)));
                h = text(GroupMZ(MaxErri),GroupRT(MaxErri),[' ',Barcode{I(MaxErri)},'
',GeneralID{I(MaxErri)},' MZ',num2str(MZ(I(MaxErri))),'
RT',num2str(RT(I(MaxErri)))],'color','r');
            end
            hold off;
        end
    end
end
xlabel('M/Z');
ylabel('Retention time');
set(gca,'box','on');
end

function PrismScriptAuto(analysis_Data,output_Location,date_m8)
% takes analysis_data
% (2 + number of samples) by (6 + number of species) cell with pmol (db)
% ------------------------------------------------------------------------
%                      initialization + dictionnary
% ------------------------------------------------------------------------
% x_Adjust : double, corrects iterable x for deleted sheets
% analysis_Button : characters, holds yes or no, leads to assign
% analysis_Data : 6+transitions-by-2+samples cell containing the data to be
%                 graphed, up to 5 features and the name of each samples,
%                 and the ID and the MZ of all the transitions to be graphed
% analysis_Stop : double, holds 0 (continue scripting) or 1 (stop scripting)
% date_m8 : 1-by-1 cell, contains string to name Output folder (will
%           be concatenated to "_Output")
% data_Index : cell containing row-vector of doubles corresponding to the
```

```
%              index of each sample (in analysis_Data) in each analysed group
% analysis_Location : cell containing a string specifying the output folder
%                     path
% transition_Names : 1-by-transitions cell containing the ID of each
% output_Location : character string specifying the output folder path
% analysis_Counter : double, counting the prism files (different analyses)
% sheet_Counter : double, counting the sheets (within one analysis)
% transition_Max : double, number of transitions to analyse
% analysis_Type : cell containing a string specifying the type of analysis
% script : cell containing the lines (strings) of prism script output
% script_Counter : double, counting the script files (within one analysis)
% ------------------------------------------------------------------------
TypeOfAnalysis={};
analysis_Location={};
Traits={};
%------------------------------------------------------------------------
for x=3:size(analysis_Data,1)
    for y=3:size(analysis_Data,2)
        if isnan(analysis_Data{x,y})==1
            analysis_Data{x,y}={};
        end
    end
end
% % date_m8=inputdlg('Please enter the date in yyyymmdd format.','Hey you',1);
% % output_Location = uigetdir('','Where should I create the output folder?');
% % mkdir(output_Location,char(strcat(date_m8,'_Output')))
%--------------
analysis_Stop=0;
analysis_Counter=0;
while analysis_Stop==0
    clear selection1;clear selection2;clear selection3;clear row;clear col;
    analysis_Counter=analysis_Counter+1;
    %------------------------------------------------------------------------
    analysis_Button = questdlg('Please choose the statistical test','Test
selection','T-test','One-way anova','Two-way anova','Two-way anova');
    switch analysis_Button
        case 'T-test'
            a=1;
        case 'One-way anova'
            a=1;
        case 'Two-way anova'
            a=2;
        case 'Stop the analysis'
            error('Stopping...')
    end
    %------------------------------------------------------------------------
    if a==1 %One-way or t-test
        clear data_Index
        trait={};
        traits={};
        for a=1:5
            temp=unique(analysis_Data(a,3:end));
            for b=1:size(temp,2)
                trait(a,b)=temp(b); % gets the traits
                if b==1
                    traits{a}=strcat([trait{a,b}]);
                elseif b==size(temp,2)
                    traits{a}=strcat([traits{a} ', ' trait{a,b} '.']);
                else
                    traits{a}=strcat([traits{a} ', ' trait{a,b}]);
                end
            end
        end
```

