Scalable machine learning approaches for chromatographic pattern extraction in large-scale mass spectrometry data

Olga Permiakova

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ GRENOBLE ALPES

Spécialité : MBS - Modèles, méthodes et algorithmes en biologie, santé et environnement

Arrêté ministériel : 25 mai 2016

Présentée par

Olga PERMIAKOVA

Thèse dirigée par **Thomas BURGER**, CR CNRS, Université Grenoble Alpes

préparée au sein du **Laboratoire Biologie à grande echelle** dans **l'École Doctorale Ingénierie pour la Santé la Cognition et l'Environnement**

Méthodes d'apprentissage automatique pour l'extraction de motifs chromatographiques dans des gros volumes de données de spectrométrie de masse

Scalable machine learning approaches for chromatographic pattern extraction in large-scale mass spectrometry data

Thèse soutenue publiquement le **3 mai 2021**,
devant le jury composé de :

Monsieur THOMAS BURGER
CHARGE DE RECHERCHE HDR, CNRS DELEGATION ALPES, Directeur de thèse
Monsieur FREDERIC BERTRAND
PROFESSEUR DES UNIVERSITES, UNIVERSITE DE TROYES, Rapporteur
Monsieur BLAISE HANCZAR
PROFESSEUR DES UNIVERSITES, UNIVERSITE EVRY VAL D'ESSONNE, Rapporteur
Madame FLORENCE FORBES
DIRECTEUR DE RECHERCHE, INRIA CENTRE GRENOBLE-RHONE-ALPES, Présidente
Monsieur LAURENT JACOB
CHARGE DE RECHERCHE, CNRS DELEGATION RHONE AUVERGNE, Examinateur

Acknowledgements

I am deeply thankful to Prof. Frédéric Bertrand from Université de Technologie de Troyes and Prof. Blaise Hanczar from Université d'Evry who accepted to review this thesis; I would also like to thank the other members of my jury: Prof. Florence Forbes from INRIA Grenoble Rhone-Alpes and Dr. Laurent Jacob from Université Lyon 1, who both accepted to evaluate my work.

I would like to express my sincere appreciation to my supervisor, Thomas Burger, for his guidance and encouragements, without which this work would not have been possible. I am truly thankful for his ongoing support, his perseverance and for our fruitful brainstorming. During these years, he was consistent during our discussions, he was always present for me. He taught me a lot in the fields of machine learning and proteomics, but also "know-hows", ranging from result presentation and scientific writing to work-planning. I am grateful to Thomas for the opportunities he gave me to participate to conferences and workshops. I am also sincerely grateful for the fact that during all these years, he considered me as his colleague, trusting me and relying on me.

I would also like to express my deepest gratitude and respect to Thomas Fortin, for our productive and fruitful discussions. Thanks to these discussions, I improved my knowledge in linear algebra and mathematical analysis.

My appreciation also extends to all my dear colleagues. It was a great pleasure to work at EDyP team. I am thankful for their encouragements, interesting discussions, sudoku competitions during coffee breaks and wonderful moments in the mountains. At EDyP team, I have never felt myself as a foreigner and learned a lot about the french culture and

traditions. I thank my co-authors Anne-Marie Hesse and Alexandra Kraut for their help in result interpretation and for their implication in our joint publication. The same concern goes to Romain Guibert, whose involvement during his internship was a real asset. I would also like to thank Céline Fleury, EDyP team's secretary, who helped me constantly in all my administrative deals; I thank her for both her prompt replies and her patience. I would like to express my gratitude to Virginie Brun, the head of EDyP, for her encouragements and valuable recommendations.

I wish to thank my colleagues and friends Hélène Borges, Lucid Belmudes, Julia Novion Ducassou, Sabrina Ferré, Marion Crespo and Sara El Kennani. Their presence in the lab certainly brightened my everyday life. Moreover, they taught me about mass spectrometry, and some ideas blossomed during our discussions were indeed implemented in my final data process algorithms.

Finally, I am incredibly grateful to my husband, Sergey Shcherbanev, who was always there for me. I am thankful for his unwavering support, for his unfailing patience. He has never let me down. I would also like to thank my family: my mother and father, for their sincere faith in me.

Abstract

English version

Proteomic analysis consists in determining which proteins are contained in biological samples and in which quantity. Such analysis is often required in fundamental or clinical research, to find proteins differentially expressed between several conditions, a.k.a. biomarkers. Modern proteomics largely relies on analytical chemistry techniques, and notably, on mass spectrometry (MS) coupled with high-pressure liquid chromatography (LC). To increase the depth and coverage of proteomics analyses, multiplexed LC-MS acquisitions are increasingly relied on, despite the subsequent challenges in data processing. Recently, it has been shown that some of these challenges could be addressed using chromatogram libraries, which consist of elementary chromatographic profile collections corresponding to different protein fragments present in the samples. The current state-of-the-art approaches propose to construct the chromatogram library by means of additional (and costly) mass spectrometry experiments. The aim of this work is to construct it numerically, through the direct analysis of the LC-MS data using innovative machine learning approaches. Two approaches have been developed. The first one, referred to as CHICKN (Chromatogram Hierarchical Compressive K-means with Nyström approximation), proposes to cluster the observed elution profiles (defined as the columns of the matrix containing the LC-MS data) and to construct the library using the consensus chromatograms resulting from these clusters. This clustering method operates on a data sketch, as defined in the compressive learning theory. Furthermore, the algorithm is compatible with the kernel trick, which is accelerated thanks to Nyström kernel approximation. Finally, we have derived two new kernel functions, based on the Wasserstein-1 distance. We have established on real proteomics data that these kernel functions lead to better capturing the LC-MS data specificities. The second approach developed in this thesis is an online dictionary learning algorithm, referred to as SSDL (Sketched Stochastic Dictionary Learning), so as to use the trained dictionary as a chromatogram library. This method also relies on the compressive learning theory. In addition, its computational efficiency is strengthened by a stochastic version of Nesterov accelerated gradient descent method. The performance of both methods has been assessed on real LC-MS data. We demonstrated that both of them lead to the construction of meaningful chromatogram libraries, satisfying all LC-MS data requirements (notably physical interpretability). Moreover, they have small computational cost and are efficient to build extremely large chromatogram libraries, as required for complex biological samples.

French version

L'analyse protéomique consiste à déterminer les identités et quantités des protéines contenues dans des echantillons biologiques. Une telle analyse est souvent nécessaire en recherche fondamentale ou clinique, pour trouver des protéines différentiellement exprimées entre plusieurs conditions, communément appelées « biomarqueurs ». La protéomique moderne s'appuie principalement sur des techniques de chimie analytique, et notamment, sur la spectrométrie de masse (MS) couplée à la chromatographie liquide haute pression (LC). Pour augmenter la profondeur et la couverture des analyses protéomiques, le multiplexage des acquisitions est de plus en plus utilisé, malgré les défis que cela soulève ensuite lors du traitement des données. Récemment, il a été montré que certains d'entre eux pouvaient être résolus à l'aide d'une « bibliothèque de chromatogrammes », c'est-à-dire une collection de profils chromatographiques élémentaires correspondant à différents fragments de protéines présents dans les échantillons. Les approches de l'état de l'art s'appuient sur des expériences complémentaires (et coûteuses) de spectrométrie de masse pour construire cette bibliothèque de chromatogrammes. L'objectif de ce travail a donc été de s'affranchir de ces expériences et d'appliquer des méthodes d'apprentissage automatique innovantes pour construire in silico cette bibliothèque. Deux méthodes ont été développées. La première, appelée CHICKN (Chromatogram Hierarchical Compressive K-means with Nyström approximation), propose de partitionner les profils d'élution observés (définis comme les colonnes de la matrice contenant les données LC-MS) en plusieurs groupes en fonction de leur forme, puis de construire la bibliothèque en utilisant un représentant de chaque groupe. Afin d'être calculatoirement efficace, l'étape de partitionnement s'appuie sur la théorie de l'apprentissage compressif, qui permet de traiter un *sketch* des données (un résumé de taille fixe) plutôt que les données complètes. Par ailleurs, l'algorithme ainsi obtenu est compatible avec l'astuce du noyau, qui est accélérée grâce à l'approximation de Nyström. Enfin, nous avons proposé deux nouveaux noyaux à partir de la distance Wasserstein-1. Nous avons établi sur des données protéomiques réelles que ces deux noyaux permettent de mieux appréhender les spécificités des données LC-MS. La deuxième méthode développée dans cette thèse est constituée d'un algorithme d'apprentissage de dictionnaire, baptisé SSDL (Sketched Stochastic Dictionary Learning); afin d'utiliser ensuite le dictionnaire ainsi appris comme bibliothèque de chromatogrammes. Cette méthode repose également sur la théorie de l'apprentissage compressif. De plus, son efficacité computationnelle est renforcée par une version stochastique de la méthode de descente de gradient accélérée de Nesterov. Les performances des deux méthodes ont été évaluées sur des données LC-MS réelles. Nous avons démontré que les deux méthodes conduisent effectivement à la construction de bibliothèques de chromatogrammes qui satisfont toutes les exigences de données LC-MS (dont, notamment, l'interprétabilité physique). En outre, elles ont un faible coût de calcul, ce qui leur permet de construire efficacement les très grandes bibliothèques de chromatogrammes qui sont nécessaires à l'analyse d'échantillons biologiques complexes.

Foreword

The book (universe) cannot be read until we have learnt the language and become familiar with the characters in which it is written. It is written in mathematical language, and the letters are triangles, circles and other geometrical figures, without which means it is humanly impossible to comprehend a single word.

Galileo Galilei

Nowadays, it has become obvious that without mathematics and thorough algorithmic investigations, even with the largest computational capacity and the most performing instruments, many scientific problems would remain unsolved: Because of the volume of data generated by modern experimental settings, manual processing and classical statistical validation are intractable.

Few decades ago, a new field of artificial intelligence called machine learning appeared, but it has been witnessing an extreme popularity increment for only the last 8 years. Today machine learning methods allow not only to analyze newly produced massive data, but more importantly, to rely on the preexisting and ever-growing information ocean to predict new facts, or to extract hidden knowledge.

Machine learning applications are endless and limited only by developers' vivid imagination. To date, machine learning has fertilized technological developments that are already highly integrated in our lives: Customized services; Personalized medicine; Biological or physical scientific investigations; Climate change; etc. Among them, proteomics is no exception and an increasing number of machine learning methods are involved in the improvement of proteomics data analysis techniques, as notably shown in this manuscript.

To citizens as well as scientists from other domains, the unprecedented ascent of machine learning popularity may be surprising. However, this discipline is grounded on theories and methods, which thrived many decades ago in different unrelated domains of mathematics: probabilities and statistics, numerical optimization, signal processing, algorithmic and linear algebra.

This work is expected to target both the proteomics researchers, for the practical interest of the presented tools, as well as the machine learning community, for our methodological proposals. To adapt to this interdisciplinary readership, a pedagogical angle has been used whenever possible. I hope machine learning experts will as pleasingly discover proteomics and the kingdom of proteins; as biologists and analytical chemists will unveil an application to their former mathematical education.

Contents

CHAPTER 1

Introduction

1.1 Overview

Proteomics analysis aims at identifying and quantifying proteins (*i.e.* life building blocks) in a biological sample. While this discipline has long been at the center of biological and medical investigation, it has also heavily relied on analytical chemistry (the branch of science dedicated to analyzing samples, as opposed their synthesis). Over the last decades, analytical chemistry has made tremendous progresses, making it possible to orchestrate multiple instruments in a high-throughput workflow, where thousands of compounds are analyzed by the hour. Although this instrumentation has unleash proteomics analysis capabilities in terms of sensitivity, robustness and application domain diversity, it has brought new challenges; as for the first time, producing experimental data has become easier than storing, processing and interpreting them. This big data breakthrough has made proteomics even more interdisciplinary: it is now a research field at the crossing of biology, analytical chemistry, computer science and mathematics. Notably, developing new methodologies inspired by the recent trends of artificial intelligence with the objective to extract in an automated fashion meaningful knowledge from

this massive data has become an active research field.

The work presented in this document focuses on this research direction. Over the last three years we have tried to leverage the smoothness of some time-varying biochemical measurements to: (1) Improve the clustering of various signals according to the biological molecules that originated them; (2) Learn libraries of patterns that will be instrumental in the future to better exploit more complex experiments (due to their noise level, or to the multiplexing strategy used to increase their coverage). This work has led to two research articles, either accepted or still under review, that constitute the two main chapters of this thesis. In addition, peripheral works were also valorized: Notably, a letter format article addressing a homonymous definition between applied mathematics and proteomics (available in Appendix C); and a research article in biology involving data processing and analysis.

