Mathematical Biosciences and Engineering

http://www.aimspress.com/journal/mbe

MBE, 20(9): 17197–17219.
DOI: 10.3934/mbe.2023766
Received: 11 May 2023
Revised: 06 August 2023
Accepted: 28 August 2023
Published: 04 September 2023

Research article

Column storage enables edge computation of biological big data on 5G networks

Miaoshan Lu[1,2,3,4,†], Junjie Tong[4,†], Weidong Fang[5], Jinyin Wang[1], Shaowei An[6], Ruimin Wang[6], Hengxuan Jiang[4] and Changbin Yu[4,*]

[1] Zhejiang University, Hangzhou 310009, Zhejiang Province, China
[2] School of Engineering, Westlake University, Hangzhou, China
[3] Institute of Advanced Technology, Westlake Institute for Advanced Study, Hangzhou, China
[4] Shandong First Medical University & Shandong Academy of Medical Sciences, Jinan, China
[5] Shanghai Institute of Microsystem and Information Technology, Chinese Academy of Sciences, Shanghai, China
[6] Fudan University, Shanghai, China

[†] These authors contributed equally to this work.

[*] **Correspondence:** Email: yu_lab@sdfmu.edu.cn.

Abstract: With the continuous improvement of biological detection technology, the scale of biological data is also increasing, which overloads the central-computing server. The use of edge computing in 5G networks can provide higher processing performance for large biological data analysis, reduce bandwidth consumption and improve data security. Appropriate data compression and reading strategy becomes the key technology to implement edge computing. We introduce the column storage strategy into mass spectrum data so that part of the analysis scenario can be completed by edge computing. Data produced by mass spectrometry is a typical biological big data based. A blood sample analysed by mass spectrometry can produce a 10 gigabytes digital file. By introducing the column storage strategy and combining the related prior knowledge of mass spectrometry, the structure of the mass spectrum data is reorganized, and the result file is effectively compressed. Data can be processed immediately near the scientific instrument, reducing the bandwidth requirements and the pressure of the central server. Here, we present Aird-Slice, a mass spectrum data format using the column storage strategy. Aird-Slice reduces volume by 48% compared to vendor files and speeds up the critical computational step of ion chromatography extraction by an average of 116 times over the test dataset. Aird-Slice provides the ability to analyze biological data using an edge computing architecture on 5G networks.

Keywords: column storage; edge computing; mass spectrum; aird-slice; data compression

1. Introduction

In the life sciences, huge amounts of biological data are produced every day. These data are generated by advanced scientific instruments, which are huge and complex to analyze. High-performance central servers are often used to analyze this data. When biometric data becomes too large, centralized servers are often overwhelmed. Network transmission speed, data processing speed, data privacy protection and other issues will be amplified. The privacy of biological data has always been an important issue [1, 2]. Large-scale biological data generated by scientific instruments can be partially processed locally rather than all transferred to a central server. This can effectively enhance the speed of data analysis and ensure data privacy and security. The edge computing architecture is used to migrate some of the complex computing work to the terminal computer. This is an important solution to optimize the architecture of biological big data computing [3].

However, the detection of biological data usually involves converting macroscopic samples into digital samples. Scientific instruments record the detected microscopic objects in digital form. Thus, even a single sample of biological data can be enormous. In this case, the data storage strategy, compression method and information reading strategy will greatly affect the scope of edge computing. Column storage and edge computing are both long-standing technical ideas [4–8]. Compared with cloud computing, edge computing can provide services with faster reaction times and higher quality. By reducing IO overhead and improving data read efficiency, column storage can reduce CPU overhead. As a result, traditional small computing devices can perform efficient calculations as well. However, column storage is not necessarily suitable for all biological data. The introduction of column storage technology also requires a deep understanding of the prior characteristics of the biological data. This makes it much more difficult to transform biological data with column storage.

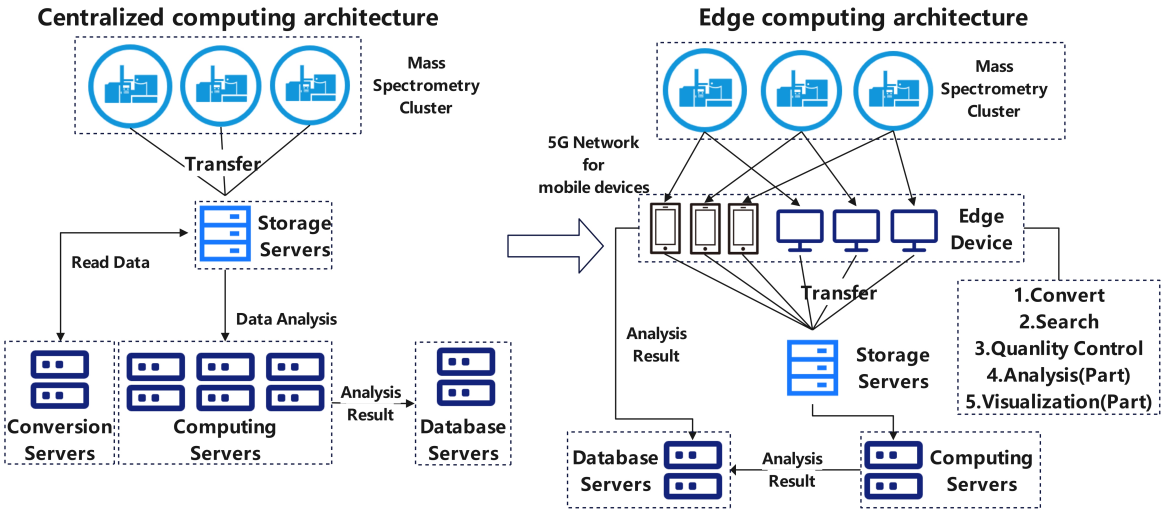

Figure 1. From a central computing architecture to an edge computing architecture. Mass Spectrometry data using column storage can be analyzed on a small device. This can effectively reduce the computing pressure of the central server. Furthermore, due to the physical distance between the edge computing device and the mass spectrometer, the data transmission and calculation costs are lower.

Taking mass spectrometry (MS) data as an example, we describe how column storage is introduced into MS data. Thus, the analysis of MS data can evolve away from the central calculation, and make it work properly on small devices. The devices used in edge computing can be connected to mobile computing devices via 5G networks in addition to conventional small computing devices. This also enables edge computing frameworks based on MS data (see Figure 1). Mass spectrometry is a widely recognized technology that has found applications in different fields of applied sciences. MS-based proteomics and metabolomics are two techniques that have emerged as critical tools in frontier science. Despite its potential, the challenge of unraveling critical information from the massive amount of data generated by MS remains a significant hurdle. As a result, researchers have invested considerable effort in optimizing both MS data formats and data processing platforms to uncover the key findings.

There are numerous MS data formats produced by various MS manufacturers. Nevertheless, these formats are typically proprietary and cannot be directly accessed. The software that lands with the mass spectrometer must be required to read and analyze the data. With the growing prevalence of open data formats, more vendors are offering proprietary software development kits (SDK) to enable researchers to read their data content [9]. However, developing general-purpose MS data-processing software that uses vendor-formatted data can be time-consuming due to the lack of standardization among vendors [10]. In addition, as new formats evolve, new software will cease to support older formats and these obsolete data formats will become "rotten data" [11, 12]. Therefore, the development of a standardized format for the MS files has become an industry consensus. Since 2003, the Human Proteome Organization-Proteomics Standards Initiative (HUPO-PSI) has proposed mzData format to help researchers better preserve and read MS files in order to establish a unified, standardized structure of MS files [13]. The mzData format aims to consolidate existing open formats used to store peak list information, such as PKL, DTA, MGF, etc. and aims to become a data standard for manufacturers of MS instruments. Due to the high design flexibility of mzData, the overarching framework must be continuously updated to accommodate new scenarios, making it difficult for the format to be compatible with older ones. During the same period, the institute for system biology (ISB) proposed the mzXML format, which prefers a more strict structure and is less flexible [14]. Both formats use different data coding principles to write the same information. However, the coexistence of the two formats is considered a distraction for software developers and a source of confusion for users. To solve this problem, PSI and ISB created a new data format called mzML in 2009 [15]. mzML incorporates the popular features of mzData and mzXML, making it one of the most popular mass spectrometry data formats available today.

With the advancement of MS-based omics technology and MS instruments, the size and number of MS files have been significantly increasing. A total of 34,233 datasets were deposited in the open repository ProteomeXChange in 2022. This number has more than doubled compared to three years ago when it was 14,169 and continues to rise each month [16, 17]. In addition to proteomics, metabolomics data repositories like MetaboLights are also expanding annually and their growth shows no sign of slowing down [18, 19]. With the extensive use of MS technology in clinical research, the number of high-quality MS files will certainly increase.

