Insights from Automated and Untargeted Marine Microbial Metabolomics

William Kumler

2025-06-11

# Acknowledgements

* Lab
  + Anitra
  + Laura
  + Josh
  + Susan
  + Angie/Katherine/Frank
  + Regina
  + Natalie/Iris/Claudia/Raisha/Bryn
  + Leland/Lindsey/Everetta/Raafay/Natalie/Anna/Amy/Alec
* Broader Ocean/UW
  + Committee
  + Armbrust lab
  + Cohort
  + Bowling team
  + OCEAN front office
  + OSB cleaning staff
  + eScience (Dave and Bryna esp.)
  + Fisheries friends (SEAS+Helena/Zoe)
* Research things
  + Funding (ARCS/SCOPE)
  + SCOPE folks
    - Dave K/Sonya/Ed/Angel/Matt/Ben/Daniel
    - SCOPE Ops
    - Captain and crew of the KM
    - PERISCOPE team
  + R packages xcms/data.table/tidyverse/knitr/plotly/shiny/vegan/broom/arrow/duckdb/xml2/XML/Rcolorbrewer/viridis/ggh4x/ggtext/patchwork/pbapply/future/furrr
    - RStudio/Posit, R Markdown
  + Sisweb/Chemcalc/Metlin/HMDB/GNPS/Metabolights/Metabolomics Workbench
* Friends and family
  + Not all my time was spent on boats, in the lab, or the office
    - Stone Gardens/Edgeworks/Seattle Public Libraries/Seattle Aquarium
    - Land acknowledgement
  + Past mentors
    - Bethanie, Jim, Mimi, Pete
  + Kaylyn/Hannah/Shawnee/Wave/Kate/Sarra/Rachel Liu/Rachel Lai/Tasha/Sylvia/Colby/Allison/Este/Cassie?/Carly?/Tori?
  + Mom/Dad/Ben/Maggie/Gmas
* And you, dear reader?

# Chapter 1: Introduction

## Marine microbes

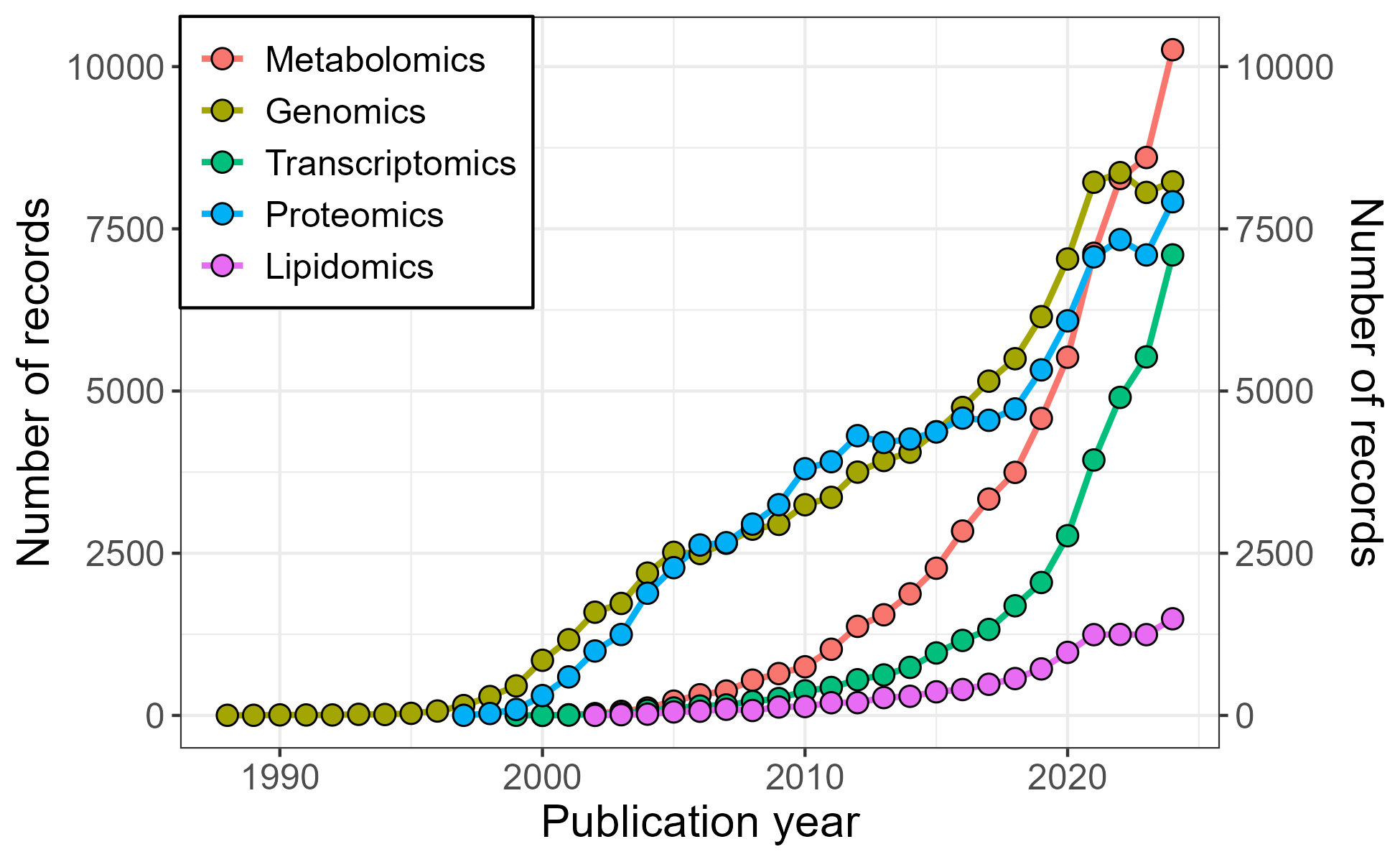
Marine carbon fixation happens at an incredible rate. In the blink of an eye (~100 milliseconds), the ocean converts a blue whale’s mass of atmospheric carbon into biomass and has performed this continuously for at least the last two billion years (Falkowski 1994; Ligrone 2019). Most of this is performed by single-celled organisms too small to see with the naked eye known as microbes (Falkowski 1994; Falkowski, Fenchel, and Delong 2008). The process by which they transform air and nutrients into food is the base of the marine food web and regulates Earth’s climate, with many fates available to the fixed carbon. A large fraction of this particulate matter will be transformed back into CO2 via respiration within the surface ocean, either by the phytoplankton themselves or the rest of the food chain. A smaller fraction makes it out of the euphotic zone via the biological pump and is sequestered for hundreds to thousands of years, while an even smaller fraction survives to the seafloor and can be sequestered for millenia in marine sediments (Iversen 2023; Siegel et al. 2023).

The pathway a particular atom of carbon travels is determined by the structure of the molecule it composes and the environment in which it’s found. Highly labile compounds such as sugars and amino acids can be converted almost instantaneously back into CO2, while ultra-refractory compounds can persist for thousands of years (Moran, Ferrer-González, et al. 2022). Our understanding of the marine environment’s biogeochemistry and community composition has vastly expanded in the past few decades thanks to the establishment of long-term ecological time series and advances in genetic tools, while our characterization of organic carbon lags far behind (Moran, Kujawinski, et al. 2022; Longnecker, Kido Soule, and Kujawinski 2024). Determining the molecular composition of marine carbon and its fluxes through the environment is therefore paramount in improving our ability to accurately model the microbial marine ecosystem (Jones et al. 2024).

## Metabolites and metabolomics

Metabolites are defined simply as the products of cellular metabolism, but this uncomplicated definition belies the dizzying complexity of microbial processes. While technically all biologically produced molecules could fall within this category, the conventional usage refers to the small (<1000 Dalton) organic molecules that act as currencies within the cell while excluding macromolecules such as proteins and lipids. Metabolites are often the reaction intermediates and building blocks of larger molecules but have several important roles of their own, including nutrient and energy storage (Van Mooy et al. 2009; Becker et al. 2018; Mojzeš et al. 2020), antioxidation (Narainsamy et al. 2016), osmotic balance (Yancey et al. 1982; Yancey 2005), buoyancy (Yancey 2005; Boyd and Gradmann 2002), and cell signaling (both beneficial and antagonistic interactions) Thukral, Allen, and Petras (2023). There are likely hundreds of thousands of individual molecules composing the metabolome in the environment, making their comprehensive analysis challenging (Schrimpe-Rutledge et al. 2016).

Nonetheless, metabolomics attempts to do so. The study of “all” small molecules in the cell is a rapidly growing field with over 10,000 publications in 2024 and recently eclipsed all other “omics” fields of study according to a topic search in Web of Science (Figure 1.1, Patti, Yanes, and Siuzdak (2012), Edwards (2023)). These publications span a massive swath of disciplines, with contributions from medicine, polymer chemistry, astronomy, and oceanography. This interdisciplinary nature has resulted in the construction of expansive databases linking organisms’ genetic potential to their realized state (Bauermeister et al. 2022; Kanehisa 2000; Karp et al. 2019).



*Figure 1.1: Number of publications indexed by Web of Science yearly since 1988 across different ’omics disciplines. Data were generated by searching the term in the legend as a topic and tabulated as a bar chart using the WOS Analyze Results option for Publication Years. All data rows were exported to CSV and plotted here using R’s ggplot2 library.*

Quantifying all small molecules in the cell is challenging for many reasons. First, metabolites span a wide range of chemical properties that cannot all be extracted simultaneously or separated on the same type of chromatography (Kido Soule et al. 2015; Cajka and Fiehn 2016; Gika et al. 2019). Second, their wide range of roles in the cell mean that annotating signals is more difficult than proteomics or lipidomics because their building blocks are not shared (Schrimpe-Rutledge et al. 2016). Third, the diversity and novelty of many compounds makes pure standards often unavailable, let alone isotopically-labeled versions necessary for the construction of the gold-standard multipoint internal calibration curve (Patti, Yanes, and Siuzdak 2012; Cajka and Fiehn 2016).

The problems listed above are exacerbated in marine microbial metabolomics. Primarily this is due to their incredibly low concentrations in both the particulate and dissolved phases, with typical values in the picomolar to nanomolar range (Heal et al. 2021; Sacks et al. 2022; Moran, Kujawinski, et al. 2022; Longnecker, Kido Soule, and Kujawinski 2024). An additional problem is the way the salty matrix of seawater behaves similarly to many metabolites during chemical analysis but numerically dominates their abundance by 105 to 1010 molecules per liter. (Boysen et al. 2018; Longnecker, Kido Soule, and Kujawinski 2024). In contrast to other metabolomics specialties where the organism of interest is well studied and genetically documented, environmental metabolomics struggles with a lack of genetic representation and less than 5% of the genetic diversity in the ocean has been captured by reference genomes (DeLong 2005; Salazar and Sunagawa 2017). Certainly fewer than 5% of the organisms in the ocean have been cultured in the lab and their metabolites documented, though work to improve this is underway (Heal et al. 2021; Durham et al. 2022; Kujawinski et al. 2023). Finally, the general inaccessibility of the open ocean results in chronic undersampling and significantly reduced sample sizes relative to land-based metabolomics, resulting in low-power analyses that are only able to detect the largest signals (Karl and Church 2017).

Despite these challenges, marine microbial metabolomics shows significant promise for characterizing the composition of seawater and the organisms that live within it. Metabolites have been used to describe the latitudinal variation in marine particles (Heal et al. 2021; Johnson et al. 2023, 2020), the response of the microbial community to nutrient and vitamin availability (Sañudo-Wilhelmy et al. 2014; Heal et al. 2017; Bertrand et al. 2015; Wilson et al. 2019; Dawson et al. 2020), and the response of phytoplankton to changes in temperature and salinity (Dawson et al. 2023) as well as their response over the diel cycle (Muratore et al. 2022; Boysen et al. 2021). Additionally, recent work on metabolites dissolved in seawater has begun to unlock the vast diversity of organic carbon and nitrogen in the ocean (Sacks et al. 2022; Widner et al. 2021; Johnson, Kido Soule, and Kujawinski 2017). All of these efforts have implications for the way the smallest molecules in the ocean affect its ability to cycle energy and matter through the globe.

## Automated and untargeted liquid-chromatography mass spectrometry

Mass spectrometry (MS) is the dominant analytical platform in metabolomics (Cajka and Fiehn 2016; Gika et al. 2019). Commonly, this technique is paired with chromatographic separation to allow isomers to be quantified independently and to provide additional information about the chemicals’ nature. The disadvantage of this pairing is that the signal must then be integrated in retention time to provide an accurate reconstruction of the original quantity. With noisy signals such as those produced by hydrophilic interaction columns (HILIC, Buszewski and Noga (2012)) and compounds near the limit of detection, this becomes a challenge. The conventional solution is manual integration, in which a mass-spectrometrist manually reviews the extracted chromatograms and determines the start and end of chromatographic peak for integration, often via graphical user interface (GUI). However, this method is time consuming (scaling with the number of compounds and the number of samples) and cannot be guaranteed to be reproducible. This has led to the use of software for automatic peak detection and integration.

Automatic peakpicking and annotation software has been developed in parallel for the better part of two decades by both open-source and commercial endeavours (C. A. Smith et al. 2006; Tautenhahn, Böttcher, and Neumann 2008; Heuckeroth et al. 2024; Schmid et al. 2023; Tsugawa et al. 2015; Rafiei and Sleno 2015; Coble and Fraga 2014; Hohrenk et al. 2020). The focus of these tools is typically on untargeted metabolomics (including proteomics and lipidomics), which uses a data-driven approach to compound detection rather than approaching the dataset with a list of anticipated compounds (Gika et al. 2019). This approach is particularly useful for marine microbial metabolomics, where many compounds are yet to be discovered and the additional features detected produce more powerful statistics to compensate for small sample sizes. The untargeted method also comes with significant drawbacks, with imperfect integrations by the peakpickers, multiple signals due to adducts and isotopes, and low-confidence annotations still requiring extensive manual review (Myers et al. 2017a). While untargeted analysis is traditionally associated with hypothesis generation because of its compound discovery capability (Giera, Yanes, and Siuzdak 2022; Thukral, Allen, and Petras 2023), it is perfectly qualified for testing of a well-formed hypothesis as well.

I highlight here the distinction between untargeted MS and automated MS because they are orthogonal philosophies often conflated. It is entirely possible (and often desired!) to have an automatic targeted workflow where specific compounds of interest are quantified with the speed and reproducibility of an algorithm without expanding the analysis to unknowns. Similarly, it is possible to perform untargeted metabolomics with traditionally targeted tools such as Proteowizard’s Skyline (Adams et al. 2020) or even Microsoft Excel as long as the data is used to drive discovery instead of a priori knowledge about the expected compounds. For example, one could imagine an Automated Data Analysis Pipeline (ADAP) type algorithm (Myers et al. 2017b) that recursively extracts the largest intensities in a file and nearby *m/z* values for manual integration. These “alternate” MS methods (targeted automation and manual untargeted) are underutilized simply because the tools for their use have not yet been implemented or documented sufficiently.

## Overview of projects

This thesis presents a body of work spanning data science and oceanography. In the first chapter, I discuss how MS data can be enormously simplified by converting it into a “tidy” format in the sense of Wickham (2014). This allows for the rapid exploration and reproducible analysis that I use in the rest of the thesis. Chapter 3 logically extends this framework into proper database systems which mitigates Chapter 2’s major problems with memory usage. I additionally compare multiple database systems with emphasis on modern column-oriented and online analytical processing methods that show particular promise. The particular strength of these methods is their ability to look at data *across* files rather than within a single one, something that I heavily leverage in later chapters.

Chapter 4 demonstrates the utility of allowing for rapid raw data access by showing how novel peak metrics calculated from the raw data can significantly reduce the rate of false positives in existing peakpicking software. This “cleaned” data set then shows interesting differences between marine microbial samples taken from different depths that were not apparent in the original. In the appendix, I also illustrate how raw data itself can be treated as a multidimensional array with the largest “signals” being those of high-quality peaks, allowing dimensionality reduction techniques to group MS features for rapid quality annotation.

Chapters 5 and 6 are applications of the above philosophy to oceanographic data collected from the North Pacific Subtropical Gyre (NPSG) near Station ALOHA. The NPSG is the largest biome on the planet and, like most of the surface ocean, is limited by the bioavailability of nitrogen despite large standing stocks of dinitrogen gas and DON in addition to the constant upwelling of nitrate from the deep (Moore et al. 2013; Karl and Church 2017). Since nitrogen limits the amount of carbon fixation and export possible, understanding the forms and fluxes of nitrogen-containing molecules and the organisms they compose directly affects our ability to predict marine carbon cycling. As the majority of the nitrogen flux is through small, polar molecules (Moran et al. 2016; Moran, Kujawinski, et al. 2022), metabolomics is particularly well suited to describing and quantifying these elemental cycles.

Chapter 5 documents an exploratory metabolomics dataset collected in the NPSG across two sets of mesoscale eddy features of opposing polarity. In many ways this chapter felt like a return to the Challenger era of observational oceanography which required the use of complex ecological statistics to unravel the impacts of sea level anomaly on the ocean’s metabolome and reported several compounds for the first time in the open ocean. Chapter 6, in contrast, was a deeply-nested experimental framework using short-term incubations with isotopically-labeled nitrogen substrates to test specific hypotheses about microbial nutrient acquisition and use. In both cases, the rapid and intuitive exploration of select chromatograms as well as access to the raw data was key for constructing a confident and coherent narrative of the microbial role in ocean biogeochemistry.

# Chapter 2: Tidy Data Neatly Resolves Mass-Spectrometry’s Ragged Arrays

## Abstract[[1]](#footnote-29)

Mass spectrometry (MS) is a powerful tool for measuring biomolecules, but the data produced is often difficult to handle computationally because it is stored as a ragged array. In R, this format is typically encoded in complex S4 objects built around environments, requiring an extensive background in R to perform even simple tasks. However, the adoption of tidy data (Wickham 2014) provides an alternate data structure that is highly intuitive and works neatly with base R functions and common packages, as well as other programming languages. Here, we discuss the current state of R-based MS data processing, the convenience and challenges of integrating tidy data techniques into MS data processing, and present RaMS, a package that produces tidy representations of MS data.

