Mass Spectra alignment using virtual lock-masses

Francis Brochu[1,2*], Pier-Luc Plante[1,3], Alexandre Drouin[1,2], François Laviolette[1,2], Mario Marchand[1,2], Jacques Corbeil[1,3]

1 Big Data Research Center, Laval University, Québec, Qc, Canada

2 Département d'Informatique et Génie Logiciel, Université Laval, Québec, Qc, Canada

3 Centre de Recherche du CHU de Québec, Université Laval, Québec, Qc, Canada

¤Current Address: 2325 rue de l'Université, Big Data Research Center, Université Laval, Québec, Qc, Canada

* francis.brochu.2@ulaval.ca

Abstract

Mass spectrometry is a valued method to evaluate the metabolomics content of a biological sample. The recent advent of rapid ionization technologies such as Laser Diode Thermal Desorption (LDTD) and Direct Analysis in Real Time (DART) has rendered high-throughput mass spectrometry possible. It can now be used for large-scale comparative analysis of populations of samples. In practice, many factors resulting from the environment, the protocol, and even the instrument itself, can lead to minor discrepancies between spectra, rendering automated comparative analysis difficult. In this work, a sequence/pipeline of algorithms to correct variations between spectra is proposed. The algorithms correct multiple spectra by identifying peaks that are common to all and, from those, computes a spectrum-specific correction. We show that these algorithms increase comparability within large datasets of spectra, facilitating comparative analysis, such as machine learning.

Author summary

Mass spectrometry is a widespread technology used to measure the chemical content of samples. This measurement technique is often used with biological samples for diverse applications, such as protein sequencing, metabolomic profiling or quantitative measurements. However, with the increasing throughput of mass spectrometry technologies and methodologies, the resulting datasets are becoming larger. This reveals slight shifts in mass measured by the instruments, in the case of Time-of-Flight (ToF) mass spectrometers. These shifts render spectra harder to compare and analyze in large datasets. In this article, we propose algorithms that counter mass shifts and variations in datasets of ToF mass spectra. These algorithms use no external reference points, instead calculating spectrum-specific corrections by finding peaks present in all spectra of a dataset. Applying these algorithm yields a representation of the mass spectra that can then easily be used for statistical or machine learning analyses.

Introduction

Mass spectrometry (MS) is a widely used technique for acquiring data on the metabolome or the proteome of individuals [1] [2]. Proteomics applications can consist, among others, of typing of microbial organisms [3], imaging MS [4], quantitative comparisons [5], and peptide sequencing [6] [7]. For metabolomics applications, the two main approaches fall into the categories of targeted and untargeted studies. In comparison with targeted studies, untargeted studies acquire data using a shotgun approach. Therefore, this type of study is a good option for novel biomarker discovery and hypothesis generation [8] [9].

Through recent years, novel ionization technologies have emerged, facilitating the high-throughput acquisition of mass spectra [10]. Technologies such as Laser Diode Thermal Desorption (LDTD) or Direct Analysis in Real Time (DART), allow for the rapid acquisition of large datasets. These methods often preclude or bypass the time separation process used in Liquid Chromatography (LC) or Gas Chromatography (GC) [11]. Thus, without any time separation, a single mass spectrum will often be represented as lists of peaks, composed of the mass-to-charge ratio of the ion (m/z

value) and its intensity.

With the rise of larger datasets, multiple problems of comparability between spectra have emerged. Datasets are acquired in multiple batches over numerous days, on different instruments in multiple locations, with recalibrations of the instruments occurring between batches [12]. These factors induce variations in the spectra that hinders their comparison.

In the past, three algorithms have been proposed to address this problem, which mainly affecting Time-of-Flight mass spectrometers. These include the work of Tibshirani *et al.* (2004) [13], Jeffries (2005) [14], and Tracy *et al.* (2008) [15]. Tibshirani's algorithm relies on a clustering algorithm to align peaks that are present in multiple spectra and picks them for further statistical analyses. However, unlike the algorithms proposed in this article, it does not address the problem of inter-batch variations. Jeffries' algorithm is more appropriate for this problem. This method uses cubic splines to recalibrate spectra, based on the shifts between observed peaks and known reference masses. A similar algorithm has been proposed by Barry *et al.* (2013) for Fourier-Transform Mass Spectrometry [16]. This approach uses ambient ions in order to correct the spectra using known reference masses. One limitation of these algorithms is that they require known reference masses. The algorithm presented in this work alleviates this constraint, by automatically detecting such reference points. Another algorithm of interest for MALDI-ToF spectra has been proposed by Tracy *et al.* [15]. In this case, commonly occurring peaks within the dataset are used to correct the spectra and determine the binning distance used. However, this method computes a single constant correction factor for the entire spectrum, while the method proposed in this work computes correction factors that vary across the m/z axis of the spectra in order to obtain a more accurate correction.

The algorithms proposed in this article aim to render spectra more comparable prior to peak selection and statistical analyses. We draw inspiration from the interal lock mass approach and exploit the fact that spectra of samples of the same nature (i.e., blood plasma samples, urine samples, etc.) are very likely to share common peaks (i.e., compounds that are present in each sample). For example, human blood plasma contains compounds, such as glucose and amino acids [17]. Similarly, urine contains urea, creatinine, citric acid, and many more [18]. Hence, we propose to correct the

spectra based on the position of peaks that are detected to be consistently present in samples of the same nature. We call these peaks "virtual lock masses" (VLM) and propose an algorithm to detect them automatically. This idea is similar to the one proposed by Barry *et al.* [16], but the peaks are not limited to ambient ions. In this work, we show that our algorithm allows the detections of tens to hundreds of peaks that can be used as reference points to re-align the spectra and reduce inter-batch variations. Our approach is fully compatible with the classical lock mass approach, which can be used complementarily. Moreover, we show that a slight modification to the VLM detection algorithm can produce an alignment algorithm that can be used to further correct the spectra.

Hence, our key contributions are: an algorithm that automatically detects reference points in mass spectra, an algorithm that corrects the spectra based on these points, and an alignment algorithm to align large sets of spectra. In the next section, we present results supporting the accuracy of our reference point detection algorithm. Moreover, we show that the proposed algorithms are a key component of machine learning analysis performed on ToF mass spectra. Subsequently, we discuss these results and their implications and finally, present the details of the algorithms and their implementation.

Results

A consistent set of virtual lock masses can be detected in different batches

This experiment was conducted on the *Days* dataset (see *Methods*), which consist of 192 samples of pooled blood plasma. Half of the samples were acquired on a given day and the others were acquired on a separate day. Since the samples are of the same nature, we expect a high similarity apart from inter-batch variations. The goal of the experiment was to determine whether the same virtual lock masses are detected among similar datasets, and within parts of the same dataset.

The VLM detection algorithm was independently applied to 1) every spectrum in the dataset, to 2) only the spectra acquired on the first day, and to 3) only the spectra acquired on the second day. The algorithm was applied with the same window size of 40

ppm in all cases. This window size was determined by the procedure described in the Methods section, being the w that yielded the largest number of isolated VLMs on the entire dataset.

The detected VLMs were then compared in the following manner. We define that if we have two sets of spectra A and B, their detected VLMs will be $\mathcal{V}_A$ and $\mathcal{V}_B$. Each element of $\mathcal{V}_A$ is a VLM v_A that is composed of a single peak per spectrum for the spectra in A. If $B \subset A$, then a VLM $v_A \in \mathcal{V}_A$ and a VLM $v_B \in \mathcal{V}_B$ are homologous if the peaks forming v_B are a subset of the peaks forming v_A. Additionally, we can define comparisons between the VLMs of subsets of A. If we have sets of spectra A,B and C, where $B \subset A$ and $C \subset A$, then we can define that VLMs v_B and v_C are homologous if v_B if homologous to v_A and v_C is homologous to v_A.

We compared the peak groups forming the VLMs in all spectra with the spectra acquired on the first day, and saw that the 113 VLMs detected on all spectra have homologues in the set of 148 VLMs detected on the first day. Conversely, we observed that the 113 VLMs also have homologues within the set of 118 VLMs detected in the spectra acquired on the second day.

Hence, the algorithm finds common VLM points in all settings, corresponding to different days and multiple instrument recalibrations. This suggests that it correctly identifies landmark compounds that are present in a particular type of sample, which can be used as a common basis for correction. We therefore conclude that our detection algorithm behaves as expected.

Virtual lock mass correction improves machine learning analysis

Machine learning experiments were conducted on four binary classification tasks. The first two tasks consist of the detection of a single compound spiked in blood plasma samples from the *Clomiphene-Acetaminophen* dataset (see *Methods*). The third task is the detection of malaria infection in red blood cell culture samples from the *Malaria* dataset (see *Methods*). The fourth and final task consists of distinguishing plasma samples of patients with and without breast cancer in the *Cancer* dataset (see *Methods*).