Along with the growth in MS file numbers, the volume of MS files has also increased significantly due to improvements in MS instrument detection accuracy and acquisition methods. It has proven fairly challenging to preserve and interpret this MS data effectively. The issue of data standardization and scenario compatibility for mass spectrometer instruments, platforms and acquisition methods is resolved by mzML. However, the efficacy of mzML in storage or computing scenarios is inadequate, severely limiting

the computational conditions necessary for MS data analysis. Researchers have an urgent need for high compression rate and high reading performance data format. In 2012, Mathias et al. proposed the lossless compression format called mz5 [20], which uses HDF5 [21] as a new container to store information in mzML. Due to the built-in compression capability of the HDF5, the file size of mz5 is significantly smaller than the mzML. In 2014, Teleman et al. proposed MSNumpress [22], a data compression algorithm used in mzML. MSNumpress includes both lossy and lossless compression for MS data, which can effectively reduce the mzML file size. In 2015, David et al. proposed the format mzDB [23], which uses SQLite [24] as a storage container, a cross-platform storage framework. mzDB provides three different data acquisition strategies for DDA and DIA acquisition to further enhance the data reading speed, resulting in improved read performance advantage in compute-oriented scenarios. In 2019, Yang et al. proposed a lossless compression algorithm for MS data called MassComp [25]. MassComp uses a special algorithm to compress the m/z and intensity data, which improves the compression ratio. Toffee [26], a lossless compression format proposed by Brett et al. in 2020, also uses HDF5 as a storage container. Besides, Toffee uses a specific algorithm targeting for time-of-flight (TOF) mass spectrometer to achieve a high compression rate. The mzMLb format [27] proposed by Ranjeet et al. in 2021 also uses HDF5 as a storage container. Unlike the mz5 format, mzMLb uses a mixed storage mode, which keeps the metadata in the full PSI standard mzML format and save the numerical data in HDF5. This allows mzMLb to maintain the integrity of the mzML format while still having the container benefits of HDF5. In the same year, Felix presented a compression algorithm called mspack [28], which utilizes the similarity of adjacent mass spectra and bucket sorting algorithm to compress mass spectra directly. The mspack algorithm achieves an extremely high compression ratio. However, mspack does not support random spectrum reading, which makes it more appropriate for archive-oriented scenarios. In 2022, Lu et al. reported the Aird format. Aird is a compression format created specifically for a computing scenario (Aird-ZDPD) [29] and a storage scenario (Aird-StackZDPD) [30]. Aird improves compression rates while simultaneously enhancing decompression rates and random access speed. Aird offers extraordinarily high reading performance for single spectrum or spectra blocks by creating unique indices for distinct MS acquisition methods. In contrast to other formats, Aird provides a generic combined compression infrastructure called ComboComp [31] by combining an integer compressor with a generic compressor. Among all the extended formats of Mzmine3 [32], its file load time is much higher than other formats.

In terms of application scenarios, these MS data formats can be divided into three major categories:

- **Standardization-oriented formats**, including mzData, mzXML, mzML and mzMLb. They are designed to store the complete metadata, compress the spectra one by one and support random spectrum reading. The interoperability of read and write capabilities across platforms and the completeness of the stored data are typically greater concerns in standard formats.
- **Archive-oriented formats**, including mspack and Aird-StackZDPD. They prioritize extreme compression over random spectrum reading. These formats usually do not store spectrum sheet by sheet, but further leverage similarities between spectra for high-efficacy storage.
- **Computation-oriented formats**, including Aird-ZDPD and mzDB. Similar to the standardization oriented format, they also compress the spectrum one by one and support random spectrum reading. In addition, they build different indexes based on different acquisition methods expedite the random reading of spectra or spectra blocks. The speed of the following analysis of the data corresponding to the acquisition method can be significantly accelerated owing to this distinctive index design.

It can be observed that the format development of mass spectrometry started to shift from standardized storage to vertical domain optimization gradually, which is tightly linked to the analytical method of the mass spectrometry file.

Here, we present Aird-Slice. Aird-slice is a column compression format based on the Aird [29], which adopts the benefits of combination compressors in the Aird format. Aird-Slice is specifically designed for the scenarios where researchers need to quickly review a few *m/z* from hundreds of MS files within seconds. When confronted with tens of thousands of MS files in public data repositories, the repository can manage to organize these files based on their specific tags, but researchers cannot quickly access the extracted ion chromatogram (XIC) result of a molecule in large batch of raw files. Although some mass spectrometry data warehouses do XIC pre-extraction of incoming mass spectrometry data, it can be quickly displayed during subsequent searches for an *m/z*. However, this kind of pre-extraction is not accurate due to the limitations of the XIC algorithm, and often misses some key and effective points. Aird-Slice, on the other hand, retains all data points completely and is a real-time computation XIC algorithm. XIC is the key procedure for mass spectrometry and is usually the initial step in MS data processing. In order to create an XIC for a specific *m/z*, each spectrum must be decompressed. However, only a part of the data is really used in this process, which wastes the efficiency of the disk reads and negatively affects performance while looking for the correct *m/z* range. This manuscript employs column compression by storing every point of every target *m/z* within the entire MS file, thereby changing the composition of information in the MS file from retention time (RT)-(*m/z* array, intensity array) mode to *m/z*-(rt array, intensity array) mode. This transformation applied in Aird-Slice simplifies the XIC construction process in the presence of *m/z* and increases data reading efficacy from less than one thousandth to one hundred percent. Furthermore, we compress each column using the combined compressor provided by the AirdPro [29], again achieving an acceptable compression ratio. Therefore, the column compression format with *m/z* as storage unit is called Aird-Slice. The introduction of Aird-Slice reduces the maximum memory requirement for MS data analysis and greatly improves disk reading efficiency. It solves the limitation that a high throughput MS file must be analyzed using a high performance and large-memory server, which brings a brand-new experience for MS data analysis in proteomics or metabolomics. There is great potential for building mz-oriented search engines in data warehouse-based scenarios as well as in streaming computing-based scenarios. High-throughput MS data processing can even be moved to small mobile devices due to the drastically decreased CPU and memory requirements. Small IoT devices can even complete independent computing in the streaming computing scenario, which is crucial for applications like real-time quality control, target integration correction and target aggregation search.

2. Materials and methods

2.1. Core computing step

Calculating the XIC results of target molecules is the most common analytical procedure to obtain quantitative and qualitative results. The mass spectrum contains an array of key-value pairs of *mz* and intensity values, which represents the ions and corresponding abundance information when the mass spectrum is produced. XIC, on the other hand, is the process of searching for the abundance value of a given ion from each mass spectrum to form a key-value pair of RT and intensity values. Because of the difference in the resolution of the mass spectrometers and the distribution of the natural isotopes, we

need to approximate the cumulative intensity value using a very small *mz* window when conducting the XIC result. Figure 2 is a typical XIC process and illustrates the procedure for deriving the cumulative intensity of a particular *mz* from a single mass spectrum. It is evident that two ions with a difference in *m/z* of just 0.01 can have drastically different intensity sums.

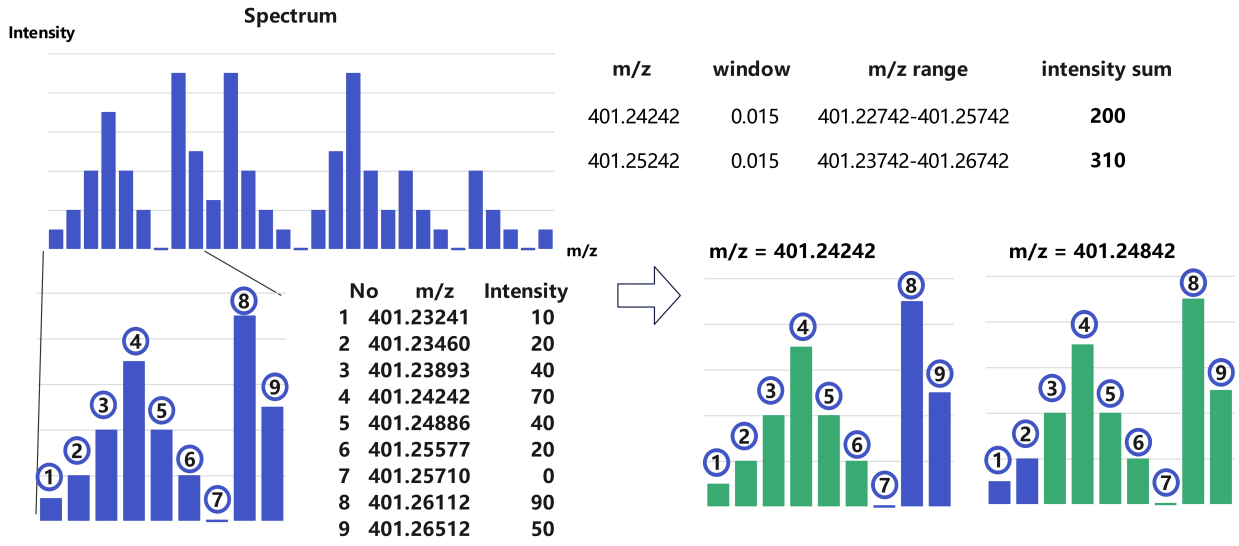

Figure 2. The schematic representation of the XIC procedure. XIC process for *m/z* of 401.24242 and 401.25242 are shown respectively. We use the *m/z* tolerance of 0.015 for calculating the intensities sum. Then, the search range of the two *m/z* becomes (401.22742, 401.25742] and (401.23742, 401.26742]. The four boundaries are searched by binary search algorithm, and the final search result is determined to be between column-1 to column-9 as shown in the figure.

As a result, we discovered that in order to get the cumulative intensity value for a single spectrum, we must first perform a direct summing or weighted summation using the binary search algorithm to perform boundary search four times. A simple summing operation consists of four steps:

- **Step1.** Read spectra data from the disk;
- **Step2.** Decompress the spectra data;
- **Step3.** Binary search to confirm summation boundary on every spectrum;
- **Step4.** Sum the intensity values in the boundary for every spectrum.

Obviously, Step1 and Step2 are the most time-consuming steps. If one MS file needs to be analyzed comprehensively, full file reading and decompressing are unavoidable. This makes searching for a single *m/z* particularly inefficient.