## Introduction

Mass-spectrometry (MS) is a powerful tool for identifying and quantifying molecules in laboratory and environmental samples. It has grown enormously over recent decades and has been responsible for countless advances in chemical and biological fields. It is often paired with liquid chromatography (LC) to separate compounds by retention time and improve detection limits. The large quantity of data produced by increasingly rapid and sensitive instruments has facilitated the adoption of computational methods that use algorithms to detect, identify, and quantify molecular signatures.

Many mass-spectrometrists have some exposure to programming, often in R, and this familiarity is expected to increase in the future as computational methods continue to become more popular and available. However, these researchers typically focus on results and the conclusions that can be drawn from them rather than the arcane details of any particular language or package. This produces a demand for simple data formats that can be quickly and easily understood by even a novice programmer. One such representation is the “tidy” data format, which is rapidly growing in popularity among R users for its consistent syntax and large library of supporting packages (Wickham 2014). By formatting MS data tidily, the barrier to entry for novice programmers is dramatically reduced, as tidyverse functions learned elsewhere will function identically on MS data.

This article begins by reviewing the current theory and implementation of MS data handling, as driven by three major questions. First, why is it difficult to access and interpret MS data? Second, why should it be easier to do this? Finally, why don’t current algorithms make it trivial to do this? In the latter portion of this article, we introduce a new package, called R-based access to Mass Spectrometry data (RaMS) that provides tidy access to MS data and will facilitate future analysis and visualization.

## Why is it difficult to access mass-spectrometry data?

Mass spectrometers produce data in the form of ragged (also sometimes called “jagged”) arrays. These data structures contain an unequal number of columns per row because any number of ion masses (*m/z* ratios) may be observed at a given time point. This data is typically managed in a list-of-lists format, with a list of time points each containing a list of the ions observed and their abundances. While this is an effective way to preserve the data structure as it was produced by the instrument, it is less helpful when performing analysis. Typically, analysis (both manual and computational) iterates over *m/z* windows rather than time. The main focus is the extracted ion chromatogram (EIC) which represents all time points for a given mass, and the spectrum of masses obtained at a given time point is less useful during the preliminary review and initial discovery phases. This nested syntax, often itself contained within S4 objects and encoded as an environment, makes it difficult to extract EICs quickly and intuitively.

Even so, “difficult” is a relative assessment. Veteran R programmers have little difficulty writing elegant code that embraces these ragged arrays and the list-of-lists syntax. Indeed, the dominant MS processing package in R, MSnbase currently uses the S4 object system to great effect. However, MS experts are rarely also R experts and have a working familiarity with R rather than a comprehensive background in computer science. This working knowledge typically includes creating plots, subsetting data, and manipulating simple objects but does not extend to the nuances of the S4 object system or methods for rewriting package code. Thus, a package capable of converting these complex data structures into a familiar format appears to be very much in demand.

Finally, it should be noted that existing MS data processing packages are designed to be holistic pipelines which accept raw data and output definitive results. There is very little room for a user’s customization beyond the provided function arguments despite the enormous variability in MS setups, usage, and data quality. It is often challenging to access intermediate objects as a way to debug unexpected results, and published code is rarely easy to edit safely due to poor documentation and unit test coverage. These issues are compounded by the agglomerative nature of R packages that build extensively upon other R packages; the popular xcms processing package has over a hundred dependencies installed from across CRAN and Bioconductor, with further functionality provided by unregulated code from GitHub and SourceForge. When combined with additional issues from C++ compilers, versioning, and operating system discrepancies, MS data analysis becomes very much a “black box” with functioning pipelines treated as fragile rather than simple, robust, and reproducible.

## Why should it be easier to access mass-spectrometry data?

Mass-spectrometry data is fundamentally simple. In LC-MS full-scan mode, each data point has three coordinates corresponding to the time, molecular mass, and intensity dimensions. Even the more complex fragmentation data requires only a single additional dimension, fragment mass. While this ignores the large quantity of critical metadata associated with each file that must also be stored somewhere, a core part of MS research is driven by the data alone. In this preliminary stage of analysis, metadata is less relevant than simple exploratory questions about which molecules can be detected and preliminary assessments of data quality. This exploratory phase is driven by rapid, ad hoc discovery and hypothesis testing that typically requires visualizing chromatograms and the raw data to assess quality: this appears to be one of the reasons why R and its built-in plotting ability is so popular for MS analysis (Gatto, Gibb, and Rainer 2021). These queries should be trivial to implement, even for beginning R users, but current data storage methods make them difficult and often time-consuming. Currently, the easiest questions to answer about MS data are metadata-based queries about the instrument that the analyst is usually already able to answer. This is an artifact of information storage in most raw data files, with metadata available readily at the top level and measurements buried deep within.

Raw MS data is typically converted from vendor-specific formats into open-source versions that can be parsed without proprietary software. The modern standard is the mzML document, which has been designed to combine the best aspects of precursor standards in a single universal format (Deutsch 2010). These XML documents have well-defined schema built around a controlled vocabulary to enable consistent parsing. Most critically, the development of the modern mzML format established accession numbers for each attribute which (according to the specification document) should never change. This stability means that the data can be accessed robustly with any XML parser. Older formats, such as mzXML, are currently deprecated and will not undergo further development, making them equally stable.

Finally, simple data formats make it easier to work within existing frameworks rather than developing exclusive functions. Tidy data interacts neatly with the entire tidyverse thanks to its shared design philosophy and it’s simple to upgrade basic data frames to data.tables for improved access speed. More crucially, however, simple formats make it possible to port MS data to other languages and interfaces. It is straightforward to convert an R data frame to Python’s pandas version via the reticulate package, encode it as a SQL database, or export it as a CSV file to be viewed in Excel or other familiar GUIs. The same cannot be said for R’s environments and S4 objects. This connectivity ensures that the best tools possible can be applied to a problem, rather than the subset available in a given package or programming language. Simplifying access to and working storage of MS data is a critical step for the further development of fast, accurate algorithms for the detection and quantification of molecules across many areas of science.

## Why isn’t it already easier to access mass-spectrometry data?

Of course, there are challenges that make simplification difficult and a trade-off must be made between speed, storage, and sanity. Tidy data favors code readability and intuitiveness over computational efficiency: for example, a list-of-lists model is more memory efficient than the proposed rectangular data structure because each time point is stored once rather than repeated in each row. When multiple files are analyzed simultaneously, tidy data also requires that the filename be repeated similarly, resulting in essentially a doubling of object size in the computer memory. Given that most MS experiments involve tens or hundreds of large files, this is a major concern and current packages handle memory carefully, either reading from disk only what is needed or running files in batches. There are several ways to resolve this problem within the tidy data model as well. During the exploration phase, it is rarely necessary to load all data from files simultaneously, but viewing some portion of the data is still critically important for quality control. With the tidy model, it’s not required to import all the data in a single comprehensive step. Instead, quality control files or pooled samples can be viewed as representative of the whole run and rarely challenge memory requirements. Additionally, tidy data makes it easy to subset only the masses of interest for targeted analyses, and the remainder of the data can be discarded from memory. For the final comprehensive analysis, it is much simpler to encode MS data into an external database for access via SQL or other query language when formatted tidily than it is to wrangle current implementations into some accessible object that can handle project sizes larger than the computer’s memory.

Theoretically, the ideal data structure for MS data processing speed would invert the current list-of-lists schema by constructing a list of unique *m/z* values, each containing the time points at which that mass ratio was observed and the corresponding intensity. However, this method is complicated by the instrumental error inherent in measuring molecular masses. The same molecule may be measured to have a slightly different mass at each time point, and “binning” these masses together across all time points for a single consensus value risks incorporating nearby masses together even at hypothetical sub-ppm mass accuracy (Kind and Fiehn 2006). Instead, *m/z* values are continuous rather than discrete, making it difficult to encode the data in this way. A tidy framework resolves part of this issue by storing the time and *m/z* values in columns that can be indexed by a binary search, such as the one implemented by data.table. This allows for rapid subsetting by both time and *m/z*. Finally, it is worth noting that computers have rapidly grown faster and larger while human intuition has not grown as quickly. This indicates that concerns with processing time and memory will lessen over time and that in the long run, sanity should be prioritized over speed and storage.

There are other reasons that a tidy approach has not yet been implemented for MS data. MS files include large amounts of metadata which should not be discarded, but are challenging to encode efficiently in a rectangular format. A proper tidy approach requires that a separate table be constructed to hold this per-file metadata, with a key such as file name that permits joining the metadata back to the original information. Compared to the monolithic S4 objects constructed by traditional workflows, managing multiple tables may be unappealing. S4 objects also excel at recording each process that is performed on the data, and a specific “processes” slot is found in some objects to record exactly this. However, with the emergence of code sharing and open-source projects it becomes less critical that the data itself records the process because the source code is available.

Finally, a significant history exists for today’s methods. MSnbase, the first widely-used R package designed to process MS data, implemented S4 objects as a way to hold entire MS experiments in memory, and dependent packages extend this MSnExp object in various ways rather than discarding it entirely. This development history and connected network of packages is incredibly useful and represents an extensive process of innovation and refinement. We would like to emphasize that the concerns raised here and the package introduced below are not designed to critique or replace this significant effort. Instead, our goal is to function alongside prior work as a way to enable rapid, interactive, and preliminary exploration. Following initial investigation, we recommend using the existing pipelines and extensive package network to establish a reproducible, scripted process of MS data analysis.

## The RaMS package

The RaMS package implements in R a set of methods used to parse open-source mass-spectrometry documents into the R-friendly data frame format. Functions in the package accept file names and the type of data requested as arguments and return rectangular data objects stored in R’s memory. This data can then be processed and visualized immediately using base R functions such as plot and subset, passed to additional packages such as ggplot2 and data.table, or exported to language-agnostic formats such as CSV files or SQL databases.

###Installation

The RaMS package can be installed in two ways:

The release version from CRAN:

Or the development version from GitHub:

### Input arguments

RaMS is simple and intuitive, requiring the memorization of a single new function grabMSdata with the following usage:

Where files is a vector of file paths to mzML or mzXML documents, which can be located on the user’s computer, a network drive, FTP site, or even at a URL on the Internet. Further parameters are documented below in Table 2.1:

| Parameter | Description |
| --- | --- |
| grab\_what | Specifies the information to extract from the mzML or mzXML file. Can currently accept any combination of “MS1”, “MS2”, “EIC”, “EIC\_MS2”, “metadata”, and “everything” (the default). |
| verbosity | Controls progress messages sent to the console at three different levels: no output, loading bar and total time elapsed, and detailed timing information for each file. |
| mz | Used when grab\_what includes “EIC” or “EIC\_MS2”. This argument should be a vector of the *m/z* ratios interesting to the user, if the whole file is too large to load into memory at once or only a few masses are of interest. |
| ppm | Used alongside the mz argument to provide a parts-per-million error window associated with the instrument on which the data was collected. |
| rtrange | A length-two numeric vector with start and end times of interest. Often only a subset of the LC run is of interest, and providing this argument limits the data extracted to those between the provided bounds. |

*Table 2.1: Parameters accepted by the grabMSdata function.*

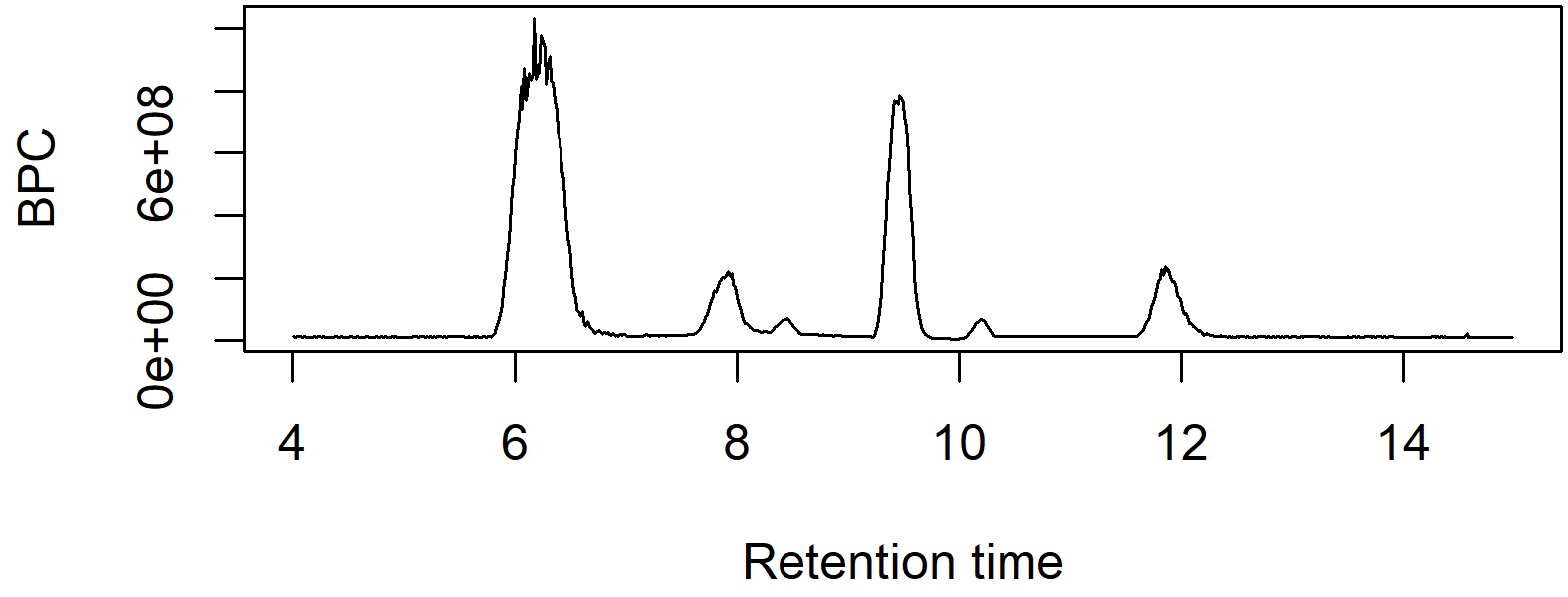
### Usage

Extracting data with grabMSdata returns a list of tables, each named after one of the parameters requested. A grab\_what argument of "MS1" will return a list with a single entry, the MS1 (i.e. full-scan data) for all of the files:

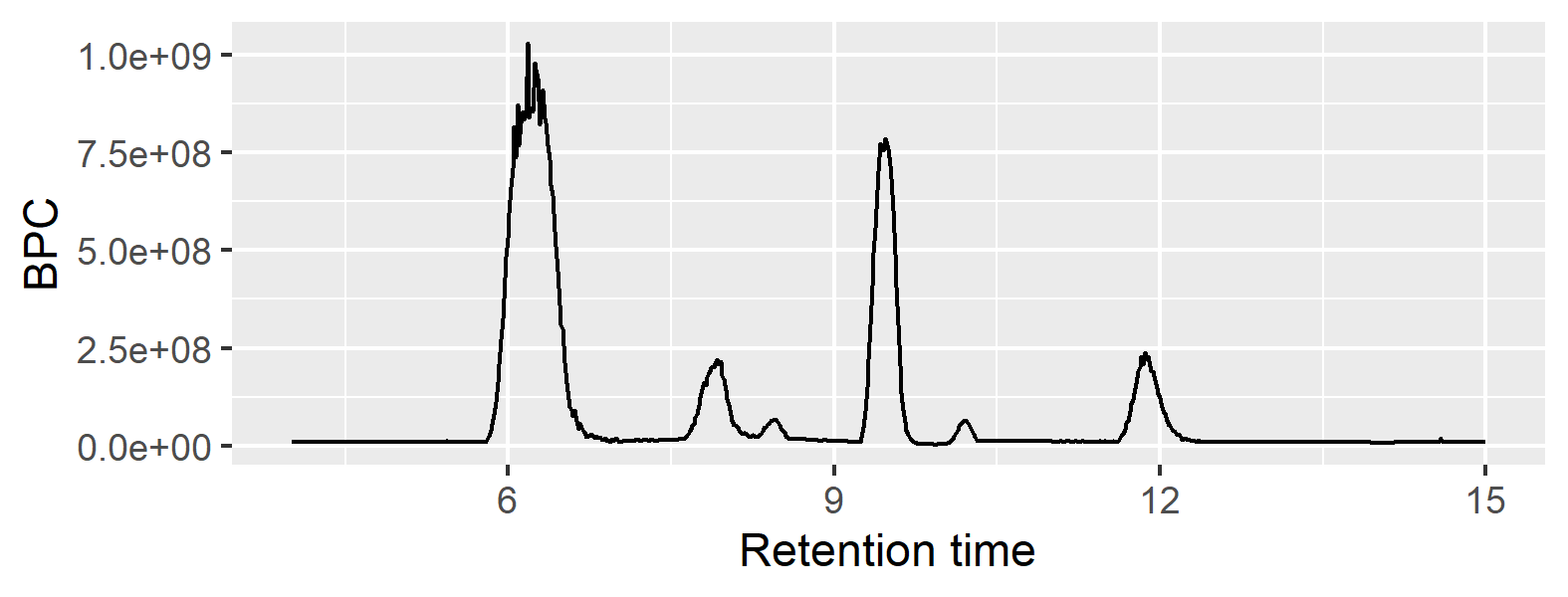
| rt | mz | int | filename |
| --- | --- | --- | --- |
| 4.009 | 104.0710 | 1297755.000 | LB12HL\_AB.mzML.gz |
| 4.009 | 104.1075 | 140668.125 | LB12HL\_AB.mzML.gz |
| 4.009 | 112.0509 | 67452.859 | LB12HL\_AB.mzML.gz |
| 4.009 | 116.0708 | 114022.531 | LB12HL\_AB.mzML.gz |
| 4.009 | 118.0865 | 11141859.000 | LB12HL\_AB.mzML.gz |
| 4.009 | 119.0837 | 9636.127 | LB12HL\_AB.mzML.gz |

*Table 2.2: Tidy format of RaMS output showing columns of MS1 data, with columns for retention time (rt), mass-to-charge ratio (mz), intensity (int) and name of the source file (filename). Note that this is a subset - the actual object contains 8,500 entries.*

This table is already tidied, ready to be processed and visualized with common base R or tidyverse operations. For example, it’s often useful to view the maximum intensity observed at each time point: this is known as a base peak chromatogram or BPC. Below are two examples of calculating and plotting a BPC using base R and the tidyverse.