Machine learning algorithms

Multiple machine learning algorithms were applied to the spectra. The first algorithm used is the AdaBoost ensemble method [19]. This method learns a weighted majority vote on a set of simple pre-defined classifiers. A linear Support Vector Machine (SVM) [22] was used with a L1-norm regularizer. The latter is to ensure that the predictions are based on a small subset of the peaks. We also used decision tree [20] and Set Covering Machine classifiers [21]. These algorithms have the advantage of producing interpretable classifiers that consist of a very small combination of simple rules on peak intensities. We used the Scikit-learn implementations for AdaBoost, CART, and the L1-regularized SVM [23]; whereas we used our own implementation of the SCM. This implementation is available at `https://github.com/aldro61/pyscm`.

Experimental protocol

For each experiment, the spectra were randomly partitioned into a training set and a test set. For the compound detection tasks (clomiphene and acetaminophen), the test set consisted of 50 selected samples. For the cancer detection task, the same number of samples were included in the test set. Finally, for the malaria detection task, 100 samples were selected for the test set. The hyper-parameters of each learning algorithm were chosen by 5-fold cross-validation on the training set (refer to Elements of Statistical Learning [24]). Each experiment was repeated 10 times independently on different partitions of the data.

Two different experimental protocols were tested which are illustrated in Fig (1). First, the correction and alignment algorithms were applied in the transductive learning setting [25]. In this setting, the whole dataset is exposed to the pipeline of proposed algorithms (VLM detection + VLM correction + alignment point detection). The training and testing sets are then partitioned randomly. The second experimental protocol was conducted as the inductive learning setting, in which the pipeline of proposed algorithms were only applied to the training set. Hence the set of alignment points is found from the training set only. For the inductive learning protocol, the percentile parameter of the alignment algorithm is considered an hyper-parameter and is thus cross-validated on the training set. For the transductive learning protocol, the

percentile parameter is set at 95%. The features shown to the machine learning algorithms are the alignment points and their associated intensity values.

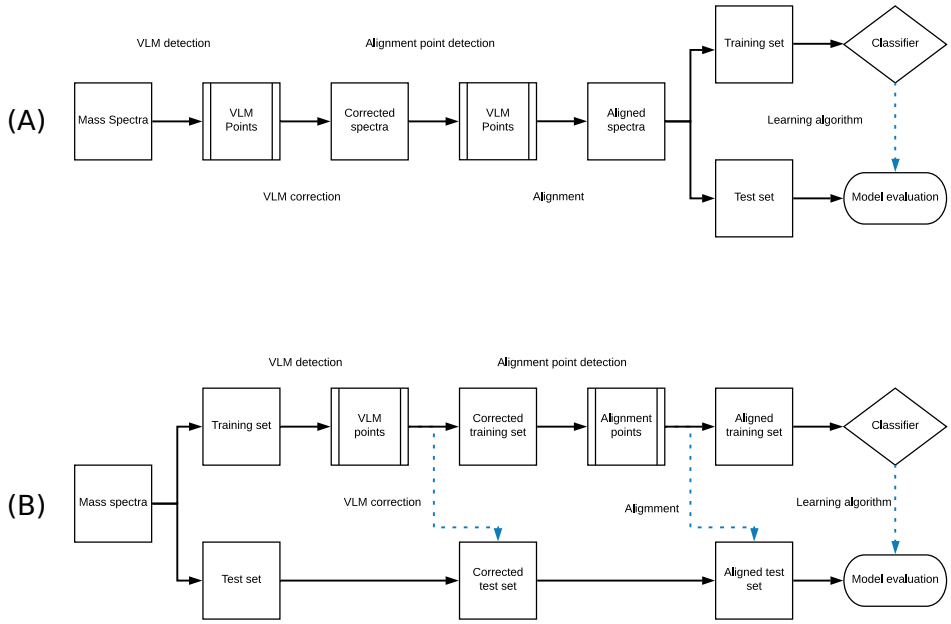

Fig 1. Transductive and inductive workflows. Subfig. (A) shows the transductive workflow. Subfig. (B) shows the inductive workflow. The dotted blue arrows show where the algorithms were applied on unseen data, while the whole black arrows show the workflow of the training data.

For each task, we compared the performance of classifiers according to their preprocessing. We thus compared (a) simply binning the spectra, (b) using the VLM detection and correction algorithms and then binning the mass spectra and (c) using the VLM detection and correction algorithms before using the alignment algorithm. Binning is a commonly used technique in mass spectrometry analysis consisting in grouping peaks and intensities found in a larger bin on the m/z axis into a single point or peak [26].

Results for transductive learning

Table 1 shows the results of the machine learning experiments in the transductive setting for different tasks. Let us first consider the case of the clomiphene detection task. In all conditions, we observe excellent results, with accuracies over 90% in almost every case. However, we know that the solution to this problem is the appearance of a

single additional molecule and its fragments in the spectra, since a solution of water and clomiphene is added in the plasma samples. Thus, it is expected that a single peak (feature) should be sufficient to classify the spectra. Considering this information, we see that a single peak is used for classification only when applying the VLM correction and alignment algorithms when using the Decision Tree and SCM. We also see a decrease in the number of features used for the AdaBoost classifier when using the VLM correction and alignment algorithms. In the case of the L1-regularized SVM, the sparsest solution (with an average of 2.6 features used) was obtained when the VLM correction algorithm was applied in addition to binning.

Condition	AdaBoost	Decision Tree	SCM	L1 SVM
Clomiphene Detection				
Binning only	98.0% (4.7)	98.6% (1.8)	95.2% (1.1)	89.6% (52.0)
VLM + Binning	98.2% (4.9)	97.0% (2.3)	97.0% (1.2)	**93.6%** (**2.6**)
VLM + Alignment	**98.8%** (**2.3**)	**99.4%** (**1.0**)	**99.4%** (**1.0**)	92.8% (138.6)
Acetaminophen Detection				
Binning only	99.2% (1.0)	99.2% (**1.0**)	99.2% (1.2)	97.6% (97.5)
VLM + Binning	99.2% (**1.0**)	99.2% (**1.0**)	**99.4%** (**1.0**)	99.0% (121.0)
VLM + Alignment	**99.8%** (**1.0**)	**100.0%** (**1.0**)	**99.4%** (**1.0**)	**99.6%** (**63.4**)
Malaria Detection				
Binning only	92.4% (51.8)	82.5% (**4.3**)	84.6% (2.2)	92.6% (150.1)
VLM + Binning	93.3% (**39.7**)	**88.7%** (4.6)	**89.4%** (**2.0**)	**95.4%** (133.2)
VLM + Alignment	**93.8%** (65.3)	86.1% (4.8)	85.4% (2.3)	95.2% (**131.4**)
Cancer Detection				
Binning Only	**70.4%** (69.2)	**63.8%** (6.4)	55.6% (**1.9**)	56.8% (**113.6**)
VLM + Binning	70.2% (43.9)	61.6% (4.8)	53.6% (2.2)	69.4% (138.6)
VLM + Alignment	67.4% (**30.0**)	62.6% (**2.3**)	**59.6%** (2.2)	**74.6%** (135.2)

Table 1. Machine learning results in the transductive setting. The percentage in each column is the average accuracy of classifiers on 10 repeats of the experiment. The number shown in parentheses is the average number of features used by the classifiers.

Consider now the results for acetaminophen detection. In this case, an acetaminophen pill was added to the blood plasma samples. Thus, it is expected that multiple molecules and their fragments appear in the spectra in this case, at extremely high concentration not normally found in physiological blood plasma. It is then not surprising that most algorithms can identify acetaminophen with the use of a singe feature (peak). Note that in the case of the L1-regularized Support Vector Machine, the best results, both in terms of accuracy and sparsity, are obtained when the VLM correction and alignment algorithms were used.

The next two tasks represent more realistic problems with unknown solutions. Let

us then consider the malaria detection task. For each algorithm, applying the VLM correction algorithm yields an increase in prediction accuracy. For the AdaBoost classifier, we observe an increase of about 1% and the best sparsity in the case of the VLM correction applied before binning, with a slight increase in accuracy with the alignment algorithm. The Decision Tree classifier increases its accuracy by approximately 5% with the VLM correction algorithm, both with alignment and with binning. We see a similar increase in accuracy for the Set Covering Machine in the case of VLM correction with binning. Finally, the L1-regularized SVM obtains a 3% increase in accuracy with the VLM correction algorithm applied, and a better sparsity.

Finally, let us consider the results for the cancer detection task. This classification problem is much harder, with few machine learning algorithms having a prediction accuracy over 70%. Still, both the AdaBoost and Decision Tree classifiers have similar results in all cases, with slight losses in accuracy but improved sparsity with the proposed algorithms applied. The Set Covering Machine sees its accuracy increased by 4% with both correction and alignment algorithms applied and with comparable sparsity. However, in the case of the L1-regularized SVM, the classifier accuracy increases of almost 20% with the proposed algorithms compared to binning only.