2.2. Column storage transformation

The majority of current MS data formats compress and store mass spectra on a sheet-by-sheet manner. If we were only interested in a handful of *m/z* values, we would have to read the entire file into memory. Nonetheless, almost all the data read in is irrelevant for these target *m/z* points and incurs additional

computational costs in Step3. One mass spectrum usually contains 5,000–50,000 different *m/z*-intensity pairs. One MS file usually contains 7,000–70,000 mass spectra, while the actual number of *m/z* points used to sum the intensity values is typically no more than 10.

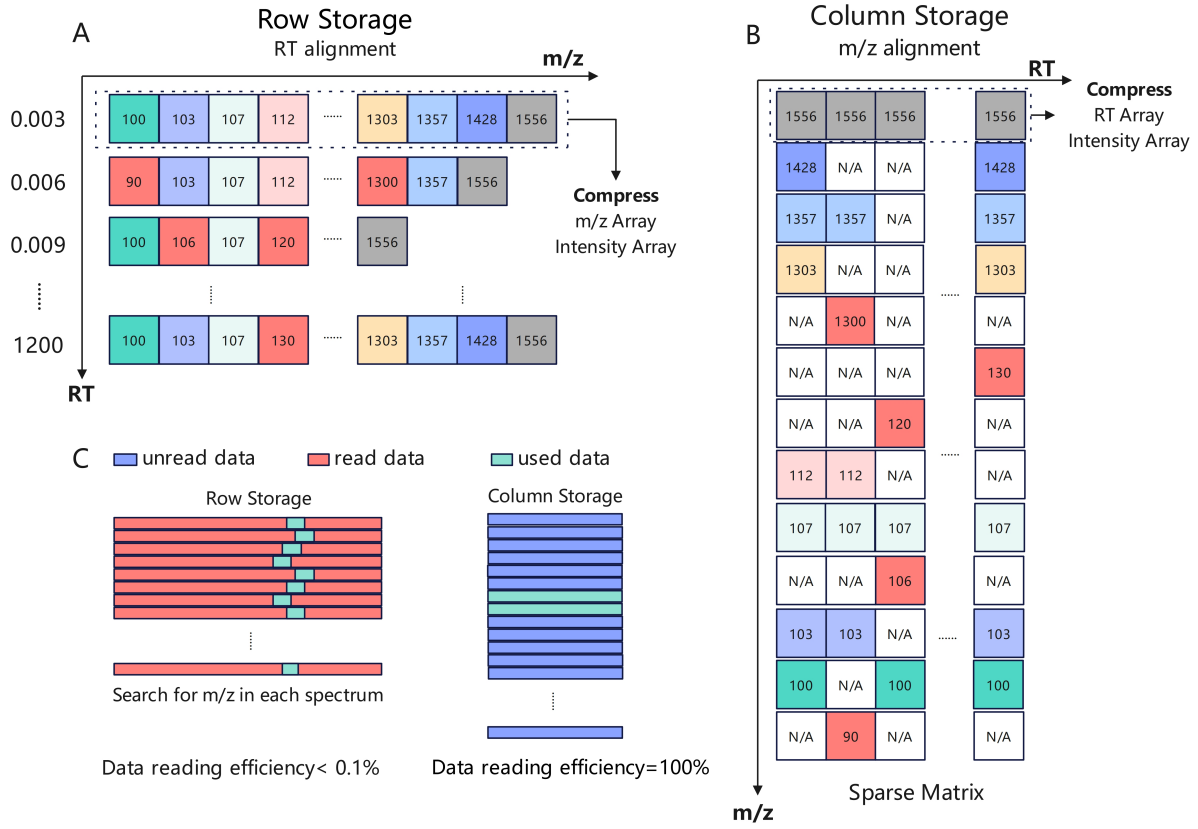

Figure 3. (A) The storage model in the traditional MS data format. Spectrum is compressed and stored sheet by sheet. (B) The storage model of Aird-Slice, using column storage compression, *m/z* is the basic organizational unit. (C) Advantages of column storage for XIC procedures. The data reading efficiency of traditional storage structure is less than one thousandth, while the data reading efficiency of column storage is 100%.

Figure 3(A) is the traditional MS file storage structure. This can be recognized as a rt-alignment structure. Each row represents a spectrum arranged inline with the scanning time of the mass spectrometer. Since the length of each mass spectrum is not the same, the data structure formed is a collection of multiple unequal length arrays. If the mass spectrum file contains n spectra, then all key-value pairs in an MS file can be expressed as

$$P_n = \sum_{i=1}^{n} \sum_{j=1}^{L_i} p_{ij} \tag{2.1}$$

where L_i is the length of the $spectrum_i$. Since each *m/z* involves two boundaries when searching, the

time cost of binary searching in Step3 on this MS file is:

$$O(P_n) = 2 \times \sum_{i=1}^{n} O(log_2 L_i) = 2 \times O(log_2 \prod_{i=1}^{n} L_i) \tag{2.2}$$

Figure 3(B) is the Aird-Slice storage structure. Aird-Slice uses column compression to store MS data. The first step is to calculate all the different *m/z* in the entire file. Since the different spectra usually contain a large number of duplicate *m/z*, the final de-duplicated *m/z* array(here defined as $T_{m/z}$) is usually much smaller than *m/z* points in P_n. Due to the fact that *m/z* typically does not contain a response intensity value in every RT, we populate the positions where the intensity is absent with a value of zero to obtain a matrix. The height of the transformed matrix (here defined as H) is the array length of $T_{m/z}$, while the width of the matrix (here defined as n) is the total number of spectra in the MS file. The matrix here is defined as $T_{H\times n}$. Each column represents the XIC information of *m/z* in the $T_{m/z}$. All the points can be expressed as:

$$T_{H\times n} = \sum_{i=1}^{H} \sum_{j=1}^{n} p_{ij} \tag{2.3}$$

From the definition of $T_{m/z}$, $\forall i \leq n$ in P_n (see Eq (2.1)), $L_i \ll H$. Therefore, the converted matrix is a typical sparse matrix. Figure 3(C) shows a comparison of the difference in data reading efficiency for the XIC process between the two different storage structures. When the specified *m/z* is analyzed, the majority of read data is left unused when using the RT-alignment mode. However, under the *m/z*-alignment mode, all data read is used in the final calculation. In *m/z*-alignment mode, the binary search algorithm only needs to be executed twice. The procedure used to calculate the XIC is changed to the following steps:

- **Step1.** Read $T_{m/z}$ data from the disk;
- **Step2.** Decompress the $T_{m/z}$ data;
- **Step3.** Binary search to confirm summation boundary on $T_{m/z}$;
- **Step4.** Read and decompress all specified *m/z* columns by the calculated boundary of Step3;
- **Step5.** Add up all the columns.

Therefore the time complexity used to determine the *m/z* search range is:

$$O(T_{H\times n}) = 2 \times O(log_2 H) \tag{2.4}$$

Comparing Eqs (2.2) and (2.4), it is clear that Eq (2.4) is almost a constant value. Equation (2.2), on the other hand, varies according to the total number of mass spectra and the length of every mass spectra. Here is an example to compare the time complexity of the two modes. Suppose there is an MS file that contains 10000 spectra, and each spectrum has 5000 *m/z* values. Since the detection *m/z* in proteomics or metabolomics data is generally in the range of 0–3000, the maximum number of points after 5 decimal places to be precise is 300 million (actually much lower than this number). Here, the theoretical maximum is substituted. Then the Eq (2.2) can be expressed as $O(2 \times 10000 \times log_2 5000) = O(242754)$. The Eq (2.4) can be expressed as $O(2 \times log_2 300000000) = O(56)$. The difference in time complexity between the two modes are more than 4000 times.

In terms of data reading efficiency, RT-alignment mode needs to read and decompress all the data file, whereas *m/z*-alignment mode reads only specified *m/z* columns. Therefore, no matter how the original

file volume changes, the number of *m/z* columns that need to be read only depends on the *m/z* tolerance. The amount of data to be read is usually only a few kilobytes (KB). It is also a thousand times more complex in time compared to a comprehensive read of a file in RT-alignment mode.

In general, XIC calculation using *m/z*-alignment is thousands of times faster than RT-alignment for a small group of *m/z*. Furthermore, columnar storage enables streaming analysis of mass spectrum data.

2.3. Compression method

After *m/z*-alignment transformation, the original MS data is transformed into a sparse matrix. Column compression is plainly more suited as Aird-Slice is aimed for column slicing. Based on the sparse feature of the matrix, Aird-Slice only store the non-zero points in the matrix. Each non-zero point can be expressed as (column number,intensity) pairs. Storing data for each *m/z* column is actually storing an array of column numbers array and intensity array.

In addition to saving these non-zero points, Aird-Slice must also save the actual mapping value of the matrix. The width of the matrix corresponds to the $T_{m/z}$ array, with the height refers to the RT array. Aird-Slice compresses these four types of arrays using the combined compressor framework from Aird [29].