```
        selection1 = listdlg('ListString',traits,'Name','Trait
selection','SelectionMode','single','PromptString',[{'Please select the trait groups
you want to use for your'};{'t-test or one-way anova.'};{'Make sure the trait contains
two or more groups.'}],'ListSize',[300 160]);
        % selection1 = index of the trait groups in trait (row)
        selection2 = listdlg('ListString',char(trait{selection1,:}),'Name','Groups
selection','PromptString',[{'Please select the groups you want to use for your'};{'t-
test or one-way anova.'};{'Make sure you select two or more groups (hold the control
key to select multiple groups).'}],'ListSize',[300 160]);
        % selection2 = index of the groups in trait (column)
        if size(selection2,2)>2 % more than 2 groups, one-way anova
            analysis_Type={'oneway'};
        elseif size(selection2,2)==2 % 2 groups, t-test
            analysis_Type={'ttest'};
        else
            error('Why')
        end
        col(1) = listdlg('ListString',char(trait{selection1,selection2}),'Name','First
column','SelectionMode','single','PromptString',[{'Please select the group you want to
use as your first column.'};{'Note that it will be your control group for multiple
comparisons.'}],'ListSize',[300 160]);
        % col(1) = index of selection2 (which column index of selection2) being the
control group (goes 1st)
        for a=2:size(selection2,2)
            col(a) =
listdlg('ListString',char(trait{selection1,selection2}),'Name','Column
selection','SelectionMode','single','PromptString',[{'Please select the group you want
to use as your'};{'next column. Make sure to not select one group
twice!'}],'ListSize',[300 160]);
        end
        sheet_Counter=0;
        while (size(analysis_Data,1)-6)-(249*sheet_Counter)>0
            sheet_Counter=sheet_Counter+1;
            mkdir(char(strcat(output_Location,'\',date_m8,'\')),
char(strcat('forprism-',num2str(analysis_Counter),'-',analysis_Type,'-
',num2str(sheet_Counter))))
            temp=selection2(col(1));
            for a=2:size(selection2,2)
                temp=[temp selection2(col(a))];
            end
            t1=cell2table(trait(selection1,temp));
            writetable(t1,char(strcat(output_Location,'\',date_m8,'\forprism-
',num2str(analysis_Counter),'-',analysis_Type,'-
',num2str(sheet_Counter),'\t1.csv')),'Delimiter',',','WriteVariableNames',0)
            if size(analysis_Data,1)-6<(249*sheet_Counter)
                transition_Max=(size(analysis_Data,1)-6)-249*(sheet_Counter-1);
            else
                transition_Max=249;
            end
            for x=1:transition_Max
                for a=1:size(selection2,2) % index in analysis_data for each group

data_Index{a,:}=find(strcmp(t1{1,a},{analysis_Data{selection1,3:end}}));
                end
            end
            script={''};
            script = vertcat(script, ['SetPath "'
char(strcat(output_Location,'\',date_m8,'\forprism-',num2str(analysis_Counter),'-
',analysis_Type,'-',num2str(sheet_Counter))) '"']);
            script = vertcat(script, 'Open Template.pzfx');
            x_Adjust=0; % substracted from the sheet numbers, adjusting for the
missing sheets due to NaN
            for x=1:transition_Max
```

```
                script = vertcat(script, ['go to d' num2str(x-x_Adjust)]);
                script = vertcat(script, 'cleartable -1,:,:'); %only removes data and
column titles NOT ROWS (make sure template has properly labeled rows)
                script = vertcat(script, 'import T1.csv, 1');
                temp4=0; % print in script if at least one group contains data
                for a=1:size(selection2,2) % for each group
                    temp=analysis_Data((x+249*(sheet_Counter-1))+6,2+data_Index{a,:});
                    temp=analysis_Data((x+249*(sheet_Counter-1))+6,2+data_Index{a,:});
                    temp2={''};
                    temp3=0; % don't print in script if containing no data [ . . .]
                    for b=1:size(temp,2)
                        if isempty(temp{b})==1 % '.' missing value
                            temp2=[temp2, ' .'];
                        else % number
                            temp3=1; % print in script if contains data
                            temp2=[temp2, {' '}, num2str(temp{b})];
                        end
                    end
                    if temp3==1 % print in script if contains data
                        temp4=temp4+1; % print in script if contains data
                        temp=[temp2{:}];
                        script = vertcat(script, ['InsertData,1,' num2str(a)]);
                        script = vertcat(script, '<data>');
                        script = vertcat(script, temp);
                        script = vertcat(script, '</data>');
                    end
                end
                if temp4~=0 % print if at least one group contains data
                    script = vertcat(script, ['Goto G' num2str(x-x_Adjust)]);
                    script = vertcat(script, ['setGraphTitle "'
analysis_Data{(x+249*(sheet_Counter-1))+6,2} '" <k>']);
                    script = vertcat(script, ['go to d' num2str(x-x_Adjust)]);
                    script = vertcat(script, ['setTitle "'
analysis_Data{(x+249*(sheet_Counter-1))+6,2} '" <k>']);
                    script = vertcat(script, ['setSheetTitle "'
analysis_Data{(x+249*(sheet_Counter-1))+6,2} '" <k>']);
                    if x~=transition_Max
                        script = vertcat(script, 'duplicatefamily"", ');
                    end
                else
                    script(end-2:end,:)=[];
                    x_Adjust=x_Adjust+1;
                end
            end
            a=0;
            script_Counter=0;
            while a<size(script,1)
                nbytes_tot=0;
                script_Counter=script_Counter+1;
                testscript = fopen(char(strcat(output_Location,'\',date_m8,'\forprism-
',num2str(analysis_Counter),'-',analysis_Type,'-
',num2str(sheet_Counter),'\testscript',num2str(script_Counter),'.pzc')),'w');
                formatSpec = '%s\n';
                while nbytes_tot<60000 && a<size(script,1)
                    a=a+1;
                    nbytes=fprintf(testscript,formatSpec,script{a,:});
                    nbytes_tot=nbytes_tot+nbytes;
                end
                fclose(testscript);
            end
            traits_for_table=table2array(t1);
            traits_for_table=strrep(traits_for_table,'_',' ');
            traits_for_table=strjoin(traits_for_table,' / ');
```