This introductory chapter is structured as follows: Section 1.2 focuses on the proteomics context: it introduces the basic proteomics terminology, explains the role of proteins, and presents the goals, challenges, and applications of proteomics. Section 1.3 contains descriptions about analytical chemistry tools used for proteomics analysis and discusses the advantages and drawbacks of classical data acquisition workflows. Section 1.4 describes bioinformatics tools for data analysis (protein identification and quantification, as well as validation approaches). Section 1.5 is dedicated to the machine learning context: It provides general machine learning notions and concepts; but also an overview of more specific approaches, on which our work has thrived. Finally, Section 1.6 formulates and formalizes the problems that are addressed in this manuscript.

1.2 Object and goals of proteomics analysis

This section summarizes a large background in molecular biology. Interested readers can read the following references [1–5] which we relied

on to gather the material assembled and presented in the following sections.

1.2.1 The kingdom of proteins

Proteins are complex molecular compounds, which play crucial roles in all processes of living organisms. For instance, enzymes are a type of proteins, which catalyze the biochemical reactions in cells[1]. Other proteins protect our body from physical and chemical damages; as for example, collagen and keratin, which form muscles, bones and skin; or thrombin, which performs blood clotting; or liver enzymes, which carry out the detoxification. The immune system uses proteins (antibodies a.k.a. immunoglobulins) to neutralize pathogens, such as viruses or bacteria. Proteins also transport and store atoms or small molecules: ferritin stores iron in liver; hemoglobin transfers oxygen. Messenger proteins transmit signals between different cells to coordinate biological processes; such as insulin, which regulates the metabolism of carbohydrates in the organism. Regulatory proteins, such as transcription factor proteins, control the gene expression by activating or deactivating their transcription.

Whatever its functions, a protein is essentially a chain of amino acids, linked together through so-called "peptide bounds", as illustrated in Figure 1.1. Amino acids are often called protein building blocks, and the list of amino acids composing a protein is referred to as its *primary structure*. Any amino acid is composed of an amino group, a carboxyl group and a side chain (referred to as its R group), which is specific to each amino acid and which defines its chemical properties.

A great number of different amino acids can be defined (depending on the R group). However, 22 of them are sufficient to define all the proteins of living organisms. Thus, proteins can be understood as chemical words written in an alphabet of amino acids. Assembling specific amino acids

[1]Notably, trypsin is an enzyme which cleaves other proteins into pieces, making it an extremely interesting analytical chemistry tool

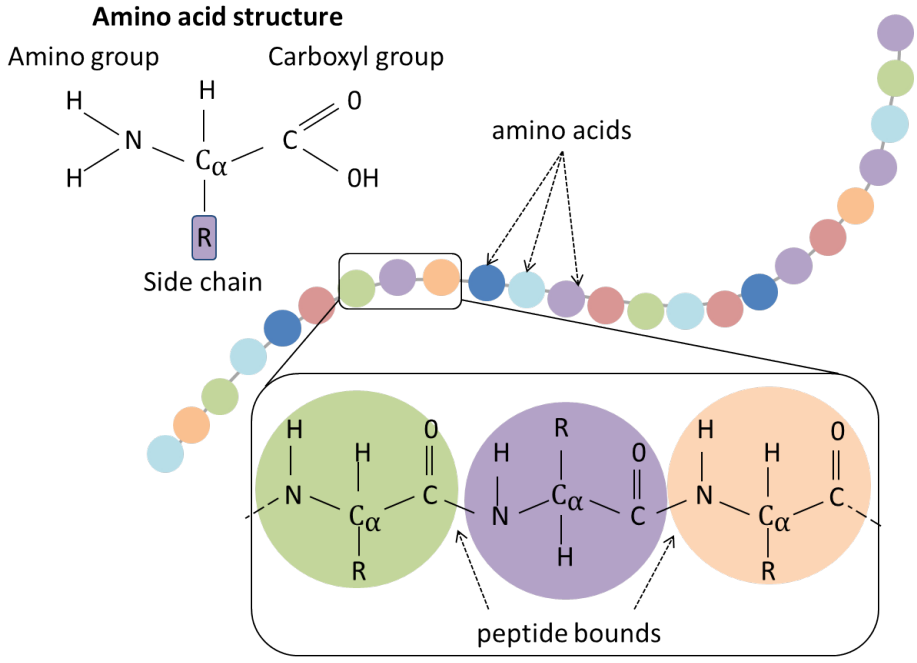


Figure 1.1: The primary structure of a protein. The amino acids are linked together into a chain by the peptide bounds, as depicted in the zoom plot. The general structure of an amino acid is presented in the left upper corner. The alpha carbon connects together amino, carboxyl and R (side chain) groups. This figure has been inspired by the materials found at https://www.technologynetworks.com/applied-sciences/articles/essential-amino-acids-chart-abbreviations-and-structure-324357.

in the correct order to obtain well-formed proteins is possible thanks to the genetic code. By analogy with the term *genome* (the entire genetic material of an organism), one has defined the *proteome*. However, the gene-to-protein translation being rather complex, the genome and the proteome do not exactly mirror one another. As a result, two proteome definitions coexist: It can either correspond to a set of proteins expressed by a genome; or a set of proteins, expressed in a particular cell or tissue at a given time under given condition. In this document, we will rely on the second one, which is more adapted to the analytical approach underlying this work.

1.2.2 Proteomics as a discipline

Proteomics is a field of molecular biology, which studies proteins. In the lab hosting my PhD studies (EDyP[2]), the main objective is to understand how they change over time or under different conditions. In this context, one seeks to identify and quantify the maximum amount of proteins among those present in a given sample. However, proteomics can also encompass larger studies, about *e.g.* protein structure, their functions, protein-protein interactions, or the discovery of new proteins. In order to avoid any confusion in the terminology, it should be noticed that the term *discovery proteomics* is generally not related to the discovery of new proteins (which nowadays are scarce), but to the large-scale and high-throughput identification (and possibly quantification) of proteins in a biological sample (*i.e.* with the objective to discover, or unveil, the sample content). In other words, EDyP essentially focuses on discovery proteomics, and consequently, my PhD project has thrived on this discipline.

Many factors make proteomics challenging: First, the proteome varies depending on cell type, time and environmental conditions. Second *post-translational modifications* (PTMs) make protein identification difficult: PTMs are chemical compounds added to one or several amino acids of the protein. PTMs occur at any time of the protein life cycle (after its biosynthesis) and can significantly change its mass, structure and function. The chemical compounds involved in PTMs are manifold, but some of them are extremely frequent: phosphoryl group (the corresponding PTM is termed phosphorylation), a methyl group (leading to methylation), a carbohydrate molecule (glycosylation). Finally, proteins may interact between each other, leading to complexes that are more difficult to analyze.

Proteomics has many applications, but most of them can be classified into two main categories: medical applications and fundamental researches in biology.

[2] Exploring the Dynamics of Proteomes

In medical researches, proteomics is an essential tool of biomarker discovery, *i.e.* of the identification of proteins which presence or concentration can be the marker of a disease or of its severity.

In fundamental researches, proteomics is essentially a tool that biologists rely on to have a comprehensive picture of the protein expression landscape in the context of their experiments.

1.3 Analytical chemistry tools and workflows for proteomics

While proteomics is of utmost importance for biomedical investigations, performing a modern proteomic analysis requires analytical chemistry tools, for it involves a series of specific high-throughput instruments. Broadly, an analysis relies on three steps: (1) the sample preparation, which most important step is to digest proteins into smaller molecules; (2) the elution of the sample by means of liquid chromatography (LC); and (3) the analysis itself, by means of tandem mass spectrometry (MS/MS). Figure 1.2 schematically depicts LC-MS/MS pipeline.

This section does not provide an exhaustive description of all existing analytical chemistry approaches and workflows used in the proteomics analysis, but on the contrary, focuses on the very pipeline used in our project. More details about mass spectrometers and the analytical chemistry techniques for proteomics can be found in [6–14].

1.3.1 LC-MS/MS pipeline

1.3.1.1 Digestion

The protein digestion is an enzymatic cleavage of proteins into shorter sub-sequences referred to as *peptides*. Generally, the digestion is performed using trypsin. This enzyme cuts proteins after any lysine or arginine: The frequency of both amino acids is such that many resulting

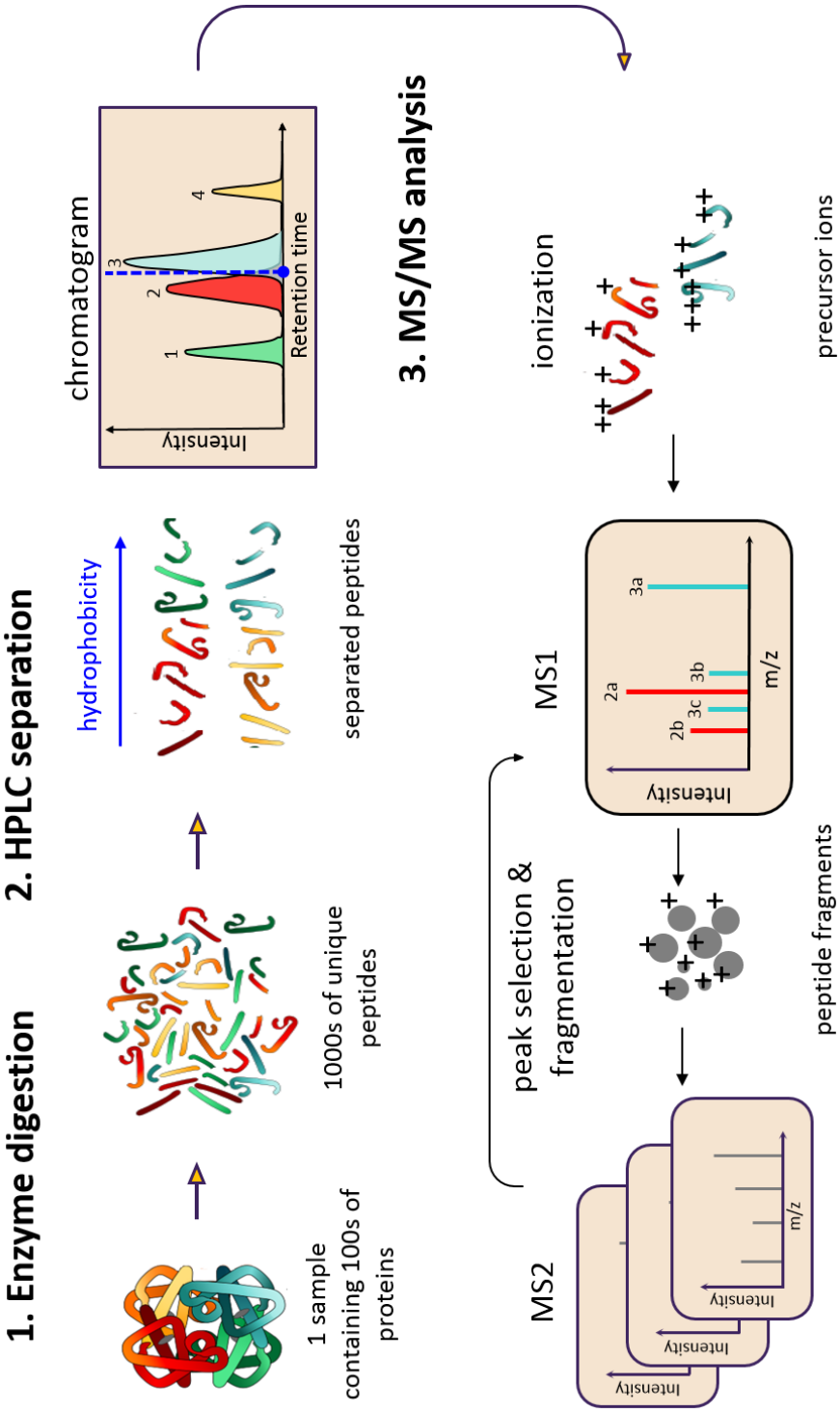


Figure 1.2: Illustration of the LC-MS/MS pipeline. The proteomics analysis is achieved in three steps: protein digestion, peptide separation by liquid chromatography and MS/MS analysis. Only a single cycle of the mass spectrometry analysis is detailed, more precisely the one which is carried out at the retention time depicted with a dotted blue line. At this time stamp, two peptides co-elute (peptides 2 and 3, depicted with blue and red colors). During the MS/MS analysis, these peptides are first ionized, then the m/z values of their ions (labeled as 2a, 2b and 3a, 3b, 3c respectively) are recorded in the MS1 spectrum. Finally, after the peak selection and fragmentation of some of these ions, their MS2 spectra are sequentially acquired.

peptides have an optimal size for mass spectrometry analysis (6 to 30 amino acids). However, for some specific experiments, other enzymes than trypsin can be used.

A proteomics workflow starting with a protein digestion step is called *bottom-up*. Relying on a digestion step makes the mixture even more complex (as each protein is replaced by several peptides). However working on peptides instead of intact proteins (referred to as *top-down* proteomics), gives considerable advantages: broadly speaking, as peptides are simpler molecules, they are easier and faster to analyze, which largely balance the sample complexity increment.