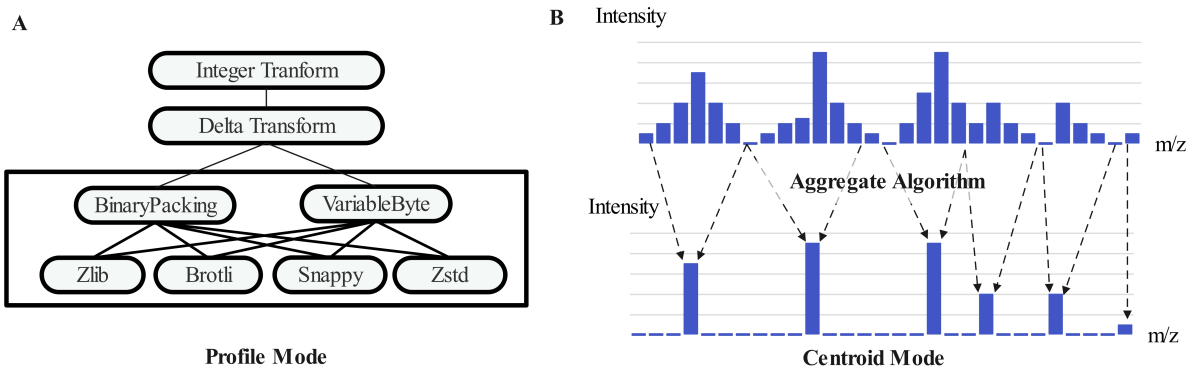

Figure 4. (A) The principle of the combined compression framework provided by Aird. (B) The process of converting the Profile mode mass spectrometry to centroid mode.

Figure 4(A) is the basic structure of the combined framework. The framework offers two integer-purpose compressors (BinaryPacking and Variable Bytes) [33] and four general-purpose compressor(Zlib, Zstd [34], Snappy [35] and Brotli [36]). The two arrays must first undergo a micro loss conversion before being compressed, which converts the array from a double floating-point type to an integer type. The *m/z* array is converted with a reasonable absolute error of 10^{-5} (5 decimal places, 5dp) [37], while the absolute error of the RT conversion is 10^{-3}. Aird-Slice offers three precision modes of 3, 4 and 5dp for different scenarios.

2.4. Data centroid processing

Spectrum organization can be divided into two modes: Profile mode and Centroid mode. Centroid mode is a result that appears after the profiled spectrum has been algorithmically processed. Figure 4(B)

is the diagram of data centroid algorithm. The centroided data is obviously getting more succinct. Although it may cause some possible valuable signal loss, it can greatly reduce the cost of data storage, thus improving the efficiency of subsequent analysis.

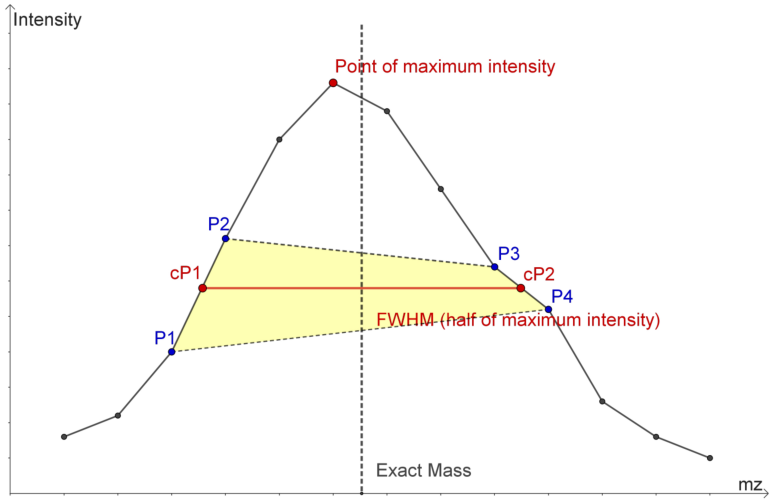

Figure 5. Diagram of exact-mass algorithm used by Aird-Slice.

Aird-Slice provides a common data centroid algorithm called exact-mass, which is also used in Mzmine3 [32]. For profile MS data, the exact-mass algorithm is highly recommended. See Figure 5. This algorithm calculates the exact mass of a peak using the FWHM (full width at half maximum) concept. First, the algorithm locates four data points at half the highest intensity that are closest to the peak center. The four points can be expressed as $(P_i(x_i, y_i), i = 1, 2, 3, 4)$. The $l_{P_iP_j}$ can be written as determinant:

$$\begin{vmatrix} y - y_i & y_j - y_i \\ x - x_i & x_j - x_i \end{vmatrix} = 0 \tag{2.5}$$

The cross points of $y = halfmax(hf)$ with $l_{P_1P_2}$ and $l_{P_3P_4}$ are $cP_1(x_{c1}, hf)$ and $cP_2(x_{c2}, hf)$, which define the width of the peak. The values of x_{c1} and x_{c2} are deduced from the following equation:

$$\begin{vmatrix} hf - y_1 & y_2 - y_1 \\ x_{c1} - x_1 & x_2 - x_1 \end{vmatrix} = 0 \tag{2.6}$$

$$\begin{vmatrix} hf - y_3 & y_4 - y_3 \\ x_{c2} - x_3 & x_4 - x_3 \end{vmatrix} = 0 \tag{2.7}$$

The exact mass is then obtained as the center of the width, and we get:

$$ExactMass = \frac{X_{c2} - X_{c1}}{2} \tag{2.8}$$

The exact mass algorithm is an optional algorithm for format conversion.

2.5. Storage strategy for DDA and DIA

There are two main acquisition methods for high-throughput mass spectrometry: Data dependent acquisition (DDA) and data independent acquisition (DIA) In either acquisition method, the step of

constructing XIC result is necessary. However, in the process of data analyzing for DDA, spectra similarity comparison is another important step for ion identification. Such spectra consist of specific *m/z* fragmentation and are labeled as secondary spectrum (MS2). The spectra similarity comparison algorithm needs to read the entire mass spectrum. It is obvious that spectrum-centered row storage is more suitable for this algorithm step. Figure 6(A) shows the spectra structure of MS1 and MS2 in the DDA file. Each MS1 contains a number of *m/z* and intensity key-value pair signals. In the DDA method, the signal in MS1 is screened according to certain conditions to determine which ions need to construct the MS2. For example, ions with top 20 intensity values were selected to generate MS2. Due to the randomness of this selection, the MS2 collected by DDA is not analyzed by constructing XIC, but is used as an MS "fingerprint" for ion identification. Since MS1 requires calculation of XIC for subsequent analysis and MS2 requires similarity comparison algorithm, Aird-Slice uses a mixed storage mode to achieve the best read performance. For the DIA acquisition method, since both MS1 and MS2 in DIA files use XIC as the basis step for subsequent algorithms. Therefore, column storage can be fully used for DIA file conversion.

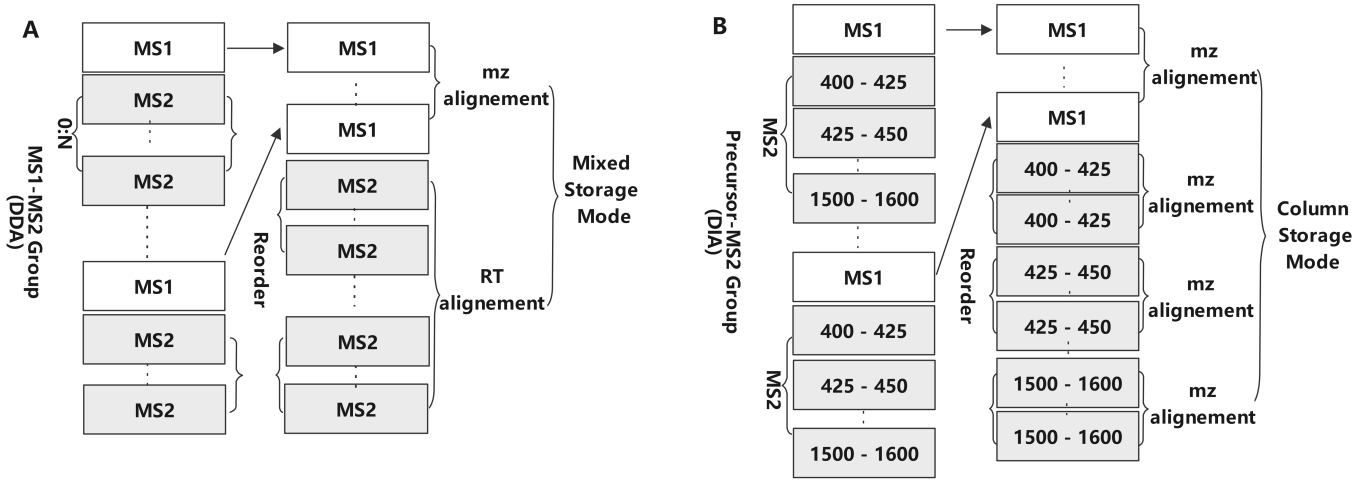

Figure 6. (A) The DDA spectra structure. Since the MS2 spectra are only used for similarity comparison, Therefore, hybrid compression mode is used for DDA storage. (B) The DIA spectra structure. MS2 spectra in DIA are also used in XIC building. However, because each MS2 block's search space is independent of the others. As a result, the column storage strategy is more appropriate for the MS1 block and each MS2 blocks.

2.6. Index building and compression

With column storage, some of the metadata described with controlled vocabulary (CV) is meaningless for column information. To ensure the integrity of MS data while not interfering with Aird-Slice format reading performance, we store the original metadata information separately from the column metadata information. The newly described column storage metadata is stored as a JSON file with the extension cjson. The index mostly records the global RT array, the global unique *m/z* array and the spectrum number array and intensity array corresponding to each column of *m/z*. The key fields are described in Table 1.

Table 1. Description of Aird-Slice column index.