Results for inductive learning

In Table 2, we compare the effect of using the proposed algorithms in the transductive setting versus the inductive setting. For the compound detection tasks, there is very little difference between the two approaches for both clomiphene detection and acetaminophen detection. The inductive setting yields slightly sparser classifiers, but the results are very similar. For the malaria detection task, the difference in sparsity is not significant for the Decision Tree and Set Covering Machine algorithms. The AdaBoost classifier is sparser for the inductive setting, while the L1 SVM has a significant advantage in the transductive setting. The results are also very similar in terms of accuracy for both settings, with very slightly better accuracies in the transductive setting. Finally, the transductive setting appears to be the best setting for cancer detection. The AdaBoost classifier is sparser in this case, with a slight decrease in accuracy. The Decision Tree and Set Covering Machine have better accuracies in the transductive setting, though the SCM is sparser in the inductive setting. The

L1-regularized SVM is, on the other hand, much more accurate and slightly sparser in the transductive setting, with an increase in accuracy of about 6%.

Condition	AdaBoost	Decision Tree	SCM	L1 SVM
Clomiphene Detection				
Transductive	98.8% (2.3)	99.4% (1.0)	99.4% (1.0)	92.8% (138.6)
Inductive	99.4% (1.0)	99.4% (1.0)	96.4% (1.0)	93.4% (90.0)
Acetaminophen Detection				
Transductive	99.8% (1.0)	100.0% (1.0)	99.4% (1.0)	99.6% (63.4)
Inductive	100.0% (1.0)	99.2% (1.0)	99.6% (1.0)	98.6% (30.0)
Malaria Detection				
Transductive	93.8% (65.3)	86.1% (4.8)	85.4% (2.3)	95.2% (131.4)
Inductive	92.9% (54.3)	87.8% (4.7)	84.2% (2.2)	95.1% (151.0)
Cancer Detection				
Transductive	67.4% (30.0)	62.6% (2.3)	59.6% (2.2)	74.6% (135.2)
Inductive	69.2% (63.9)	61.2% (6.7)	57.4% (1.6)	68.2% (145.4)

Table 2. Comparison of transductive and inductive learning of the VLM and Alignment algorithms

Finally, and perhaps not surprisingly, we can see (for AdaBoost and L1-SVM) that cancer and malaria detection need far more features then clomiphene and acetaminophen detection.

Stability of virtual lock masses in datasets

This experiment was conducted in order to verify that virtual lock masses detected on a given dataset will be found in unseen spectra of the same type. The algorithm for VLM detection was also cross-validated on the *Days* Dataset and the *Clomiphene-Acetaminophen* Dataset. Each dataset was randomly split into k folds. The VLM detection algorithm was applied to the first $k-1$ folds, the training folds. The detected VLMs on the training folds are then used for VLM correction of the spectra in last remaining fold, the testing fold. When the correction is applied, we note if every VLM is found in the spectra of the testing fold. The algorithm is scored according to the ratio of detected VLMs on the training folds that are also found in the testing fold. This process is repeated k times so that each fold serves as a test fold once. Multiples values of k were used in the experiment, such that $k \in \{3, 5, 8, 10, 15, 20\}$.

In each case, we found that *every* VLM point detected on the training set was detected on the testing set. This thus results in a ratio of VLMs found in the testing set over the VLMs detected on the training set of 100% in all cases. This provides empirical

evidence of the stability of VLM points across different sets of spectra.

Learning Curves

A learning curve experiment was performed in order to evaluate the behavior of the VLM detection and correction algorithms on varying numbers of samples. In a first step, the VLM detection algorithm followed by the VLM correction algorithm was performed on the whole set of spectra. 25 spectra were randomly selected as a test set. These test spectra will be considered the "ground truth", i.e., the best correction that the algorithm can achieve for these 25 spectra.

The algorithm was subsequently applied once again to a number of spectra. The number of spectra the algorithm was exposed to was gradually increased, from 10 spectra to 160 spectra. At each point, the uncorrected test spectra are corrected and compared to the ground truth spectra. The difference in m/z value between the homologous peaks is calculated in ppm. Then, the difference is squared and summed for all test spectra. Finally, this sum is divided by the number of peaks in the test spectra and the square root is taken. The difference in correction is thus expressed as the Root Mean Squared Error (RMSE) in ppm units for each peak. This experiment was repeated 50 times, with randomly re-sampled test sets, in order to obtain statistically significant results.

Fig 2 shows the learning curves obtained on three different datasets. In each case, the trends is similar. When sub-sampling a low number of spectra as a training set for the VLM detection and correction algorithms, a higher number of lock masses is found. As the number of training spectra increases, the number of virtual lock masses found diminishes and starts to plateau near the number of lock masses found in the whole dataset. This is explained by the fact that when few spectra are in the training set, there is a higher number of candidates. As new spectra are added in the training set, there is a probability that one of the new spectra are missing at least one peak that was previously considered a virtual lock mass. These peaks could be missing because of strong noise, either on the m/z axis or in terms of intensity, rendering its intensity too small to be considered a VLM. A peak could also be missing simply because the compound or fragment generating that peak is not present in all samples.

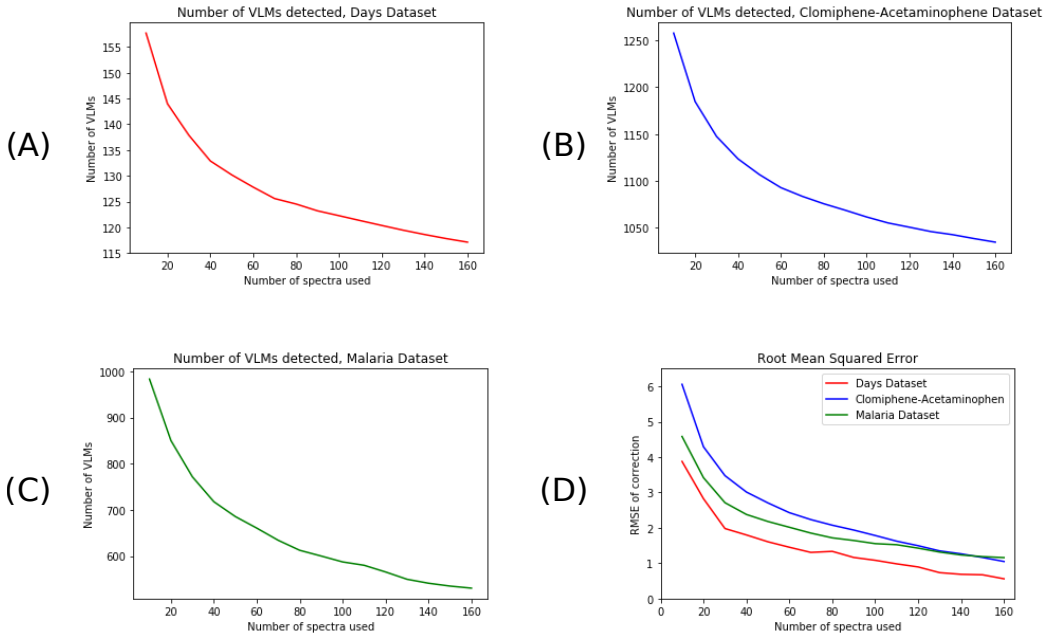

Fig 2. Learning Curves of Virtual Lock Mass Detection and Correction. Subfigures (A), (B) and (C) show the learning curves for three different datasets (Days , Clomiphene-Acetaminophen, Malaria). Subfigure (D) shows the RMSE of VLM Correction for these datasets.

The same trend is found in all three datasets for the Root Mean Squared Error (RMSE) in Subfigure (D). The error is initially high when few spectra are in the training set, but as more spectra are added in the training set it gradually decreases. In the case of the *Days* Dataset, the final average RMSE when using 160 spectra to train the algorithm is 0.56 ppms. For the other two datasets (*Clomiphene-Acetaminophen* and *Malaria*), the final RMSEs are approximately 1.10 ppms. In each case, the RMSE drops under 2.0 ppms when using 100 spectra or more to train the correction algorithm. In conjunction with the results of inductive learning shown above, these results suggest that the VLM detection and correction algorithms can generalize the virtual lock masses and correction it learns to unseen spectra of the same nature, such as those of a new test set.

Discussion

The algorithms proposed in this article aim to render mass spectra more comparable for large datasets acquired in single or multiple batches. The VLM detection algorithm is

stable and detects virtual lock masses reliably in datasets. It also detects peaks that are present in mass spectra of the same type but that are not part of the training set. In addition, applying the proposed pipeline of algorithms (VLM detection + VLM correction + alignment point detection) on sets of mass spectra before statistical and machine learning analyses generally yields classifiers with increased accuracy and sometimes with increased sparsity, leading to interpretable models that could serve for biomarker discovery. The proposed pipeline of algorithms has a very low running time complexity of $O(n \log m)$ for a collection of m spectra containing a total of n peaks which, as argued, cannot be surpassed by algorithms based on clustering (with the current state of knowledge).