*Figure 2.1: A simple chromatogram plotted using base R. This plot shows the retention time of all compounds in a sample plotted against the maximum intensity at each timepoint. Base graphics were used so the plot is fully customizable with normal graphics options.*



*Figure 2.2: A simple chromatogram plotted using the ggplot2 package. This plot shows the same data as Figure 1 of retention time by maximum intensity across compounds but uses ggplot2 syntax and defaults.*

Importantly, note that the creation of these plots required no special knowledge of the S3 or S4 systems and the plots themselves are completely customizable. While similar packages provide methods for plotting output, it is rarely obvious what exactly is being plotted and how to customize those plots because the data is stored in environments and accessed with custom code. RaMS was written with the beginning R user in mind, and its design philosophy attempts to preserve the most intuitive code possible.

RaMS uses data.table internally to enhance speed, but this also allows for more intuitive subsetting in mass-spectrometry data. With data.table, operations are nearly as easy to write in R as they are to write in natural language, leveraging the user’s intuition and decreasing the barrier to entry for non-coder MS experts. For example, a typical request for MS data might be written in natural language as

“All MS data points with *m/z* values between an upper and lower bound, from start time to end time.”

This request can be written in R almost verbatim thanks to data.table’s intuitive indexing and %between% function:

Most importantly, this syntax doesn’t require the mass-spectrometrist to have an understanding of how the data is stored internally. Current implementations use S4 objects with slots such as “chromatograms” and “spectra” or derivatives of these, despite their inconsistent usage across the field and unclear internal structure. (R. Smith, Ventura, and Prince 2015)

RaMS enhances the intuitive nature of data.table’s requests slightly by providing the pmppm function, short for “plus or minus parts-per-million (ppm)”. Masses measured on a mass-spectrometer have a certain degree of inherent deviation from the true mass of a molecule, and the size of this error is a fundamental property of the instrument used. This means that mass-spectrometrists are often interested in not only the data points at an exact mass, but also those within the ppm error range. MS data exploration often makes requests for data in natural language like:

“All MS data points with *m/z* values within the instrument’s ppm error of a certain molecule’s mass”

Which can again be expressed in R quite simply as:

### Internals

Fundamentally, RaMS can be considered an XML parser optimized for mzML and mzXML documents. The rigorous specification and detailed documentation make it possible for a generic XML parser to efficiently extract the document data. In R, the xml2 package provides modern parsing capabilities and is efficient in both speed and memory usage by calling C’s libxml2 library, making it an attractive choice for this processing step. Much of RaMS’s internal code consists of a library of XPath expressions used to access specific nodes and extract the (often compressed) values. Table 2.3 below provides several examples of XPath expressions used to extract various parameters from the mzML internals:

| Parameter of interest | mzML XPath expression |
| --- | --- |
| Fragmentation level | //spectrum/cvParam[@name="ms level"] |
| Retention time | //scanList/scan/cvParam[@name="scan start time"] |
| *m/z* values | //binaryDataArrayList/binaryDataArray[1]/binary |
| Intensity values | //binaryDataArrayList/binaryDataArray[2]/binary |
| Polarity (for positive mode) | //spectrum/cvParam[@accession="MS:1000130"] |

*Table 2.3: A few example parameters extracted from the mzML file and the corresponding XPath expression used to extract it.*

These sample expressions illustrate the controlled vocabulary of the mzML parameters (the cvParam elements above) and the remarkable stability of the specification that permits optimization. While the “polarity” parameter for positive mode is the only one above that is specified via its accession number (“MS:1000130”), it’s worth noting that the other parameters also have unique accession number attributes that could be used but instead have been foregone in favor of readability.

MS data files are often highly compressed and the *m/z* and intensity data is typically encoded as base 64 floating point arrays. MS data extracted from the binary data array must then first be decoded from base64 to binary using the base64enc package, then decompressed if necessary using R’s base memDecompress function, and finally cast to double-precision floating point values via base R’s readBin.

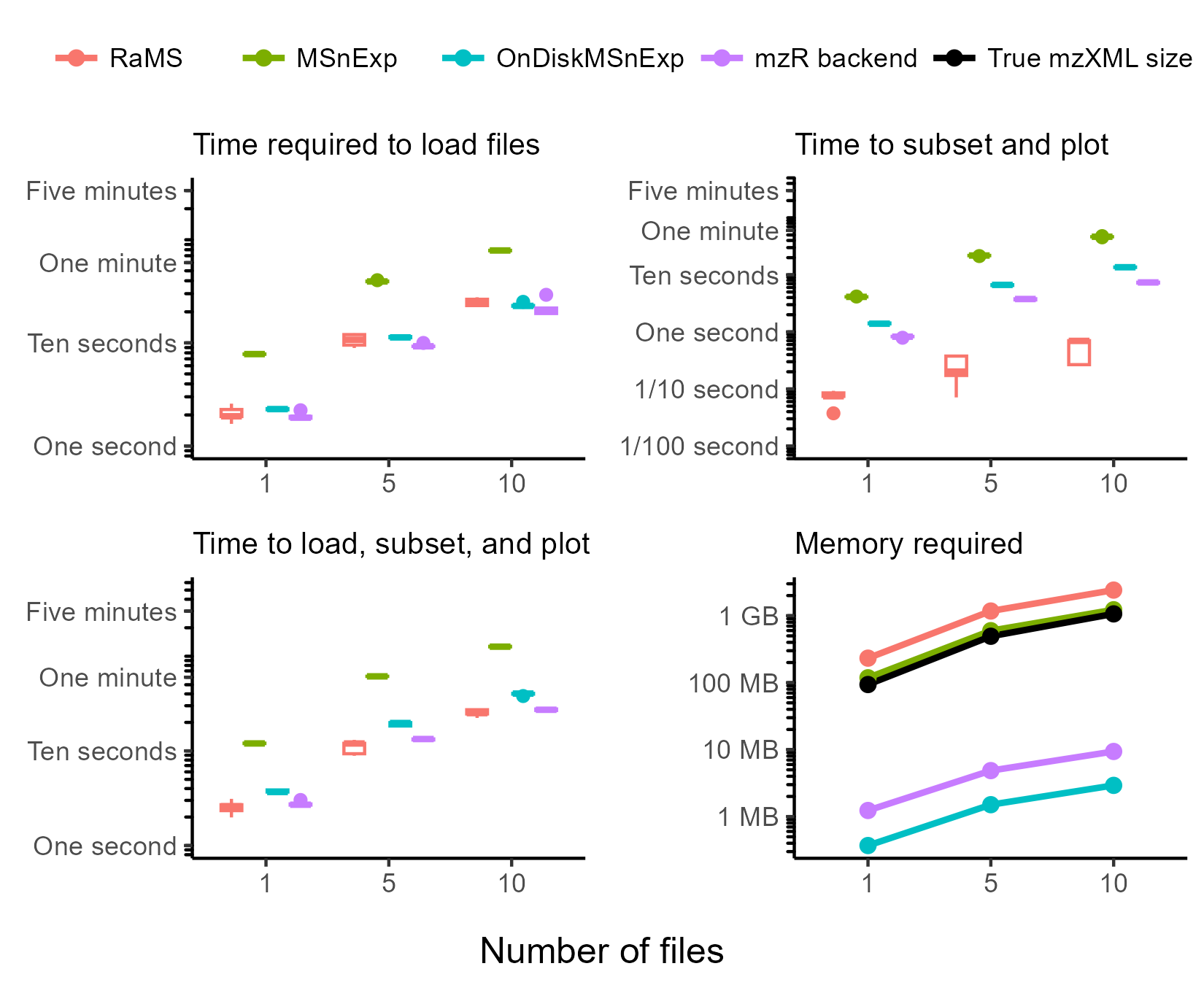
After the data has been extracted from the XML document, RaMS uses the data.table package to provide fast aggregation and returns data.table objects to the user. This is also the step which converts the data from a ragged array format into a tidy format, and neatly illustrates the strength of tidy data. Rather than continuing to store the data as a list-of-lists and preserving the nested data structure, this step creates separate columns for retention time (rt) and *m/z* (mz) values. This allows the user to perform rapid binary searches on both the retention time and *m/z* columns and can greatly accelerate the extraction of individual masses of interest, as is often the goal when analyzing MS data.

### Comparison to similar packages

While many packages exist to process MS data within R, very few can be found that actually read the raw data into the R environment. The dominant package by far is MSnbase, which describes itself as providing “infrastructure for manipulation, processing and visualisation of mass spectrometry and proteomics data”, and is thus very similar to RaMS. MSnbase itself calls the Bioconductor package mzR to provide the C++ backend used to parse the raw XML data. Other packages include readMzXmlData and MALDIquantForeign, both developed by Sebastian Gibb and hosted on CRAN. One additional package to note is the caMassClass package that no longer exists on CRAN but code from which can be found in the CorrectOverloadedPeaks package and only parses the deprecated mzXML format. Finally, the Spectra package is under active development by the RforMassSpectrometry initiative and represents a useful comparison for other cutting-edge frameworks that will be expanded in the future (Rainer et al. 2022). However, all of these packages preserve the list-of-lists format and none produce naturally tidy representations.

This section illustrates how RaMS compares to MSnbase as the current dominant processing package and Spectra as the next iteration of MS processing. MSnbase has undergone constant revision since its inception in 2010, while Spectra has been under development since 2020. The most recent version of MSnbase as of this writing was announced in 2020 and focuses on the new “on-disk” infrastructure that loads data into memory only when needed. This new infrastructure and the legacy storage mode released in the first version of MSnbase provide useful comparisons for RaMS in terms of memory usage and speed and the Spectra package will provide a useful future-oriented comparison. As noted above, however, RaMS has different goals from either of these packages. RaMS is optimized for raw data visualization and rapid data exploration while MSnbase and Spectra are designed to provide a solid foundation for more streamlined data processing and these packages all can work neatly in concert rather than replacing each other.

To compare the different methods, ten MS files were chosen from the MassIVE dataset MSV000080030 to mimic the large-experiment processing of (Gatto, Gibb, and Rainer 2021). Methods were compared in terms of memory usage, time required to load the data into R’s working memory, and the time required to subset an EIC and plot the data. Due to the differences in method optimization, we expected MSnbase to be significantly faster when loading the data, RaMS to be significantly faster during subsetting and plotting, and MSnbase to have the smallest memory footprint. The Spectra package’s capabilities were less well known in advance but should represent a consistent improvement over MSnbase. These expectations were well-validated by the results shown in Figure 2.3.



*Figure 2.3: Time and memory required by RaMS compared to the MSnbase and Spectra methods across 1, 5, and 10 mzXML files. The top-left plot shows the time required to load the mzXMLs into memory (RaMS and MSnExp) or construct pointers (OnDiskMSnExp, Spectra’s mzR backend) with the MSnExp object taking approximately an order of magnitude longer than the other methods. The top-right plot shows the time required to subset the data by m/z to a single chromatogram and plot that subset after the object has already been created. The RaMS package performs this approximately an order of magnitude faster than the other packages and the Spectra package is second-fastest, with RaMS taking less than a second for up to 10 mzXMLs and the Spectra package taking between one and ten seconds depending on the number of files to be subset. The bottom-left plot shows a combination of the two plots above by timing each package as it performs the full object construction, subsets to a single chromatogram, and plots it with RaMS again the fastest among the packages. The bottom-right plot shows the memory required for each package across different numbers of files as well as the size of the original mzXML documents as a benchmark. Both RaMS and the MSnExp objects occupied more space in RAM than the original file size (RaMS occuying approximately 2x as much memory, MSnExp closer to 1.1x), while the OnDiskMSnExp and mzR backend were consistently two orders of magnitude smaller. Times were obtained by the microbenchmark package and object sizes were obtained with pryr. Note the log-scaled y-axes.*

RaMS performed better than expected on the data load-time metric, taking approximately the same amount of time as the new on-disk MSnbase backend and the Spectra package and significantly less than the old in-memory method. This was surprising because while RaMS is performing the physical I/O process essentially equivalent to the creation of the MSnExp, both the OnDiskMSnExp method and the Spectra object instead create a system of pointers to the data and don’t actually read the data into memory. However, the new backend begins to perform better as the number of files increases and proportional improvements are expected with even larger file quantities. The Spectra package, as expected, shows consistent improvements over both MSnbase backends.

For the subsetting and plotting metric, our expectation that RaMS would be the fastest method was validated by times approximately two orders of magnitude smaller than those obtained by MSnbase (note the log scale used in the figure). These results also validated earlier results demonstrating the superiority of the new on-disk method (Gatto, Gibb, and Rainer 2021) and the improvements in the new Spectra package. The sub-second subset and plot times of RaMS are so much smaller than the other timings recorded in this trial that RaMS essentially has a single fixed cost associated with the initial data import, making it ideal for the exploratory phase of data analysis where files are loaded once and then multiple chromatograms may be extracted and reviewed. This design also aligns with the user’s expected workflow in which data import is accepted as a time-consuming task, but subsequent analysis should be relatively seamless and instantaneous.

The greatly reduced subsetting and plotting time required by RaMS and the observation that file load times and data plotting times were approximately equal for MSnbase led to the creation of the bottom-left graph in Figure 2.3. This follow-up analysis highlights that the slightly increased file load time of RaMS combined with the very short subsetting and plotting phase is actually less than the total time required by MSnbase and Spectra to read, subset, and plot, establishing RaMS as the fastest option even if the end goal is to extract a single chromatogram. This follow-up also demonstrates the largest improvements of the new MSnbase on-disk method over the old one and the clearest improvements in Spectra.

As expected, this speed comes at a cost. RaMS has a larger memory footprint than even the old in-memory MSnExp object. While all three objects grew approximately linearly with the number of files processed, the RaMS object was approximately 2 times larger than the in-memory MSnbase object and several orders of magnitude larger than the new, on-disk version. This was expected because RaMS stores retention time and filename information redundantly in the tidy format while the list-of-lists method only stores that information once. In fact, the RaMS object size was larger than the uncompressed mzXML files themselves! However, this trade-off can be minimized through the use of RaMS’s vectorized grab\_what = "EIC" and grab\_what = "EIC\_MS2" functions that can extract a vector of masses of interest and discard the remainder of the data to free up memory for analyses where the specific ions of interest are known beforehand. The general lesson from this analysis seems to be that if the memory is available and a quick and intuitive interaction is desired, RaMS is now the top contender. For other purposes, MSnbase or Spectra remain the obvious choices depending on expected workflow.

### Broader interactions

RaMS is intentionally simple. By encoding MS data in a rectangular, long data format, RaMS facilitates not only R-specific development but contributes to MS analysis across languages and platforms. At the most basic level, subsets of interest can be exported as CSV files for use in any language that can read this ubiquitous format. Even users with zero programming background are familiar with Excel and other spreadsheet GUIs, so this method of export and data-sharing improves transparency by allowing anyone to open the raw data corresponding to compounds of interest.

The list-of-tables format that RaMS returns was inspired by traditional relational databases, and this provides a slightly more complex method of storing data with several advantages over CSV export. The dominant convenience of relational databases is that they can grow almost indefinitely, rather than being limited by computer memory. While existing packages perform admirably when operating on files that fit into RAM, there are few good solutions for the MS experiments that can exceed hundreds of gigabytes in size. Both batching and subset analysis face issues with systematic inter-sample variation rarely controlled for across subsets. Additionally, an external relational database can be easily appended with additional files as experiments continue to be performed, rather than demanding that all samples be run before any analysis can begin. RaMS output can be easily written to SQL databases using existing packages such as DBI and RSQLite:

Finally, with reticulate, R data frames can be directly coerced into Pandas DataFrames. This allows for an unprecedented degree of interaction between R and Python for MS data analysis, reducing the need for parallel development in both languages and allowing the optimal functions to be used at each step rather than the limited selection that have already been implemented in R or Python. As MS data exploration and analysis continues to grow increasingly machine-learning heavy, allowing R to interact elegantly with Python enables the best of R’s extensive MS analysis history with Python’s powerful interfaces to deep learning frameworks such as TensorFlow and Pytorch.

## Summary

In this paper, we discussed the current paradigm of MS data analysis in R and identify an area where tidy data techniques significantly improve user experience and support increased interaction with other packages and software. We also present RaMS as a package that fills this gap by presenting MS data to the R user in a tidy format that can be instantly queried and plotted.

## Acknowledgements

We are grateful to members of the Ingalls Lab and other labs at the University of Washington who gave invaluable feedback on early versions of this package and the philosophy behind it. Katherine Heal and Laura Carlson generated the data used in the demo files and were early adopters, and Angie Boysen and Josh Sacks provided crucial testing and application of the package. We also thank both anonymous reviewers for their insightful commentary and suggestions that improved both the manuscript and the CRAN package. This work was supported by grants from the Simons Foundation (329108, 385428, and 426570, A.E.I.).

# Chapter 3: Databases Are an Effective and Efficient Method for Storage and Access of Mass-Spectrometry Data

## Abstract

Current mass spectrometry (MS) data formats lack accessibility, interoperability, and performance. This study evaluates 10 recent MS file formats and readers across several exploratory MS analysis metrics and compares them to a simple database representation implemented in SQLite, DuckDB, and Parquet. We found that most existing formats severely lack the documentation required for adoption and that no existing format offers a balanced combination of speed, storage space, and simplicity. In contrast, our data storage schema improved data discovery and extraction by multiple orders of magnitude with minimal overhead. We argue that these database systems offer a performant and transparent way to store MS data for exploratory analysis while reducing technical debt and allowing mass spectrometrists to leverage recent advances in data science as our own computational complexity continues to grow.

## Introduction

Mass spectrometry (MS) still lacks a performant data access format. The mzML file type (Martens et al. 2011), a result of over a decade of interlaboratory collaboration and workshopping, struggles to provide rapid computational access to the *m/z* and intensity pairs. This is the crucial component in nearly all mass spectrometry analysis, but mzML’s text-based XML format requires time-consuming decompression performed one scan at a time. This is largely due to its preservation of the scan as the unit of transaction while the field moves increasingly away from single-scan analysis (Hannes L. Röst et al. 2014; Ting et al. 2015).