Field Name	Type	Description
level	Integer	Label the spectra type from which the *m/z* data came
ptr	Long	Position of the total column stored data in Aird
range	Array	Range of the target *m/z*
mzListPtr	Long	Position of the compressed unique *m/z* array in Aird
rtListPtr	Long	Position of the compressed global RT array in Aird
specrtaIdListPtr	Long	Position of the compressed offset array for spectra ids
intensityListPtr	Long	Position of the compressed offset array for intensities

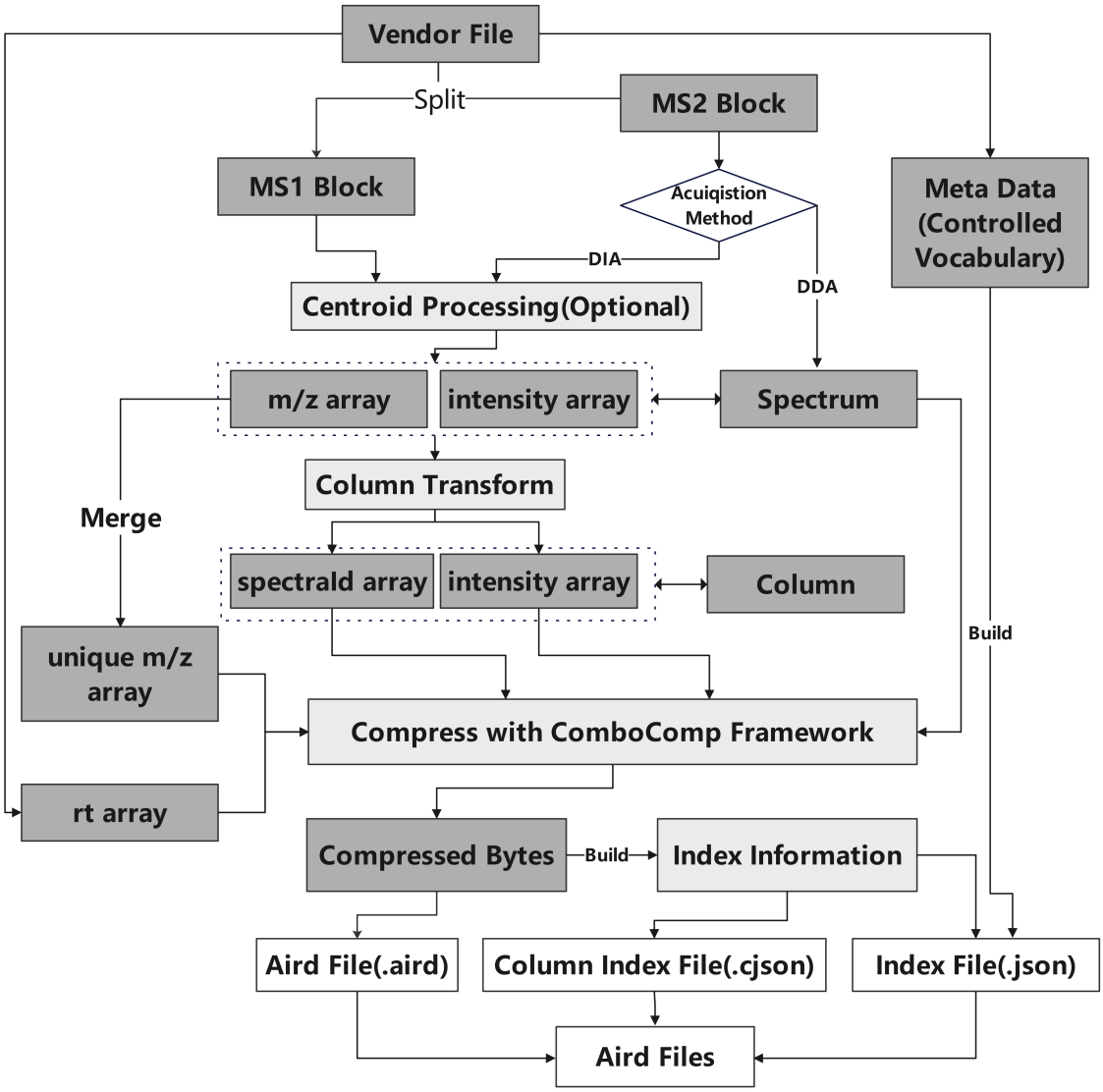

Figure 7. The main conversion workflow for Aird-Slice. DDA and DIA modes are both supported by Aird-Slice at present. Aird-Slice sets up cjson files to define column information in addition to the fundamental description data based on the CV structure.

The range of the unique *m/z* array is ordered and usually between 0 and 3000. As a result, the difference between neighbors is minimal. Due to the mass spectrometer's extraordinarily high and constant scanning frequency, the global RT array is ordered and the distinction between RTs is also relatively minor. Using the ComboComp framework to compress these two arrays can achieve an excellent compression ratio. We only store the volume of each spectraId and intensity compressed array rather than their starting and ending positions. In order to minimize disk reading costs for index files, Aird-Slice compress the offsets arrays as well.

Figure 7 shows the whole conversion process of Aird-Slice. First, you need to separate MS1 and MS2 blocks in the MS file. Then, DDA files and DIA files are then converted according to different policies (see Section 2.5). Aird-slice finally consists of three files, Aird binary data file, metadata and spectra information description index file and column data index file.

2.7. Test datasets

To conduct a comprehensive test on the performance of Aird-slice, we selected and collected 16 raw MS files sourced from others laboratories Table 2.

The test dataset includes 11 metabolomics files obtained from Metabolights and 5 proteomics files acquired from ProteomeXchange [16]. These files are obtained from various mainstream mass spectrometer manufacturers (Thermo, SCIEX, Agilent), representing a wide range of types of instruments. Of these files, 12 are acquired by DDA, while the rest were in DIA format. For the file size, they are ranging from tens to hundreds of megabytes (MB), with 10 out of 16 files being smaller than 400MB, one file being over 700MB and five files being over 1 gigabyte (GB).

The raw files were first converted to the mzML and mzMLb formats using msConvert (version 3.0.22290 and version 3.0.20059) with parameters '–mz32 –inten32' [27, 38]. Then, we further converted raw files to the Aird-Slice format by AirdPro (version 4.4.0.0) [29]. The parameters used were Scene in search mode, m/z precision at 3, 4 and 5 decimal places, auto decision and either profile or centroid mode. Therefore, each raw file generated six Aird-Slice files, with both profile and centroid modes and encoded at different storage precisions (3, 4 and 5dp).

This manuscript compares Aird-Slice format file sizes with other data formats, and compares Aird-Slice file sizes with different precision and preprocessing conditions. Additionally, the speeds of searching through numerous files for a tiny group of *m/z* are contrasted. The comparison reading speed test was performed on a laptop. The CPU uses Intel i9-12900H, 32GB memory and 2TB SSD disks. To conduct a comprehensive test on the performance of Aird-slice, we selected and collected 16 raw MS files sourced from others laboratories Table 2.

The test dataset includes 11 metabolomics files obtained from Metabolights and 5 proteomics files acquired from ProteomeXchange [16]. These files are obtained from various mainstream mass spectrometer manufacturers (Thermo, SCIEX, Agilent), representing a wide range of types of instruments. Of these files, 12 are acquired by DDA, while the rest were in DIA format. For the file size, they are ranging from tens to hundreds of megabytes (MB), with 10 out of 16 files being smaller than 400MB, one file being over 700MB and five files being over 1 gigabyte (GB).

The raw files were first converted to the mzML and mzMLb formats using msConvert (version 3.0.22290 and version 3.0.20059) with parameters '–mz32 –inten32' [27, 38]. Then, we further converted raw files to the Aird-Slice format by AirdPro (version 4.4.0.0) [29]. The parameters used were Scene in search mode, m/z precision at 3, 4 and 5 decimal places, auto decision and either profile or centroid

mode. Therefore, each raw file generated six Aird-Slice files, with both profile and centroid modes and encoded at different storage precisions (3, 4 and 5dp).

We compare Aird-Slice format file sizes with other data formats, and compares Aird-Slice file sizes with different precision and preprocessing conditions. Additionally, the speeds of searching through numerous files for a tiny group of *m/z* are contrasted. The comparison reading speed test was performed on a laptop. The CPU uses Intel i9-12900H, 32GB memory and 2TB SSD disks.

Table 2. Vendor MS files used for test.

No.	Repository	Files	Instruments
1	MTBLS733	SA1.raw	QE HF
2	MTBLS736	SampleA_1.wiff	TripleTOF 6600
3	MTBLS1108	PestMix1_8Plasma DDA20-50.wiff	TripleTOF 6600
4	MTBLS2267	jaeger3_0001.d	6520 Series QTOF
5	MTBLS2421	1.raw	LTQ Orbitrap XL
6	MTBLS2421	02122020_QCPool_Set1_P_2.raw	QE
7	MTBLS2469	SID599_H.raw	QE
8	MTBLS4861	RP POS-QC-1.raw	QE hybrid
9	MTBLS5615	MS01.raw	QE Hybrid
10	MTBLS6402	EL01_pos.raw	QE Plus
11	MTBLS6742	2_feces_fibrosis1_positive.raw	Q Exactive plus
12	PXD034709	20220818_HFX_cy_MCF_phospho_TNF_DDA_1_120min.raw	QE HF-X
13	PXD034709	20220818_HFX_cy_MCF_phospho_TNF_DDA_2_120min.raw	QE HF-X
14	PXD034709	20220905_HFX_cy_MCF_phospho_TNF_DIA_1_120min.raw	QE HF-X
15	PXD034709	20220905_HFX_cy_MCF_phospho_TNF_DIA_3_120min.raw	QE HF-X
16	PXD034709	20220905_HFX_cy_MCF_phospho_TNF_DIA_4_120min.raw	QE HF-X

3. Results

3.1. Read speed comparison

Reading and decoding of MS data is the first and most time-consuming step of MS data analysis. Typically, XIC step will be processed after MS data decoding. Similar to the mzML format, Aird-CC is a spectrum-oriented format, but it has been substantially optimized for MS data compression and reading speed, making it 3 times faster than mzML format [29]. Aird-Slice focuses on scenarios that search and build XIC result for a single or small number of *m/z*. The speed of Aird-CC versus Aird-Slice when reading a single mz is contrasted in this section. Here, the total time for reading, decoding and XIC building is evaluated at 1, 10 and 100 *m/z*, respectively.