However, the algorithms, as presented, have a number of drawbacks. Since the virtual lock masses are assigned the average m/z value of the peaks associated to it, the correction algorithm does not correct the peaks to the exact m/z value of the ion. The alignment algorithm has also the same property. However, the virtual lock mass approach is compatible with any external lock masses added to the spectra. Thus, by applying both methods, any shift away from the exact (and known) m/z value of an external lock mass can be corrected. Some situations are also unsuitable for the proposed algorithms. In order for the VLM detection algorithm to function properly and detect virtual lock masses, the mass spectra forming the dataset must be of the same "nature" so that the algorithm can detect a sufficient number of peaks that are common to all spectra. Additionally, the correction algorithm works best in a situation where there are more peaks than spectra. In the cases where each spectrum contains very few peaks, there is a much lower probability that that algorithm can find peaks present in all spectra of the set.

Future works The algorithms, as presented here, can only be applied to mass spectra represented by a list of peaks of the form (μ, ι) where μ is the m/z value of the peak and ι its intensity. Hence, the algorithms are currently not applicable with mass spectra having additional dimensions for the peaks, such as ion mobility. It is also not applicable to chromatogram where multiple mass spectra are acquired over time for a single sample and where each spectra contains different peaks due to the chromatographic separation. It is thus relevant to investigate if the proposed approach,

based on virtual lock masses, can be extended to incorporate these extra dimensions.

Materials and methods

In this section, we present the mathematical basis of the proposed methodology. First, the problem of virtual lock-mass identification is addressed. A formal definition of VLM peaks is introduced, along with an highly efficient algorithm capable of identifying such peaks in a set of mass spectra. Second, a methodology for correcting mass spectra based on a set of identified virtual lock masses is described. Thirdly, an algorithm for mass spectra alignment based on the previous algorithm is proposed. Finally, the datasets used and the experimental methodologies are presented.

Definitions

Let us first recall that a *set* is an un-ordered collection of elements whereas a *sequence* is an ordered collection of elements. Hence, in a sequence we have a first element, a second element, and so on. If A is a sequence or a set, $|A|$ denotes the number of elements in A.

Let $\mathcal{S} \stackrel{\text{def}}{=} \langle S_1, ..., S_m \rangle$ be a sequence of mass spectra. Each spectrum S_i is a sequence of peaks, where each peak is a pair (μ, ι) with an m/z value μ and a peak intensity ι.

We will use intervals that are centered on a peak that are measured in units relative to the m/z value of the peak. Hence, a *window* of size $2w$ centered on peak (μ, ι) covers the m/z interval that starts at $\mu(1-w)$ and ends at $\mu(1+w)$. In that case, we say that the window size parameter w is expressed in *relative units*. The reason for using window sizes in relative units follows from the fact that the m/z uncertainty of ToF mass spectrometers increases linearly with the m/z value of a peak. This is also confirmed by our experiments (Figure (4) displayed in the subsection *From VLM correction to spectra alignment*). Moreover, we are normally interested in values for w that typically lie within the range from 10^{-6} to 10^{-5}. When $w = z \times 10^{-6}$, it is customary to write that $w = z$ ppm.

Given a sequence $\mathcal{S}$ of mass spectra and a window size parameter w expressed in relative units, a *virtual lock mass* (VLM) with respect to $(\mathcal{S}, w)$ is a point v on the m/z axis such that there exists a sequence $\mathcal{P}$ of peaks from $\mathcal{S}$ that satisfies the following properties

1. $\mathcal{P}$ contains exactly one peak from each spectrum in $\mathcal{S}$.
2. The average of the m/z values of the peaks in $\mathcal{P}$ is equal to v.
3. Every peak in $\mathcal{P}$ has a m/z value located in the interval $[v(1-w), v(1+w)]$.
4. No other peak in $\mathcal{S}$ has an m/z value that belongs to $[v(1-w), v(1+w)]$.
5. Every peak in $\mathcal{P}$ has an intensity superior to a threshold t.

Whenever these criteria are satisfied, we say that $\mathcal{P}$ is the sequence of peaks associated to the VLM v.

The threshold on intensity t is added as a criterion because a peak with a higher intensity will tend to have a higher mass accuracy. Hence, in principle, a VLM is defined only with respect to (w.r.t.) $(\mathcal{S}, w, t)$. However, we will drop the reference to t to simplify the notation.

A crucial aspect of the definition of VLM is the fact that it holds only w.r.t. a given value of a window size w. Indeed, consider Fig (3) which represents the peaks coming from three different spectra. We clearly see a grouping within a window size w_1. Hence, this set of 3 spectra identifies 4 VLM points w.r.t. w_1. Note that we could not unambiguously define a VLM point as a sequence of $|\mathcal{S}|$ peaks (without referring to a window size) since, in that case, the peaks surrounded by window w_2 would also qualify as being a VLM and, consequently and ambiguously, each peak would belong to more than one VLM. Note that if the chosen window size is too large, as it is the case for w_2, then the different VLM sequences will overlap. If the chosen window size is too small (i.e. smaller than w_1), we will have no VLM point. If it is too large however, we will see too many but, in this case, they will overlap and we can thus identify that case. Hence, this motivates the following definition of overlapping VLM points. Given $(\mathcal{S}, w)$, a VLM v_i w.r.t. $(\mathcal{S}, w)$ is said to *overlap* with another VLM v_j (with respect to $(\mathcal{S}, w)$) if and only if there exists a non empty intersection between the m/z interval $[v_i(1-w), v_i(1+w)]$ and the m/z interval $[v_j(1-w), v_j(1+w)]$. Moreover, we say that a VLM v w.r.t. $(\mathcal{S}, w)$ is *isolated* from all all other VLM with w.r.t. $(\mathcal{S}, w)$ if and only if there does not exists any other VLM v' w.r.t. $(\mathcal{S}, w)$ that overlaps with v. For a given window size w, the algorithm that we present in the next subsection identifies *all*

isolated VLM points w.r.t. $(\mathcal{S}, w)$. Consequently, the best value for w is one for which the number of isolated VLM points is the largest.

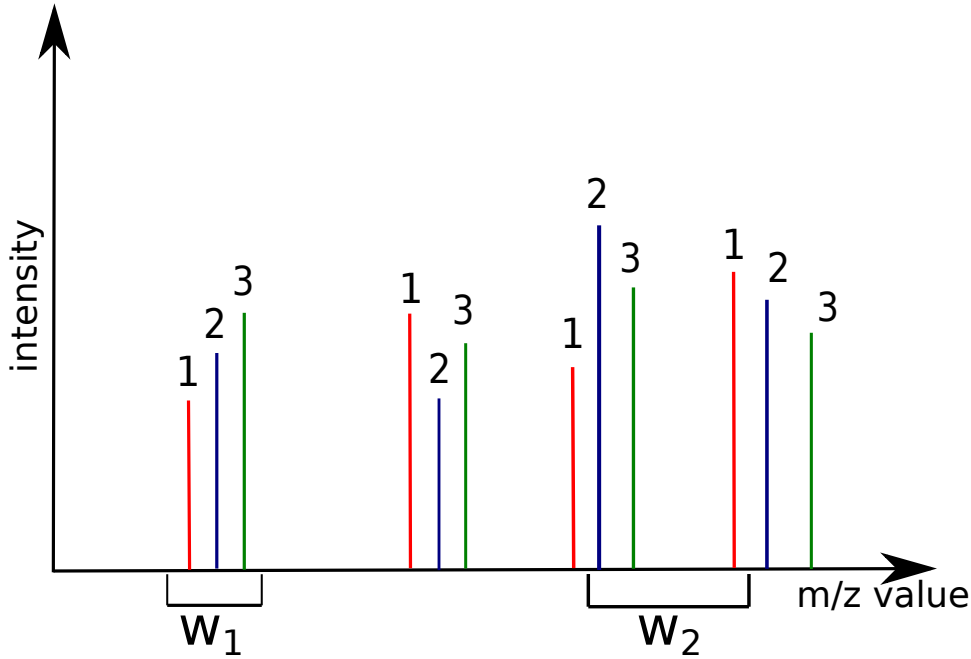

Fig 3. Peaks coming from three different spectra (identified as 1, 2, and 3).

An Algorithm for Virtual Lock Mass Identification

Given a sequence $\mathcal{S} = \langle S_1, \ldots, S_m \rangle$ of m spectra, each peak is identified by a pair (σ, ρ) where $\sigma \in \{1, \ldots, m\}$ is the index of its spectrum of origin and $\rho \in \{1, \ldots, n_\sigma\}$ is the index of the peak in spectrum S_σ containing n_σ peaks. Given that we have a total of n peaks in $\mathcal{S}$, we have that $\sum_{\sigma=1}^{m} n_\sigma = n$. For the description of the algorithm, $\mu(\sigma, \rho)$ denotes the m/z value of peak (σ, ρ). Finally, we assume that the peaks in each spectra S_i are listed in increasing order of their m/z values.

The proposed algorithm uses two data structures: a *binary heap* and a so-called *active sequence*. A binary heap is a classical data structure used for priority queues which are useful when one wants to efficiently remove the element of highest priority in a queue. In our case, the heap will maintain, at any time, the next peak of each spectra to be processed by the algorithm. Hence, given a sequence $\mathcal{S}$ of m spectra, the heap generally contains a set of m peaks, where each peak belongs to a different spectrum of $\mathcal{S}$. The "priority value" for each peak (σ, ρ) in the heap is given by its m/z value

$\mu(\sigma, \rho)$; a peak with the smaller mass is always on top of the heap. Given a sequence $\mathcal{S}$ of m mass spectra, a heap H containing the first peak of each spectrum can be constructed in $O(m)$ time. Moreover, we can read the m/z value at the top of the heap in constant time; we can remove the peak (σ, ρ) of the top of the heap and replace it with the next available peak in the spectrum S_σ in $O(\log m)$ time [1].