Alternative file formats aimed at improving data access are proposed nearly every year. These include direct improvements to the mzML format with indexing (Hannes L. Röst et al. 2015) and better internal encoding of the data (Bhamber et al. 2021), HDF5-based alternatives (Bhamber et al. 2021; Wilhelm et al. 2012; Bilbao et al. 2023; Tully 2020; Askenazi, Ben Hamidane, and Graumann 2017), relational databases (Shah et al. 2010; Bouyssié et al. 2015; Handy et al. 2017; Yang et al. 2022; Beagley et al. 2009), or fully custom alternatives (Römpp et al. 2011; Lu et al. 2022). Fundamentally, these alternatives exchange ease of use for access speed and/or size on disk with clever compression algorithms and modern data structures that move away from the human-readable format of the mzML. These optimized formats are inherently more difficult to understand and usually lack comprehensive documentation or examples (particularly across programming languages) making it difficult for new users to enjoy their benefits or extend their functionality. This steep learning curve, coupled with a lack of support in conversion tools such as Proteowizard’s msconvert (Chambers et al. 2012), has prevented widespread adoption of these new formats despite their clear computational advantages. Such formats are also fragile in the sense that without community support, their continued development depends entirely on the original developers and easily become deprecated (as is the case with YAFMS, Shaduf, and mz5, all of whom have links in their papers that currently redirect to missing webpages). A simple, speedy, and small MS data format remains very much in demand.

Relational databases are not new for MS workflows (see references above) and compete predominantly with HDF5-based methods. Both of these systems are widely used for big data and can be applied to MS data in a plethora of ways, leading to the proliferation of implementations we see today. Both backends provide excellent universality, larger-than-memory support, and rapid access to data, but HDF5-based systems excel at self-description and hierarchical structures (Askenazi, Ben Hamidane, and Graumann 2017) while the relational database model is optimized for multi-table queries using a consistent syntax (Codd 1970). Relational databases are increasingly seen in MS workflows for both raw and processed data, with SQLite backends now supported in the popular peakpicking software xcms (C. A. Smith et al. 2006) via the Spectra package (Rainer et al. 2022) (though in-memory and HDF5 options are also supported) and on MetabolomicsWorkbench (Sud et al. 2016) while the development of MassQL (Jarmusch et al. 2022) demonstrates the increasing comfort that MS analysts have with the adoption of SQL.

Relational databases also have several distinct advantages over hierarchical or text-based systems, particularly in performing searches for subsets of data via indices. Importantly, this indexing differs from the byte-offset indexes that already exist in the indexed mzML and HDF5 formats because the search for a particular subset cannot be done efficiently with a byte-offset index when the *m/z* data is encoded (whether by numpress, zlib, or just base64), though access to a particular scan can be incredibly rapid. Additionally, data from multiple samples can be stored together in a single database table to permit queries of all dataset samples to be performed without looping through each file in turn. This differs from existing formats like mzDB, mzTree, and mzMD and thereby avoids the associated computational overhead and query complexity.

SQL databases also allow mass spectrometrists to access the continual improvements and long-term stability produced by the industries who specialize in these. While HDF5 is a common scientific data format, databases are constantly under development by industry titans deeply invested in their maintenance and optimization. Online analytical processing (OLAP) methods are particularly well suited for MS data given their optimization for read speed under the assumption of infrequent transactions, making modern systems such as DuckDB (Raasveldt and Mühleisen 2019) or Apache’s Parquet formats highly appealing while preserving the familiar file-based serverless approach.

Our previous work showed how the ragged arrays of MS data can be converted into a tidy database table in memory (Kumler and Ingalls 2022) and we now logically extend that method into proper database storage on disk. Here, we test the hypothesis that a “vanilla” implementation of a relational database which exposes the raw *m/z* and intensity pairs is an intuitive and performant way of storing MS data for exploratory analysis, visualization, and quality control. We compare the time and space required to extract a representative data subset under six conditions and perform these tests on multiple databases as well as mzML and other MS data formats. Our specific questions were:

1. Is there a simple database schema that enables exploratory MS queries with basic SQL statements?
2. How expensive in time and (disk) space is it to access MS data in SQLite, DuckDB, and Parquet formats?
3. How does this cost compare to more complex MS storage formats that have been previously proposed?

## Experimental section

We chose to focus on liquid-chromatography mass-spectrometry (LC-MS) data given its widespread use and fundamentally simple raw data structure as tuples consisting of retention time, *m/z*, and intensity, though the tidy framework here can be extended easily to other MS data (Supplemental Figure 1). We performed a literature search for mass-spectrometry data formats that have been published in the last 15 years and attempted to find or construct parsers for each format in Python, a popular high-level interpreted language. Each parser was written to perform three common exploratory data analysis operations on full-scan data and three common operations on MS/MS fragmentation data. Full scan queries consisted of 1) single scan extraction by scan number, 2) retention time range extraction of all scans within a specified retention time range, and 3) chromatogram extraction, which collects the ions within a specified parts-per-million (PPM) error of a known mass. These queries generally correspond to the methods used in Bouyssié et al. (2015) Bouyssié et al. (2015), which performed similar tests benchmarking the mzDB format against mz5 and an mzML parser. Note that the chromatogram extraction does not extract a precompiled chromatogram of the sort commonly found at the end of mzML files or as a result of SIM/PRM analysis but instead refers to sifting through the raw data for data tuples with an *m/z* value between specified bounds. MS/MS queries involved extracting three relevant subsets, consisting of 1) a single scan extraction by scan number similar to that of the full scan, 2) extraction of all the fragments associated with a precursor *m/z* within a given PPM, and 3) extraction of all fragments with *m/z* values within a given PPM.

We explored the available documentation on PyPI and Github for each mass spectrometry data format and either identified existing functions and packages that would perform the above queries or wrote our own functions if necessary.

### Mass-spectrometry files and software used

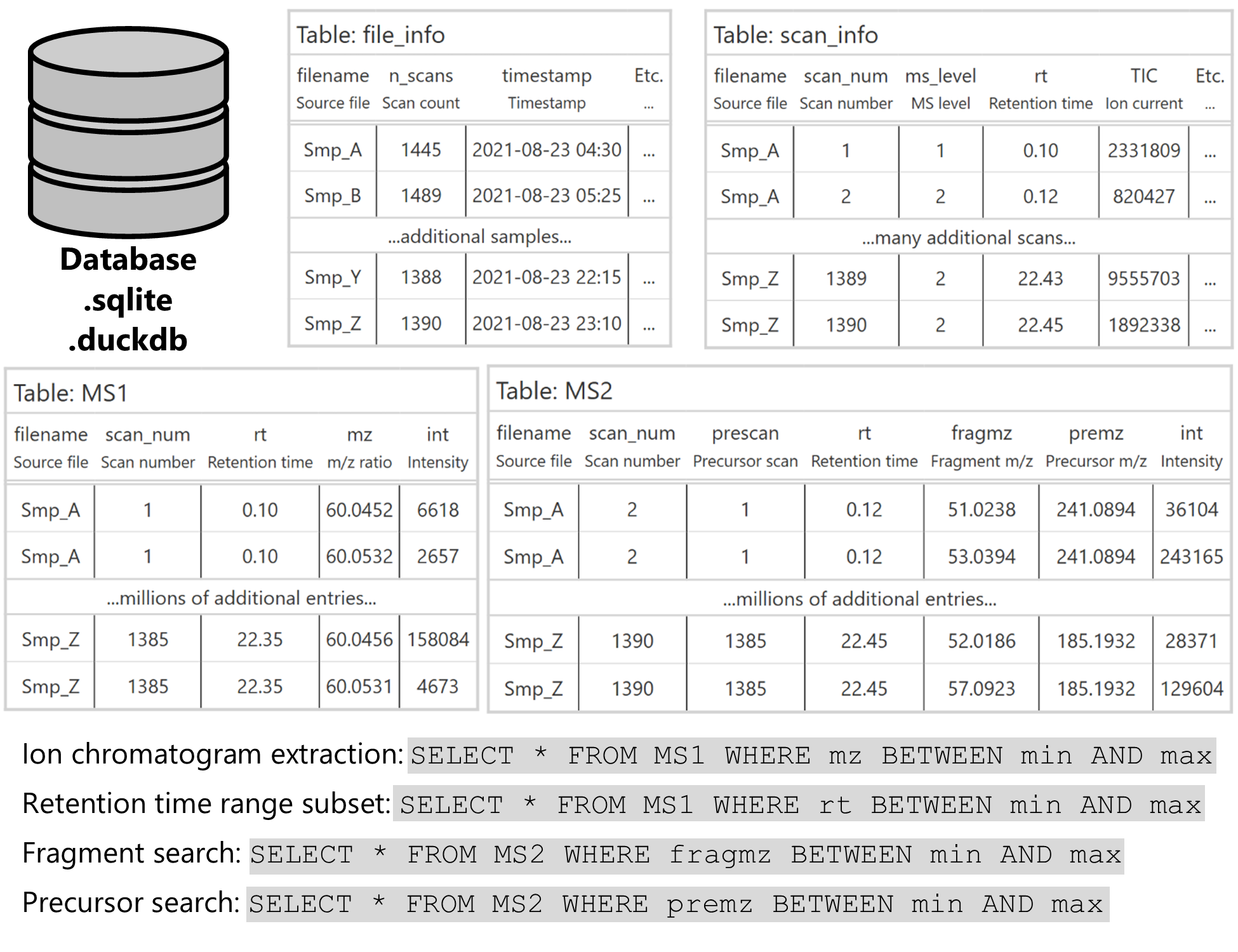
We browsed Metabolights (Haug et al. 2019) for suitable LC-MS datasets, looking for studies that included 100+ gigabytes of data from both full scan and MS/MS analysis. We were also restricted to the Thermo Scientific .raw file format, as it was the most widely supported by alternative MS storage methods. We also excluded polarity-switching data as it is unclear whether all converters would be able to separate scans based on polarity.

Files were downloaded as .raw. mzML, mz5, and mzMLb were all natively supported by Proteowizard’s msconvert software (version 3.0.25009) while MZA (v1.24.11.16) and mzDB (v0.9.10\_build20170802) had separate extensions to this executable enabling their own conversion. MzTree and mzMD were converted via their GUI which did not have release or versioning information available but were downloaded from Github (https://github.com/optimusmoose/MZTree and https://github.com/yrm9837/mzMD-java, respectively) and built via Maven (v3.9.9) for Java (v21.0.6). SQL databases were built using Python 3.11.11 with SQLite (v3.48.0) via the Python sqlite3 package (v2.6.0), DuckDB via the duckdb package (v1.1.3), and Parquet files via the pyarrow package (v19.0.0).

mzML access was done with Python’s pyteomics package (v4.7.5), the pymzml package (v2.5.10), and the pyopenms package (3.0.0.dev20230306). MZA files were accessed via the mzapy library (v1.8.dev4 from the no\_full\_mz\_array branch on Github) and via custom code built around the h5py package (3.12.1). Custom parsers were required for mzDB, mz5, MzTree, and mzMD.

### Database schema

The “vanilla” database style proposed here abandons a 1:1 representation of the original vendor-specific file. This decision was made after discussion with a wide variety of experts, all of whom preserved the original MS files even after conversion to another file type, indicating that a highly-performant addition is more important than direct replacement. Here, we map MS concepts (retention time, drift time, spatial coordinate, *m/z*, intensity, etc.) directly to database fields to make downstream processing as intuitive as possible. Metadata is stored separately in file\_info and scan\_info tables that are linked by filename and scan number (Figure 3.1). We do not force compression of any of these fields because we find that decoding compressed data is both a slow and unintuitive step, though automatic compression is supplied by the DuckDB and Parquet file types.



*Figure 3.1: Database schema for an example MS/MS dataset showing the organization of mass-spectrometry data into tables. Fields of interest are easily queryable with simple SQL commands as shown in the table at bottom.*

### Time and space testing

We randomly sampled a single file out of the full dataset for comparison across metrics and file formats (20220923\_LEAP-POS\_QC04). We sampled 100 random scan numbers using SQLite’s ORDER BY RANDOM() function and pulled out the largest ions for chromatogram extraction using a 10 ppm mass range window. Retention time ranges were the highest-intensity retention time of each ion chromatogram plus or minus one minute. Similarly, the largest fragments by intensity were used for the MS/MS metrics with a 10 ppm mass range exclusion window.

Timing was performed via Python’s timeit library and the timeit.repeat function, with the various file formats as the innermost loop to ensure bias over time was distributed equally among function calls. File sizes were estimated using Python’s os library with os.path.getsize. We did not exhaustively monitor memory usage, though our heuristic exploration of the timing scripts did not ever indicate that memory was a constraint.

Timing data was obtained on an Intel Xeon CPU with two X5650 (@2.67 GHz) processors and 24 total cores running Windows 10 Pro (64 bit version). 96 gigabytes of RAM (DDR3 @ 1333 MHz) were available and a solid-state drive was used for disk storage.

## Results

We settled on a large dataset of gut microbiota LC-MS files published in Portlock et al. (2025) Portlock et al. (2025) and available on Metabolights under accession number MTBLS10066.

### All existing MS data formats demand a high level of domain knowledge

We were able to obtain or write parsers for seven different existing mass spectrometry (MS) data formats: mzML, mzMLb, mz5, mzDB, MZA, MzTree, and mzMD. Multiple Python packages exist for the mzML data format so we used each of the three dominant packages (pyteomics, pyOpenMS, and pymzml) and compared their timing results as well. We failed to produce parsers for the YAFMS and Shaduf file types due to complete deprecation (links to these no longer exist), the toffee file type due to its application solely to time-of-flight (TOF) data-independent acquisition (DIA) data, the Aird file type due to its current deprecation in Python and C#, and the UIMF format due to a complete lack of interface documentation.

#### File conversion support varied enormously

Conversion from the initial Thermo .raw file type to the open-source .mzML format was seamlessly performed by Proteowizard’s msconvert library. Similarly, Proteowizard support for the .mz5 and .mzMLb file types made their conversion trivial.

mzDB and MZA both had extensive documentation, providing self-contained extensions to msconvert for ease of conversion. However, both converters provide limited coverage, with mzDB missing support for Waters and Agilent .d files while MZA currently lacks support for AB Sciex .wiff and Bruker .baf files. Both converters are only available via binary executable (.exe), restricting their use to Windows platforms. Additionally, both parsers appear to be unable to separate scans from a polarity-switching experiment or support any of the other filters available natively in msconvert, as additional arguments passed to the executable throw errors instead of being passed along to the original software.

MZTree and its derivative, mzMD, provided significantly less documentation about the conversion process than the other file types. This documentation consisted solely of the README available in the associated Github repositories and their installation and deployment required rebuilding the Java applet, of which the bare-bones instructions make several assumptions about the user’s PATH environmental variable. In the case of mzMD, no documentation for installation and build was provided and this instead needed to be deduced from MZTree. Additionally, we ran into issues with hardware acceleration once the GUI was launched that required extensive debugging. The GUI conversion, however, is straightforward once the app is correctly compiled and launched, albeit requiring a manual entry of a single file at a time with no apparent batch processing available.

The Aird file type was straightforward to convert on Windows via the executable available on Github (v6.0.0) but was not available for other operating systems, much like MZA and mzDB. The Python package designed to allow an interface to the file type has been deprecated and we were unable to install or use it and were unable to reverse-engineer the file type sufficiently to compare it here. The UIMF file type from the Pacific Northwest National Lab (PNNL) provided documentation exclusively in the form of C# commands and did not supply instructions for file conversion, making it unclear what input formats were supported. The toffee format provided no documentation for conversion from other formats and was restricted to time-of-flight (TOF) data independent acquisition (DIA) MS data. Thus, we were unable to directly compare any of these three file types to the others.

#### Universal lack of support for the six relevant queries

Despite the relative simplicity and relevance of our queries, none of the available mass spectrometry (MS) formats had existing functions or examples of all six queries. The mzML file type had the most extensive coverage but documentation and prebuilt functionality was still sparse. The pyteomics package provides four “combined examples” that focus on the spectrum visualization and annotation common to proteomics research but provide minimal guidance about chromatogram or retention time range extraction. Pyteomics also provides native support for the mzMLb file type and was the only one of the three Python packages to do so, deserving praise for the minimal disruption that mzMLb files placed on existing pipelines if they were to switch from mzML to mzMLb. The pyopenms package provides similarly extensive documentation for proteomic and scan-based analysis but again lacks information about subsetting in the retention time direction, though the existence of an undocumented parser (get2DPeakDataLong) provides a simple way to do this for MS1 data. Additionally, pyOpenMS required installing an old version of the package (3.0.0), Python itself (3.11) and the numpy package (<2.0) due to more recent builds requiring AVX support which was unavailable on our hardware. Pymzml is intentionally a lightweight parser focused exclusively on reading mzML files but does not supply any functions for the queries other than scan extraction by number and the “Spectrum and Chromatogram” documentation module was empty at the time of writing (February 2025).

mz5’s documentation was sparse, especially for one of the earliest mzML formats with support from Proteowizard. The original paper (Wilhelm et al. 2012) contains links to a website (https://software.steenlab.org/mz5) which currently returns an HTTP error 500. A Python library (pymz5) exists but requires an old version of Python (2.7 or 3.2), has not been updated in 12 years, and is predominately a simple fork of h5py (Collette et al. 2017) with three mz5-specific commits on top. Most problematically, we were unable to determine how mz5 stores precursor *m/z* ratios, making the fragment and precursor searches impossible. This was largely due to the variable-length nested compound structures mz5 that are not supported in all APIs, e.g. Java.(Bhamber et al. 2021)

mzDB access was hamstrung by several issues, primarily the outdated repository that implies Python and R support via a port from Rust but was unavailable at the time of development, though we are grateful for the responsive developer who notified us that this implementation was not feature-complete. This required that we deduce the SQLite BLOB type compression format from scratch when writing a parser and spend extensive time reading through the documentation to determine how best to link the various tables provided in the mzDB file. Scan metadata in this file type is stored as raw XML strings, producing the worst of both worlds in requiring both SQLite knowledge in their extraction and XML processing to obtain the necessary information. Additionally, mzDB seems to dump all MS/MS data into a single bounding box, meaning that we were unable to use the scheme to avoid parsing every MS/MS spectrum when performing precursor and fragment searches.