1.3.1.2 Peptide separation by Liquid chromatography

Considering the number of peptides classically present in a complex biological sample, it is not possible to analyze them simultaneously with mass spectrometry. This is why, *liquid chromatography* is used to separate peptides and to serialize them before analysis. Liquid chromatography exploits hydrophobicity, *i.e.* the peptide propensity to avoid contact with water molecules. The device encompasses a high-pressure pump, which pushes a mixture of water and organic solvents (*mobile phase*) containing the sample of interest through a column, filled with a solid adsorbent material, referred to as a *stationary phase*. The separation is achieved by gradually changing the water / solvents proportion in the mobile phase (this varying proportion being called a *gradient*). That leads to different peptide interactions with the stationary and mobile phases. Concretely, when traversing the column, peptides are adsorbed by the hydrophobic stationary phase. With an important water proportion, lesser hydrophobic peptides detach and move with the mobile phase. Conversely, the gradual increase of the solvent concentration results in the progressive detaching of more hydrophobic peptides, until the mobile phase hydrophobicity exceeds the one of stationary phase.

The time taken by a peptide to pass through the chromatography column is referred to as its *retention time*. A curve that depicts the

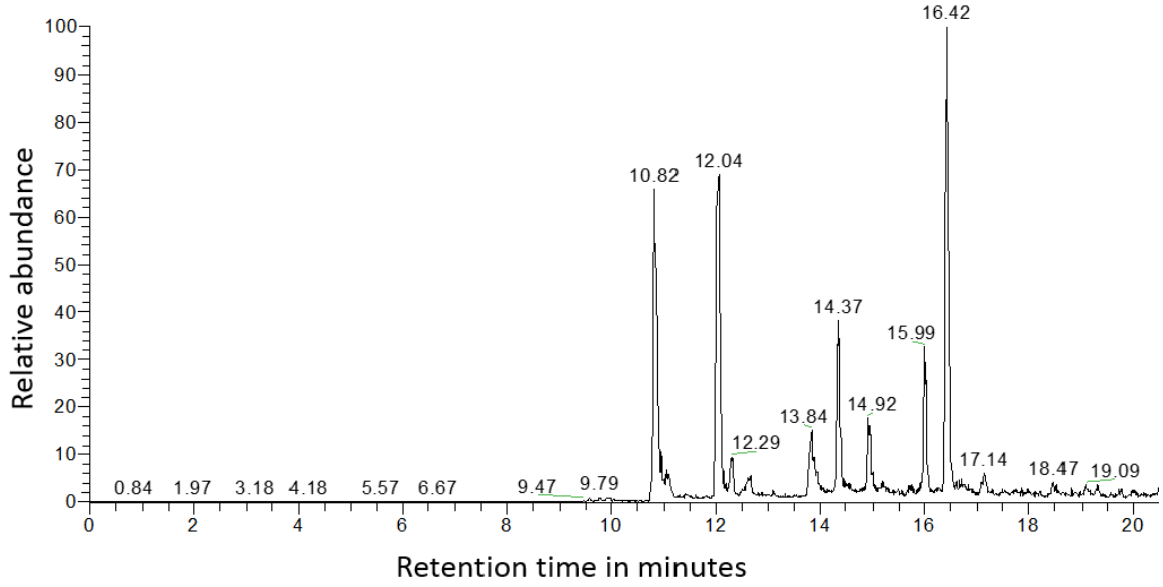


Figure 1.3: Example of a real chromatographic signal acquired along a 20 minutes gradient. Each annotated peak corresponds to an individual chromatographic profile.

molecular flow of a given peptide at any retention time is called a *chromatogram* or a *chromatographic profile*. A good liquid chromatography must prevent the chromatographic profile of different peptides to overlap. A real chromatogram is supposed to form a narrow, Gaussian-like, but slightly asymmetric curve, as the one depicted in Figure 1.3. Different peptides pass through the column with different flow rates, so their chromatographic profiles have different shapes. The chromatographic profile uniqueness is the fundamental property, on which this project relies.

1.3.1.3 Mass spectrometry analysis

Mass spectrometry is an advanced analytical chemistry technology, which is used for measuring the masses of molecules presented in a sample, thereby determining the sample composition. It can be used to analyze other type of substances than protein mixtures, yet, for each analysis, an appropriate spectrometer must be use (notably proteomics mass spectrometry analysis has its own particularities, as detailed in Section 1.3.2.1)

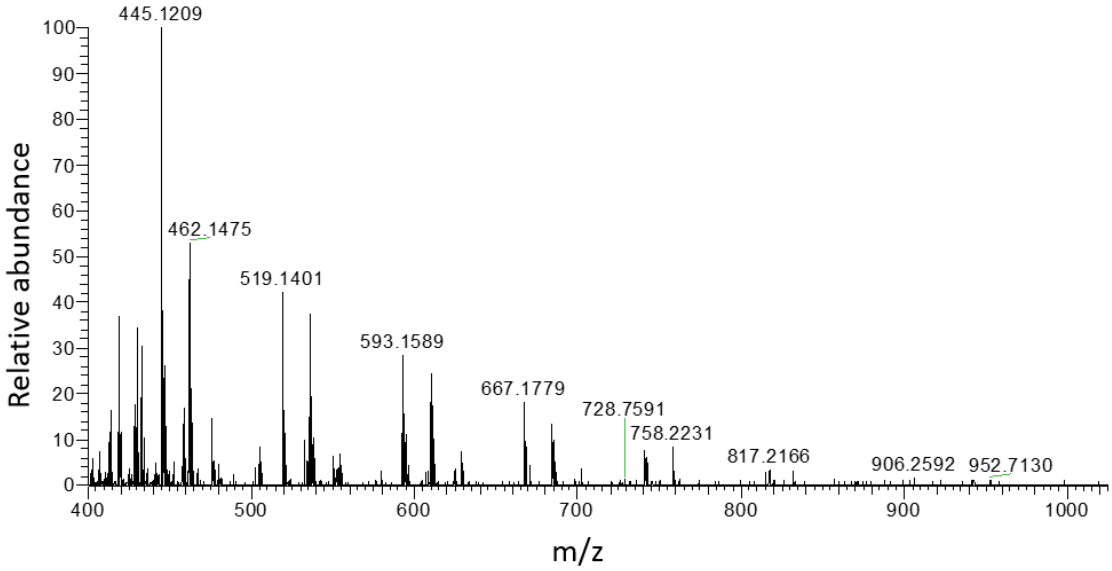


Figure 1.4: Examples of a mass spectrum; the annotated values represent the m/z values of as many precursor ions.

There are many different types of mass spectrometers, but all of them have the same objective (mass measurement), and broadly the same structure. The classical structure of a mass spectrometer is the following: an ionization source, a mass analyzer and a mass detector. The ionization source aims at converting the sample components (in our case peptides) into ions (positively charged molecules) by the addition of protons to the molecules. It is a necessary step, because only a mass to charge ratio (m/z) can be measured by a mass spectrometer. After ionization, ionized peptides are transferred to a mass analyzer, which measures their m/z values and separates them according to these values. Finally, the mass detector registers the number of ions at each m/z value and records a *mass spectrum*, that is a diagram which has ion m/z values on the x-axis and the corresponding ion intensity on the y-axis (see Figure 1.4).

A Q Exactive HF mass spectrometer was used to generate the data on which this work relies. This instrument is equipped with an electrospray ionization source (evaporation ionization technique)[3] and with a mass

[3]The dissolved sample is sprayed from a heated needle into an electromagnetic field. The progressive evaporation of the solvent from the created droplets results in the production of charged peptides

analyzer called Orbitrap. Orbitrap mass analyzer traps ions between two electrodes: a central in the form of a spindle and an outer barrel-like. Ions turn around the central electrode with circular trajectories, whose axial frequencies depend on the ion m/z only. Then, m/z values are computed from frequencies using Fourier transforms.

1.3.2 Comparative description of the classical acquisition modes

1.3.2.1 Fragmentation and Tandem mass spectrometry

Ionized peptides are too complex molecules to have their identity recovered by mere m/z measurements. Two different peptides can have identical masses, when they are composed of the same, but differently ordered amino acids. To cope for this, mass spectrometers are used in tandem mode: First, ionized peptides (a.k.a. *precursor ions*) have their m/z measured (this is the MS1 mode; MS standing for Mass Spectrometer). Then peptides are fragmented. Finally, the peptide fragments have their m/z measured (MS2 or MS/MS mode).

Concretely, peptide fragmentation is performed as following: A mass analyzer (it can be the same analyzer, which measures the masses or an additional one) isolates precursor ions, which m/z values lay in a selected range. These ions are injected in the collision chamber, where they react with an inert gas. This reaction causes the destruction of the peptide bounds along the amino acid chain leading to precursor fragmentation. Only fragments carrying the charge are detected. However, as different molecules of the same peptide are cut at different peptide bounds, it should in theory leads to all possible fragments (in practice the majority of fragments is presented).

As the peptides from sample progressively reach the mass spectrometer, based on their specific retention time, it is necessary to alternate between the MS1 mode, which globally provides the m/z value of all the attending precursors, and the MS2 mode, where fragmentation patterns

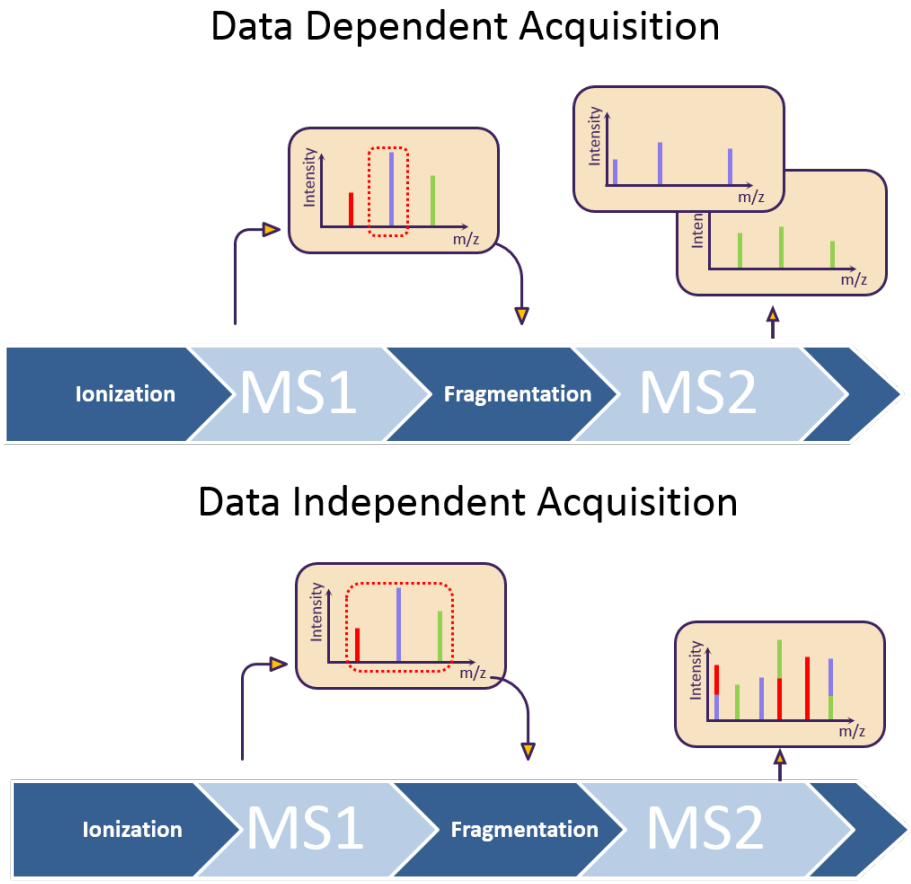


Figure 1.5: Comparison of DDA (top figure) and DIA (bottom figure) modes. In the DDA mode, peptides are selected and fragmented one after the other. The peptide to spectrum correspondence is preserved, but only the MS2 spectra of the most abundant peptides (depicted as blue and green bars) are produced. In the DIA mode, all detected peptides are selected and fragmented simultaneously. Thus, the resulting MS2 spectrum contains peaks resulting from all the peptides, including the least abundant ones (red one). As a side effect, the peaks from different peptides, but with equal m/z values are summed up, making the peptide identification non-trivial.

are registered. The two principal acquisition methods basically differ according to MS1/MS2 pattern the instrument cycle on, see Figure 1.5.