The second data structure is, what we call, the *active sequence* A. At any time, A contains a sequence of peaks, listed in increasing order of their m/z values, which are currently being considered to become a VLM sequence. That data structure uses a doubly linked list L and a boolean-valued vector B of dimension m. The linked list L is actually containing the sequence of peaks to be considered for the next VLM and the vector B is such that, at any time, $B[\sigma] = True$ if and only if a peak from spectrum S_σ is present in L. The active sequence A also maintains the m/z value μ_l of the last peak that was removed from L, the average m/z value μ_A of the peaks in L, and a copy w_A of the window size w chosen by the user. Since L is a linked list, we can read the front (first) and back (last) values of L in constant time, as well as obtaining its size (number of peaks). Removing the value at the front of L is also performed in constant time.

We now present a short description of the algorithm for virtual lock mass identification. The detailed description is provided in Supplementary Information 1.

Validation of an active sequence For this step, we use a method, call $A.isValid()$, that returns $True$ if and only if the peaks in the active sequence A satisfies all the criteria enumerated in the definition of a VLM. A precondition for the validity of this method is that L contains only peaks that belong to distinct spectra of $\mathcal{S}$. This precondition holds initially for an empty list L and will always be enforced each time a new peak gets inserted in A, as we will see below. Thus, this step of the algorithm checks first that the active sequence contains exactly $|\mathcal{S}|$ peaks, thus one peak from each spectrum in the set. Then, if there are still peaks in the heap, we verify that the peak at the top of H (thus, the peak immediately following the active sequence) has a m/z value that is out of the interval $[\mu_A(1-w), \mu_A(1+w)]$. Similarly, it is verified that the peak whose m/z value immediately precedes the active sequence also has an m/z value

[1] We refer here to the well-known running times (available from any introductory textbook on data structures and algorithms) for heap construction, removal of its top element, and insertion of a new element.

outside of $[\mu_A(1-w), \mu_A(1+w)]$. If either peak lies inside this window, then the property (4) of a VLM is violated, as the window contains more than $|\mathcal{S}|$ peaks. Finally, we ensure that the first and last peaks in the active sequence A are both within the window $[\mu_A(1-w), \mu_A(1+w)]$. If all checks pass, then the current sequence is considered a potential virtual lock mass.

Advancing the active sequence This step tries to insert at the end of the list L of A the peak (σ, ρ) located on top of H. The insertion succeeds if the resulting A still have some probability that the peak sequence can become a VLM after zero or more future insertions. Thus, we first verify if another peak from spectrum $\mathcal{S}_\sigma$ is present in A. If that is the case, then the insertion fails. Otherwise, we compute the new value μ'_A that μ_A will have after the insertion. If the peak at the front of L (the peak in A having the smallest m/z value) and the new peak (σ, ρ) have masses that are within the window $[\mu'_A(1-w_A), \mu'_A(1+w_A)]$, then the insertion succeeds. The peak is inserted, and H is updated by removing the peak (σ, ρ) and adding the next peak from the spectrum $\mathcal{S}_\sigma$. Thus, this step ensures that we can insert a new peak in A and still have some probability that the sequence can become a VLM after zero or more future insertions.

Whenever we have an insertion failure, it means that the active sequence cannot become a valid VLM and that we must remove from A the peak having the smallest m/z value (which is located in the front of L) in order to have a chance that the sequence of peaks in A becomes a valid VLM.

Advancing the lower bound This step is called to remove the peak (σ, ρ) at the front of L until a valid insertion can be made. First, it updates $B[\sigma]$ to $False$, as peak (σ, ρ) is about to be removed and no peak from S_σ will be in the active sequence A anymore. The m/z value of peak (σ, ρ) is copied in μ_l, and the peak is then removed from L. If L is empty at this point, its average m/z value μ_A is set to 0. Otherwise, μ_A is set to the average value of the peaks remaining in the active sequence.

Removing overlapping virtual lock masses The final step of the algorithm removes all overlapping VLMs. As described in Appendix 1, a Boolean vector (with a number of components equal to the number of VLMs found) is initialized to $False$. Then, we simply iterate over all the VLM points found and assign the corresponding

vector entry to $True$ whenever a VLM point (with m/z value μ) is found such that its window $[\mu(1-w), \mu(1+w)]$ overlaps with that of its neighboring VLMs. Only the VLMs whose entry in the vector is $False$ are kept.

```
virtualLockMassDetection(S, w);
Input: S = ⟨S_1, S_2, ..., S_m⟩, a sequence of mass spectra.
Input: w, a window size parameter in relative units.
Output: The sequence of all isolated VLM points with respect to (S, w).
Data: H, a heap initialized with H.init(S); thus containing the first peak of each
      spectra in S.
Data: A, an active sequence initialized with A.init(H, w); hence initially empty.
Data: U, a sequence of m/z values, initially empty.
found ← False;
while H.empty() = False do
    if A.isValid(H) = True then found = True;
    if A.insert(H, S) = false then
        if found = True then
            U.append(A.getμ_A());
            found ← False;
        end
        A.advanceLowerBound();
    else
        if H.empty() = True then
            while A.empty() = False do
                if A.isValid(H) = True then
                    U.append(A.getμ_A());
                    break;
                end
                A.advanceLowerBound();
            end
        end
    end
end
return deleteOverlaps(U, w);
```

Algorithm 1: The Virtual Lock Mass Detection Algorithm.

Main algorithm Having described the data structures used and their methods, we are now in position to present the main algorithm for virtual lock mass detection, which is described by Algorithm (1). The task of this algorithm is to find all the isolated VLM points w.r.t. $(\mathcal{S}, w)$. To achieve this, the central part of the algorithm is to find find the sequence $\mathcal{U} = \langle \mu_1, \ldots, \mu_{|\mathcal{U}|} \rangle$ of all possible VLM points w.r.t. $(\mathcal{S}, w)$. This sequence may contain several pairs of overlapping VLMs. The strategy to achieve this central task is to use $A.insert(H, \mathcal{S})$ to try to insert in A (consequently in L) the next unprocessed peak of $\mathcal{S}$, which is always located on the top of the heap H. Initially, the first such peak of

$\mathcal{S}$, a peak having the smallest m/z value among those in $\mathcal{S}$, gets eventually inserted in an empty A by $A.insert(H, \mathcal{S})$. Next, after verifying with $A.isValid(H)$ if the content of A satisfies the criteria to be a valid VLM sequence, we try to insert again in A the next available peak. On each insertion failure, we test if, before this insertion, the content of A was a valid VLM sequence. This is done with the Boolean variable $found$ (which is set to $True$ as soon as the content of A is a valid VLM sequence and which is set to $False$ immediately after the average m/z value μ_A of A's content is appended to $\mathcal{U}$). Hence, for each considered peak in $L.front()$, we try to insert one more peak in L and test after the insertion if L's content is a valid VLM sequence. If we cannot insert an extra peak in L with the current peak in $L.front()$ this means that there is no possibility of finding one more VLM sequence with the current peak in $L.front()$. In that case we remove that peak from L with $A.advanceLowerBound()$ and, consequently, $L.front()$ now becomes the peak that was next to $L.front()$ in L. Hence, with this strategy, the algorithm attempts to find the largest consecutive sub-sequence of peaks from $\mathcal{S}$ that starts with any given peak in $\mathcal{S}$ and that forms a valid VLM sequence[2]. In addition, note that in the **else** branch of Algorithm (1), we verify if H becomes empty after a successful insertion. In that case, we need to check if we can find a valid VLM sequence by incrementing sequentially the lower bound $L.front()$ and then append to $\mathcal{U}$ the first VLM found. Then, we can safely exit the **while** loop since any other possible VLM sequence will be a subset of the one already found. Without this **else** branch, a VLM sequence that ends with the last peak presented by H would be missed by the algorithm. As explained in Appendix 1, the running time of Algorithm (1) (i.e., the VLM detection algorithm) is in $O(n \log m)$ for a sequence $\mathcal{S}$ of m spectra that contains a total of n peaks. This, however, is for a fixed value of window size w. Note that in order to obtain the most accurate correction (by interpolation) for the spectra in $\mathcal{S}$, we should use the largest number of isolated (i.e., non-overlapping) VLMs we can find. Consequently, the optimal value for w is the one for which Algorithm (1) will give the largest number of isolated VLMs. Moreover, note that if w is too small, very few VLMs will be detected as w will not be able to cover exactly one peak per spectra. If,

[2]This may appear to be a strategy a bit too complicated than necessary in view of the fact that the largest (and smallest) such sub-sequence must contain exactly m peaks to be a valid VLM. However, we will see below that a significant advantage of using the proposed strategy is the fact that the same algorithm, modulo some very small and trivial modification, can also be used to detect the alignment points of $\mathcal{S}$.

on the other hand, w is too large, a large number of the VLMs found in the first phase of the algorithm will overlap and the remaining isolated VLMs will be rare. Consequently, because of this "unimodal" behavior, one can generally find rapidly the best value for w. In our case, we never needed to tried more than 20 different values.