```
            TypeOfAnalysis=[TypeOfAnalysis;analysis_Type];
            analysis_Location=[analysis_Location;char(strcat('forprism-
',num2str(analysis_Counter),'-',analysis_Type,'-',num2str(sheet_Counter)))];
            Traits=[Traits;traits_for_table];
        end
        %-------------------------------------------------------------------------
    elseif a==2 % TWO-WAY
        analysis_Type={'two-way'};
        clear data_Index
        trait={};
        traits={};
        for a=1:5
            if ischar(analysis_Data{a,3})==1
                temp=unique(analysis_Data(a,3:end));
                for b=1:size(temp,2)
                    trait(a,b)=temp(b); % gets the traits
                    if b==1
                        traits{a}=strcat([trait{a,b}]);
                    elseif b==size(temp,2)
                        traits{a}=strcat([traits{a} ', ' trait{a,b} '.']);
                    else
                        traits{a}=strcat([traits{a} ', ' trait{a,b}]);
                    end
                end
            end
        end
        selection1 = listdlg('ListString',traits,'Name','Trait
selection','PromptString',[{'Please select the two trait groups you want to use for
your'};{'two-way anova'};{'(hold the control key to select multiple groups).'};{'Make
sure the trait contains two or more groups.'}],'ListSize',[300 160]);
        % selection1 = index of the two trait groups in trait (rows)
        selection1col = listdlg('ListString',char(traits{selection1}),'Name','Column
factor','PromptString',[{'Please select the trait group you want to use'};{'as your
column factor in your two-way anova.'}],'ListSize',[300 160]);
        selection1row = listdlg('ListString',char(traits{selection1}),'Name','Row
factor','PromptString',[{'Please select the trait group you want to use'};{'as your
row factor in your two-way anova.'}],'ListSize',[300 160]);
        selection2 =
listdlg('ListString',char(trait{selection1(selection1col),:}),'Name','Column factor,
trait 1','PromptString',[{'Please select the groups you want to use as your'};{'column
factor in your two-way anova.'};{'Make sure you select a trait with two or more
groups'};{'(hold the control key to select multiple groups).'}],'ListSize',[300 160]);
        col(1) =
listdlg('ListString',char(trait{selection1(selection1col),selection2}),'Name','First
column','SelectionMode','single','PromptString',[{'Please select the group you want to
use as your first column.'};{' '}],'ListSize',[300 160]);
        % col(1) = index of selection2 (which column index of selection2) being the
control group (goes 1st)
        for a=2:size(selection2,2)
            col(a) =
listdlg('ListString',char(trait{selection1(selection1col),selection2}),'Name','Column
selection','SelectionMode','single','PromptString',[{'Please select the group you want
to use as your'};{'next column. Make sure to not select a group
twice!'}],'ListSize',[300 160]);
        end
        % selection2 = index of the groups in trait1 (column)
        selection3 =
listdlg('ListString',char(trait{selection1(selection1row),:}),'Name','Row factor,
trait 2','PromptString',[{'Please select the groups you want to use as your'};{'row
factor in your two-way anova.'};{'Make sure you select a trait with two or more
groups'};{'(hold the control key to select multiple groups).'}],'ListSize',[300 160]);
        row(1) =
listdlg('ListString',char(trait{selection1(selection1row),selection3}),'Name','First
```