The test code is implemented in the Java language, and the entire Java virtual machine (JVM) is restarted for each file tested to ensure there is no caching impact. Additionally, to guarantee that the Java code is always hot loaded each time, every test is preheated with additional files outside the test dataset.

See Figure 8(A), since the spectrum-oriented format requires going through all the spectra, the majority of the consumption time is in reading and decoding the spectra, and the difference between computing 1 *m/z* or 100 *m/z* is relatively small. See Figure 8(B), when calculating a single *m/z*, the total computation time for Aird-Slice is significantly reduced. Even 10 *m/z* calculation has a significant time advantage,

which can be seen from Figure 8(C). In 16 test files, Aird-Slice increased the speed by 20–300 times, with an average increase of 116 times. The computing time in all files is less than 20 milliseconds.

Figure 8. (A) Comparison of XIC time consumption for 1,10 and 100 *m/z* using Aird-CC format. (B) Comparison of XIC time consumption for 1,10 and 100 *m/z* using Aird-Slice format. (C) Comparison of XIC time consumption for single *m/z* between Aird-Slice and Aird-CC.

3.2. File volume comparison

In this study, we converted sixteen raw files with varying vendor file sizes, ranging from 12.2 to 2242.6 Mb, mentioned in Table 2 into four types of formats, mzML, mzMLb, Aird-ComboComp (CC) and Aird-Slice by msConvert and AirdPro. Aird-CC is an Aird-based computation-oriented format, which is converted through AirdPro with Scene option in Computation and Auto Decision is selected. To evaluate the compression performance of each format, we compared the file sizes across these formats.

For the Aird-CC and Aird-Slice format, each file was stored at three storage precisions (3, 4 and 5dp) in profile mode, but Aird-Slice was further tested in centroid mode Figure 9. Compared with Aird-CC, the average file sizes of profile-mode are larger at different storage accuracies, where 3dp is 124%, 4dp is 170% and 5dp is 190% larger. However, the Aird-Slice format in centroid mode illustrated the smallest file sizes in most files, except for files 2, 3 and 4, which have similar file sizes to Aird-CC. These centroid-mode Aird-Slice showed a relative smaller file size than its corresponding file in profile mode, resulting file sizes were reduced to 36, 32 and 30% for 3, 4 and 5dp, respectively.

We further compared Aird-Slice with raw files as well as other popular open formats (Figure 10). For the mzML and mzMLb format, almost all files showed a larger file size than raw files, with the exception for three DIA proteomics data (i.e., Files 14, 15 and 16) with a similar file size to raw files. More specifically, mzML and mzMLb files were on average of 185 and 286% larger than raw files, respectively. Here, we chose the 5dp for Aird-Slice and Aird-CC, since this storage precision is the recommended setting. Unlike mzML and mzMLb, most Aird-CC and Aird-Slice file sizes were significantly smaller than raw files, except for a few files in three DDA metabolomics data (Files 2, 3 and 4) obtained from non-Thermo instruments. More specifically, Aird-Slice files generated by Thermo instrument with 5dp precision showed smaller file sizes than raw files, with an average reduction 48% for profile-mode and 83% for centroid-mode. Furthermore, Aird-Slice had a larger file size than Aird-CC at 5dp precision and in profile-mode, with an average rate of 148%. In the non-Thermo files (Files 2, 3 and 4), the file size of Aird-Slice-5dp is 3-times larger than raw files and is about 4-times larger than Aird-CC-5dp, but is similar to mzML and only half of mzMLb.

To conclude, Aird-Slice-5dp files showed a smaller file size in 15 out of 16 files than mzML and mzMLb formats, except for file4. Furthermore, Aird-Slice-5dp centroid files were significantly smaller than all other format files.

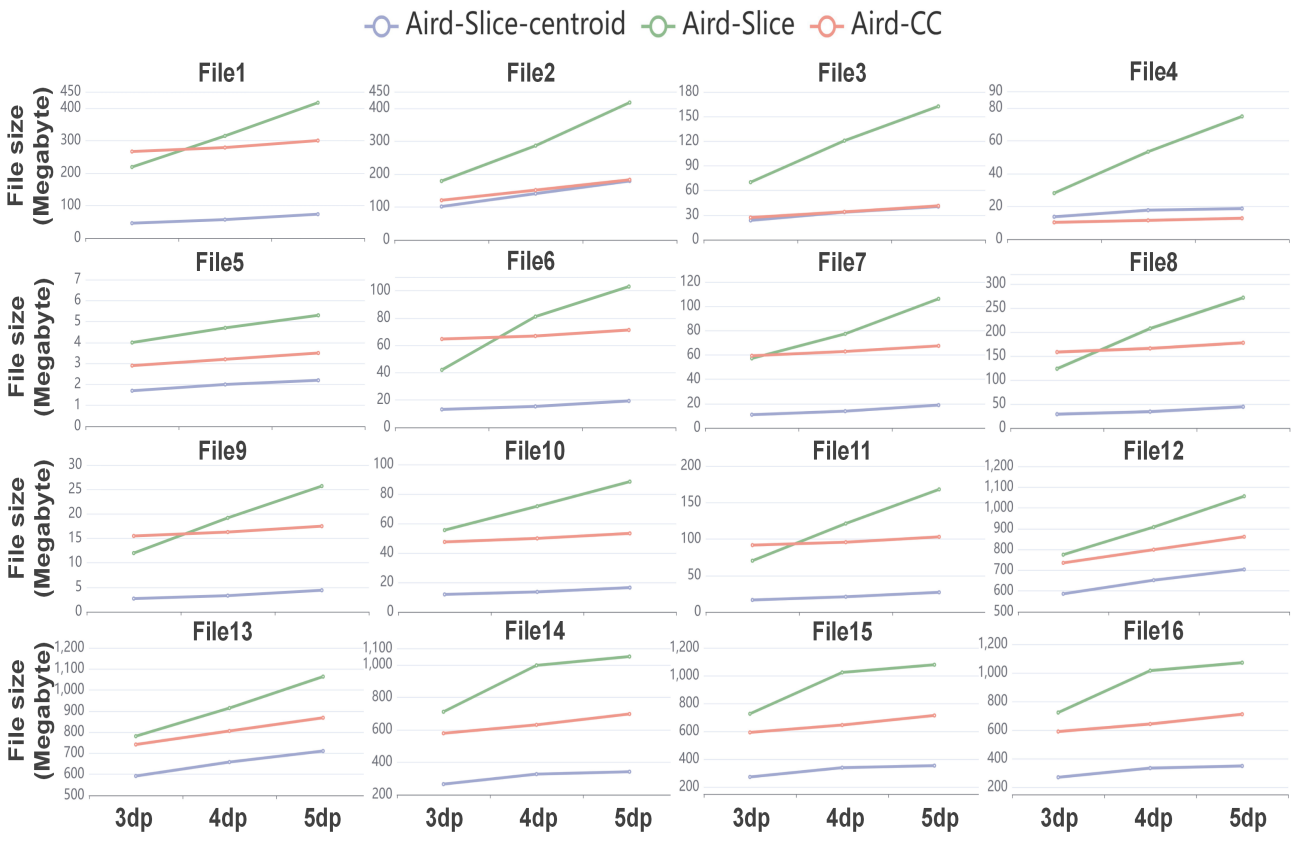

Figure 9. File Size Comparison between Aird-Slice and Aird-CC formats at different storage precisions. Aird-CC files were stored at data precision of 3, 4 and 5dp. Aird-Slice files in both profile and centroid mode were stored at precisions of 3, 4 and 5dp.