An Algorithm for Virtual Lock Mass Correction

Given a sequence $\mathcal{S}$ of spectra and a widow size parameter w expressed in relative units, once the sequence $\mathcal{V}$ of all isolated VLM points w.r.t. $(\mathcal{S}, w)$ has been determined, the individual spectra in $\mathcal{S}$ can be corrected in a manner similar as it is usually done with traditional lock masses. Algorithm (2) performs the correction needed for each peak in a spectrum $S \in \mathcal{S}$.

virtualLockMassCorrection$(S, \mathcal{V}, w)$;
Input: $S = \langle(\mu_1, \iota_1), (\mu_2, \iota_2), ..., (\mu_m, \iota_m)\rangle$, a spectrum
Input: $\mathcal{V} = \langle v_1, v_2, ..., v_n\rangle$, a sequence of m/z values (VLMs) sorted in increasing order
Input: w, a window size parameter in relative units
Output: A spectrum $S' = \langle(\mu'_1, \iota_1), (\mu'_2, \iota_2), ..., (\mu'_m, \iota_m)\rangle$ where each μ'_j is the corrected m/z value for the peak $(\mu_j, \iota_j) \in S$
Data: $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)$, a vector of indexes (natural numbers)
//construction of $\boldsymbol{\alpha}$
$i \leftarrow 1$;
for $j = 1$ **to** m **do**
  if $\mu_j \in [v_i(1-w), v_i(1+w)]$ **then**
    $\alpha_i \leftarrow j$ //peak (μ_j, ι_j) is associated to VLM v_i;
    $i \leftarrow i + 1$;
  end
end
//correct each μ_j such that $\alpha_1 \leq j \leq \alpha_n$
$j \leftarrow \alpha_1$;
$i \leftarrow 1$;
while $i < n$ **do**
  //linear interpolation correction of μ_j when $\alpha_i \leq j \leq \alpha_{i+1}$
  $slope \leftarrow \frac{v_{i+1} - v_i}{\mu_{\alpha_{i+1}} - \mu_{\alpha_i}}$;
  $b \leftarrow v_i - slope \times \mu_{\alpha_i}$;
  while $j \geq \alpha_i \wedge j \leq \alpha_{i+1}$ **do**
    //correction of μ_j
    $\mu'_j \leftarrow slope \times \mu_j + b$;
    $j \leftarrow j + 1$;
  end
  $i \leftarrow i + 1$;
end

Algorithm 2: Virtual Lock Mass Correction Algorithm

First, in the **for** loop, we identify each peak of S corresponding to a lock mass point $v_i \in \mathcal{V}$. Since $S \in \mathcal{S}$ and v_i is a VLM point w.r.t. $(\mathcal{S}, w)$, we are assured to find exactly one such peak $p_j \in S$ with an observed m/z value of μ_j such that μ_j lies in the interval $[(1-w)v_i, (1+w)v_i]$. For such μ_j, we assign the index j to α_i so that $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)$ is a vector of n indexes, each pointing to the peak in S associated to a VLM point. Note that for $\mu_j \in [(1-w)v_i, (1+w)v_i]$, its corrected m/z value must be equal to v_i. Instead of performing these corrections immediately in the **for** loop, we delay them to the linear interpolation step where all peaks having a m/z value μ_j such that $\alpha_1 \leq j \leq \alpha_n$ will be corrected.

Next, for each VLM v_i, we correct by linear interpolation all the m/z values μ_j such that $\alpha_i \leq j \leq \alpha_{i+1}$. To explain precisely this procedure, let $\mu'(\mu_j)$ denote the corrected value of μ_j. Linear interpolation consists at looking for a correction of the form

$$\mu'(\mu_j) = a\mu_j + b\,,$$

where a is called the *slope* and b is the *intercept*. By imposing that $\mu'(\mu_j) = v_i$ for $j = \alpha_i$ and $\mu'(\mu_j) = v_{i+1}$ for $j = \alpha_{i+1}$, we find that

$$a = \frac{v_{i+1} - v_i}{\mu_{\alpha_{i+1}} - \mu_{\alpha_i}}\,,$$

and $b = v_i - a\mu_{\alpha_i}$. The nested **while** loops of the algorithm performs exactly these linear interpolation corrections for all μ_j such that $\alpha_i \leq j \leq \alpha_{i+1}$ for $i = 1$ to $n-1$.

Once all m/z values μ_j such that $\alpha_1 \leq j \leq \alpha_n$ have been corrected, the algorithm is done. Hence, we have decided not to correct any m/z value of S that is either smaller that $v_1(1-w)$ or larger than $v_n(1+w)$ because such a peak has only one adjacent VLM and, consequently, could only be corrected by extrapolation, which is much less reliable than interpolation[3]. Finally, the intensities of the peaks remain unchanged. The running time complexity of this algorithm is $O(m)$ where m is the number of peaks in the spectrum S (see the full details in Appendix 1).

[3]Therefore, we recommend removing all these peaks from S to perform statistical analyses or machine learning experiments.

From VLM correction to spectra alignment

After running the VLM detection and correction algorithms, all the peaks associated with VLM points will be perfectly aligned in the sense that each peak in different spectra associated to a VLM point v will have exactly the same m/z value v. However, all the other peaks corrected by Algorithm (2) will not be perfectly aligned in the sense that a molecule fragment responsible for a peak in different spectra will not yield exactly the same mass after correction. This is due to possibly many uncontrollable phenomena that vary each time a sample gets processed by a mass spectrometer, and by the fact that the correction of each peak was performed by an approximate numerical interpolation. However, if all the peaks have been corrected by Algorithm (2), we expect that the peaks corresponding to the same molecule fragment f across different spectra will have very similar masses and will all be localized within a very small window of m/z values. Moreover, we also expect that the m/z values of the peaks coming from another molecule fragment g having a different mass will not cross the m/z values coming from molecule fragment f.

More precisely, suppose that we have executed Algorithms (1) and (2) with a window size parameter w (in relative units) on a sequence $\mathcal{S}$ of mass spectra. In addition, suppose that a molecule fragment f gives rise to a peak of m/z value μ_1 in spectrum S_1, and a peak of m/z value μ_2 in spectrum S_2, and so on for a sub-sequence of spectra in $\mathcal{S}$. Let $\mathcal{M}_f = \langle \mu_1, \mu_2, \ldots \rangle$ be the sequence of these m/z values. Moreover, let μ_f be the average of the m/z values in $\mathcal{M}_f$. Then, we expect that there exists a window size θ in relative units, such that $0 < \theta \ll w$, and for which we have $\mu_i \in [\mu_f(1-\theta), \mu_f(1+\theta)]$ for all $\mu_i \in \mathcal{M}_f$. Moreover, if θ is sufficiently small, we expect that the sequence $\mathcal{M}_g$ referring to peaks produced by another molecule fragment g having a different mass will be such that each $\mu_j \in \mathcal{M}_g$ will not be located within $[\mu_f(1-\theta), \mu_f(1+\theta)]$.

Motivated by this hypothesis, let us introduce the following definitions. Given that Algorithms (1) and (2) have been executed on a sequence $\mathcal{S}$ of mass spectra with window size parameter w in relative units, and given that we have another window size parameter $\theta \ll w$ in relative units, we say that a m/z value μ_f is an *alignment point* w.r.t. $(\mathcal{S}, \theta)$ if there exists a sequence $\mathcal{M}_f$ of peaks from $\mathcal{S}$ that satisfies the following properties.

1. Every peak in $\mathcal{M}_f$ comes from a different spectrum of $\mathcal{S}$.
2. The average of the m/z values of the peaks in $\mathcal{M}_f$ is equal to μ_f.
3. Every peak in $\mathcal{M}_f$ has an m/z value in $[\mu_f(1-\theta), \mu_f(1+\theta)]$ and all other peaks of $\mathcal{S}$ have an m/z value outside this interval.
4. There does not exist another peak in $\mathcal{S}$ that we can add to $\mathcal{M}_f$ and still satisfy the above properties.

Whenever these criteria are satisfied, we say that $\mathcal{M}_f$ is the *alignment sequence associated* to alignment point μ_f. Given $\mathcal{S}$ and θ, an alignment point μ_f w.r.t. $(\mathcal{S}, \theta)$ is said to *overlap* with another alignment point μ_g w.r.t. $(\mathcal{S}, \theta)$ if and only if there exists a nonempty intersection between the m/z intervals $[\mu_f(1-\theta), \mu_f(1+\theta)]$ and $[\mu_g(1-\theta), \mu_g(1+\theta)]$.