```
row','SelectionMode','single','PromptString',[{'Please select the group you want to
use as your first row.'};{' '}],'ListSize',[300 160]);
        % row(1) = index of selection3 (which column index of selection3) being the
control group (goes 1st)
        for a=2:size(selection3,2)
            row(a) =
listdlg('ListString',char(trait{selection1(selection1row),selection3}),'Name','Row
selection','SelectionMode','single','PromptString',[{'Please select the group you want
to use as your'};{'next row. Make sure to not select a group twice!'}],'ListSize',[300
160]);
        end
        % selection3 = index of the groups in trait2 (row)

t1=strcat(trait(selection1(selection1col),col(1)),'_',trait(selection1(selection1row),
row(1)));
        c=0;
        for a=1:size(selection2,2)
            for b=1:size(selection3,2)
                if c==0
                    c=c+1;
                else

temp=strcat(trait(selection1(selection1col),col(a)),'_',trait(selection1(selection1row
),row(b)));
                    t1=[t1 temp];
                end
            end
        end
        t1=cell2table(t1);
        sheet_Counter=0;
        while (size(analysis_Data,1)-6)-(249*sheet_Counter)>0
            sheet_Counter=sheet_Counter+1;
            mkdir(char(strcat(output_Location,'\',date_m8,'\')),
char(strcat('forprism-',num2str(analysis_Counter),'-',analysis_Type,'-
',num2str(sheet_Counter))))
            %
writetable(t1,char(strcat(output_folder_path,'\',date_m8,'_Output\forprism-
',num2str(prism_analysis_counter),'-',prismtype,'-
',num2str(prism_sheet_counter),'\t1.csv')),'Delimiter',',','WriteVariableNames',0)
            temp=selection2(col(1));
            for a=2:size(selection2,2)
                temp=[temp selection2(col(a))];
            end
            t2=cell2table(trait(selection1(selection1col),temp));
            writetable(t2,char(strcat(output_Location,'\',date_m8,'\forprism-
',num2str(analysis_Counter),'-',analysis_Type,'-
',num2str(sheet_Counter),'\t2.csv')),'Delimiter',',','WriteVariableNames',0)
            temp=(trait(selection1(selection1row),selection3(row(1)) ));
            for a=2:size(selection3,2)
                temp=[temp; (trait(selection1(selection1row),selection3(row(a)) ))];
            end
            t3=cell2table(temp);
            writetable(t3,char(strcat(output_Location,'\',date_m8,'\forprism-
',num2str(analysis_Counter),'-',analysis_Type,'-
',num2str(sheet_Counter),'\t3.csv')),'Delimiter',',','WriteVariableNames',0)
            if size(analysis_Data,1)-6<(249*sheet_Counter)
                transition_Max=(size(analysis_Data,1)-6)-249*(sheet_Counter-1);
            else
                transition_Max=249;
            end
            for x=1:transition_Max
                november{x}=strrep(analysis_Data{(x+249*(sheet_Counter-1))+6,2},'
','_');
```

```
                for a=1:size(selection2,2) % for each 1st group

data_Index{a,1}=find(strcmp(trait(selection1(selection1col),selection2(col(a))),{analy
sis_Data{selection1(selection1col),3:end}}));
                    for b=1:size(selection3,2) % for each 2nd group

data_Index{b,2}=find(strcmp(trait(selection1(selection1row),selection3(row(b))),{analy
sis_Data{selection1(selection1row),3:end}}));
                    end
                end
            end
            script={''};
            script = vertcat(script, ['SetPath "'
char(strcat(output_Location,'\',date_m8,'\forprism-',num2str(analysis_Counter),'-
',analysis_Type,'-',num2str(sheet_Counter))) '"']);
            script = vertcat(script, 'Open Template.pzfx');
            %-------------------------------------------------------------------------
-
            x_Adjust=0;
            for x=1:transition_Max
                script = vertcat(script, ['go to d' num2str(x-x_Adjust)]);
                script = vertcat(script, 'cleartable ');
                script = vertcat(script, 'import T3.csv,1,-1');
                script = vertcat(script, 'import T2.csv,-1,1');
                temp4=0; % print in script if at least one group contains data
                for a=1:size(selection2,2) % for each 1st group
                    for b=1:size(selection3,2) % for each 2nd group