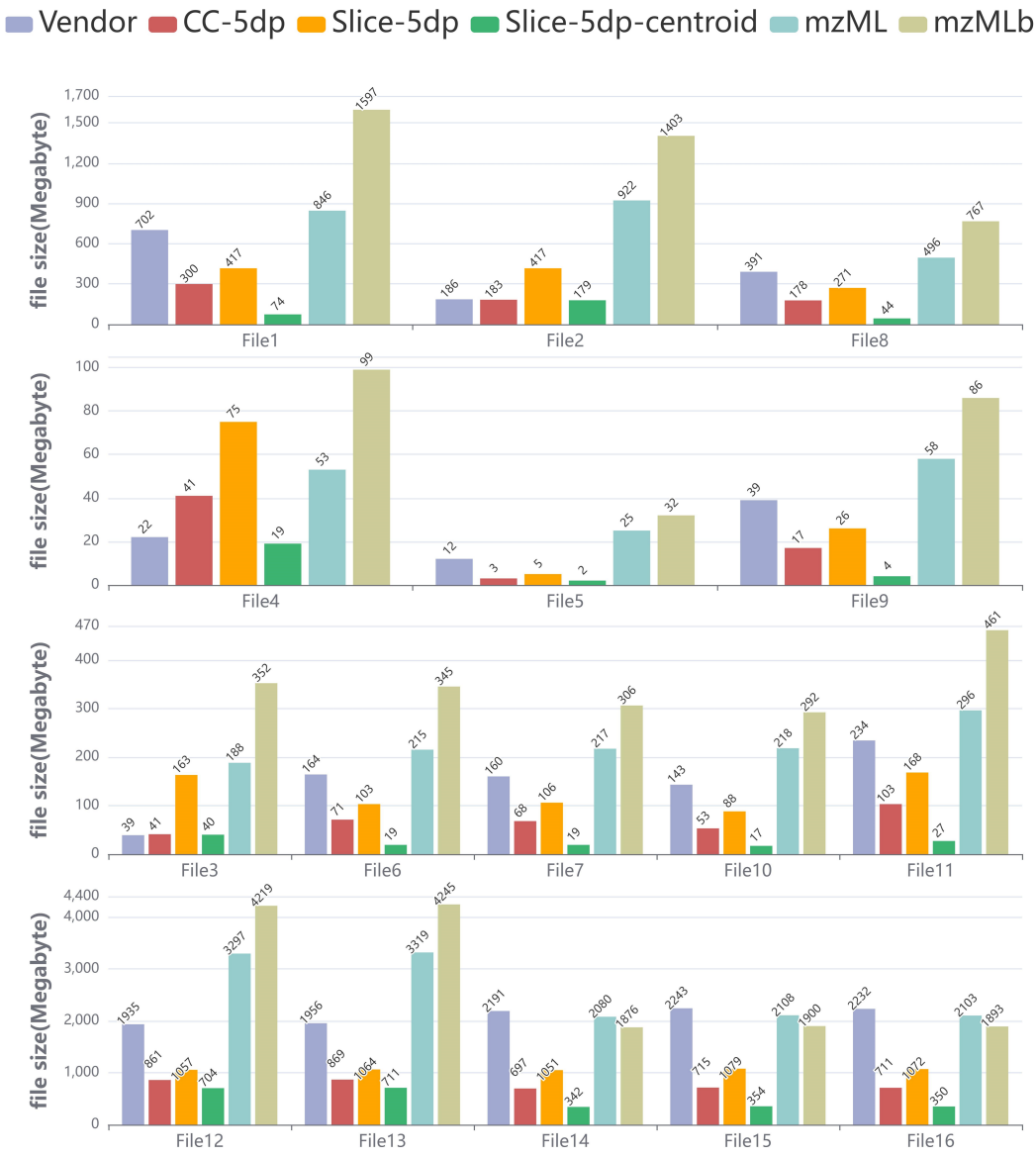

Figure 10. File Size Comparison between raw, mzML, mzMLb, Aird-Slice (Slice) and Aird-CC (CC) formats. Each raw file was converted into four different formats (Aird-CC, Aird-Slice, mzML and mzMLb). Aird-CC and Aird-Slice (profile- and centroid-mode) files with storage precision of 5dp were compared to raw, mzML and mzMLb files.

3.3. AirdPro and AirdSDK

AirdPro is a Graphical User Interface (GUI) client for converting MS vendor formats to Aird-Slice format. It can also generate conversion tasks by commands through Redis middleware. AirdPro is opensource and can be visited at https://github.com/CSi-Studio/AirdPro. AirdSDK is the SDK for reading the Aird-Slice format. Currently, AirdSDK provides Java and C# languages. The project can be visited at https://github.com/CSi-Studio/Aird-SDK.

4. Discussion

To optimize the process of MS data exchange, dissemination and analysis in the MS community, a number of open data formats has been developed. Among them, standardization-orient formats aim to store complete data and improve the capability of data read and write cross-platforms, and archive-oriented formats focus on store data with an optimized compression rate. Upon these formats, we developed Aird, a computation-oriented format with significant compression rate, to accelerate the following data processing [29]. Furthermore, we integrated Combcomp, an auto-decision compressor, into AirdPro and generated Aird-CC with optimized compression rate. However, we have noticed that researchers often only need to inspect a few *m/z* points that they are interested, requiring to read and decompress all spectra to the disk, which is a time- and computation-consuming process. Therefore, we developed Aird-Slice, a column compression format, from Aird format to fill the gap.

As a search-oriented format, Aird-Slice has not demonstrated a higher compression rate relative to Aird-CC, but it still showed a significant smaller file size than raw, mzML and mzMLb formats. More specifically, Aird-Slice in profile mode can save half (48–66% in 3–5dp storage precision) of the storage space compared to the raw files, expanding the rate to 83 to 88% in the centroid mode. Compared with the popular formats of mzML and mzMLb, Aird-Slice tends to show a higher compression rate.

In the scenario of glancing a few molecules from a whole MS dataset, Aird-Slice demonstrates the ability to quickly locate data points and construct XICs due to its high read speed. Moreover, we believed that the high compression rate could be another strong reason for researchers to choose a Aird-Slice format. Previously, MS data formats compressed spectra with RT-alignment structure and utilized the distinctive index for acceleration of data processing. Aird-Slice format leverages the transformation of data structure, from RT-alignment to mz-alignment, to improve the data reading efficiency from less than 0.1 to 100%. Moreover, we built an additional index of column storage while storing the original metadata, accelerating the mz point search and maintaining the integrity of MS data.

Although Aird-Slice is characterized by its high read speed and compression rate, some limitations continue to exist. Aird-Slice has an obvious advantage when searching a few points from MS data files, while the advantage is losing and the number of points is increasing. However, in most cases, researchers would search one point once, which maintains the strength of high read speed. The characteristic of search in Aird-Slice is suitable for deploying across Internet of Things (IoT) devices, resulting a quick visual query of a molecule.

This manuscript demonstrates that rethinking the storage technology of biological data requires a great deal of specialized knowledge, which is not a simple task. Column storage technology has been widely used in conventional database scenarios. Edge computing is also not a new concept. However, these methodologies remain difficult to put into practice in biological big data. To complete the task more effectively, assistance of scholars from multiple disciplines are required. It is hoped that such cross-learning can provide more possibilities for data computation in the field of life science.

5. Conclusions

Storing by column can significantly minimize the amount of data read from a disk while computing XIC with a single mz, enhancing search performance significantly. It also has the lowest CPU and memory consumption, which allows MS data analysis to be performed on small IoT devices. Column storage loses

its speed advantage when faced with hundreds or thousands of *m/z* as the data read steadily gets closer to reading the complete data file. Nevertheless, there are many situations in which it is sufficient to look through existing MS files just for several relevant chemical targets rather than fully analyze the entire file. Typically, this takes place during the data reanalysis stage. Aird-Slice provides a feasible solution for fast data search capability in MS data centers. When a new molecular target is hypothesized, the Aird-Slice can be used to quickly verify it in existing MS files, in addition to real-time search. Due to the fact that each column of data carries the whole signal of an *m/z*, XIC calculation can be performed directly during data transmission. This also provides technical feasibility for streaming computing.

Mass spectrometer, as a technology that converts biological samples into digital samples, can effectively preserve valuable biological information. However, current algorithms and software are difficult to use for all the signals in the MS file effectively. This means that current MS files may have new discoveries in the future. How to quickly search for the signal of the target molecule in an existing MS file has important biological significance. In addition, MS data analysis using Aird-Slice has minimal hardware requirements and can be quickly applied in IoT devices. This makes decentralized MS data analysis possible.

Compression and reading of biological big data is becoming an increasingly important topic. As the precision of scientific instruments continues to increase and the number of deployments continues to grow, centralized computing architectures for biological data are stretched thin. Column storage has great performance advantages for biological data based on time series acquisition. It enables partial queries or calculations to be performed on small devices and promotes the diversified design of network architecture in biological data analysis. However, using such methods on biological data requires a thorough comprehension of the underlying biology. It requires considerable engineering to assure compatibility and actual performance. With the use of mass spectrum data as a case study, this work describes in detail how column storage is integrated into MS data storage and promotes mass spectrum data analysis to avoid a strong reliance on centralized computing mode, making edge computing architecture conceivable. We hoped that the work can pave the way for introducing related technologies into more biological data.

Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

We thank all the technicians in the CSi Studio for their support in working on the paper at a critical time. This work is in part supported by the Shandong Provincial Natural Science Fund(2022HWYQ-081) and the Academic promotion project of SDFMU.

Conflict of interest

The authors declare there is no conflict of interest.