1.3.2.2 Data Dependent Acquisition

In the Data Dependent Acquisition (DDA) method, each MS1 scan is used to measure the m/z value of all the precursor ions that are concomitantly present in the instrument. Then, a fixed number of precursors (usually, the 20 most abundant ones) are selected and fragmented, one after the other, leading to as many fragmentation spectra

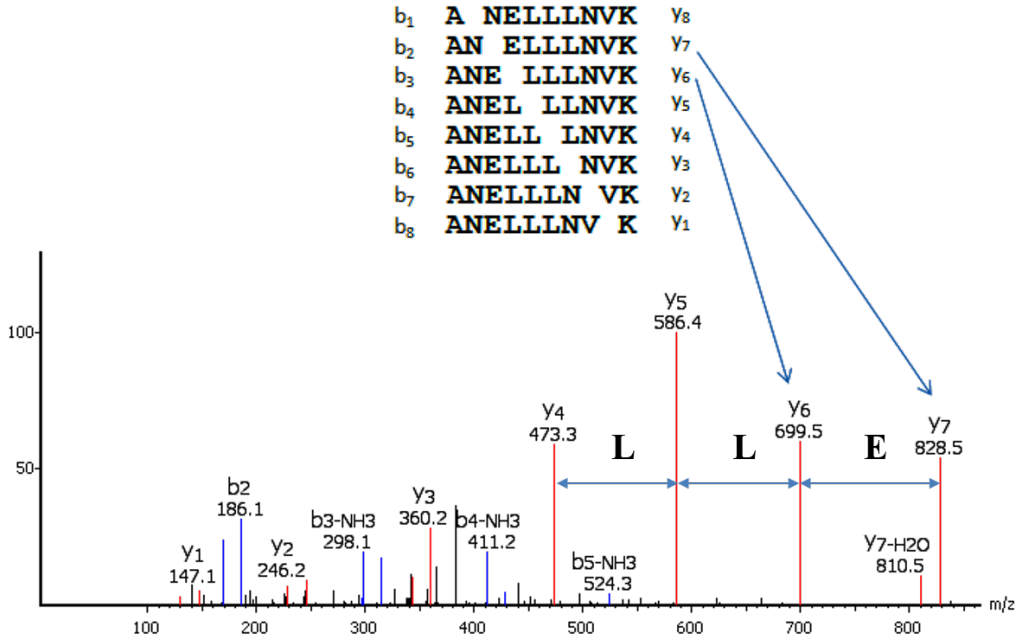


Figure 1.6: Manual interpretation of the MS2 spectrum resulting from the ANELLLNVK peptide. The peptide sequence is deciphered by computing the mass difference between consecutive y-type fragments. The b-type and y-type fragments are highlighted by blue and red colors, respectively. The list of all theoretical b-type and y-type fragments is provided above the annotated mass spectrum. It can be observed that b_1, b_6, b_7, b_8, y_8 fragments are not identified in the mass spectrum. The example is taken from https://www.bioinfor.com/denovo-tutorial/

(or MS2). As each fragmentation spectrum only contains the peaks of a single precursor, it is possible to identify its constituting amino acid chain, leading to the peptide identification: indeed we can manually find which peak in the MS2 spectrum correspond to which fragment; and the difference in masses between fragment peaks amounts to the mass of amino acids. An example of the manual interpretation of the MS2 spectrum is illustrated in Figure 1.6. The fragments are annotated as b_n (if the charge was located on the left side of the peptide bond break) or y_n (if on the right), where the subscript n indicates the number of amino acids in the fragment. The amino acid identities are determined by computing the mass differences between consecutive y-type or b-type fragments.

As a result, with DDA, peptides can be directly identified from MS2 spectra, and numerous tools exist to automatize such task (see

Section 1.4.1). It is important to understand that the number of MS2 spectra between two MS1 spectra is fixed to guarantee a short enough cycle: if too many MS2 spectra were acquired, the peptides eluting next would be missed. As a side effect, if too many peptides reach simultaneously the instrument, some of the resulting precursors will not be fragmented. Thus, not all detectable peptides can be analyzed, leading to a loss of peptide coverage. Despite this drawback DDA is the most classical acquisition mode and it is mostly used at EDyP for regular proteomics analysis. However, it has recently been challenged by an alternative acquisition mode, referred to as DIA (Data Independent Acquisition).

1.3.2.3 Data Independent Acquisition

DIA was proposed to overcome the main disadvantage of DDA, the limitation of the number of fragmented peptides. To do so, instead of selecting a single peptide for fragmentation, all precursors within a given m/z range are co-fragmented. Therefore, a given MS2 spectrum contains fragments from all different co-fragmented precursors. No peptide is lost, however, as a drawback, the direct links between an individual peptide and a MS2 scan are lost and consecutive peak differences does not related anymore to amino acid masses, because the peaks can potentially correspond to fragments from different peptides. Therefore, alternative and more elaborated strategies are necessary to exploit data resulting from DIA analyses.

1.4 MS data analysis tools and methods

There exists a great variety of bioinformatics tools to process the data resulting from proteomics analyses. For sakes of brevity, this section is not exhaustive and only focuses on the most popular ones; as well as the ones that are most frequently used in our laboratory. The section follows the classical processing chronology: peptide identification,

protein inference and peptide/protein quantification.

1.4.1 Peptide identification in DDA

This step aims at assigning peptide sequences to the acquired MS2 spectra. The most commonly used strategy is to perform a so-called *database search*. It consists in comparing the acquired spectra with those from a reference protein database. In practice, this database is derived from a genomic database of the organism(s) which peptides are potentially in the sample, by applying the gene-to-protein translation rules as well as an *in-silico* simulation of the digestion and fragmentation processes. The database search provides, for each spectrum, a list of possible peptide sequences, ranked by decreasing similarity scores.

There exist many searching engines [15–21]. They principally differ in their scoring function, as well as, to some extent, in the way they account for PTMs. We dwell upon three widely used ones: Mascot [15] (the tool generally used in our laboratory), X!Tandem [16] and SEQUEST [17].

Mascot score first computes the probability that the observed peaks match by chance those of the theoretical spectra. Then, this probability of a random peptide match, denoted p, is converted into a peptide-spectrum match (PSM) score reading:

$$S = -10 \log_{10}(p). \tag{1.1}$$

X!Tandem score calculates the dot product between the theoretical (T) and experimental (E) spectra: $S(T, E) = \sum_{i=1}^{N} I_i \cdot \delta(T, E_i)$, where I_i is the fragment ion intensity and $\delta(T, E_i) \in \{0, 1\}$ indicates whether an experimental peak E_i is presented or not in the theoretical spectrum.

SEQUEST relies on two scores: the preliminary score is used to reduce the set of peptide candidates and the second to evaluate the similarity between spectra. The preliminary score (S_p) is the weighted peak intensity sum, where weights take into account the peak continuity and the peptide length. The main SEQUEST score, $Xcorr$, amounts to the

cross-correlation between the theoretical spectrum and the normalized experimental spectrum.

1.4.2 Peptide identification from DIA data

In DIA data, direct database search cannot be conducted because of the simultaneous fragmentations. Methods for peptide identification follow different strategies that can be classified into library-based and library-free approaches.

The library-based tools, such as Skyline [22], OpenSWATH [23], Spectronaut [24], DIA-NN [25], use a comparison with a spectral library to identify peptides. A spectral library is generated using several DDA analyses of related samples, where as many fragmentations as possible are considered, and the precursor identifications from the DIA experiments are guessed using those of the library which have compatible m/z and retention time. The main limitation of the library-based approaches is its strong dependence on the preliminary DDA experiments and their subsequent peptide identification results. Only peptides, presented in the library, can be detected, making the approach only useful to streamline a large number of very precise quantitative analyses of similar samples (with known peptides).

The principle of library-free DIA approaches consists in first demultiplexing the measured MS2 spectra into pseudo-DDA spectra (*i.e.* restoring links peptide - its fragment ions). A pseudo-DDA spectrum is a computationally constructed fragmentation spectrum, where all the peaks derived from a same precursor have been grouped, to the exclusion of peaks from other precursors. Having pseudo-DDA spectra makes it possible to on standard DDA database search engines to perform peptide identification. The spectrum-demultiplexing problem is challenging, as for combinatorial reasons, there is a huge number of ways to cluster a DIA spectrum into a set of pseudo-DDA spectra. In many library-free approaches, such as DIA-Umpire [26] and Group-DIA [27],this is tackled by relying on the fact that the chromatograms of a precursor and its

fragment ions must have the same shapes and their chromatographic peaks must be aligned (*i.e.* they must coelute; *i.e.* they have similar retention time). Concretely, these tools compute the correlation between the fragment and peptide chromatograms and construct groups accordingly to build pseudo-DDA spectra. Another tool called Specter [28] is based on the hypothesis that a multiplexed DIA spectrum is a linear combination of DDA spectra. It looks over DDA spectra of some spectral library to find which of them can best reconstruct the measured DIA spectrum. It uses also several chromatographic peak scores to measure the peptide identification quality. Since this approach employs spectral libraries, it is not considered as a purely library-free tool, but an in-between approach. Finally, PECAN [29] directly compares the multiplexed MS2 spectra with the predicted fragmentation patterns. It does not use any spectral library, but it requires a prejudiced list of peptides.

1.4.3 Identification validation and protein inference

As proteins are the biologically meaningful entities, peptide identification is not sufficient: It is necessary to infer the identities of the sample's proteins from the peptide identities. The protein inference task is made difficult by (1) the presence of shared peptides (*i.e.* peptides that cannot be attributed to a unique protein; for evolutionary reasons, many related proteins indeed share homologue subsequences); and by (2) the fact that some protein-specific subsequences may not be detectable by LC-MS/MS (inadequate peptide size, poorly ionizable peptides, etc.). To cope for the resulting ambiguity, the principle of parsimony is often applied: one reports the smallest set of proteins that can explain the set of identified peptides. However, other strategies exist and there is still no consensus on the best way to conduct protein inference [30–32]. Often, a subset of identified peptides confidently points toward several homologous proteins, among which refinement is not possible, so that equivalence classes are defined (and termed *protein groups*). Mainly

tools are available to perform protein inference. Some of them also propose to aggregate the peptide scores into protein scores, so that a confidence measure can also accompany the protein level results: INTERACT [33], DTAselect [34] and CHOMPER [35] simply provide the list of the identified proteins, without computing any confidence protein identification measure. However, Qscore [36] computes protein confidence level on top of the SEQUEST scores (see Section 1.4.1). The tool called ProteinProphet [31] uses a Bayes rule–based scoring scheme to estimate the probability that a protein is presented in a sample relying on the list of the identified peptides.

To avoid corrupting the biological conclusions with unreliably identified proteins, stringent validation rules are usually applied, either at peptide or protein level, possibly both. The most mainstream approach is to compute a false discovery rate at peptide spectrum matching (PSM) level. The *false discovery rate* (FDR) [37] is defined as a conservative estimate of the expected ratio between the number mismatches and the total number of PSMs that have an identification score larger than a user-defined threshold (or conversely, the threshold is adjusted to meet a specific FDR, such as for instance, 1%). In this approach, the most difficult part is to confidently estimate the likely number of mismatches. To cope for this, the most classical approach is the target-decoy one. Its purpose is to artificially generate mismatches by using a protein sequence database that has been shuffled or reversed, hereby containing non-existing peptides (termed "decoys", as opposed to the real "target" ones). The seminal assumption of target-decoy approaches is that the probability of reporting an incorrect target identification is the same as that of reporting a decoy one. Thus, the number of PSMs mapping to the decoy database serves as an estimate of the number of false positive matches. However, it has recently been demonstrated that this seminal assumption may not always hold, notably with newer more-resolutive mass spectrometers, so that more traditional approaches to FDR (such as Benjamini-Hochberg ones) should be used instead [38].

The protein-level validation is even more challenging, since the error in the peptide identifications may propagate to protein ones. In line with the peptide identification validation, target-decoy strategies can be also used to estimate protein identification false discovery rates (see for instance [39]), however, this remains an active research field which moving state-of-the-art is of secondary importance for the work presented here.

1.4.4 Quantification

Quantification is the process of determining the amount, or the abundance, of a particular peptide or protein in a sample. The existing quantitative techniques can be divided in two categories: label-free and label-based. In the label-based quantification, peptides are chemically bound with some compounds that allows to calculating their abundances, such as stable isotopes (SILAC [40], ICATs [41] or an isobaric tag labeling (TMT [42], iTRAQ [43]). However, label-free approaches are more suited to discovery proteomics, as labelling becomes laborious and time-consuming with the sample complexity increment. In addition, labelling requires expensive reagents. On the other hand, a main drawback of label-free approaches is restriction to relative quantification: it can only be used to compute variations of peptide abundances across different samples, and not to derive the exact peptide concentrations in each sample. However, let us note recent attempt of interest for approaching absolute quantification within a label-free setting [44].

The most known label-free quantification tools are Mascot Distiller (http://www.matrixscience.com), Viper [45], MaxQuant [46]. However, at EDyP, we use an in-house software, Proline [47] to perform quantification. All these software tools broadly follow the same strategy to estimate peptide abundances, that is to integrate the area under the peptide chromatogram.