data_Index{b,2}=find(strcmp(trait(selection1(selection1row),selection3(row(b))),{analy
sis_Data{selection1(selection1row),3:end}}));
                        temp=analysis_Data((x+249*(sheet_Counter-
1))+6,2+intersect(data_Index{a,1},data_Index{b,2}));
                        temp2={''};
                        temp3=0; % don't print in script if containing no data [ . .
.]
                        for c=1:size(temp,2)
                            if isempty(temp{c})==1 % '.' missing value
                                temp2=[temp2, ' .'];
                            else % number
                                temp3=1; % print in script if contains data
                                temp2=[temp2, {' '}, num2str(temp{c})];
                            end
                        end
                        if temp3==1 % print in script if contains data
                            temp4=temp4+1; % print in script if contains data
                            temp=[temp2{:}];
                            script = vertcat(script, ['InsertData,' num2str(b) ','
num2str(a)]);
                            script = vertcat(script, '<data>');
                            script = vertcat(script, temp);
                            script = vertcat(script, '</data>');
                        end
                    end
                end
                if temp4~=0 % print if at least one group contains data
                    script = vertcat(script, ['Goto G' num2str(x-x_Adjust)]);
                    script = vertcat(script, ['setGraphTitle "'
analysis_Data{(x+249*(sheet_Counter-1))+6,2} '" <k>']);
                    script = vertcat(script, ['go to d' num2str(x-x_Adjust)]);
                    script = vertcat(script, ['setTitle "'
analysis_Data{(x+249*(sheet_Counter-1))+6,2} '" <k>']);
                    script = vertcat(script, ['setSheetTitle "'
analysis_Data{(x+249*(sheet_Counter-1))+6,2} '" <k>']);
```

```
                    if x~=transition_Max
                        script = vertcat(script, 'duplicatefamily"", ');
                    end
                else
                    script(end-3:end,:)=[];
                    x_Adjust=x_Adjust+1;
                    if x==transition_Max
                        if strcmp(script{end,1},'duplicatefamily"", ')
                            script(end,:)=[];
                        end
                    end
                end
            end
            %-------------------------------------------------------------------------
-
            a=0;
            script_Counter=0;
            while a<size(script,1)
                nbytes_tot=0;
                script_Counter=script_Counter+1;
                testscript = fopen(char(strcat(output_Location,'\',date_m8,'\forprism-
',num2str(analysis_Counter),'-',analysis_Type,'-
',num2str(sheet_Counter),'\testscript',num2str(script_Counter),'.pzc')),'w');
                formatSpec = '%s\n';
                while nbytes_tot<60000 && a<size(script,1)
                    b=0;
                    while b==0
                        if a<size(script,1)
                            a=a+1;
                        end
                        if contains(script{a,1},'cleartable')
                            b=1;
                        elseif a==size(script,1)
                            b=1;
                        else
                            b=0;
                        end
                        nbytes=fprintf(testscript,formatSpec,script{a,1});
                        nbytes_tot=nbytes_tot+nbytes;
                    end
                end
                fclose(testscript);
            end
        end
        traits_for_table=table2array(t1);
        traits_for_table=strrep(traits_for_table,'_',' ');
        traits_for_table=strjoin(traits_for_table,' / ');
        TypeOfAnalysis=[TypeOfAnalysis;analysis_Type];
        analysis_Location=[analysis_Location;char(strcat('forprism-
',num2str(analysis_Counter),'-',analysis_Type,'-',num2str(sheet_Counter)))];
        Traits=[Traits;traits_for_table];
        %-------------------------------------------------------------------------
    end
    %-----------------------------------------------------------------------
    if analysis_Counter>0
        analysis_Button = questdlg('Do you want to create prism scripts for
statistical analysis in Prism?','Prism scripts','Yes','No','No');
        switch analysis_Button
            case 'Yes'
                analysis_Stop=0;
            case 'No'
                analysis_Stop=1;
        end
```

```
    end
end
statsT=table();
statsT.TypeOfAnalysis=vertcat(TypeOfAnalysis);
statsT.Location=vertcat(analysis_Location);
statsT.Traits=vertcat(Traits);
writetable(statsT,char(strcat(output_Location,'\',date_m8,'\StatsSummary.csv')),'Delim
iter',',')
```