Let $m \stackrel{\text{def}}{=} |\mathcal{S}|$. Note that there are only two differences between the definition of alignment point (and its associated alignment sequence) and the definition of VLM point (and its associated VLM sequence). The first difference is the fact that a VLM sequence must contain exactly m peaks, whereas an alignment sequence can contain any number of peaks between 1 to m (since the peaks in an alignment sequence may originate from a molecule fragment which is not present in all the samples for which we have a spectrum in $\mathcal{S}$). Hence, if we remove the constraint that each virtual lock mass must be formed of $|\mathcal{S}|$ peaks from the validation step, Algorithm (1) then finds all the maximum-length sub-sequence of peaks that satisfy the 4 criteria for a valid alignment sequence when it reaches the overlap deletion step. The second difference is that there is no intensity threshold t applied to the peaks for alignment, as we wish to align every peak in the spectra if possible. Note that, generally, a lower intensity threshold is still applied to the peaks in order to remove peaks that are the result of background noise. Consequently, with that very minor change,

$$\textbf{virtualLockMassDetection}(\mathcal{S}, \theta)$$

finds all isolated alignment points w.r.t. $(\mathcal{S}, \theta)$ in $O(n \log m)$ time, where n is the total number of peaks in $\mathcal{S}$.

If the window size parameter θ is too large, then many alignment points will overlap and Algorithm (1) will return very few isolated alignment points. If θ is very very small, then, in contrast with the VLM identification case, Algorithm (1) will return a very large number of isolated alignment points associated to aligned sequences that contain only one point. Hence, in contrast with the VLM identification case, the best parameter θ is not the one for which we obtain the largest number of alignment points. What should then be the choice for θ? To answer this question, we consider each VLM point (and its associated sequence of peaks) found by Algorithm (1). If we leave out one VLM point v_i from the correction algorithm (2) and use this algorithm to correct all the m/z values of the peaks associated to this VLM point, the maximum deviation from v_i among these m/z values will give us the smallest window size θ_i such that each m/z value will be located within $[v_i(1-\theta_i), v_i(1+\theta_i)]$. Essentially, this window size θ_i is the smallest one for which we can still recognize all the peaks associated to the same VLM v_i. It would then certainly be a very good choice for θ in that region of m/z values. We can then repeat this procedure for all isolated VLM points (except the VLMs with the smallest and largest m/z values) found by Algorithm (1) to obtain a sequence of θ_i values. One interesting possibility for θ is the maximum among the θ_i values. However, this is clearly an overestimate of the maximum spreading of peaks associated to the same molecule fragment since all the VLMs will be used for the correction, including the one that was left out. Moreover, as we can see in Figure (4), we can recover a large fraction of the non-overlapping VLMs if we use a significantly smaller window size than the $\max_i \theta_i$. For that reason, we have decided to use, for the window size θ, the smallest value covering 95% of the non-overlapping VLMs, i.e., the 95th percentile. Alternatively, to attempt to maximize the accuracy of a learning algorithm, a percentile z can be selected by cross-validation along with the selection of the hyperparameters of the learning algorithm.

If we have r VLM points, each θ_i associated to the ith VLM point is found in $O(m)$ time for a sequence $\mathcal{S}$ of m spectra; thus implying a running time in $O(mr)$ to find every θ_i. Then, the 95th percentile is found by sorting the vector of θ_is in $O(r \log r)$ time. Assuming that we always have $\log(r) < m$, the total running time to find θ is in $O(mr)$, and hence in $O(n)$ when $\mathcal{S}$ contains a total of n peaks.

Once the window size θ is found, we can then run Algorithm (1) just once on the full

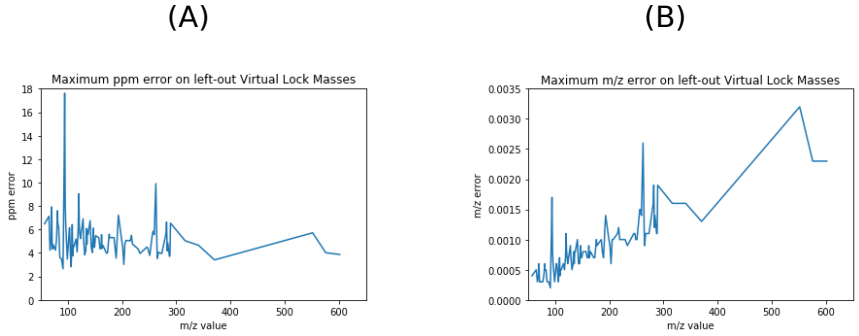

Fig 4. Error in ppm versus mass units. Subfigure (A) shows the error on left-out VLMs in ppms, while Subfigure (B) shows the error in Daltons.

sequence $\mathcal{S}$ of spectra with that value of θ in $O(n \log m)$ time. Consequently, the total running time of the alignment algorithm, which includes the running time to find θ and to find all non overlapping alignment points w.r.t. $(\mathcal{S}, \theta)$, is in $O(n \log m)$.

Once we have the VLM points and the alignment points, these are used to provide a *representation* of the spectra which is well suited for running machine learning algorithms on them. Indeed, consider Fig (5). For any new spectrum S, the VLM points are first used correct the m/z value of each peak of S and, following that, the intensity of any corrected peak that fall into the window associated to an alignment point give a feature of S. Hence, the vector of these intensities provides a new representation of the spectrum S that we will use for the input into a classifier to predict the label of S.

Finally, it might be tempting to use a clustering approach to solve the problem of finding the isolated alignment points. After all, the alignment sequences are just clusters of peaks belonging to different spectra. However, we have to keep in mind that current trends lead to the processing of hundreds of spectra, each potentially containing thousands of peaks. The total number of peaks to be processed can thus reach a million peaks or more. In our case, the total running time of the full pipeline (finding all isolated VLMs, correcting all the mass spectra with the VLMs, and finding all the isolated alignment points) is in $O(n \log m)$. Hence, any algorithm running in $\Omega(n^2)$ time, will be completely surpassed by the proposed pipeline of algorithms. As far as we know, the running times of popular of-the-shelf clustering algorithms such as K-means

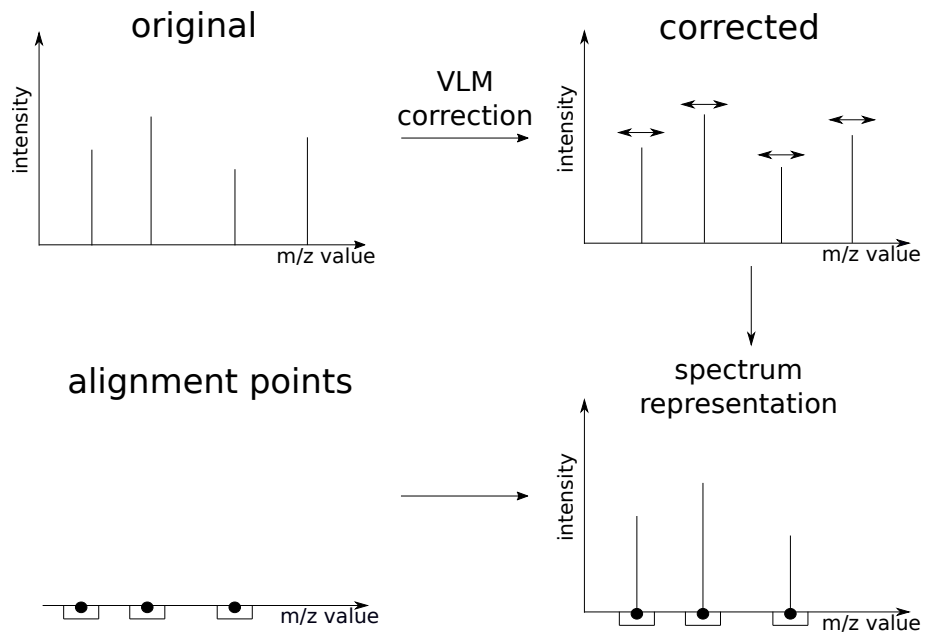

Fig 5. A representation of the original spectra is given by the intensities of the VLM-corrected peaks that fall into the windows associated to alignment points.

and linkage-based clustering algorithms all require a running time in $\Omega(n^2)$. Moreover, all the clustering algorithms that we know have at least one parameter to tune, which often includes the number of desired clusters. Hence, with the current state of knowledge, a clustering-based algorithm is bound to be substantially less efficient than the proposed pipeline of algorithms.

An implementation of the algorithms for Python is available at `https://github.com/francisbrochu/msvlm`.

Dataset descriptions

***Days* Dataset** Plasma from 20 healthy persons was equally pooled together. The pooled plasma was aliquoted and kept at -20 °C.

Two consecutive days, an acetonitrile crash was performed using 9 parts of acetonitrile (Fisher Optima) for 1 part of unfrozen plasma pool. The crash solution was centrifuged at 4000 rpm for 5 minutes. 2 μl of the solution was spotted on every well of a 96 wells Lazwell plate (Phytronix). The same experiment was repeated the next day.

***Clomiphene-Acetaminophen* Dataset** One pill of acetaminophen (500 mg) was diluted in 50 mL of methanol and water (50:50). The solution was put in a sonicating bath for 20 minutes. The resulting solution was centrifuged and diluted 1:100 in water. For clomiphene, we used a solution of 100μg/ml of clomiphene in methanol (Phytronix).