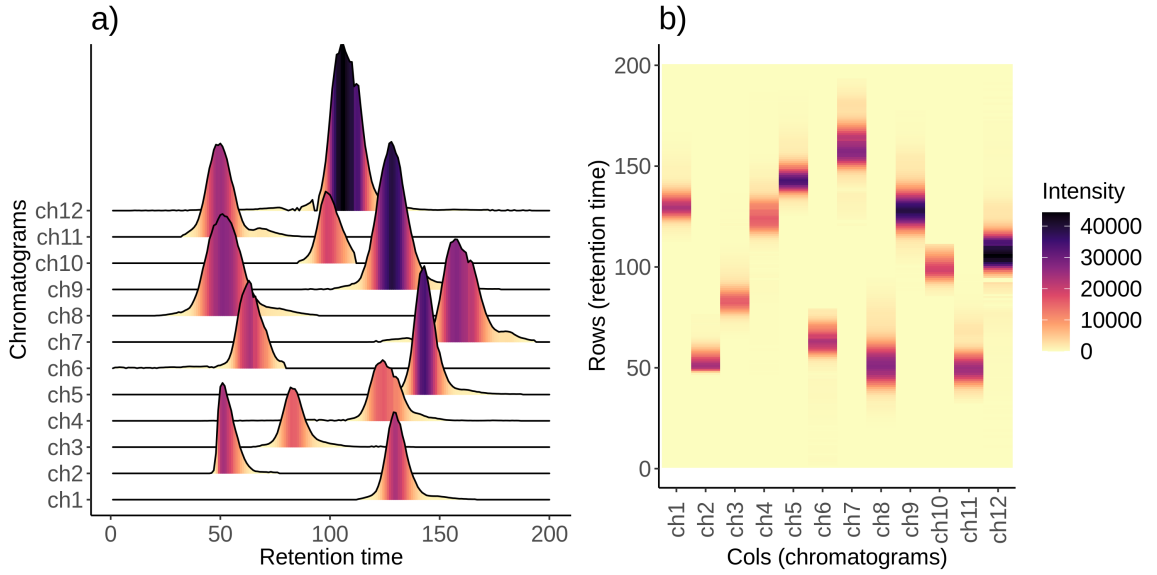


Figure 1.7: (a) 12 discretized chromatographic profiles. (b) A data matrix, constructed by storing these chromatographic profiles in the matrix columns. Matrix rows correspond to the discrete retention time stamps.

1.5 Machine learning approaches

1.5.1 Background notions

1.5.1.1 Machine learning model, loss function and empirical risk minimization

Let us consider a *data collection*, hereafter denoted as $X = \{x_1, \ldots, x_N\}$. A single data instance x_i is usually represented as a vector in $\mathbb{R}^n$ and it is called an *observation*. In the case of LC-MS data, X corresponds to a collection of discretized chromatographic profiles, which form a data matrix with chromatograms in columns, which is denoted as well as X. An example of such matrix is presented in Figure 1.7.

Machine learning aims at exploring an observed data collection through the viewpoint of a user-selected parametrized function, referred to as a *machine learning model*, and to use it for a variety of tasks; Such as for instance performing predictions about unseen (future) data, or extracting hidden structures from the current data to improve decision making. The specificity of machine learning is that the model is not

fully specified by the expert. An algorithm is used to find the parameter values that best fit the observed data: Doing so is referred to as *learning* or *training*. To quantitatively estimate which combination of parameter values yields the best data description, one measures the model error for each data instance using another function (the *loss function* or the *objective function*). The smaller value of the loss function, the better the fit.

In theory, to find the best model fit, we need to compute the entire model error, *i.e.* to evaluate the loss function for all possible data instances (not only the observable ones at hand), which is obviously impossible in practice. However, if the sample of observations is not biased (*i.e.* it represents well the set of possible observations) the exact model error can be approximated by the average of losses over the observed data collection. This approximation is formally defined as the *empirical risk function*:

$$ER(p, X) = \frac{1}{N} \sum_{i=1}^{N} L(p, x_i), \tag{1.2}$$

where $p = (p_1, \ldots, p_K)$ denotes model parameter vector, $L(p, x_i)$ is a loss function computed on the observation x_i. In this context, the best model is given by minimizing the empirical risk function with respect to model parameters:

$$\min_{p_1,\ldots,p_K} ER(p_1, \ldots, p_K, X) \tag{1.3}$$

Most machine learning tasks can be cast in this minimization framework, which explains why machine learning is so grounded on optimization theory; and why it exploits so many minimization methods (see Section 1.5.1.2). In this context, one generally makes a difference between *supervised learning* and *unsupervised learning*.

Supervised learning consists in analyzing labelled data, *i.e.* observations that are endowed with an output variable (such as for instance, a label category). In this case, the goal is to infer a function, which

will propose an output value or label for any new unlabeled observation. Classification and regression are examples of supervised learning problems. Due to the random nature that most often underlies real world's observations, two prediction errors must be simultaneously controlled: the bias and the variance. The *bias* is a difference between the expected estimator value and the true value (broadly speaking, it can be pictured as a systematic offset in the prediction). High bias can cause an algorithm to miss the relevant relations between inputs and target outputs (*underfitting*). The *variance* measures how the prediction accuracy varies from an instance to another one. A model with high variance shows good performances on the training set, but the performances will not generalize well to other observations (a phenomenon called *overfitting*). The objective is thus to minimize both: accurately capture the regularities in the training data and generalize well to unseen data. Unfortunately, the simultaneous minimization of both errors is impossible, because reducing one generally leads to increasing the other, and an in-between is sought: the so-called *bias/variance trade-off*. In practice, detecting an important bias is easy, as the resulting model will not capture well the data distribution. However, detecting overfitting is more difficult, as it relates to generalization capabilities on other data than those at hand. To cope for this it is recommended to divide in two sets the labelled data: the training set (*i.e.* data on which the model is trained), and the test set, (*i.e.* data used to estimate the prediction error).

In unsupervised learning, the input data have no corresponding outputs. The goal is thus to learn more about the underlying structure of the data (distribution, subgroups, repetitive patterns, etc.). Thus, unsupervised learning does not necessarily target generalization capabilities, although it sometimes appears. The learning algorithms developed in this project (which perform cluster analysis and dictionary learning, see below), belong to the unsupervised category.

Over the last decade, many problems that share similarities between

supervised and unsupervised problem has been proposed, leading to a continuum with respect to the presence (and the use) of labelled data amongst the unlabeled ones: semi-supervised learning; Bayesian learning; active learning; on-line learning; multi-view learning; etc. These problems being beyond the scope of this work, we simply refer to the adequate literature [48–51].

1.5.1.2 Objective function optimization

Many methods exist to efficiently solve a minimization problem akin to Eq. (1.3). The most well-known is referred to as gradient descent. It relies on the gradient function property: The gradient vector indicates the direction of the greatest increase of the function, and the gradient magnitude measures the increase rate of this function in the gradient direction.

Concretely, to perform a minimization such as in Eq. (1.3), the gradient descent algorithm starts from a random initialization of the parameter vector $p^0 = (p_1^0, \ldots, p_K^0)$ and then iteratively approaches to the function minimum by following the direction opposite to the gradient one and making steps proportional to the gradient vector. More precisely, at each iteration the parameter values are updated according to the following formula:

$$p^{t+1} = p^t - \gamma \frac{1}{N} \sum_{i=1}^{N} \nabla_p L(p^t, x_i), \tag{1.4}$$

where $\nabla_p L(p^t, x_i) = \left(\frac{d}{dp_1} L(p^t, x_i), \ldots, \frac{d}{dp_K} L(p^t, x_i) \right)$ is the gradient vector of the loss function computed on the current parameter values and the observed data, and where γ is a parameter called learning rate.

The gradient method is simple and efficient, but not universal. Notably, if the function to minimize has several local minima and a single global minimum (*i.e.* in mathematical terms, the function is not convex), the gradient descent approach does not guarantee an optimal minimization. Generally, convex optimization problems (*i.e.* which address the

minimization of a convex function[4] over convex sets of constraints) are easier to minimize: As any local minimum of a convex function is also a global minimum, simple minimization algorithms (such as gradient descent) are sufficient to provide an optimal solution.

Often, the optimization problems occurring in machine learning are ill-posed, *i.e.* their solution is not unique. To overcome this issue, a classical solution is to regularize the objective function, by adding to it some *regularization term* (or *penalty term*). Doing so allows promoting a solution with the desired properties and penalizing other ones. Many efficient regularization methods propose to use a penalty proportional to the norm of the solution. For example, when relying on the l_2 norm (given by $\|x\|_2 = \sum_{i=1}^{n} x_i^2$), the regularization is referred to as *Tikhonov regularization*. Alternatively, using the l_0 pseudo-norm (which counts the non-zero coordinates of vector $\|x\|_0 = \sum_{i=1}^{n} [x_i \neq 0]$) leads to promoting sparse model, a frequently required feature in many applications. Unfortunately, an l_0 regularization yields non-convex problems, however, as demonstrated with the LASSO regression (see [52]) it can easily be relaxed with an l_1 norm: $\|x\|_1 = \sum_{i=1}^{n} |x_i|$, (*i.e.* the l_1 norm provides a convex approximation of a l_0 penalized problem). A more detailed discussion about sparse representation can be found in Chapter 3.

1.5.1.3 k-means clustering

Cluster analysis aims to partition observations into groups in such way that the similarities within any given group are higher than between different groups. The choice of the similarity measure strongly influences the clustering result and it is guided by the problem objective and the data type (see Section 1.5.2.1). Cluster analysis is frequently used in many applications, including mass spectrometry data analysis: For instance, DIA-Umpire (see Section 1.4.2) uses clustering to group co-eluting fragments and precursors; Another tool, Xnet [53] (discussed

[4] That is a line connecting any two points lies above the function graph between these points

in Chapter 2) performs a chromatogram clustering to recover isotopic envelopes (different chemical form of a given precursor ion). Many clustering methods coexist. This work focuses on a very popular approach (among others), referred to as k-means clustering. This approach covers many different algorithms, yet all seek to optimize the so-called k-means objective function, which reads:

$$\min_{\substack{C_1,\dots,C_k \\ \mu_1,\dots,\mu_K}} \sum_{i=1}^{k} \sum_{x \in C_i} \|x - \mu_i\|_2^2, \tag{1.5}$$

where μ_i is a mean of points in cluster C_i and it is called a cluster centroid[5]. In other words, one searches for k cluster centroids such that the sum of the squared distances between the data instances and a cluster centroid, to which they are assigned, is minimal. As similarity decreases as a function of the distance, the k-means objective function provides the partitioning into k clusters, where the data instance within each cluster are more similar than between clusters. It should be noted that k-means objective function is non-convex, so that there is no polynomial-time algorithm that can find its global minimum. The most popular algorithm to solve Eq. (1.5) is Lloyd's algorithm (which is often improperly referred to as the classical k-means method). It works as follows: (i) k cluster centroids are randomly initialized; (ii) the clusters are formed by assigning the data instances to the closest cluster centroid; (iii) cluster centroids are recomputed as a mean of data instances in the newly constructed clusters; steps (ii) and (iii) are repeated while the objective function can be still improved. This method has some drawbacks: (1) Different random initializations of the cluster centroids lead the algorithm to converge towards different local minima, yielding different clustering results; (2) It is not adapted for large-scale data, since the repeated distance computations (between all data instances and cluster centroids) at each iteration is time consuming. Even so

[5] Let us note that in the mass spectrometry community, when the data points are chromatograms, then, the μ_i's are often referred to as *consensus chromatograms*

in practice a few iterations are sufficient to reach a local minimum, the theoretical complexity of Lloyd algorithm is super-polynomial; (3) It requires specifying the number of expected clusters (which may be unknown, notably on LC-MS data, see Chapter 2); (4) It cannot cope of the presence of outliers, as all data points are necessary assigned to one and only one cluster. Many approaches (other than Lloyd's algorithm) are available to minimize the k-means objective function. For instance k-means++ [54] (which improves on Lloyd's algorithm), discretizing the eigenvectors of the covariance matrix in a PCA-like way (see [55]) or using Orthogonal Matching Pursuit to have a decent initializer and then perform a global optimization (see [56]), and notably, the algorithm proposed in Chapter 2 relies on one of these. However, whatever the approach, the last two drawbacks (number of cluster definition and outlier management) remain, as they are inherent to the objective function.

Hierarchical clustering is another type of cluster analysis approach, which is intended to construct a cluster hierarchy. The most well-known hierarchical clustering strategy is agglomeration of clusters together going up in the hierarchy. Concretely, at the first hierarchy level, each data instance belongs to its own cluster; then cluster are pairwise merged, relying on their similarities. At the last step, all data are merged in a single cluster.

It must be noted that in the proteomics community, the clustering of mass spectra is often incorrectly referred to as the spectral clustering. However, spectral clustering has long been a particular clustering algorithm, which uses the spectrum (set of eigenvalues) of the data similarity matrix to identify clusters [57].