MZA provides a complementary Python package, mzapy, for access to MZA files. Here again we ran into several issues with its installation and use stemming largely from the deployed package requiring TOF bins for parsing, though a separate Github branch provides a workaround and the rapid developer response was appreciated. The mzapy package provides a clear example of chromatogram extraction as well as a method for retention time range extraction, though there exists no clear function for the extraction of a single spectrum by scan number despite the internal file structure being highly optimized for this purpose. mzapy also provides good support for ion mobility extraction but fails to index MS/MS information or provide any clear way to extract fragments by *m/z* or precursor.

MZTree and mzMD provide a slightly strange interface to MS data, requiring a separate Java server that can then be queried via an HTTP API. For users without prior knowledge of HTTP request methods or exposure to programming APIs, the README is entirely unhelpful because it simply documents the API’s endpoints and provides no complete query strings as examples to guide the user. This combination of GUI server and command-line HTTP request inverts the typical paradigm of GUI for exploration and command line for construction to convoluted effect, though the structure of the data returned by the server is impressively simple. More problematically for this analysis, the API provides no apparent way to access MS/MS data or query the files by scan number, with only RT and *m/z* bounds controlling the subset of data extracted. Finally, the GUI provides no way to open multiple files simultaneously or iterate through files programmatically and instead requiring point-and-click interaction with the GUI each time a file is opened or closed, preventing us from making reasonable comparisons in tests requiring multiple files.

### SQL-based parsers were simple to write and use

We then used custom code to convert the mzML files into SQLite and DuckDB databases using a simple schema for full scan (MS1) and MS/MS (MS2) data. The MS1 table consisted exclusively of fields for filename, scan index, retention time, *m/z* ratio, and intensity. The MS2 table consisted of the same fields except that the *m/z* column was separated into precursor and fragment *m/z*. Although we did not extend these databases to include the metadata associated with each file and scan, the logical framework could be easily extended in future work and the metadata typically represents a small fraction of the total space within the file, allowing us to make reasonable comparisons about file size between the databases and the metadata-rich other file types. We also converted each file’s MS1 and MS2 table into Parquet representations for comparison using the same field/column schema.

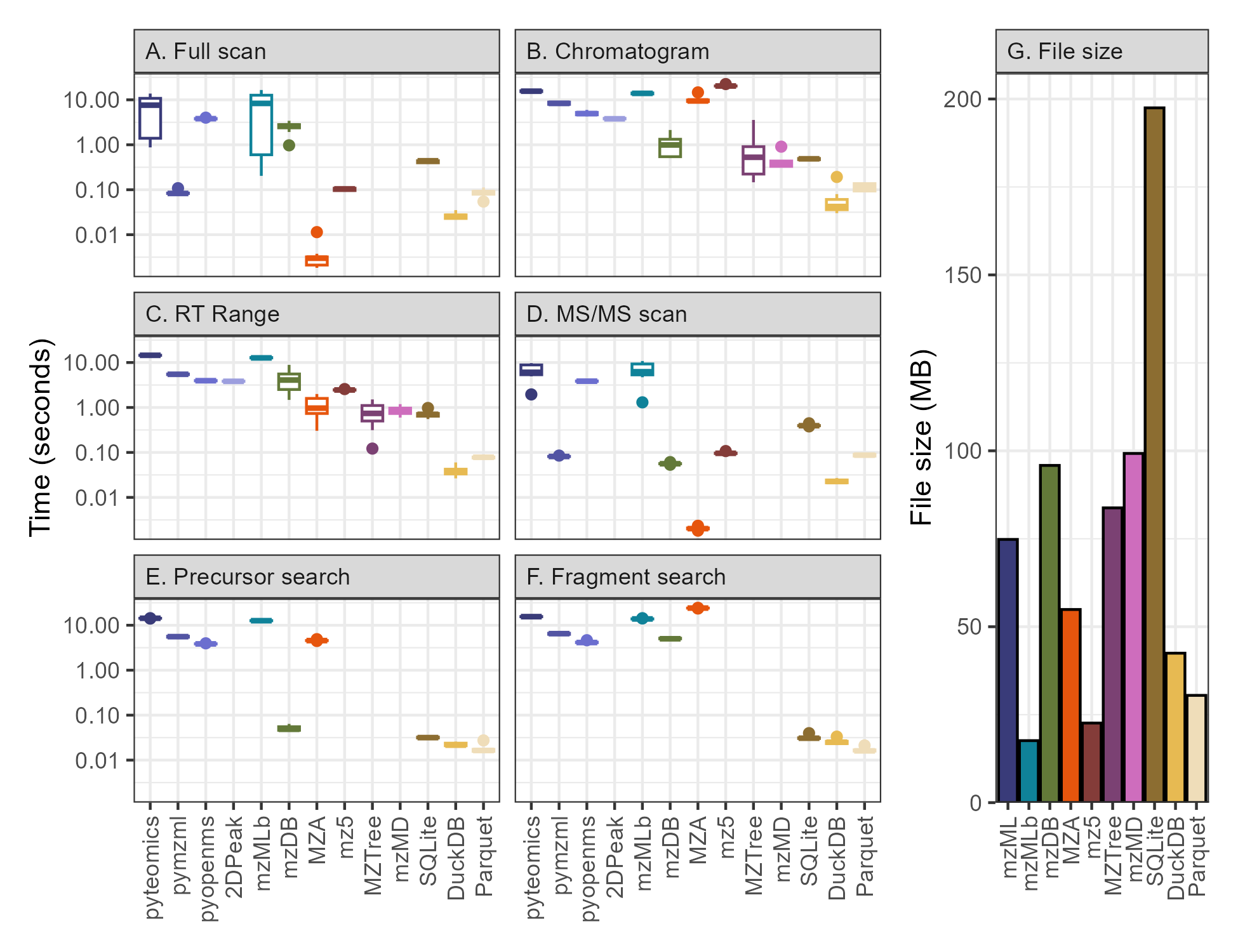
We found that the documentation for SQLite, DuckDB, and Parquet file formats in Python far exceeded the documentation available for any mzML parser. This is unsurprising given that these file formats are used widely outside of MS research and are developed and maintained by dedicated teams. Additionally, the use of a consistent SQL syntax for table creation and insertion meant that the same code could be used to write to both SQLite and DuckDB, as well as any other databases supported in Python. The use of packages such as SQLAlchemy could be used to additionally streamline this process to any additional database by simply swapping in a new database engine.

Querying the MS1 and MS2 tables was also very straightforward. After establishing a connection to the database, the six queries could be asked using nearly human-readable SQL syntax. Requesting the thousandth MS1 scan by number consisted simply of SELECT \* FROM MS1 WHERE id = 1000 passed along to the pandas.read\_sql\_query function. More complicated queries such as retention time range (SELECT \* FROM MS1 WHERE rt BETWEEN 6 AND 8) and a precursor mass search (SELECT \* FROM MS2 WHERE premz BETWEEN 118.086 AND 118.087) were similarly intuitive.

### Time and space requirements for a single DDA file across formats

#### Spectrum extraction

The simplest and most abundantly documented query was the extraction of a single spectrum. In many ways, this is the fundamental unit of mass spectrometry and thus many formats are highly optimized for its extraction into manipulatable data (Figure 3.2A and 3.2D). Here, we found that the mzML and mzMLb file types were consistently the slowest to parse and required multiple seconds, likely highlighting inefficiencies in the pyteomics package used to parse both file types. pyopenms also struggled to open and extract a specific scan, requiring several seconds due in large part to the expensive initiation function, after which requests were orders of magnitude faster (Supplemental Figure 2). It is also worth noting that while both of these packages provided rapid extraction of a *random* spectrum, a significant overhead was introduced by needing to scan through the file to find a *specific* spectrum by scan number. Scans are not always consecutive and no metadata was obviously available that would have allowed using the index directly to a specific scan number.



*Figure 3.2: Query time for the six data extraction methods and the associated file sizes for all 13 methods explored in this paper. The left six panels show boxplots representing the time required in seconds to extract a full scan spectrum (A), an ion chromatogram (B), all data within a retention time range (C), an MS/MS scan (D), the fragments of a specified precursor (E), and all precursors with a specified fragment (F). The error in the boxplot is composed of timing information for 10 repeated queries, each of a different target scan number, retention time (RT), or m/z. The right panel (G) shows a barplot of the size on disk in megabytes (MB) occupied by each file type.*

The pymzml package was able to extract both MS1 and MS2 spectra from the mzML file nearly two orders of magnitude faster than the other mzML parsers, largely due to its use of naming the scans by their number and thus avoiding the expensive scan number extraction step. mzDB had the only notable difference between MS1 and MS2 scans, performing slightly better than pyteomics and pyopenms methods for MS1 data and significantly better for MS2 data, placing it approximately on par with pymzml in taking about a tenth of a second. The simple database methods (SQLite, DuckDB, and Parquet) also fell in this ~0.1 second range, with SQLite performing most poorly and DuckDB ~10x faster. Finally, both mz5 and MZA were an additional order of magnitude faster than any other method, returning the data within the spectrum in thousandths of a second. This shows the power of the HDF5 file system for data access when its location within the file is known in advance.

#### Chromatogram extraction and subsetting by retention time range

Ion chromatogram extraction and retention time range subsets were a key metric for us, corresponding to essential tasks in chromatographic peakpicking and adduct, isotope, and in-source fragment detection (Figure 3.2B and 3.2C). EIC query times here were universally slower than those for a single spectrum extraction, reflecting the way in which a scan-based file type is sub-optimal for chromatogram extraction because each scan must be parsed to find data within a given *m/z* range. MZA and mz5 particularly suffered, with this query type entirely negating the advantages of the HDF5 file structure.

MzTree and mzMD are both file types optimized exclusively for chromatogram extraction and performed very well on the EIC metric and were two orders of magnitude faster than those parsing mzMLs, with mzMD surprisingly less performant than the older MzTree file type it was based on. However, we also note that both Java-based applications have a slow initial file load step that must be done through a GUI and therefore could not be counted in the timing comparison, the inclusion of which would likely mitigate any advantage for a single chromatogram extraction. The mzDB file type is also optimized for chromatogram extraction and was an order of magnitude faster than the other existing file types for which all queries could be run (MzTree and mzMD do not provide interfaces for spectrum extraction or MS/MS data).

The SQLite, DuckDB, and Parquet formats were just as speedy as mzMD and MzTree with SQLite taking half a second, Parquet requiring a tenth of a second, and DuckDB reaching query times of hundredths of a second, far outstripping the seconds or even minutes typically expected of this task and resulting in a functionally instantaneous interaction for the user.

Retention time range extraction times were an average of the single-spectrum extraction and the chromatogram extraction times across the board, potentially hinting at a major predictive factor in timing estimation being the total amount of scan parsing required.

#### MS/MS precursor and fragment search

We also investigated the efficacy of the various MS data formats for MS/MS data and found that support for fragmentation data searches was lacking or absent from the documentation and exposed functionality of each of these file types, requiring custom implementations every time. Despite both precursor searches (where all the precursors of a given fragment are found) and fragment searches (where all the fragments of a given precursor are found) representing intuitive and useful methods of MS/MS data processing, these timings were consistently among the slowest of the six query types for the non-database methods (Figure 3.2E and 3.2F).

All existing MS data types required multiple seconds to perform a single fragment search (Figure 3.2F), representing a significant bottleneck for any downstream analysis requiring the data associated with the fragments of a given precursor. The SQL-based parsers, on the other hand, all took fractions of a second and consistently returned the relevant data hundreds of times more quickly than existing methods. The same was true for a precursor search across all methods aside from mzDB (Figure 3.2E), which benefited significantly from constructing a single bounding box for all MS/MS information that requires a single decoding into computer memory, though this strategy will fail for any file with sufficiently large MS/MS data.

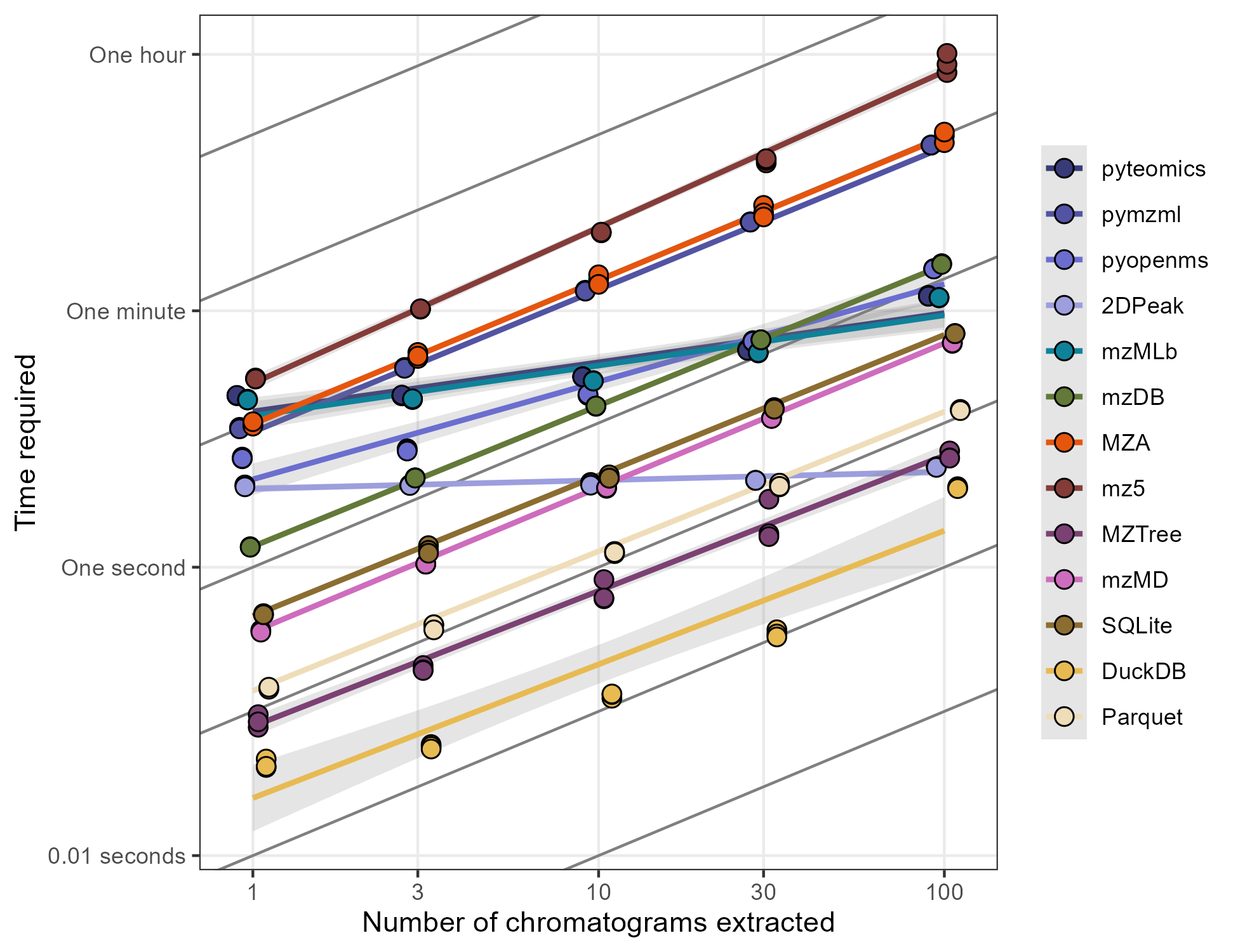
#### File sizes

File size is another important constraint on the efficacy of various MS formats. We measured the size on disk of each of the file types and found that they varied by approximately an order of magnitude, with HDF-based file types hovering around one-third the size of the mzML (mzML size = 75 megabytes (MB), mzMLb = 18 MB, mz5 = 23 MB) while mzDB and mzMD were larger (95 MB and 99 MB, respectively). The SQLite object was the largest on disk of all the file types, nearly tripling the mzML’s size at 197 MB, while DuckDB improved on MZA at two-thirds of the mzML (42 MB and 55 MB, respectively) and Parquet improved slightly upon that again (30 MB total) with its columnar-based storage format (Figure 3.2G).

However, these comparisons are not perfect because not all files store exactly the same data. MZTree and mzMD appear to entirely lack the MS/MS information in the sample DDA file, representing a potentially significant size reduction that’s difficult to estimate though the extraction of the same file via msconvert containing only MS1 scans was 58 MB, a 23% size reduction. The SQLite, DuckDB, and Parquet formats also lack the extensive scan and file metadata that’s present in the other file types, though it is difficult to estimate the fraction of disk space allocated for this (and which will depend upon the precise definition of metadata).

### Timings for multiple chromatograms

The single-file, single-metric case discussed above and shown in Figure 2 is largely a worst-case scenario for many MS data systems that have a slow initial setup step to make downstream analysis faster. To compare these systems more fairly to our database schema, we also tested timings across multiple chromatograms. In each case, this was implemented as a for loop iterating over an increasing number of chromatograms corresponding to the largest intensity ions in the file (Figure 3.3).