A pool of plasma was crashed as previously described. The solution was split in 3 parts. One received 10 μl of the acetaminophen solution, another 10 μl of the clomiphene solution and the last one stayed unmodified. Each type of sample was spotted 32 times.

***Malaria* Dataset** *Plasmodium falciparum* parasites were put in culture in red blood cells and tightly synchronized. Culture was performed for 28-36 hours, until parasites are in the trophozoite stage and parasitaemia reached 5-10%. In the same conditions, red blood cells were kept uninfected. Cells were diluted to 2% hematrocrit by adding the correct amount of pelleted cells to complete RPMI media. 200 μL of the cell suspensions was deposited in a 96 well plate in order to have 40 samples of infected cells and 40 samples of uninfected cells.

After 4 hours at 37 °C, the plate was spin at 800x g for 5 minutes. 180 μL of culture media was removed. Pellet was resuspended in the remaining 20 μL and 10 μL was transferred to a new 96 well plate. 100 μL of ice-cold methanol was quickly added to the plate and put on dry-ice to stop any metabolic reaction. The plate was vortexed 3 times, for 15 seconds each, over 15 minutes incubation on dry-ice. The plate then was placed for sonication in a water bath for 5 times 1 minute with 2 minutes breaks on dry-ice. Finally, the plate was centrifuged at 3200 rpm for 5 minutes at 4°C. 30μL of the supernatant was transferred to another plate and kept at -80°C until LDTD-MS analysis.

For analysis, 2 μl of the metabolomic extract was spotted on a 96 well Lazwell plate and left at room temperature until dryness.

***Cancer* Dataset** Plasma from patients diagnosed for breast cancer and from healthy patients were individually treated using the same acetonitrile crash protocol. A total of 96 samples from breast cancer patients were acquired. In addition, 96 plasma samples from healthy patients were also acquired in order to have control samples.

Data acquisition All data were acquired on a Synapt G2-Si mass spectrometer. The instrument was operated in high resolution mode. Except if stated otherwise, data acquisition was performed in positive ionization. The acquisition method was MS^e with a scan time of 0.1 second. Calibration of the instrument was performed daily before data acquisition using a solution of sodium formate 0.5 mM. The instrument was operated with Mass Lynx software. The source is a LDTD 960 ion source (Phytronix). The laser pattern used is the following: 2 seconds at 0%, ramp up to 65% in 6 seconds, hold at 65% for 2 seconds and back at 0% in 0.1 second.

Data conversion Raw files produced by the mass spectrometer were converted to ion list using a continuous to centroid approach using the ProcessKernel software (Waters Corporation) using only the first function (low energy) present in the files. The resulting centroided peak list were used for data analysis.

For all experiments presented in this article, the t threshold on intensity for virtual lock mass detection was set at 1000 counts.

Acknowledgments

We thank Water Corporation for their support and expertise that helped to the design of the proposed algorithms and for the support using their instruments. We also thank Phytronix Technologies Inc. for their support with their instruments. Computations were made on the supercomputer Colosse from Université Laval, managed by Calcul Québec and Compute Canada. The operation of this supercomputer is funded by the Canada Foundation for Innovation (CFI), the ministère de l'Économie, de la science et de l'innovation du Québec (MESI) and the Fonds de recherche du Québec - Nature et technologies (FRQ-NT). Computations were also made on the supercomputer Graham from Waterloo University, managed by Compute Canada. Finally, we thank Dr. Dave Richard for providing us with the malaria dataset samples and Dr. Francine Durocher for providing the breast cancer dataset samples.

References

1. K. Dettmer, P. A. Aronov, B. D. Hammock, Mass spectrometry-based metabolomics, Mass spectrometry reviews 26 (1) (2007) 51–78.

2. X. Han, A. Aslanian, J. R. Yates III, Mass spectrometry for proteomics, Current opinion in chemical biology 12 (5) (2008) 483–490.

3. C. Fenselau, P. A. Demirev, Characterization of intact microorganisms by maldi mass spectrometry, Mass spectrometry reviews 20 (4) (2001) 157–171.

4. R. M. Caprioli, T. B. Farmer, J. Gile, Molecular imaging of biological samples: localization of peptides and proteins using maldi-tof ms, Analytical chemistry 69 (23) (1997) 4751–4760.

5. J. Cox, M. Mann, Quantitative, high-resolution proteomics for data-driven systems biology, Annual review of biochemistry 80 (2011) 273–299.

6. D. F. Hunt, J. R. Yates, J. Shabanowitz, S. Winston, C. R. Hauer, Protein sequencing by tandem mass spectrometry, Proceedings of the National Academy of Sciences 83 (17) (1986) 6233–6237.

7. J. S. Cottrell, U. London, Probability-based protein identification by searching sequence databases using mass spectrometry data, electrophoresis 20 (18) (1999) 3551–3567.

8. A. Alonso, S. Marsal, A. Julià, Analytical methods in untargeted metabolomics: state of the art in 2015, Frontiers in bioengineering and biotechnology 3 (2015) 23.

9. W. B. Dunn, A. Erban, R. J. Weber, D. J. Creek, M. Brown, R. Breitling, T. Hankemeier, R. Goodacre, S. Neumann, J. Kopka, et al., Mass appeal: metabolite identification in mass spectrometry-focused untargeted metabolomics, Metabolomics 9 (1) (2013) 44–66.

10. A. Römpp, U. Karst, Current trends in mass spectrometry imaging (2015).

11. M.-Z. Huang, C.-H. Yuan, S.-C. Cheng, Y.-T. Cho, J. Shiea, Ambient ionization mass spectrometry, Annual review of analytical chemistry 3 (2010) 43–65.

12. O. J. Semmes, Z. Feng, B.-L. Adam, L. L. Banez, W. L. Bigbee, D. Campos, L. H. Cazares, D. W. Chan, W. E. Grizzle, E. Izbicka, et al., Evaluation of serum protein profiling by surface-enhanced laser desorption/ionization time-of-flight mass spectrometry for the detection of prostate cancer: I. assessment of platform reproducibility, Clinical Chemistry 51 (1) (2005) 102–112.

13. R. Tibshirani, T. Hastie, B. Narasimhan, S. Soltys, G. Shi, A. Koong, Q.-T. Le, Sample classification from protein mass spectrometry, by 'peak probability contrasts', Bioinformatics 20 (17) (2004) 3034–3044.

14. N. Jeffries, Algorithms for alignment of mass spectrometry proteomic data, Bioinformatics 21 (14) (2005) 3066–3073.

15. M. B. Tracy, H. Chen, D. M. Weaver, D. I. Malyarenko, M. Sasinowski, L. H. Cazares, R. R. Drake, O. J. Semmes, E. R. Tracy, W. E. Cooke, Precision enhancement of maldi-tof ms using high resolution peak detection and label-free alignment, Proteomics 8 (8) (2008) 1530–1538.

16. J. A. Barry, G. Robichaud, D. C. Muddiman, Mass recalibration of ft-icr mass spectrometry imaging data using the average frequency shift of ambient ions, Journal of the American Society for Mass Spectrometry 24 (7) (2013) 1137–1145.

17. N. Psychogios, D. D. Hau, J. Peng, A. C. Guo, R. Mandal, S. Bouatra, I. Sinelnikov, R. Krishnamurthy, R. Eisner, B. Gautam, et al., The human serum metabolome, PloS one 6 (2) (2011) e16957.

18. S. Bouatra, F. Aziat, R. Mandal, A. C. Guo, M. R. Wilson, C. Knox, T. C. Bjorndahl, R. Krishnamurthy, F. Saleem, P. Liu, et al., The human urine metabolome, PloS one 8 (9) (2013) e73076.

19. Y. Freund, R. E. Schapire, A desicion-theoretic generalization of on-line learning and an application to boosting, in: European conference on computational learning theory, Springer, 1995, pp. 23–37.

20. L. Breiman, J. Friedman, R. Olshen, C. Stone, Classification and Regression Trees, Wadsworth and Brooks, Monterey, CA, 1984, new edition [**?**]?

21. M. Marchand, J. Shawe-Taylor, The set covering machine, Journal of Machine Learning Research 3 (Dec) (2002) 723–746.

22. C. Cortes, V. Vapnik, Support-vector networks, Machine learning 20 (3) (1995) 273–297.

23. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, Journal of Machine Learning Research 12 (2011) 2825–2830.

24. J. Friedman, T. Hastie, R. Tibshirani, The elements of statistical learning, Vol. 1, Springer series in statistics New York, 2001.

25. A. Gammerman, V. Vovk, V. Vapnik, Learning by transduction, in: Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence, Morgan Kaufmann Publishers Inc., 1998, pp. 148–155.

26. A. Nordström, G. O'Maille, C. Qin, G. Siuzdak, Nonlinear data alignment for uplc- ms and hplc- ms based metabolomics: quantitative analysis of endogenous and exogenous metabolites in human serum, Analytical chemistry 78 (10) (2006) 3289–3295.