1.5.2 Selected overview of machine learning techniques of interest to tackle LC-MS data

This section introduces a series of specific machine learning concepts or algorithms on which our results from the following chapters have thrived.

1.5.2.1 Scalar product, similarity measure, optimal transport distance

The choice of the similarity measure involved in cluster analysis is of the utmost importance, since it impacts a lot the clustering result. The k-means objective function (see Eq. (1.5), implies that the similarity is inversely proportional to the distance. When the data instances are vectors in $\mathbb{R}^n$, the space is referred to as Euclidean if its geometry is induced by a function referred to as *scalar product*:

$$\langle x, y \rangle = x^\top \cdot y = \|x\|_2 \cdot \|y\|_2 \cdot \cos(\theta), \tag{1.6}$$

where θ is an angle between vectors x and y; where $\|\cdot\|_2$ refers to the l_2 norm of a vector (*i.e.* its length computed from the vector coordinates using the Pythagorean theorem), and $\top$ is the matrix transpose operator. Conversely, the *Euclidean distance* between two vectors can be written using scalar products as follows:

$$\|x - y\|_2 = \sqrt{\langle x, x \rangle - 2 \cdot \langle x, y \rangle + \langle y, y \rangle}. \tag{1.7}$$

Beyond the classical Euclidean distance, many other distances (and thus similarities) are insightful in machine learning, depending on the targeted task and the type of data. Notably, the *Wasserstein distance* (or *optimal transport distance* [58, 59]) as recently gain a strong interest [60–62]. In optimal transport theory, one looks for a mapping Γ, which transports one distribution μ into another distribution ν with the minimal transportation cost $c(\mu, \nu)$. Let M be a metric space, the

mathematical definition of the Wasserstein-p distance is given by the so-called Kantorovich formulation [63]:

$$W_p(\mu,\nu) = \left(\inf_{\Gamma \in J(\mu,\nu)} \int_{M\times M} c(x,y)^p d\Gamma(x,y) \right)^{\frac{1}{p}}, \tag{1.8}$$

where $J(\mu,\nu)$ denotes the set of all probabilistic measures in $M \times M$, and where the cost function is classically defined on the basis of the Euclidean distance: $c(x,y) = \|x-y\|_2$. In Chapter 2, we consider a special case of Wasserstein distance with $p = 1$, classically referred to as *Earth mover's distance* [64]. We demonstrated that this distance is a good choice to define similarities between chromatographic elution profiles.

1.5.2.2 Kernel trick

Another popular way to define meaningful similarities from algebraic distances is to project the data into *reproducing kernel Hilbert space* H, using a mapping of the form $\varphi : X \to H$ (note that H is also known as the *feature space*). Finding the explicit form of the mapping φ is not necessary thanks to the so-called *kernel trick*. This trick can be applied only for algorithms, which can be reformulated to operate on the scalar products between the data instances only, without explicit use of the data vectors. An example is Lloyd's algorithm, which can thus be kernelized, leading to the popular *kernel k-means* approach.

To practically apply the kernel trick, it is necessary to define a specific function, referred to as a kernel function, which is based on distances $d(x,y)$ in the initial space, make it possible to define similarities in the feature space. Most often, the similarity function has the following form: $k(x,y) = f(d(x,y))$. However, it is not necessary. What is necessary is that the kernel function k meets the *positive semi-definiteness* (or PSD) property, which essentially make it equivalent to a scalar product. More

formally, the PSD property reads:

$$\sum_{i=1}^{m}\sum_{j=1}^{m} c_i c_j k(x_i, x_j) \geq 0, \ \forall c_1, \ldots, c_m \in \mathbb{R}, x_1, \ldots, x_m \in \mathbb{R}^n \tag{1.9}$$

This property allows using Mercer's theorem [65], which states that the kernel function equates the canonical inner product of an Hilbert space which is proven to exist (the feature space). Therefore, even if the feature space (or its mapping onto it, φ) are not explicitly defined, it is possible to exploit its specific geometry.

The most popular kernel functions are the following: the *polynomial kernel* defined as $k(x, y) = (x^\top \cdot y + c)^d$; the *Gaussian kernel* $k(x, y) = \exp\left(\frac{-d_E(x,y)^2}{2\sigma^2}\right)$, and *Laplacian kernel* $k(x, y) = \exp\left(\frac{-d_E(x,y)}{\sigma}\right)$, where $c \in \mathbb{R}_+$ (non-negative constant) and $d \in \mathbb{N}$ are polynomial kernel parameters, where $d_E(x, y)$ denotes the Euclidean distance and where $\sigma \in \mathbb{R}_+^*$ is a Gaussian or Laplacian kernel parameter (positive constant).

In Chapter 2, we proposed to define similarities between chromatographic profiles by means of such a kernelization, yet based on the Wasserstein-1 distance instead of the Euclidean one: $k(x, y) = \exp(-\gamma \cdot d_{W1}(x, y)^s)$, when $s = 2$, this leads to a Gaussian W1 kernel, and when $s = 1$, to Laplacian W1 kernel.

1.5.2.3 Data factorization methods

Blind source separation problem consists in finding a representation of a (complex) observation x as a linear combination of some (elementary, yet unknown) signals $d_1, \ldots, d_k$:

$$x \approx \alpha_1 \cdot d_1 + \cdots + \alpha_k \cdot d_k. \tag{1.10}$$

The objective is to determine coefficients $\alpha_1, \ldots, \alpha_k$, which are referred to as a *code*, as well as signals $d_1, \ldots, d_k$, (forming a matrix D) referred to as a *dictionary*. It should be mentioned that the spectrum demultiplexing problem addressed by Specter tool (see Section 1.4.2) can be reformulated

as a blind source separation problem. In the literature, many different alternative names exist: When the decomposition is performed for a collection of signals concatenated in a matrix, the problem is referred to as a *matrix factorization*. Additional requirements can be added via mathematical constraints to yield more realistic problem modeling: non-negative signals, orthogonal signals, etc. Notably, if all involved matrices are non-negative, one falls back on the well-known *non-negative matrix factorization* (or NMF) framework [66]. Alternatively, truncated singular value decomposition (SVD) is a matrix factorization approach suited to orthogonality constraints. SVD factorizes the data matrix $X \in \mathbb{R}^{n \times N}$ into a product of three matrices: $U \cdot \Sigma \cdot V^T$, where U contains right singular vectors of X, V contains left singular vectors and Σ is a diagonal matrix of singular values. U and V are orthogonal matrices. In practice, general SVD is replaced by its truncated approximation, where only a subset of the largest singular values is computed.

Matrix factorization can also be called *dictionary learning*. In this case, it is commonly assumed that the representation is *sparse*, *i.e.* only some of the code's coefficients $\alpha_1, \ldots, \alpha_k$ are non-zero. The dictionary learning problem can be formulated as a constrained optimization problem with a fixed sparsity level T:

$$\min_{\substack{A \in \mathbb{R}^{K \times N} \\ D \in \mathbb{R}^{n \times K}}} \frac{1}{2} \sum_{i=1}^{N} \|x_i - D \cdot \alpha_i\|_2^2, \qquad \|\alpha_i\|_1 \leq T, \ i = 1, \ldots, N, \tag{1.11}$$

as well as an unconstrained problem, where the regularization parameter λ controls the sparsity level:

$$\min_{\substack{A \in \mathbb{R}^{K \times N} \\ D \in \mathbb{R}^{n \times K}}} \frac{1}{2} \sum_{i=1}^{N} \|x_i - D \cdot \alpha_i\|_2^2 + \lambda \|\alpha_i\|_1, \tag{1.12}$$

where $A = \{\alpha_1, \ldots, \alpha_N\}$ denotes a code matrix.

It should be mentioned that the k-means objective function (presented above) can be reformulated as a matrix factorization problem [67, 68]. In fact, the most known dictionary learning algorithm

K-SVD is a generalization of the k-means method [69].

Relying on algebraic methods (such as singular values or eigenvalues decomposition) to factorize the data can be computationally costly, especially for large-scale ones. In general, matrix factorization is achieved by solving the dictionary learning problem (Eq. (1.12)) using an alternative optimization strategy. This strategy exploits the fact that the objective function in Eq. (1.12) is non-convex with respect to both variables A and D, (meaning it is hard and slow to solve), but it is convex on each variable taken independently. Thus, it makes sense to convert the joint minimization of Eq. (1.12) into an alternative minimization with respect to one variable where the other one is fixed, and then to swap. This leads to alternate between the minimization of the two following problems:

$$\alpha_1^*, \ldots, \alpha_N^* = \arg \min_{A \in \mathbb{R}^{K \times N}} \frac{1}{2} \sum_{i=1}^{N} \|x_i - D^* \cdot \alpha_i\|_2^2 + \lambda \|\alpha_i\|_1, \tag{1.13}$$

$$d_1^*, \ldots, d_K^* = \min_{D \in \mathbb{R}^{n \times K}} \frac{1}{2} \sum_{i=1}^{N} \|x_i - D \cdot \alpha_i^*\|_2^2, \tag{1.14}$$

where the optimal dictionary $D^* = \{d_1^*, \cdots, d_K^*\}$.

The first minimization (with respect to the code matrix A) can be handled using for example LARS method [70] or hard thresholding method [71] or Matching Pursuit [72]; As for the second one, we can rely on methods such as Coordinate Descent [73] or Stochastic Gradient Descent [74] (see Section 1.5.2.5 for details).

1.5.2.4 Nyström approximation

Proteomics data are both high dimensional and large-scale, which makes their processing with state-of-the-art machine learning techniques a challenge, as their complexity is often superlinear (to put it mildly). Therefore, our work extensively relies on dimensionality reduction and data compression techniques that are briefly recalled hereafter.

Nyström approximation [75] is a well-known kernel matrix approximation method. It is used to avoid the quadratic complexity induced by

the computation of all the pairwise similarity stored in a kernel matrix. It involves truncated SVD decomposition and data sub-sampling. Its core idea is to randomly select column indices, and then to compute these kernel matrix columns only. Finally, it uses these columns to approximate the entire matrix. More formally, the kernel matrix $K \in \mathbb{R}^{N \times N}$ is approximated as follows:

$$C \cdot W_r^{-1} \cdot C^\top = C \cdot U_r \cdot D_r^{-1} \cdot U_r^\top C^\top \tag{1.15}$$

where $C \in \mathbb{R}^{N \times l}$ is a kernel submatrix, $W \in \mathbb{R}^{l \times l}$ is a matrix, obtained from C by selecting corresponding rows as well, and $W_r = U_r \cdot D_r \cdot U_r^\top C^\top$ is r-truncated SVD of W.

Recently, Mahoney's team proposed scale-up the kernel k-means using Nyström approximation [76]. Concretely, their approach clusters the data in three steps: First, the kernel matrix K is approximated using Nyström method. According to Eq. (1.15), it results in the construction of r-dimensional feature vectors in the rows of the matrix $\Phi = C \cdot U_r \cdot D_r^{-\frac{1}{2}} \in \mathbb{R}^{N \times r}$. Second, the feature dimension is reduced to s ($s < r$) by applying an s-truncated SVD to Φ: $\Phi_s = U_s \cdot \Sigma_s \cdot V_s^\top$. Third, rows of the matrix B defined as $B = U_s \cdot \Sigma_s \in \mathbb{R}^{N \times s}$ are clustered using any classical (*i.e.* operating in the original space) k-means clustering algorithm. This approach is computationally efficient and comes with theoretical approximation error estimation. Concretely, they demonstrated that their approach with feature dimension s fixed at $\frac{k}{\epsilon}$, where k is the number of clusters, provides the cluster assignment, which is at most $1 + O(\epsilon)$ times worse than the clustering obtained by the classical kernel k-means, where $\epsilon \in (0, 1)$. Following this strategy in Chapter 2, we scaled-up another k-means algorithm (the Compressive k-means, see below) using Nyström approximation.

1.5.2.5 Stochastic and Momentum based gradient descent methods

Stochastic gradient descent (SGD [74]) is an adaptation of the classical gradient descent scheme for large-scale datasets. Let us assume a classical objective function in an empirical minimization framework (see Section 1.5.1.1): $\frac{1}{N}\sum_{i=1}^{N} L(p, x_i)$. Stochastic scheme approximates the time-consuming gradient computation on an entire dataset (see Section 1.5.1.2) by the gradient of a single randomly selected summand $L(p, x_i)$ (*i.e.* the empirical risk is estimated on a single data observation):

$$p^{t+1} = p^t - \gamma \cdot \nabla_p L(p^t, x_i). \tag{1.16}$$

Applying this strategy for all the data instances defines an *epoch*. There exist several variants of SGD: Mini-batch gradient descent performs the update on a small subset of data, called a *batch*, instead of a single vector. Other modifications, such as Momentum gradient descent [77] and Nesterov accelerated gradient descent [78], make use of an additional term at each iteration, called *momentum*, which is a weighted sum of gradient vectors computed at previous iterations.