*Figure 3.3: Scatter plot of the time required to extract multiple chromatograms using various methods on logarithmic axes. Best-fit linear models have been added for each method are shown behind triplicate timing measurements. Transparent intervals around each best-fit line show a single standard error of the mean. Chromatograms correspond to the largest intensity ions in the file. 1:1 lines have been added in black behind the data for comparison.*

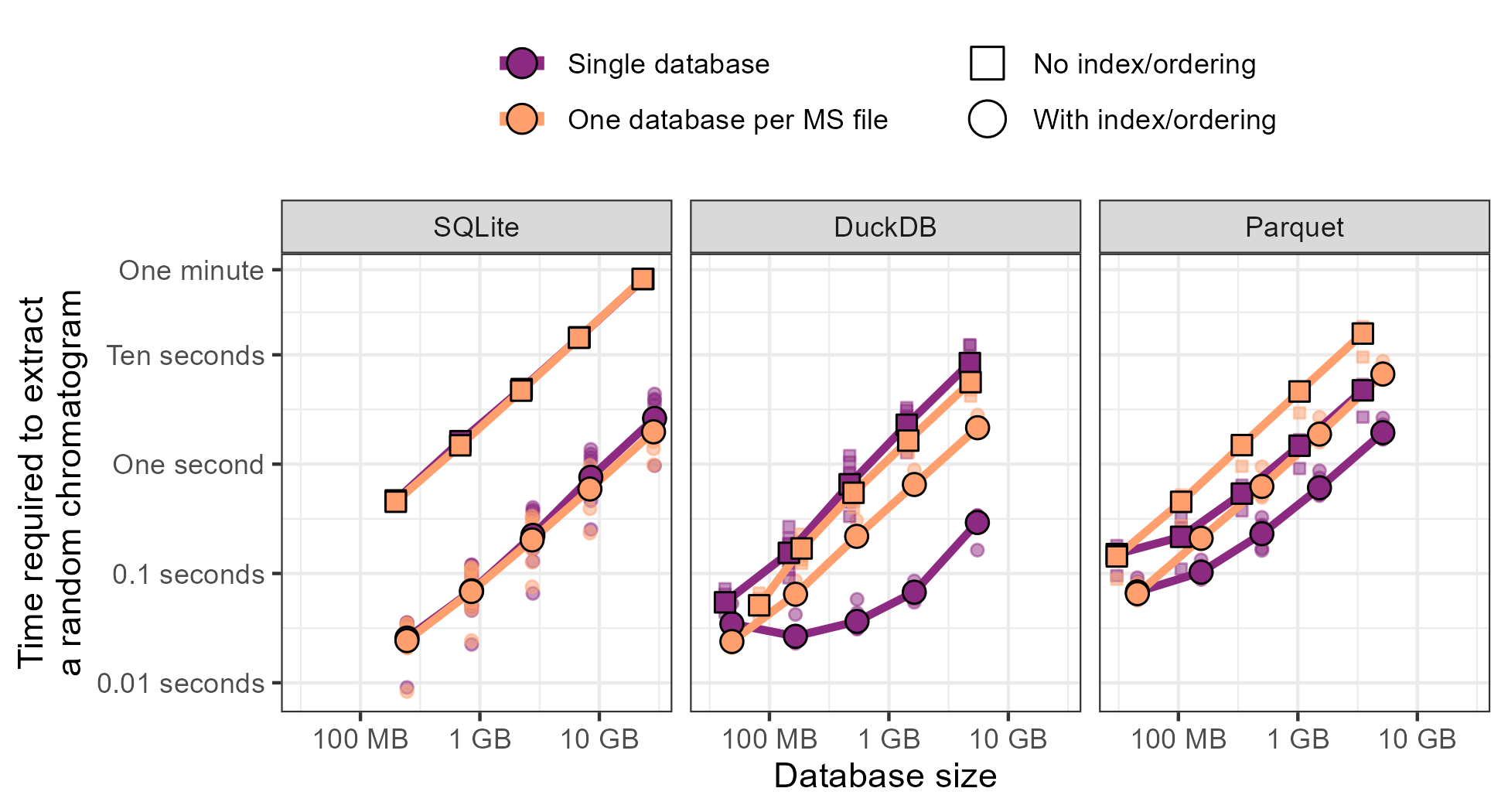
Most methods were linear extrapolations of the single chromatogram numbers shown above as expected from a simple for loop, with notable exceptions for pyteomics (and thus the mzMLb format), the 2D peak method of pyopenms, and DuckDB (Figure 3.3). Pyteomics and pyopenms both had significant overhead upon initial load resulting in faster subsequent queries that performed better than predicted from a 1:1 extrapolation, with pyopenms matching SQLite’s speed after 10 chromatograms and Parquet’s speed after 30. The methods that had a best-fit linear slope less than 1:1 also all had exponential fits, with performance at high chromatogram number worse than expected from a predicted fit to the timings for 1 and 3 chromatogram extractions (Supplemental figure 3).

We also explored whether database queries could be improved via the use of either a unified query (single SQL statement with multiple OR clauses for each ion’s *m/z* range) or a non-equi join between a peak table with *m/z* minimum and maximum columns (Supplemental Figure 4). SQLite and Parquet performed ~3-5 times faster with the unified query than with the loop method despite the necessity of and additional processing step for the looped query to correctly assign each data point to its original peak information. The opposite was true for DuckDB likely due to its optimized reader, with the unified query consistently outperformed by the non-equi join when 100 chromatograms were extracted.

### Database optimization via indices/ordering and multi-file constructions

Databases also provide multiple ways to optimize queries. SQLite allows the construction of indices for a field within a table that then speeds up queries at the cost of additional disk space. Alternatively, DuckDB and Parquet files rely predominantly on the data order when it’s written to disk and use their sophisticated row group methodology when subsetting.

We found that SQLite queries benefitted significantly from the construction of an index on the *m/z* column when extracting chromatograms, improving lookup times by an order of magnitude (dropping from 0.5 seconds to 0.03 seconds, Figure 3.4). However, because the SQLite index is stored on disk alongside the data, this improvement also increased the file size by 24%. Parquet files had the smallest improvement upon data ordering and required 50% more space, likely due to the reordering resulting in worse compression in other columns such as filename or retention time. DuckDB also improved significantly with ordered data but to a smaller degree and in contrast to SQLite or Parquet sometimes actually decreased in size when ordered.



*Figure 3.4: Time required to extract an ion chromatogram from multiple files plotted against the size of the data, broken down by the type of database used (SQLite, DuckDB, or Parquet). Points correspond to a random subset of 1, 3, 10, 30, and 100 files, respectively. Colors specify whether the data was stored as a single consolidated database (purple) or with a single database per file (orange) and the shape of the point denotes whether the database was unstructured (squares) or indexed/ordered by m/z (SQLite has indexes, DuckDB and Parquet benefit from ordering). Ten replicates of each query were performed and are shown transparently behind the mean values connected with lines.*

These improvements also persisted when multiple files were stored in a single database. We built databases consisting of between 1 and 100 individual MS files and tested the time required to extract ion chromatograms from each after an index was constructed (Figure 3.4). DuckDB was consistently the fastest ion chromatogram extraction method, with query times around 0.03 seconds for a single file and 1 second for one hundred files. SQLite had much higher variance and slower extraction times with datasets consisting of more than one file, typically an order of magnitude slower than DuckDB, while Parquet fell between the two. Importantly, only DuckDB had a slope much less than one. This is what would be expected if the database was performing a simple binary search on the index, with an expected time efficiency of O(log(# of files)). However, DuckDB’s performance degrades at larger database sizes and approaches a 1:1 slope, possibly due to the time required to read large amounts of data into memory after it’s found. We additionally compared these values to the timings obtained from converting each file into its own database and looping over each of those to confirm the linearity of that response (Figure 3.4).

## Discussion

As the gap between data scientist and mass spectrometrist continues to narrow, mass spectrometry (MS) data formats should facilitate this convergence. Instead, MS software remains relatively opaque. Documentation is sparse and data structures are complex, resulting in a landscape that is essentially restricted to the original developer’s intent. A particular pain point is the way in which MS data is stored because current methods must make trade-offs between simplicity, size, and speed. When we explored the wide range of MS file readers, we found every method had flaws that interfered with widespread adoption. The mzML file type appears to represent low-hanging fruit, with its XML-based structure that sacrifices speed and size in favor of clarity, but no alternative has yet reached a large audience of active users. Fifteen years of active competition continue to favor the highly explicit format, likely because users are leery of incomprehensible alternatives with no guarantee of continued maintenance.

Certain methods are clear winners for individual use cases but all of the existing formats failed to perform well at the full suite of exploratory data analysis tasks we attempted. Scan extraction is perhaps the most widely used query, especially for proteomics, and as a result has been extensively optimized. Here, MZA was blazingly fast thanks to the decision to index in the HDF5 file by scan number. The mzML files used here did have a precompiled index that should have made scan extraction highly efficient but this appeared to be mitigated for pyopenms and pyteomics due to their long initialization times, though pymzml performed very well at this task and the other two mzML methods were much faster after initialization (Supplemental Figure 2).

However, performant scan-based methods struggled significantly with chromatogram extraction because these axes are inherently orthogonal to each other. mzDB was competitive with the specifically optimized mzTree and mzMD formats, illustrating its success as an axis-agnostic structure. We were especially impressed with the 2DPeak method in pyopenms when extracting multiple chromatograms, as it had essentially a single setup time cost after which any number of chromatograms could be extracted for free and can therefore be highly recommended for visualization applications. Chromatograms (and retention time range queries) are of course only relevant for chromatography-based workflows but this type of analysis has become increasingly popular, making ion extraction increasingly important. None of the existing MS formats we tested performed very well on MS/MS data, despite the growing availability of fragmentation data, though our use-case is oriented more towards exploration instead of comprehensive analysis.

Finally, a complexity penalty must be noted for formats and packages requiring complex installation procedures. Pyopenms appears to be the worst offender here, with its bindings to the OpenMS C++ libraries requiring us to step back to Python 3.11 and numpy 1.26 to successfully access the data on our setup and required direct input from the maintainers. The Java applications for MzTree and, more egregiously mzMD, provided essentially zero documentation on installation consisting of a single README file without an intact HTTP request example and have not seen updates in years. Similarly, mzDB files were difficult to parse due to its opaque SQLite schema and use of the BLOB encoding type that again lacked examples or documentation outside of Java and had to be deduced iteratively. Of course, these methods eventually provided enough information that they could be parsed unlike Aird, toffee, and UIMF.

#### Timing comparison to existing literature

Novel formats are typically proposed with timing and sizing information, but the inconsistency of what’s being queried makes it difficult to directly compare across the literature. However, we mostly obtained results in line with those reported elsewhere where other intercomparisons have been performed and report here the widest set of comparisons between MS formats to our knowledge.

Wilhelm et al. (2012) Wilhelm et al. (2012) performed comparisons between mz5 and mzML with results indicating that mz5 was three times faster than mzML parsers. Their values of 0.16 seconds per million *m/z*/intensity arrays correspond to an estimated query time of 0.13 milliseconds which is faster than our measured 3 milliseconds, though our mz5 file was approximately four times smaller (20% mzML size) instead of half the mzML size.

Bouyssié et al. (2015) Bouyssié et al. (2015) also found that mz5 was about 80% smaller than the mzML and on par with their mzDB format. They reported query times of around 30 seconds for a wide (5 Da) ion chromatogram extraction and found that mz5 was about 40 times slower and that mzML was 200 times slower, in contrast to our parser which was 10 times slower for mz5 - possibly due to the large EIC width. They also were able to report mzDB scan queries on par with mz5 which we were able to replicate if the pyopenms or pyteomics libraries were used for full scan queries and pymzml for MS/MS.

MzTree (Handy et al. 2017) compared their SQLite-based system to mzML, mz5, and mzDB and reported high numbers for both EIC and RT range queries from mzML (4-1000 seconds) that were on par with the values we observed here. Their mzDB and mz5 results were unexpectedly comparable to each other at about 0.5 seconds per random EIC query, with the mzDB values equal to ours but the mz5 values much lower than our parser was able to obtain. Our disk size measurements also corresponded well with their size estimate of MzTree at approximately twice the size of the mzML, a surprising result given that our mzML file contained MS/MS information and theirs appearing to be full-scan only. Their mz5 files were much larger (80% mzML size instead of our 20%) and their mzDB much smaller (20% mzML size instead of our 110%).

The mzMD format (Yang et al. 2022) appears to be a thin wrapper around the MzTree format that applies a different philosophy for data subsetting and summarization. They report EIC queries in the 50 millisecond range, very similar to the values we obtained for the mzMD file type. They also estimate file size to be approximately 28 bytes per *m/z*/intensity tuple for a total size of 100MB in our test file which agrees reasonably well with our 72MB actual measurement. Their comparison to MzTree also agrees with our results as they report slightly larger disk usage and slightly better performance.

The mzMLb group (Bhamber et al. 2021) reports only info for spectrum access at approximately 15ms/scan which agrees with our 5-100ms/scan estimates only if the data is loaded ahead of time. They perform extensive comparisons to mzML at varying compression methods and levels but we stuck with the default options of 1024 KB chunk sizes for the mzMLb file and zlib-only compression for the mzML. This resulted in timing values very similar to those they reported when using the pyteomics library for access. They also compared to the mz5 file type and we are able to validate their results of the mzML+zlib occupying significantly more space, though our mz5 parser outperformed theirs for full scan and MS/MS data by two orders of magnitude in time.

We were also unable to test several other recent and promising formats. Aird does not report full query times for any of the metrics reviewed here, though they claim their StackZDPD algorithm (Wang et al. 2022) can improve decompression speed by three times and that the file size is 54% of the vendor file.(Lu et al. 2022) Similarly, the toffee format for time-of-flight DIA data reports sizes about equal to vendor or 60% of centroid mzml + numpress with query speeds 4 times faster for scans (spectrum-centric, 2 seconds for mzML, 0.5 for toffee) and 100 times faster for chromatograms (peptide-centric, 168 seconds for mzML, 1.8 for toffee).(Tully 2020) The Unified Ion Mobility Format (UIMF) Beagley et al. (2009) from Pacific Northwest National Laboratory format did not report direct comparisons to any of the available formats and thus we must remain unclear on its performance capabilities.

#### Fundamental inefficiencies in existing mass-spectrometry formats

We identified several fundamental inefficiencies when writing the parsers. First, scan metadata that was encoded within a scan instead of in a separate unit required looping over each scan to see whether it contained the information requested. Scan number, MS level, and retention time were all necessary bits of information that could be included in a file header or footer to relate the three to the data location within the file and allow index use instead of looping over every scan. Second, needing to decode or parse a compressed *m/z*/intensity array in each scan introduced an additional overhead that was especially punishing during ion chromatogram extraction and MS/MS search. While the *m/z* and intensity tuples are an obvious candidate for data compression, this penalty should be of significant concern to engineers. Third, looping over files is inherently slow and introduces additional complexity relative to a single unified database that encodes filename or sample ID as an additional column. A particular strength of the database system we propose is its inherent support for multi-file systems, while all other methods require looping over files.

The problems above highlight an important distinction between data *access* and data *search* that has been largely overlooked in our opinion. While HDF5 files or scan indexes excel at improving data access, they assume that the location of the data is known in advance and can be skipped to via bitwise offsets. If a search is required, however, this advantage is fully negated because each bit of information must be queried anyway. Finally, we must note that scan number is not inherently a useful bit of information. While we included it in our extraction metrics, it is entirely unclear when the scan number itself would be known in isolation. Additionally, this method often confuses the scan number with the scan’s indexes in the data structure. Scan number is not always consecutive (e.g. during polarity switching, multi-experiment samples, or if any filtering is performed during processing), so even if the first or second item in the structure can be queried speedily this is no guarantee that the item will contain the information of interest.

#### Leveraging robust, future-oriented software development with SQL

The proliferation of MS data storage formats and access algorithms illustrates the general dissatisfaction with existing alternatives to the vendor file or mzML. Formats that are faster to query or smaller on disk tend to be significantly more opaque, and those optimized for a particular method often fail to perform well on other metrics. This complexity is generally expected as optimization tends to require more complex data structures and assumptions about its use but it is not required if the complexity is outsourced to a robust and growing framework such as structured query language (SQL).

SQL is widely used for data processing outside of mass-spectrometry, though its adoption is increasing in recent years. Efforts like mzDB, the Pacific Marine Environment Laboratory’s UIMF format (Beagley et al. 2009), and the internals of MzTree hint at SQL’s suitability for MS data storage. SQL backends for the next-generation R processing package Spectra now exist (Rainer et al. 2022) and the development of MassQL (Jarmusch et al. 2022) indicates a growing comfort with SQL syntax for downstream processing, though the language itself strives for human readability in simple queries. The searching and subsetting inherent to MS data exploration represent very simple queries in database space, agnostic to high-level programming language and rarely requiring more than a single line of code. Additionally, the extensive documentation that exists across the internet means that large language models such as ChatGPT are easily able to translate queries for those unfamiliar with SQL’s syntax.

Just as the original database paper from Codd (1970) Codd (1970) argued that the same problems were being solved repeatedly, mass spectrometry data scientists are re-solving problems that have been more elegantly ironed out by dedicated teams in computer science and industry with much more extensive support. By leveraging existing optimizations in SQLite and DuckDB, we were able to create a highly performant system for storage of MS data that does not come with significant trade-offs between data extraction methods.

While SQLite is broadly used and its long history testifies to its continued utility, we can use even more modern database methods to improve further upon its analytical processing capacity. We tested both DuckDB the Apache Parquet data formats (Raasveldt and Mühleisen 2019; Vohra 2016) and found that they both performed better than SQLite in disk usage and query speed. DuckDB in particular is nearly a drop-in replacement for SQLite in many cases that’s been extensively optimized for MS-related queries given its online analytical processing (OLAP) structure. DuckDB provides automatic compression algorithms and uses zonemaps to create bounding boxes for each subset of data, bringing together existing optimizations from mz5 (delta encoding), mzDB (bounding boxes), and MzTree (axis-agnostic queries) at zero additional cost. Importantly, as with all databases, only the subset of interest needs to be written into memory, making the hardware requirements relatively lightweight.

Of course, to claim that existing frameworks should be discarded in favor of a novel method is to ignore decades of discussion. We acknowledge that our use case, that of largely exploratory and quality-control steps, is not a universal need and our lack of perfect metadata preservation in particular indicates that databases should become an auxiliary data structure alongside the vendor files or mzMLs, not substitute for them directly. Ultimately, the design decision for mass spectrometry data format will likely continue to be a point of contention and will result from a variety of factors, most crucially 1) initial vendor type, 2) programming language of the developer, 3) types of MS data included (e.g. full scan only versus MS/MS or metadata requirements), 4) whether the entire file will be processed or only a subset, and 5) how well a file type interfaces with downstream software. We intend to show with this manuscript that there is significant overlap between the goals of organizations much larger than any individual lab and that mass spectrometrists can benefit significantly from co-opting their development.