References

1. H. Lin, S. Garg, J. Hu, X. Wang, M. J. Piran, M. S. Hossain, Privacy-enhanced data fusion for COVID-19 applications in intelligent internet of medical things, *IEEE Int. Things J.*, **8** (2020), 15683–15693. https://doi.org/10.1109/JIOT.2020.3033129
2. W. Fang, C. Zhu, W. Zhang, Toward secure and lightweight data transmission for cloud-edge-terminal collaboration in artificial intelligence of things, *IEEE Int. Things J.*, (2023). https://doi.org/10.1109/JIOT.2023.3295438
3. W. Fang, C. Zhu, F. R. Yu, K. Wang, W. Zhang, A secure data aggregation strategy in edge computing and blockchain empowered internet of things, *IEEE Trans. Ind. Inform.*, **18** (2022), 4265-4274. https://doi.org/10.1109/TII.2021.3122370
4. D. Abadi, S. Madden, M. Ferreira, Integrating compression and execution in column-oriented database systems, in *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, (2006), 671–682. https://doi.org/10.1145/1142473.1142548
5. D. J. Abadi, Column stores for wide and sparse data, in *CIDR 2007 - 3rd Biennial Conference on Innovative Data Systems Research*, (2007), 292–297.
6. D. J. Abadi, S. R. Madden, N. Hachem, Column-stores vs. row-stores: How different are they really?, in *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, (2008), 967–980. https://doi.org/10.1145/1376616.1376712
7. K. Cao, Y. Liu, G. Meng, Q. Sun, An overview on edge computing research, *IEEE Access*, **8** (2020), 85714–85728. https://doi.org/10.1109/ACCESS.2020.2991734
8. Y. Ai, M. Peng, K. Zhang, Edge computing technologies for Internet of Things: a primer, *Digital Commun. Netw.*, **4** (2018), 77–86. https://doi.org/10.1016/j.dcan.2017.07.001
9. N. Hulstaert, J. Shofstahl, T. Sachsenberg, M. Walzer, H. Barsnes, L. Martens, et al., ThermoRawFileParser: Modular, scalable, and cross-platform RAW file conversion, *J. Proteome Res.*, **19** (2020), 537–542. https://doi.org/10.1021/acs.jproteome.9b00328
10. E. W. Deutsch, File formats commonly used in mass spectrometry proteomics, *Mol. Cell. Proteomics*, **11** (2012), 1612–1621. https://doi.org/10.1074/mcp.R112.019695
11. H. S. Wiley, G. S. Michaels, Should software hold data hostage?, *Nat. Biotechnol.*, **22** (2004), 1037–1038. https://doi.org/10.1038/nbt0804-1037
12. L. Martens, A. I. Nesvizhskii, H. Hermjakob, M. Adamski, G. S. Omenn, J. Vandekerckhove, et al., Do we want our data raw? Including binary mass spectrometry data in public proteomics data repositories, *Proteomics*, **5** (2005), 3501-3505. https://doi.org/10.1002/pmic.200401302
13. G. Mayer, L. Montecchi-Palazzi, D. Ovelleiro, A. R. Jones, P. A. Binz, E. W. Deutsch, et al., The HUPO proteomics standards initiativemass spectrometry controlled vocabulary, *Database*, **2013** (2013). https://doi.org/10.1093/database/bat009
14. P. G. A. Pedrioli, J. K. Eng, R. Hubley, M. Vogelzang, E. W. Deutsch, B. Raught, et al., A common open representation of mass spectrometry data and its application to proteomics research, *Nat. Biotechnol.*, **22** (2004), 3501-3505. https://doi.org/10.1038/nbt1031

15. L. Martens, M. Chambers, M. Sturm, D. Kessner, F. Levander, J. Shofstahl, et al., mzML—a Community Standard for Mass Spectrometry Data, *Mol. Cell. Proteomics*, **10** (2011). https://doi.org/10.1074/mcp.R110.000133

16. E. W. Deutsch, N. Bandeira, V. Sharma, Y. Perez-Riverol, J. J. Carver, D. J. Kundu, et al., The ProteomeXchange consortium in 2020: Enabling 'big data' approaches in proteomics, *Nucleic Acids Res.*, **48** (2020), D1145–D1152. https://doi.org/10.1093/nar/gkz984

17. E. W. Deutsch, N. Bandeira, Y. Perez-Riverol, V. Sharma, J. J. Carver, L. Mendoza, et al., The ProteomeXchange consortium at 10 years: 2023 update, *Nucleic Acids Res.*, **51** (2023), D1539–D1548. https://doi.org/10.1093/nar/gkac1040

18. N. S. Kale, K. Haug, P. Conesa, K. Jayseelan, P. Moreno, P. Rocca-Serra, et al., MetaboLights: An open-access database repository for metabolomics data, *Current Protoc. Bioinformatics*, **2016** (2016). https://doi.org/10.1002/0471250953.bi1413s53

19. K. Haug, K. Cochrane, V. C. Nainala, M. Williams, J. Chang, K. V. Jayaseelan, et al., MetaboLights: An open-access database repository for metabolomics data, *Nucleic Acids Res.*, **48** (2020), D440–D444. https://doi.org/10.1093/nar/gkz1019

20. M. Wilhelm, M. Kirchner, J. A. J. Steen, H. Steen, mz5: Space- and time-efficient storage of mass spectrometry data sets, *Mol. Cell. Proteomics*, **11** (2012). https://doi.org/10.1074/mcp.O111.011379

21. M. Folk, G. Heber, Q. Koziol, E. Pourmal, D. Robinson, An overview of the HDF5 technology suite and its applications, in *Proceedings of the EDBT/ICDT 2011 Workshop on Array Databases*, (2011), 36–47. https://doi.org/10.1145/1966895.1966900

22. J. Teleman, A. W. Dowsey, F. F. Gonzalez-Galarza, S. Perkins, B. Pratt, H. L. Röst, et al., Numerical compression schemes for proteomics mass spectrometry data, *Mol. Cell. Proteomics*, **13** (2014), 1537–1542. https://doi.org/10.1074/mcp.O114.037879

23. D. Bouyssié, M. Dubois, S. Nasso, A. G. de Peredo, O. Burlet-Schiltz, R. Aebersold, et al., MzDB: A file format using multiple indexing strategies for the efficient analysis of large LC-MS/MS and SWATH-MS data sets, *Mol. Cell. Proteomics*, **14** (2015), 771–781. https://doi.org/10.1074/mcp.O114.039115

24. T. Bhosale, T. Patil, P. Patil, SQLite: Light Database System, in *Int. J. Comput. Sci. Mob. Comput.*, **44** (2015), 882–885.

25. R. Yang, X. Chen, I. Ochoa, MassComp, a lossless compressor for mass spectrometry data, *BMC Bioinformatics*, **20** (2019). https://doi.org/10.1186/s12859-019-2962-7

26. B. Tully, Toffee–a highly efficient, lossless file format for DIA-MS, *Sci. Rep.*, **10** (2020). https://doi.org/10.1038/s41598-020-65015-y

27. R. S. Bhamber, A. Jankevics, E. W. Deutsch, A. R. Jones, A. W. Dowsey, MzMLb: A future-proof raw mass spectrometry data format based on standards-compliant mzML and optimized for speed and storage requirements, *J. Proteome Res.*, **20** (2021), 172–183. https://doi.org/10.1021/acs.jproteome.0c00192

28. F. Hanau, H. Röst, I. Ochoa, mspack: efficient lossless and lossy mass spectrometry data compression, *Bioinformatics*, **37** (2021), 3923–3925. https://doi.org/10.1093/bioinformatics/btab636

29. M. Lu, S. An, R. Wang, J. Wang, C. Yu, Aird: a computation-oriented mass spectrometry data format enables a higher compression ratio and less decoding time, *BMC Bioinformatics*, **23** (2022). https://doi.org/10.1186/s12859-021-04490-0

30. J. Wang, M. Lu, R. Wang, S. An, C. Xie, C. Yu, StackZDPD: A novel encoding scheme for mass spectrometry data optimized for speed and compression ratio, *BMC Bioinformatics*, **12** (2022), 5384. https://doi.org/10.1038/s41598-022-09432-1

31. M. Lu, J. Tong, R. Wang, S. An, J. Wang, C. Yu, Aird-ComboComp: A combinable compressor framework with a dynamic-decider for lossy mass spectrometry data compression, *bioRxiv*, **2023** (2023). https://doi.org/10.1101/2023.05.04.539411

32. R. Schmid, S. Heuckeroth, A. Korf, A. Smirnov, O. Myers, T. S. Dyrlund, et al., Integrative analysis of multimodal mass spectrometry data in MZmine 3, *Nat. Biotechnol.*, **41** (2023), 447–449. https://doi.org/10.1038/S41587-023-01690-2

33. D. Lemire, L. Boytsov, N. Kurz, SIMD compression and the intersection of sorted integers, *Software Pract. Exper.*, **46** (2016), 723–749. https://doi.org/10.1002/spe.2326

34. Google, Zstd, 1.5.2, (2022). Available from: https://github.com/facebook/zstd

35. Google, snappy, 1.1.9, (2022). Available from: https://github.com/google/snappy

36. J. Alakuijala, A. Farruggia, P. Ferragina, E. Kliuchnikov, R. Obryk, Z. Szabadka, et al., Brotli: A general-purpose data compressor, *ACM Trans. Inf. Syst.*, **37** (2018), 1–30. https://doi.org/10.1145/3231935

37. J. Tong, M. Lu, B. Peng, S. An, J. Wang, C. Yu, How much storage precision can be lost: Guidance for near-lossless compression of untargeted metabolomics mass spectrometry data, *bioRxiv*, (2023). https://doi.org/10.1101/2023.03.14.532504

38. R. Adusumilli, P. Mallick, Data conversion with proteoWizard msConvert, in *Methods in Molecular Biology*, Humana Press, (2017), 339–368. https://doi.org/10.1007/978-1-4939-6747-6_23

Appendix

The following abbreviations are used in this manuscript:
MS: Mass Spectrometry
SDK: Software Development Kits
HUPO-PSI: Human Proteome Organization-Proteomics Standards Initiative
ISB: Institute for System Biology
DDA: Data Dependent Acquisition
DIA: Data Independent Acquisition
XIC: Extracted Ion Chromatogram
RT: Retention Time
***m/z*:** mass-to-charge ratio
dp: decimal places
FWHM: Full Width at Half Maximum
MS1: Mass Spectrum Level 1
MS2: Mass Spectrum Level 2

CV: Controlled Vocabulary
GB: Gigabyte
MB: Megabyte
JVM: Java Virtual Machine
GUI: Graphical User Interface
IoT: Internet Of Things

AIMS Press