These momentum methods are based on a simple idea: If during the previous iterations, the gradient vectors frequently pointed toward a given direction, it is most likely the correct one to follow to find the minimum, and changing it would not make sense. In this context, the momentum term simply adds some inertia (by preventing excessive changes from one iteration to another). More formally, it has been demonstrated that it yields faster convergence than SGD (see [79]). The update rule for the Momentum gradient descent is given by:

$$v^{t+1} = \alpha \cdot v^t - \gamma \cdot \nabla_p L(p^t, x_i), \tag{1.17}$$

$$p^{t+1} = p^t + v^{t+1}, \tag{1.18}$$

where v^t denotes the momentum vector and α is a decaying parameter,

which controls the influence of the previously accumulated gradients.

The difference between the Nesterov accelerated scheme and the Momentum method lies in the gradient computation. In Nesterov's method, the gradient is computed at the intermediate parameter value $p^{t+\frac{1}{2}}$, which is the point indicated by the current momentum vector (Eq. (1.19)). The rest remains unchanged:

$$p^{t+\frac{1}{2}} = p^t + \alpha \cdot v^t, \tag{1.19}$$

$$v^{t+1} = \alpha \cdot v^t - \gamma \cdot \nabla_p L(p^{t+\frac{1}{2}}, x_i), \tag{1.20}$$

$$p^{t+1} = p^t + v^{t+1}. \tag{1.21}$$

1.5.2.6 Sketching learning

Recently, Remy Gribonval's team proposed an original way to scale up computationally demanding machine learning algorithm. Their method, referred to as Compressive Learning [80] is based on constructing a *data sketch*, *i.e.* a single vector of fixed size, which summarizes the data. Then, the algorithm can operate on the sketch in place of the original data, making it more efficient.

Let us assume the observed data obeys some law P referred to as the *data distribution*. With this regard, the observed data is an empirical sample from this unknown distribution $X \sim P$. The *characteristic function* of P, is defined as:

$$\varphi_{X \sim P}(w) = \int_{-\infty}^{\infty} \exp(i \cdot w^\top \cdot x) \cdot P(dx), \tag{1.22}$$

where $w \in \mathbb{R}^n$ is a frequency vector. The characteristic function has two important properties: It always exists, and it completely represents the distribution, making it the best candidate to rely on for the construction of the data sketch.

In practice, the data distribution P is unknown, and it is approxi-

mated as a sum of Diracs located at the data instances as follows:

$$P = \frac{1}{N} \sum_{i=1}^{N} \delta(x_i) \tag{1.23}$$

Thus, substituting the continuous probability distribution by the empirical one, we obtain the following formula for the characteristic function:

$$\varphi_{X \sim P}(w) = \sum_{i=1}^{N} \exp(i \cdot w^{\top} \cdot x_i), \tag{1.24}$$

Finally, the data sketch is derived by sampling the characteristic function in the frequency domain in spirit of the random Fourier features [81]:

$$SK(X) = (\varphi_{X \sim P}(w_1), \ldots, \varphi_{X \sim P}(w_m)) \in \mathbb{C}^m, \tag{1.25}$$

where $w_1, \ldots, w_m$ are frequency vectors sampled from some predefined frequency probability distribution Λ. More details about the link between the sketching operator and random Fourier features can be found in Chapter 2.

An example of learning from sketched data can be found in [56], which details an algorithm called Compressive k-means: It identifies cluster centroids $C = \{\mu_1, \ldots, \mu_K\}$ by solving the following minimization problem:

$$\min_{\substack{\mu_1,\ldots,\mu_K \\ \alpha_1,\ldots,\alpha_K}} \| SK(X) - \alpha \cdot [SK(\mu_1) \ldots SK(\mu_K)] \|_2^2, \tag{1.26}$$

where $[SK(\mu_1) \ldots SK(\mu_K)] \in \mathbb{C}^{m \times K}$ denotes the matrix resulting from the concatenation of the cluster centroid's sketches, and $\alpha = (\alpha_1, \ldots, \alpha_K) \in \mathbb{R}^K$ is a vector of cluster centroid's weights. Let us note the equivalence of the objective function of Eq. (1.26) and the signal decomposition problem (Eq. (1.10)). Thus, the cluster centroid's sketches can be seen as dictionary elements and Compressive k-means as a way to decompose the data sketch onto this dictionary. The Compressive k-means is in fact the sketched version of the dictionary learning method named

Orthogonal Matching Pursuit [82]. From an implementation viewpoint, Compressive k-means is a greedy method, which iterates four steps: (i) It expands the set of cluster centroids C with a vector μ_i, which sketch is the most correlated to the residue (at the first iteration the residue is equal to the data sketch vector):

$$\mu_i = \arg\max_{\mu \in \mathbb{R}^n} \Re \left(\left\langle \frac{SK(\mu)}{\|SK(\mu)\|_2}, r \right\rangle \right), \tag{1.27}$$

where $\Re$ denotes the real part of the complex number; (ii) It computes weights of centroids:

$$\min_{\alpha \geq 0} \left\| SK(X) - \alpha \cdot \left[SK(\mu_1) \dots SK(\mu_{|C|}) \right] \right\|_2^2; \tag{1.28}$$

(iii) It minimizes the objective function with respect both cluster centroids and weights:

$$\min_{\alpha \geq 0, C \in \mathbb{R}^{n \times K}} \left\| SK(X) - \alpha \cdot \left[SK(\mu_1) \dots SK(\mu_{|C|}) \right] \right\|_2^2; \tag{1.29}$$

(iv) It recomputes the residue:

$$r = r - \alpha \cdot \left[SK(\mu_1) \dots SK(\mu_{|C|}) \right]. \tag{1.30}$$

The second step can be achieved using any non-negative least-squares solvers, the first and the last steps are fulfilled by gradient descent. Since the objective function is non-convex with respect to both cluster centroids and weights, the global minimization of the third step is the main bottleneck of the method.

1.6 Problem statement and formulation

A recently published approach, EncyclopeDIA [83],introduces a new workflow for DIA data analysis based on the construction of a chromatogram library. EncyclopeDIA generates the chromatogram library

by using DIA with *gas-phase fractionation* ([84], a specific acquisition mode, where the same sample is injected multiple times, each one allowing the data to be acquired in different m/z windows). As covering the full m/z range requires several DIA runs on the same sample (at least six), the approach is experimentally costly. However, it makes it possible to demultiplexed DIA spectra on-the-fly, and to identify peptides in each window using a DIA library-based approach or a DIA database search engine (*e.g.* PECAN). The chromatogram library contains the following information: retention time, peak shape, peptide mass, peptide fragmentation patterns, and known interferences of detected peptides.

The authors demonstrate that using such a chromatogram library allows for the identification of 20% to 25% additional peptides, compared to both DDA analysis and library-based DIA analysis. However, EncyclopeDIA chromatogram library construction has some drawbacks: The main one is its experimental cost. Another limitation is inherited from the integrated demultiplexing methods: the constructed chromatogram library contains only peptides present either in the used spectral library or in the peptide list provided to PECAN. Chromatographic profile pattern extraction can improve DDA peptide identification as well: Recent investigations [85–87] showed that using both data types (chromatographic profiles and mass spectra) allows extracting more pieces of information from DDA data; as a result, it yields better peptide identification.

To summarize, the recently developed concept of chromatogram libraries makes it possible to significantly improve peptide identification for both DDA and DIA data. However, methods insofar available for chromatogram library construction require extensive wet-lab experiments. In this work, we have leveraged recent machine learning advances to develop computational methods (*i.e.* without additional wet-lab experiments) to construct chromatogram libraries directly from the LC-MS data of interest.

Few previously published works have already proposed to apply

machine learning algorithms on chromatographic data. However, they focus on chromatographic profile recognition from DDA data: feature recognition algorithms based on artificial neural network [88] or on Bayesian probabilistic models [89,90], as well as chromatogram extraction methods based on Wavelets transform [91, 91–93]. Moreover, their goal is closer to signal denoising than to pattern extraction.

To the best of the author's knowledge, no method has so far proposed to exactly address the problem of chromatogram library construction from DIA data.

A potential strategy to learn the chromatographic patterns from the data is to perform a chromatogram clustering and to form the chromatogram library with the cluster centroids (or consensus chromatograms). This strategy has been investigated and has led to an article which is accepted for publication in BMC Bioinformatics journal. Its content forms Chapter 2.

An alternative method is to rely on an analogy between the EncyclopeDIA approach and the formalism of dictionary learning. This suggests to directly learn the chromatogram library that best explains the LC-MS data. This idea is appealing, as in the near future, it could lead to a strategy to demultiplex DIA mass spectra, as investigated by the developers of Specter [28]. Following this approach, the considered problem would mathematically reads as:

$$\min_{\substack{D\in\mathbb{R}^{n\times K}\\ A\in\mathbb{R}^{K\times N}}} \|X - D\cdot A\|_2^2 \tag{1.31}$$

where D denotes a chromatogram library matrix (with extracted chromatogram patterns as columns), and A is a matrix with pseudo-DDA mass spectra as rows. As presented above, such an objective function is now mainstream in machine learning. However, efficiently minimizing on data as large and as noisy as LC-MS data remains a challenge. This why, our investigation on this path has led to a manuscript submission in Statistical Analysis and Data Mining, which constitutes Chapter 3.

CHAPTER 2

Chromatographic profile clustering

CHICKN: Extraction of peptide chromatographic elution profiles from large scale mass spectrometry data by means of Wasserstein compressive hierarchical cluster analysis[1]

Olga Permiakova[a], Romain Guibert[a], Alexandra Kraut[a], Thomas Fortin[a], Thomas Burger[b]

[a] Univ. Grenoble Alpes, CEA, Inserm, BGE U1038, 38000 Grenoble, France

[b] Univ. Grenoble Alpes, CNRS, CEA, Inserm, BGE U1038, 38000 Grenoble, France

Abstract

Background: The clustering of data produced by liquid chromatography coupled to mass spectrometry analyses (LC-MS data) has recently gained interest to extract meaningful chemical or biological patterns. However, recent instrumental pipelines deliver data which size, dimensionality and expected number of clusters are too large to be processed by classical machine learning algorithms, so that most of the state-of-the-art relies on single pass linkage-based algorithms.

[1] This article is accepted in *BMC Bioinformatics* journal

Results: We propose a clustering algorithm that solves the powerful but computationally demanding kernel k-means objective function in a scalable way. As a result, it can process LC-MS data in an acceptable time on a multicore machine. To do so, we combine three essential features: a compressive data representation, Nyström approximation and a hierarchical strategy. In addition, we propose new kernels based on optimal transport, which interprets as intuitive similarity measures between chromatographic elution profiles.

Conclusions: Our method, referred to as CHICKN, is evaluated on proteomics data produced in our lab, as well as on benchmark data coming from the literature. From a computational viewpoint, it is particularly efficient on raw LC-MS data. From a data analysis viewpoint, it provides clusters which differ from those resulting from state-of-the-art methods, while achieving similar performances. This highlights the complementarity of differently principle algorithms to extract the best from complex LC-MS data.

Keywords: *Large-scale cluster analysis; Liquid chromatography; Mass spectrometry; Proteomics; Wasserstein kernel; Optimal transport*

2.1 Background

Liquid chromatography coupled to mass spectrometry (LC-MS) constitute a technological pipeline that has become ubiquitous in various omics investigations, such as proteomics, lipidomics and metabolomics. Over the past decade, the MS throughput has continuously improved, leading to unprecedented data volume production. To date, processing these gigabytes of low level MS signals has become a challenge on its own, for a trade-off between contradictory objectives is sought: On the one hand, one needs to save memory and computational time with efficient encoding, compression and signal cleaning methods [94]. On the other hand, one needs to avoid too important preprocessing that systematically smoothes signals of lower magnitudes, as it is now well-established that interesting biological patterns can be found near the noise level [95]. To face this challenge, a recent and efficient investigation path has been to apply cluster analysis to LC-MS data. Cluster analysis refers to a large family of unsupervised statistical learning and multivariate analysis techniques which share a common goal: Aggregating similar data items

into clusters, so that within-cluster similarities are larger than between cluster ones. By doing so, it becomes possible to consider the various clusters independently, and thus to reduce the computational footprint without any quality loss. Moreover, as each cluster contains similar data elements, it facilitates the extraction of repetitive but small biological patterns.