## Conclusion

We propose that a simple relational database is an intuitive and performant mass spectrometry (MS) data storage format. Tables containing fields that map directly to known MS concepts means that adoption is straightforward and facilitated by the widely-understood structured query language (SQL), reducing the code required to extract subsets of interest to a single line. We show that this structure can also take advantage of regular advancements in computer science by leveraging modern data formats such as DuckDB and Parquet to reduce the disk space required while improving access times by 1-2 orders of magnitude. We hope that widespread adoption of this format alongside the metadata-heavy vendor and mzML files will reduce the barriers to data access for mass spectrometrists and provide a consistent framework that covers a majority of the exploratory use cases.

## Acknowledgements

We would like to acknowledge the University of Washington’s eScience Institute and especially Bryna Hazelton and Dave Beck for their guidance and support during this project. We are also grateful to Theo Portlock and the other authors of their 2025 manuscript for posting their metabolomics data to Metabolights and allowing us to reuse it. Finally, we would like to acknowledge Josh Sacks and other members of the Ingalls Lab for their helpful discussions and for beta-testing many parts of the project.

## Data availability

All data and code are available on the Github repository associated with this project at https://github.com/wkumler/mzsql under the manuscript\_things branch.

# Chapter 4: Picky with Peakpicking: Assessing Chromatographic Peak Quality with Simple Metrics in Metabolomics

# Chapter 5: Metabolites Reflect Variability Introduced by Mesoscale Eddies in the North Pacific Subtropical Gyre

# Chapter 6: The Form of Nitrogen Determines its Fate in the North Pacific Subtropical Gyre

# Chapter 7: Conclusions

Note that I’m unhappy I didn’t get to do more MS/MS stuff, partially due to the tools not being very good - diagnostic fragments, wildly varying ways/formats to query, Metlin going private, unclear how to create consensus spectra from multiple scans or match these to knowns, limited database availability, lack of MS/MS stuff in my own data - would love to do this next.

# Bibliography

Adams, Kendra J., Brian Pratt, Neelanjan Bose, Laura G. Dubois, Lisa St. John-Williams, Kevin M. Perrott, Karina Ky, et al. 2020. “Skyline for Small Molecules: A Unifying Software Package for Quantitative Metabolomics.” *Journal of Proteome Research* 19 (4): 1447–58. <https://doi.org/10.1021/acs.jproteome.9b00640>.

Askenazi, Manor, Hisham Ben Hamidane, and Johannes Graumann. 2017. “The Arc of Mass Spectrometry Exchange Formats Is Long, but It Bends Toward HDF5.” *Mass Spectrometry Reviews* 36 (5): 668–73. <https://doi.org/10.1002/mas.21522>.

Bauermeister, Anelize, Helena Mannochio-Russo, Letícia V. Costa-Lotufo, Alan K. Jarmusch, and Pieter C. Dorrestein. 2022. “Mass Spectrometry-Based Metabolomics in Microbiome Investigations.” *Nature Reviews Microbiology* 20 (3): 143–60. <https://doi.org/10.1038/s41579-021-00621-9>.

Beagley, Nathaniel, Chad Scherrer, Yan Shi, Brian H. Clowers, William F. Danielson, and Anuj R. Shah. 2009. “Increasing the Efficiency of Data Storage and Analysis Using Indexed Compression.” In *2009 Fifth IEEE International Conference on e-Science*, 66–71. Oxford, United Kingdom: IEEE. <https://doi.org/10.1109/e-Science.2009.18>.

Becker, Kevin W., James R. Collins, Bryndan P. Durham, Ryan D. Groussman, Angelicque E. White, Helen F. Fredricks, Justin E. Ossolinski, et al. 2018. “Daily Changes in Phytoplankton Lipidomes Reveal Mechanisms of Energy Storage in the Open Ocean.” *Nature Communications* 9 (1): 5179. <https://doi.org/10.1038/s41467-018-07346-z>.

Bertrand, Erin M., John P. McCrow, Ahmed Moustafa, Hong Zheng, Jeffrey B. McQuaid, Tom O. Delmont, Anton F. Post, et al. 2015. “Phytoplankton–Bacterial Interactions Mediate Micronutrient Colimitation at the Coastal Antarctic Sea Ice Edge.” *Proceedings of the National Academy of Sciences* 112 (32): 9938–43. <https://doi.org/10.1073/pnas.1501615112>.

Bhamber, Ranjeet S., Andris Jankevics, Eric W. Deutsch, Andrew R. Jones, and Andrew W. Dowsey. 2021. “mzMLb: A Future-Proof Raw Mass Spectrometry Data Format Based on Standards-Compliant mzML and Optimized for Speed and Storage Requirements.” *Journal of Proteome Research* 20 (1): 172–83. <https://doi.org/10.1021/acs.jproteome.0c00192>.

Bilbao, Aivett, Dylan H. Ross, Joon-Yong Lee, Micah T. Donor, Sarah M. Williams, Ying Zhu, Yehia M. Ibrahim, Richard D. Smith, and Xueyun Zheng. 2023. “MZA: A Data Conversion Tool to Facilitate Software Development and Artificial Intelligence Research in Multidimensional Mass Spectrometry.” *Journal of Proteome Research* 22 (2): 508–13. <https://doi.org/10.1021/acs.jproteome.2c00313>.

Bouyssié, David, Marc Dubois, Sara Nasso, Anne Gonzalez De Peredo, Odile Burlet-Schiltz, Ruedi Aebersold, and Bernard Monsarrat. 2015. “mzDB: A File Format Using Multiple Indexing Strategies for the Efficient Analysis of Large LC-MS/MS and SWATH-MS Data Sets \*.” *Molecular & Cellular Proteomics* 14 (3): 771–81. <https://doi.org/10.1074/mcp.O114.039115>.

Boyd, C., and D. Gradmann. 2002. “Impact of Osmolytes on Buoyancy of Marine Phytoplankton.” *Marine Biology* 141 (4): 605–18. <https://doi.org/10.1007/s00227-002-0872-z>.

Boysen, Angela K., Laura T. Carlson, Bryndan P. Durham, Ryan D. Groussman, Frank O. Aylward, François Ribalet, Katherine R. Heal, et al. 2021. “Particulate Metabolites and Transcripts Reflect Diel Oscillations of Microbial Activity in the Surface Ocean.” Edited by Jeff Bowman. *mSystems* 6 (3): e00896–20. <https://doi.org/10.1128/mSystems.00896-20>.

Boysen, Angela K., Katherine R. Heal, Laura T. Carlson, and Anitra E. Ingalls. 2018. “Best-Matched Internal Standard Normalization in Liquid Chromatography–Mass Spectrometry Metabolomics Applied to Environmental Samples.” *Analytical Chemistry* 90 (2): 1363–69. <https://doi.org/10.1021/acs.analchem.7b04400>.

Buszewski, Bogusław, and Sylwia Noga. 2012. “Hydrophilic Interaction Liquid Chromatography (HILIC)—a Powerful Separation Technique.” *Analytical and Bioanalytical Chemistry* 402 (1): 231–47. <https://doi.org/10.1007/s00216-011-5308-5>.

Cajka, Tomas, and Oliver Fiehn. 2016. “Toward Merging Untargeted and Targeted Methods in Mass Spectrometry-Based Metabolomics and Lipidomics.” *Analytical Chemistry* 88 (1): 524–45. <https://doi.org/10.1021/acs.analchem.5b04491>.

Chambers, Matthew C, Brendan Maclean, Robert Burke, Dario Amodei, Daniel L Ruderman, Steffen Neumann, Laurent Gatto, et al. 2012. “A Cross-Platform Toolkit for Mass Spectrometry and Proteomics.” *Nature Biotechnology* 30 (10): 918–20. <https://doi.org/10.1038/nbt.2377>.

Coble, Jamie B., and Carlos G. Fraga. 2014. “Comparative Evaluation of Preprocessing Freeware on Chromatography/Mass Spectrometry Data for Signature Discovery.” *Journal of Chromatography A* 1358 (September): 155–64. <https://doi.org/10.1016/j.chroma.2014.06.100>.

Codd, E. F. 1970. “A Relational Model of Data for Large Shared Data Banks.” *Communications of the ACM* 13 (6): 377–87. <https://doi.org/10.1145/362384.362685>.

Collette, Andrew, James Tocknell, Thomas A Caswell, Darren Dale, Ulrik Kofoed Pedersen, Aleksandar Jelenak, Andrea Bedini, et al. 2017. “H5py/H5py: REL: 2.7.1.” Zenodo. <https://doi.org/10.5281/ZENODO.877338>.

Dawson, H M, E Connors, N G Erazo, J S Sacks, V Mierzejewski, S M Rundell, L T Carlson, et al. 2023. “Microbial Metabolomic Responses to Changes in Temperature and Salinity Along the Western Antarctic Peninsula.” *The ISME Journal* 17 (11): 2035–46. <https://doi.org/10.1038/s41396-023-01475-0>.

Dawson, H M, K R Heal, A Torstensson, L T Carlson, A E Ingalls, and J N Young. 2020. “Large Diversity in Nitrogen- and Sulfur-Containing Compatible Solute Profiles in Polar and Temperate Diatoms.” *Integrative and Comparative Biology* 60 (6): 1401–13. <https://doi.org/10.1093/icb/icaa133>.

DeLong, Edward F. 2005. “Microbial Community Genomics in the Ocean.” *Nature Reviews Microbiology* 3 (6): 459–69. <https://doi.org/10.1038/nrmicro1158>.

Deutsch, Eric W. 2010. “Mass Spectrometer Output File Format mzML.” In *Proteome Bioinformatics*, edited by Simon J. Hubbard and Andrew R. Jones, 604:319–31. Totowa, NJ: Humana Press. <https://doi.org/10.1007/978-1-60761-444-9_22>.

Durham, Bryndan P., Angela K. Boysen, Katherine R. Heal, Laura T. Carlson, Rachel Boccamazzo, Chloe R. Deodato, Wei Qin, Rose Ann Cattolico, E. Virginia Armbrust, and Anitra E. Ingalls. 2022. “Chemotaxonomic Patterns in Intracellular Metabolites of Marine Microbial Plankton.” *Frontiers in Marine Science* 9 (September): 864796. <https://doi.org/10.3389/fmars.2022.864796>.

Edwards, Bethanie R. 2023. “Lipid Biogeochemistry and Modern Lipidomic Techniques.” *Annual Review of Marine Science* 15 (1): 485–508. <https://doi.org/10.1146/annurev-marine-040422-094104>.

Falkowski, Paul G. 1994. “The Role of Phytoplankton Photosynthesis in Global Biogeochemical Cycles.” *Photosynthesis Research* 39 (3): 235–58. <https://doi.org/10.1007/BF00014586>.

Falkowski, Paul G., Tom Fenchel, and Edward F. Delong. 2008. “The Microbial Engines That Drive Earth’s Biogeochemical Cycles.” *Science* 320 (5879): 1034–39. <https://doi.org/10.1126/science.1153213>.

Ferrer-González, Frank Xavier, Brittany Widner, Nicole R. Holderman, John Glushka, Arthur S. Edison, Elizabeth B. Kujawinski, and Mary Ann Moran. 2021. “Resource Partitioning of Phytoplankton Metabolites That Support Bacterial Heterotrophy.” *The ISME Journal* 15 (3): 762–73. <https://doi.org/10.1038/s41396-020-00811-y>.

Gatto, Laurent, Sebastian Gibb, and Johannes Rainer. 2021. “MSnbase, Efficient and Elegant R-Based Processing and Visualization of Raw Mass Spectrometry Data.” *Journal of Proteome Research* 20 (1): 1063–69. <https://doi.org/10.1021/acs.jproteome.0c00313>.

Giera, Martin, Oscar Yanes, and Gary Siuzdak. 2022. “Metabolite Discovery: Biochemistry’s Scientific Driver.” *Cell Metabolism* 34 (1): 21–34. <https://doi.org/10.1016/j.cmet.2021.11.005>.

Gika, Helen, Christina Virgiliou, Georgios Theodoridis, Robert S. Plumb, and Ian D. Wilson. 2019. “Untargeted LC/MS-based Metabolic Phenotyping (Metabonomics/Metabolomics): The State of the Art.” *Journal of Chromatography B* 1117 (June): 136–47. <https://doi.org/10.1016/j.jchromb.2019.04.009>.

Handy, Kyle, Jebediah Rosen, André Gillan, and Rob Smith. 2017. “Fast, Axis-Agnostic, Dynamically Summarized Storage and Retrieval for Mass Spectrometry Data.” Edited by Jon M. Jacobs. *PLOS ONE* 12 (11): e0188059. <https://doi.org/10.1371/journal.pone.0188059>.

Haug, Kenneth, Keeva Cochrane, Venkata Chandrasekhar Nainala, Mark Williams, Jiakang Chang, Kalai Vanii Jayaseelan, and Claire O’Donovan. 2019. “MetaboLights: A Resource Evolving in Response to the Needs of Its Scientific Community.” *Nucleic Acids Research*, November, gkz1019. <https://doi.org/10.1093/nar/gkz1019>.

Heal, Katherine R., Bryndan P. Durham, Angela K. Boysen, Laura T. Carlson, Wei Qin, François Ribalet, Angelicque E. White, Randelle M. Bundy, E. Virginia Armbrust, and Anitra E. Ingalls. 2021. “Marine Community Metabolomes Carry Fingerprints of Phytoplankton Community Composition.” Edited by Manuel Liebeke. *mSystems* 6 (3): e01334–20. <https://doi.org/10.1128/mSystems.01334-20>.

Heal, Katherine R., Wei Qin, Francois Ribalet, Anthony D. Bertagnolli, Willow Coyote-Maestas, Laura R. Hmelo, James W. Moffett, et al. 2017. “Two Distinct Pools of B12 Analogs Reveal Community Interdependencies in the Ocean.” *Proceedings of the National Academy of Sciences* 114 (2): 364–69. <https://doi.org/10.1073/pnas.1608462114>.

Heuckeroth, Steffen, Tito Damiani, Aleksandr Smirnov, Olena Mokshyna, Corinna Brungs, Ansgar Korf, Joshua David Smith, et al. 2024. “Reproducible Mass Spectrometry Data Processing and Compound Annotation in MZmine 3.” *Nature Protocols* 19 (9): 2597–2641. <https://doi.org/10.1038/s41596-024-00996-y>.

Hohrenk, Lotta L., Fabian Itzel, Nicolai Baetz, Jochen Tuerk, Maryam Vosough, and Torsten C. Schmidt. 2020. “Comparison of Software Tools for Liquid Chromatography–High-Resolution Mass Spectrometry Data Processing in Nontarget Screening of Environmental Samples.” *Analytical Chemistry* 92 (2): 1898–1907. <https://doi.org/10.1021/acs.analchem.9b04095>.

Iversen, Morten H. 2023. “Carbon Export in the Ocean: A Biologist’s Perspective.” *Annual Review of Marine Science* 15 (1): 357–81. <https://doi.org/10.1146/annurev-marine-032122-035153>.

Jarmusch, Alan K., Allegra T. Aron, Daniel Petras, Vanessa V. Phelan, Wout Bittremieux, Deepa D. Acharya, Mohammed M. A. Ahmed, et al. 2022. “A Universal Language for Finding Mass Spectrometry Data Patterns.” <https://doi.org/10.1101/2022.08.06.503000>.

Johnson, Winifred M., Melissa C. Kido Soule, and Elizabeth B. Kujawinski. 2017. “Extraction Efficiency and Quantification of Dissolved Metabolites in Targeted Marine Metabolomics.” *Limnology and Oceanography: Methods* 15 (4): 417–28. <https://doi.org/10.1002/lom3.10181>.

Johnson, Winifred M., Melissa C. Kido Soule, Krista Longnecker, Maya P. Bhatia, Steven J. Hallam, Michael W. Lomas, and Elizabeth B. Kujawinski. 2023. “Particulate and Dissolved Metabolite Distributions Along a Latitudinal Transect of the Western Atlantic Ocean.” *Limnology and Oceanography* 68 (2): 377–93. <https://doi.org/10.1002/lno.12275>.

Johnson, Winifred M., Krista Longnecker, Melissa C. Kido Soule, William A. Arnold, Maya P. Bhatia, Steven J. Hallam, Benjamin A. S. Van Mooy, and Elizabeth B. Kujawinski. 2020. “Metabolite Composition of Sinking Particles Differs from Surface Suspended Particles Across a Latitudinal Transect in the South Atlantic.” *Limnology and Oceanography* 65 (1): 111–27. <https://doi.org/10.1002/lno.11255>.

Jones, Catriona L. C., Judith Camps-Castella, Mike Smykala, Morgan S. Sobol, and Keisuke Inomura. 2024. “Seeing Through the Gray Box: An Integrated Approach to Physiological Modeling of Phytoplankton Stoichiometry.” *Frontiers in Ecology and Evolution* 12 (December): 1505025. <https://doi.org/10.3389/fevo.2024.1505025>.

Kanehisa, M. 2000. “KEGG: Kyoto Encyclopedia of Genes and Genomes.” *Nucleic Acids Research* 28 (1): 27–30. <https://doi.org/10.1093/nar/28.1.27>.

Karl, David M., and Matthew J. Church. 2017. “Ecosystem Structure and Dynamics in the North Pacific Subtropical Gyre: New Views of an Old Ocean.” *Ecosystems* 20 (3): 433–57. <https://doi.org/10.1007/s10021-017-0117-0>.

Karp, Peter D, Richard Billington, Ron Caspi, Carol A Fulcher, Mario Latendresse, Anamika Kothari, Ingrid M Keseler, et al. 2019. “The BioCyc Collection of Microbial Genomes and Metabolic Pathways.” *Briefings in Bioinformatics* 20 (4): 1085–93. <https://doi.org/10.1093/bib/bbx085>.

Kido Soule, Melissa C., Krista Longnecker, Winifred M. Johnson, and Elizabeth B. Kujawinski. 2015. “Environmental Metabolomics: Analytical Strategies.” *Marine Chemistry* 177 (December): 374–87. <https://doi.org/10.1016/j.marchem.2015.06.029>.

Kind, Tobias, and Oliver Fiehn. 2006. “Metabolomic Database Annotations via Query of Elemental Compositions: Mass Accuracy Is Insufficient Even at Less Than 1 Ppm.” *BMC Bioinformatics* 7 (1): 234. <https://doi.org/10.1186/1471-2105-7-234>.

Kujawinski, Elizabeth B., Rogier Braakman, Krista Longnecker, Jamie W. Becker, Sallie W. Chisholm, Keven Dooley, Melissa C. Kido Soule, Gretchen J. Swarr, and Kathryn Halloran. 2023. “Metabolite Diversity Among Representatives of Divergent *Prochlorococcus* Ecotypes.” Edited by Jack A. Gilbert. *mSystems* 8 (5): e01261–22. <https://doi.org/10.1128/msystems.01261-22>.

Kumler, William, and Anitra Ingalls. 2022. “Tidy Data Neatly Resolves Mass-Spectrometry’s Ragged Arrays.” *The R Journal* 14 (3): 193–202. <https://doi.org/10.32614/RJ-2022-050>.

Ligrone, Roberto. 2019. “The Great Oxygenation Event.” In *Biological Innovations That Built the World*, 129–54. Cham: Springer International Publishing. <https://doi.org/10.1007/978-3-030-16057-9_5>.

Longnecker, Krista, Melissa C. Kido Soule, and Elizabeth B. Kujawinski. 2024. “A Witches’ Brew: Dissolved Metabolites in Seawater.” In *Biogeochemistry of Marine Dissolved Organic Matter*, 69–99. Elsevier. <https://doi.org/10.1016/B978-0-443-13858-4.00007-1>.

Lu, Miaoshan, Shaowei An, Ruimin Wang, Jinyin Wang, and Changbin Yu. 2022. “Aird: A Computation-Oriented Mass Spectrometry Data Format Enables a Higher Compression Ratio and Less Decoding Time.” *BMC Bioinformatics* 23 (1): 35. <https://doi.org/10.1186/s12859-021-04490-0>.

Martens, Lennart, Matthew Chambers, Marc Sturm, Darren Kessner, Fredrik Levander, Jim Shofstahl, Wilfred H. Tang, et al. 2011. “mzML—a Community Standard for Mass Spectrometry Data.” *Molecular & Cellular Proteomics* 10 (1): R110.000133. <https://doi.org/10.1074/mcp.R110.000133>.

Mojzeš, Peter, Lu Gao, Tatiana Ismagulova, Jana Pilátová, Šárka Moudříková, Olga Gorelova, Alexei Solovchenko, Ladislav Nedbal, and Anya Salih. 2020. “Guanine, a High-Capacity and Rapid-Turnover Nitrogen Reserve in Microalgal Cells.” *Proceedings of the National Academy of Sciences* 117 (51): 32722–30. <https://doi.org/10.1073/pnas.2005460117>.

Moore, C. M., M. M. Mills, K. R. Arrigo, I. Berman-Frank, L. Bopp, P. W. Boyd, E. D. Galbraith, et al. 2013. “Processes and Patterns of Oceanic Nutrient Limitation.” *Nature Geoscience* 6 (9): 701–10. <https://doi.org/10.1038/ngeo1765>.

Moran, Mary Ann, Frank X. Ferrer-González, He Fu, Brent Nowinski, Malin Olofsson, McKenzie A. Powers, Jeremy E. Schreier, William F. Schroer, Christa B. Smith, and Mario Uchimiya. 2022. “The Ocean’s Labile DOC Supply Chain.” *Limnology and Oceanography* 67 (5): 1007–21. <https://doi.org/10.1002/lno.12053>.

Moran, Mary Ann, Elizabeth B. Kujawinski, William F. Schroer, Shady A. Amin, Nicholas R. Bates, Erin M. Bertrand, Rogier Braakman, et al. 2022. “Microbial Metabolites in the Marine Carbon Cycle.” *Nature Microbiology* 7 (4): 508–23. <https://doi.org/10.1038/s41564-022-01090-3>.

Moran, Mary Ann, Elizabeth B. Kujawinski, Aron Stubbins, Rob Fatland, Lihini I. Aluwihare, Alison Buchan, Byron C. Crump, et al. 2016. “Deciphering Ocean Carbon in a Changing World.” *Proceedings of the National Academy of Sciences* 113 (12): 3143–51. <https://doi.org/10.1073/pnas.1514645113>.

Muratore, Daniel, Angela K. Boysen, Matthew J. Harke, Kevin W. Becker, John R. Casey, Sacha N. Coesel, Daniel R. Mende, et al. 2022. “Complex Marine Microbial Communities Partition Metabolism of Scarce Resources over the Diel Cycle.” *Nature Ecology & Evolution* 6 (2): 218–29. <https://doi.org/10.1038/s41559-021-01606-w>.

Myers, Owen D., Susan J. Sumner, Shuzhao Li, Stephen Barnes, and Xiuxia Du. 2017a. “Detailed Investigation and Comparison of the XCMS and MZmine 2 Chromatogram Construction and Chromatographic Peak Detection Methods for Preprocessing Mass Spectrometry Metabolomics Data.” *Analytical Chemistry* 89 (17): 8689–95. <https://doi.org/10.1021/acs.analchem.7b01069>.

———. 2017b. “One Step Forward for Reducing False Positive and False Negative Compound Identifications from Mass Spectrometry Metabolomics Data: New Algorithms for Constructing Extracted Ion Chromatograms and Detecting Chromatographic Peaks.” *Analytical Chemistry* 89 (17): 8696–8703. <https://doi.org/10.1021/acs.analchem.7b00947>.

Narainsamy, Kinsley, Sandrine Farci, Emilie Braun, Christophe Junot, Corinne Cassier-Chauvat, and Franck Chauvat. 2016. “Oxidative-Stress Detoxification and Signalling in Cyanobacteria: The Crucial Glutathione Synthesis Pathway Supports the Production of Ergothioneine and Ophthalmate.” *Molecular Microbiology* 100 (1): 15–24. <https://doi.org/10.1111/mmi.13296>.

Patti, Gary J., Oscar Yanes, and Gary Siuzdak. 2012. “Metabolomics: The Apogee of the Omics Trilogy.” *Nature Reviews Molecular Cell Biology* 13 (4): 263–69. <https://doi.org/10.1038/nrm3314>.

Portlock, T., T. Shama, S. H. Kakon, B. Hartjen, C. Pook, B. C. Wilson, A. Bhuttor, et al. 2025. “Interconnected Pathways Link Faecal Microbiota Plasma Lipids and Brain Activity to Childhood Malnutrition Related Cognition.” *Nature Communications* 16 (1): 473. <https://doi.org/10.1038/s41467-024-55798-3>.

Raasveldt, Mark, and Hannes Mühleisen. 2019. “DuckDB: An Embeddable Analytical Database.” In *Proceedings of the 2019 International Conference on Management of Data*, 1981–84. Amsterdam Netherlands: ACM. <https://doi.org/10.1145/3299869.3320212>.

Rafiei, Atefeh, and Lekha Sleno. 2015. “Comparison of Peak-Picking Workflows for Untargeted Liquid Chromatography/High-Resolution Mass Spectrometry Metabolomics Data Analysis.” *Rapid Communications in Mass Spectrometry* 29 (1): 119–27. <https://doi.org/10.1002/rcm.7094>.

Rainer, Johannes, Andrea Vicini, Liesa Salzer, Jan Stanstrup, Josep M. Badia, Steffen Neumann, Michael A. Stravs, et al. 2022. “A Modular and Expandable Ecosystem for Metabolomics Data Annotation in R.” *Metabolites* 12 (2): 173. <https://doi.org/10.3390/metabo12020173>.

Römpp, Andreas, Thorsten Schramm, Alfons Hester, Ivo Klinkert, Jean-Pierre Both, Ron M. A. Heeren, Markus Stöckli, and Bernhard Spengler. 2011. “imzML: Imaging Mass Spectrometry Markup Language: A Common Data Format for Mass Spectrometry Imaging.” In *Data Mining in Proteomics*, edited by Michael Hamacher, Martin Eisenacher, and Christian Stephan, 696:205–24. Totowa, NJ: Humana Press. <https://doi.org/10.1007/978-1-60761-987-1_12>.

Röst, Hannes L, George Rosenberger, Pedro Navarro, Ludovic Gillet, Saša M Miladinović, Olga T Schubert, Witold Wolski, et al. 2014. “OpenSWATH Enables Automated, Targeted Analysis of Data-Independent Acquisition MS Data.” *Nature Biotechnology* 32 (3): 219–23. <https://doi.org/10.1038/nbt.2841>.

Röst, Hannes L., Uwe Schmitt, Ruedi Aebersold, and Lars Malmström. 2015. “Fast and Efficient XML Data Access for Next-Generation Mass Spectrometry.” Edited by Lennart Martens. *PLOS ONE* 10 (4): e0125108. <https://doi.org/10.1371/journal.pone.0125108>.

Sacks, Joshua S., Katherine R. Heal, Angela K. Boysen, Laura T. Carlson, and Anitra E. Ingalls. 2022. “Quantification of Dissolved Metabolites in Environmental Samples Through Cation-Exchange Solid-Phase Extraction Paired with Liquid Chromatography–Mass Spectrometry.” *Limnology and Oceanography: Methods* 20 (11): 683–700. <https://doi.org/10.1002/lom3.10513>.

Salazar, Guillem, and Shinichi Sunagawa. 2017. “Marine Microbial Diversity.” *Current Biology* 27 (11): R489–94. <https://doi.org/10.1016/j.cub.2017.01.017>.

Sañudo-Wilhelmy, Sergio A., Laura Gómez-Consarnau, Christopher Suffridge, and Eric A. Webb. 2014. “The Role of B Vitamins in Marine Biogeochemistry.” *Annual Review of Marine Science* 6 (1): 339–67. <https://doi.org/10.1146/annurev-marine-120710-100912>.

Schmid, Robin, Steffen Heuckeroth, Ansgar Korf, Aleksandr Smirnov, Owen Myers, Thomas S. Dyrlund, Roman Bushuiev, et al. 2023. “Integrative Analysis of Multimodal Mass Spectrometry Data in MZmine 3.” *Nature Biotechnology* 41 (4): 447–49. <https://doi.org/10.1038/s41587-023-01690-2>.

Schrimpe-Rutledge, Alexandra C., Simona G. Codreanu, Stacy D. Sherrod, and John A. McLean. 2016. “Untargeted Metabolomics Strategies—Challenges and Emerging Directions.” *Journal of the American Society for Mass Spectrometry* 27 (12): 1897–1905. <https://doi.org/10.1007/s13361-016-1469-y>.

Shah, Anuj R., Jennifer Davidson, Matthew E. Monroe, Anoop M. Mayampurath, William F. Danielson, Yan Shi, Aaron C. Robinson, et al. 2010. “An Efficient Data Format for Mass Spectrometry-Based Proteomics.” *Journal of the American Society for Mass Spectrometry* 21 (10): 1784–88. <https://doi.org/10.1016/j.jasms.2010.06.014>.

Siegel, David A., Timothy DeVries, Ivona Cetinić, and Kelsey M. Bisson. 2023. “Quantifying the Ocean’s Biological Pump and Its Carbon Cycle Impacts on Global Scales.” *Annual Review of Marine Science* 15 (1): 329–56. <https://doi.org/10.1146/annurev-marine-040722-115226>.

Smith, Colin A., Elizabeth J. Want, Grace O’Maille, Ruben Abagyan, and Gary Siuzdak. 2006. “XCMS: Processing Mass Spectrometry Data for Metabolite Profiling Using Nonlinear Peak Alignment, Matching, and Identification.” *Analytical Chemistry* 78 (3): 779–87. <https://doi.org/10.1021/ac051437y>.

Smith, R., D. Ventura, and J. T. Prince. 2015. “LC-MS Alignment in Theory and Practice: A Comprehensive Algorithmic Review.” *Briefings in Bioinformatics* 16 (1): 104–17. <https://doi.org/10.1093/bib/bbt080>.

Sud, Manish, Eoin Fahy, Dawn Cotter, Kenan Azam, Ilango Vadivelu, Charles Burant, Arthur Edison, et al. 2016. “Metabolomics Workbench: An International Repository for Metabolomics Data and Metadata, Metabolite Standards, Protocols, Tutorials and Training, and Analysis Tools.” *Nucleic Acids Research* 44 (D1): D463–70. <https://doi.org/10.1093/nar/gkv1042>.

Tautenhahn, Ralf, Christoph Böttcher, and Steffen Neumann. 2008. “Highly Sensitive Feature Detection for High Resolution LC/MS.” *BMC Bioinformatics* 9 (1): 504. <https://doi.org/10.1186/1471-2105-9-504>.

Thukral, Monica, Andrew E. Allen, and Daniel Petras. 2023. “Progress and Challenges in Exploring Aquatic Microbial Communities Using Non-Targeted Metabolomics.” *The ISME Journal* 17 (12): 2147–59. <https://doi.org/10.1038/s41396-023-01532-8>.

Ting, Ying S., Jarrett D. Egertson, Samuel H. Payne, Sangtae Kim, Brendan MacLean, Lukas Käll, Ruedi Aebersold, Richard D. Smith, William Stafford Noble, and Michael J. MacCoss. 2015. “Peptide-Centric Proteome Analysis: An Alternative Strategy for the Analysis of Tandem Mass Spectrometry Data.” *Molecular & Cellular Proteomics* 14 (9): 2301–7. <https://doi.org/10.1074/mcp.O114.047035>.

Tsugawa, Hiroshi, Tomas Cajka, Tobias Kind, Yan Ma, Brendan Higgins, Kazutaka Ikeda, Mitsuhiro Kanazawa, Jean VanderGheynst, Oliver Fiehn, and Masanori Arita. 2015. “MS-DIAL: Data-Independent MS/MS Deconvolution for Comprehensive Metabolome Analysis.” *Nature Methods* 12 (6): 523–26. <https://doi.org/10.1038/nmeth.3393>.

Tully, Brett. 2020. “Toffee – a Highly Efficient, Lossless File Format for DIA-MS.” *Scientific Reports* 10 (1): 8939. <https://doi.org/10.1038/s41598-020-65015-y>.

Van Mooy, Benjamin A. S., Helen F. Fredricks, Byron E. Pedler, Sonya T. Dyhrman, David M. Karl, Michal Koblížek, Michael W. Lomas, et al. 2009. “Phytoplankton in the Ocean Use Non-Phosphorus Lipids in Response to Phosphorus Scarcity.” *Nature* 458 (7234): 69–72. <https://doi.org/10.1038/nature07659>.

Vardi, Assaf, Fabio Formiggini, Raffaella Casotti, Alessandra De Martino, François Ribalet, Antonio Miralto, and Chris Bowler. 2006. “A Stress Surveillance System Based on Calcium and Nitric Oxide in Marine Diatoms.” Edited by Jeffrey Dangl. *PLoS Biology* 4 (3): e60. <https://doi.org/10.1371/journal.pbio.0040060>.

Vohra, Deepak. 2016. “Apache Parquet.” In *Practical Hadoop Ecosystem*, 325–35. Berkeley, CA: Apress. <https://doi.org/10.1007/978-1-4842-2199-0_8>.

Wang, Jinyin, Miaoshan Lu, Ruimin Wang, Shaowei An, Cong Xie, and Changbin Yu. 2022. “StackZDPD: A Novel Encoding Scheme for Mass Spectrometry Data Optimized for Speed and Compression Ratio.” *Scientific Reports* 12 (1): 5384. <https://doi.org/10.1038/s41598-022-09432-1>.

Wickham, Hadley. 2014. “Tidy Data.” *Journal of Statistical Software* 59 (10). <https://doi.org/10.18637/jss.v059.i10>.

Widner, Brittany, Melissa C. Kido Soule, Frank Xavier Ferrer-González, Mary Ann Moran, and Elizabeth B. Kujawinski. 2021. “Quantification of Amine- and Alcohol-Containing Metabolites in Saline Samples Using Pre-extraction Benzoyl Chloride Derivatization and Ultrahigh Performance Liquid Chromatography Tandem Mass Spectrometry (UHPLC MS/MS).” *Analytical Chemistry* 93 (11): 4809–17. <https://doi.org/10.1021/acs.analchem.0c03769>.

Wilhelm, Mathias, Marc Kirchner, Judith A. J. Steen, and Hanno Steen. 2012. “Mz5: Space- and Time-efficient Storage of Mass Spectrometry Data Sets.” *Molecular & Cellular Proteomics* 11 (1): O111.011379. <https://doi.org/10.1074/mcp.O111.011379>.

Wilson, Samuel T., Nicholas J. Hawco, E. Virginia Armbrust, Benedetto Barone, Karin M. Björkman, Angela K. Boysen, Macarena Burgos, et al. 2019. “Kīlauea Lava Fuels Phytoplankton Bloom in the North Pacific Ocean.” *Science* 365 (6457): 1040–44. <https://doi.org/10.1126/science.aax4767>.

Yancey, Paul H. 2005. “Organic Osmolytes as Compatible, Metabolic and Counteracting Cytoprotectants in High Osmolarity and Other Stresses.” *Journal of Experimental Biology* 208 (15): 2819–30. <https://doi.org/10.1242/jeb.01730>.

Yancey, Paul H., Mary E. Clark, Steven C. Hand, R. David Bowlus, and George N. Somero. 1982. “Living with Water Stress: Evolution of Osmolyte Systems.” *Science* 217 (4566): 1214–22. <https://doi.org/10.1126/science.7112124>.

Yang, Runmin, Jingjing Ma, Shu Zhang, Yu Zheng, Lusheng Wang, and Daming Zhu. 2022. “mzMD: Visualization-Oriented MS Data Storage and Retrieval.” Edited by Hanchuan Peng. *Bioinformatics* 38 (8): 2333–40. <https://doi.org/10.1093/bioinformatics/btac098>.

# Appendix 1: Speedy Quality Assurance via Lasso Labeling for Untargeted Mass-Spectrometry Data

1. This chapter was published as Kumler, W. and Ingalls, A.E. 2022. “Tidy Data Neatly Resolves Mass-Spectrometry’s Ragged Arrays.” *R Journal* 14 (3): 193-202. https://doi.org/10.32614/RJ-2022-050 [↑](#footnote-ref-29)