2.1.1 State of the art

To date and contrarily to the presented work, investigations have mainly focused on clustering LC-MS data across the chromatographic (or elution time) dimension, *i.e.* when the data elements are MS spectra: MS2grouper [96, 97], Pep-Miner [98], PepMerger [99], the MS-Clustering / MS-Cluster / Pride-Cluster / spectra-cluster series [100–103], Bonanza [104], CAMS-RS [105], MaRaCluster [106], N-cluster [107], and msCRUSH [108]. All these approaches propose to improve peptide identification by benefiting from the aforementioned trade-off: By grouping similar fragmentation spectra into a consensus representation, one clearly reduces the data volume. Moreover, peaks corresponding to random noise should not reinforce between spectra, while on the contrary, small but chemically consistent peaks should [109].

Clustering across the mass-to-charge ratio (m/z) dimension, *i.e.* when the data elements are chromatographic profiles (depicting the signal changes along the elution time at a given m/z value), is also insightful for many reasons: First, it proposes an original framework to construct and extract precursor ion chromatograms, which integration is essential for quantitative analysis [53]. Second, cluster centroids naturally provide consensus elution profiles which are of interest for retention time alignment [50]. Finally, elution profiles are also essential to disentangle chimeric spectra [110]. Notably if the clustering is sufficiently accurate, it can be insightful to disentangle multiplexed acquisitions (*e.g.* Data Independent Acquisition [111], or DIA), without relying on spectral libraries [28, 112]. To date, these practical problems have

been tackled in the proteomics literature by applying various heuristics which differ to some extend from the cluster analysis framework. For instance, in DIA-Umpire [26], peptide fragments' elution profiles are clustered according to their correlations with precursor profiles, so that formally, the approach is more that of classification (*i.e.* supervised) than of clustering (*i.e.* unsupervised). Similarly, in many quantification algorithms (Maxquant [46], OpenMS [113], MsInspect [114], Xnet [53]) cluster analysis aims to extract isotopic envelopes, *i.e.* to group the elution profiles of several isotopes of a given molecule, within a closed neighborhood of m/z values. As a consequence, two identical profiles in different m/z regions are not grouped together. Although this behavior (that will be referred to as the *envelope assumption simplification* in the rest of the article) concurs with the objective of isotopic envelope reconstruction, it makes the heuristic strongly attached to one objective; and non applicable to other cluster analysis problems. In contrast, we believe generic clustering algorithms would also be of interest, as different tuning would make them appropriate to deal with different objectives: *e.g.* by adding must-link/must-not-link constraints [115] so as to guide the demultiplexing task as in the DIA-Umpire case; or by incorporating an m/z difference in the similarity definition, in the case of isotopic envelope extraction; and so on.

Moreover, a refine analysis of the algorithms underlying all these (either spectrum or chromatogram) clustering techniques let appear a strong filiation between them: All rely on agglomerative and linkage-based methods, be it previously published algorithms (HAC [116, 117], DBSCAN [118] or UPGMA [119]) or *ad-hoc* procedures developed in the specific context of LC-MS data clustering (proposed in MS2grouper, Pep-Miner, PepMerger, the MS-Cluster series, Bonanza, CAMS-RS, N-cluster and XNet). Despite their unquestionable efficiency, some diversity would help. Cluster analysis is as much an art as a science [120] and there does not exist such thing as the perfect clustering – at least, on real data. Most of the time, data analysts need to rely on a toolbox

of various algorithms to extract the best of their data [121]. With this respect, MS-based omics would benefit from differently principled and complementary algorithms which have demonstrated their efficiency in data science [122]. For instance, spectral clustering [123–125] (which should not be confused with the cluster analysis of mass spectra [57]), mean shift algorithm [126, 127], and variants of the k-medoids [128] and k-means [129, 130] are of prime interest.

Finally, one observes a difference between algorithms dedicated to spectrum clustering and those dedicated to chromatogram clustering: While the former ones are mainly implemented in an independent manner, the latter ones are all embedded in computational pipelines (DIA-Umpire [26], Maxquant [46], OpenMS [113], MsInspect [114]). The only exception is Xnet [53], which makes it a unique literature reference for algorithmic and low-level comparisons. In addition, Xnet is the most recently published algorithm, and it displays interesting performances on a benchmark dataset.

In a nutshell, Xnet is a Bayesian algorithm which aims to cluster elution profiles into isotopic envelopes. More precisely, it starts from the construction of a network with chromatograms as nodes. Then, the network is decomposed into preliminary clusters. The edges within each cluster are scored by estimating the likelihood of two parameters: the correlation between chromatograms and their m/z separation. Finally, the edge validation is carried out using the scores and a chromatogram apex match verification. This leads to the final isotopic envelope construction.

Xnet has many strengths: First, it is a parameter free clustering method – the number of clusters can be inferred during the learning process. Second, the time complexity of the algorithm is linear with respect to the number of chromatograms in the data. However, it also has weaknesses: First, it cannot work on raw data and requires an important preprocessing step, referred to as *ion chromatogram extraction*, which denoizes the LC-MS map and aggregates independent measurements into well-formed *traces* (*i.e.* lists of peak intensities corresponding to a

same ion, identified in consecutive mass spectra). Concretely, starting from a raw file, it is first necessary to extract non trivial information and to store them into an input CSV file with the following columns: m/z ratios, retention times, intensities and trace labels. In addition to be time consuming, it can arguably be considered that excluding the trace construction from the algorithm amounts to transferring a bottleneck question to another preliminary processing, or to a human annotator. Second, it strongly relies on the envelope assumption simplification, making it impossible to group elution profiles which m/z difference exceeds a predefined threshold.

The third weakness is related to the generalization capabilities: As acknowledged in [53], there is not enough data to accurately train the probability model underlying Xnet, making it necessary to complement it with a Bayesian prior. This obviously questions the applicability to datasets that significantly differ from the ones that served to tune the prior. Finally, Xnet does not provide a consensus chromatogram for each cluster: Its output is a CSV file that only assigns a cluster index to each line of the input CSV file.

2.1.2 Objectives and contributions

The objective of this article is twofold: First is to propose a new cluster analysis pipeline adapted to the challenging problem of clustering multiplexed chromatographic profiles resulting from data independent acquisitions. The second objective is to build this pipeline around an algorithm which is not agglomerative and linkage-based. Concretely, we focused on k-means objective function, for two reasons: First, until recently, it was considered by the proteomics community as non-applicable to data as big as LC-MS data [100], while recent theoretical progresses have made this scaling-up possible [131] (this explains the historical predominance of agglomerative linkage-based clustering, less computationally demanding); Second, k-means can be reformulated to fit the reproducing kernel Hilbert space theory [132] (leading to the so-called

kernel k-means framework [133]), which provides new opportunities to define similarity measures that capture the biochemical specificities of LC-MS data (a challenge that has consistently been pinpointed as essential over the last fifteen years [96, 98, 99, 104–106]).

The contributions of this article are the following: First, it introduces the use of Wasserstein-1 (W1) distance (a.k.a. earth mover's distance, a.k.a. optimal transport distance) to account for similarities between elution profiles. Second, it shows that combining Nyström method and random Fourier features leads to a dramatic data compression level that makes the k-means objective function minimizable on raw and high resolution proteomics data with a multi-core machine. Finally, it demonstrates the applicability and interest of the method to process proteomics data from DIA experiments.

2.2 Methods

2.2.1 Materials

To conduct our study, we have relied on three datasets. The first one, hereafter referred to as UPS2GT, is a publicly available dataset [26]. To be used as a benchmark for Xnet, this dataset had been preprocessed and manually annotated with isotopic envelopes that can serve as ground truth [134]. Moreover, the data had been converted into *centroid* mode, *i.e.* a compressed version of the original *profile* data. In the *profile* mode, each peak of the mass spectrum is represented by intensities reported for several consecutive m/z values, so as to account for the measurement imprecision. In contrast, the *centroid* mode summarises all the values of the profile mode into a single m/z value, located at the center of the measurement distribution. It leads to significantly smaller memory footprint, at the price of blurring the differences between true signal and noise.

The second dataset, hereafter referred to as Ecoli-DIA, is the raw

output of a DIA analysis of an *Escherichia Coli* sample (containing over 15,000 peptides[2] which signals are multiplexed). To avoid any distortion or information loss, it was stored using the profile mode. The resulting file has an important memory footprint of 3.6 GB. Thus, even though chromatogram clustering operates on fraction of the data only (the so-called MS1 acquisitions, see Section 2.2.1.3), it requires adapted software tools and methods.

Finally, to account for the rapid increment of data size in proteomics (resulting from using ever longer LC and ever more resoluted MS acquisitions), we have considered a third dataset, exactly similar to the Ecoli-DIA dataset, but acquired as Full-MS instead of as DIA. This means that 100% of the acquisition time was dedicated to MS1 signals, so as to mimick the extraction of a much larger DIA dataset resulting from more time- and m/z-resoluted acquisitions. This so-called Ecoli-FMS dataset has a memory footprint of 3.2 GB. Even though of equivalent size, this dataset is in fact 16 bigger than Ecoli-DIA (four times more MS1 spectra which are four times more resoluted), see Section 2.2.1.3.

2.2.1.1 UPS2GT benchmark dataset

The UPS2GT dataset [134] resulted from the liquid chromatography coupled to mass spectrometry analysis of 48 human proteins of the Proteomics Dynamic Range Standard (UPS2) on a AB Sciex TripleTOF 5600 instrument using data dependent acquisition with an MS1 ion accumulation time of 250 ms [26].

The 28,568,990 detected points in the resulting LC-MS map were annotated according to their intensity value, either as informative or as noisy. Over 1,2 million informative points were segmented into 57,140 extracted ion chromatograms referred to as *traces*. Then, the traces were grouped into 14,076 isotopic envelopes. These envelopes constitute the dataset ground truth (therefore, the objective of the clustering task

[2] We consider that a peptide is characterized by a triplet: its amino acid sequence, a list of post-translational modifications and their localization on the sequence. Accordingly, different isotope measurements can be grouped into a single peptide definition.

would be to re-build the envelopes from the traces). The final fully annotated data were stored in a CSV file, where each row depicts one LC-MS point with four pieces of information: its mass to charge ratio, retention time, intensity, trace label and envelope label. The points that were assumed noise were given -1 or 0 as trace label.

2.2.1.2 Ecoli datasets: wet-lab analysis

Escherichia Coli bacteria were lysed with BugBuster reagent (Novagen, final protein concentration 1μg/μL). Around 560 μg of proteins were stacked in the top of a 4 - 12% NuPAGE ZOOM gel (Life Technologies) and stained with R-250 Coomassie blue. Gel was manually cut in pieces before being washed by six alternative and successive incubations in 25 mM NH_4HCO_3 for 15 min, followed by 25 mM NH_4HCO_3 containing 50% (v/v) acetonitrile. Gel pieces were then dehydrated with 100% acetonitrile and incubated with 10 mM DTT in 25 mM NH_4HCO_3 for 45 min at 56 °C and with 55 mM iodoacetamide in 25 mM NH_4HCO_3 for 35 min in the dark. Alkylation was stopped by the addition of 10 mM DTT in 25 mM NH_4HCO_3 (incubation for 10 min). Gel pieces were then washed again by incubation in 25 mM NH_4HCO_3 followed by dehydration with 100% acetonitrile. Modified trypsin (Promega, sequencing grade) in 25 mM NH_4HCO_3 was added to the dehydrated gel pieces for incubation at 37 °C overnight. Peptides were extracted from gel pieces in three sequential extraction steps (each 15 min) in 30 μL of 50% acetonitrile, 30 μL of 5% formic acid, and finally 30 μL of 100% acetonitrile. The pooled supernatants were aliquoted and dried under vacuum.

The dried extracted peptides were resuspended in 5% acetonitrile and 0.1% trifluoroacetic acid and 500ng were analyzed by online nanoliquid chromatography coupled to tandem mass spectrometry (LC-MS/MS) (Ultimate 3000 RSLCnano and the Q-Exactive HF, Thermo Fisher Scientific). Peptides were sampled on a 300 μm 5mm PepMap C18 precolumn (Thermo Fisher Scientific) and separated on a 75 μm 250

mm C18 column (Reprosil-Pur 120 C18-AQ, 1.9 μm, Dr. Maisch HPLC GmbH). The nano-LC method consisted of a 120 minute multi-linear gradient at a flow rate of 300 nl/min, ranging from 5 to 41% acetonitrile in 0.1% formic acid. The spray voltage was set at 2 kV and the heated capillary was adjusted to 270°C. For the Ecoli-FMS dataset, survey full-scan MS spectra (m/z from 400 to 1,400) were acquired with a resolution of 240,000 after the accumulation of $3 \cdot 10^6$ ions (maximum filling time 200 ms). For the Ecoli-DIA dataset, survey full-scan MS spectra (m/z from 400 to 1,400) were acquired with a resolution of 60,000 after the accumulation of $3 \cdot 10^6$ ions (maximum filling time 200 ms) and 30 successive DIA scans were acquired with a 33Th width and a resolution of 30,000 after the accumulation of $2 \cdot 10^5$ ions (maximum filling time set to auto). The HCD collision energy was set to 30%. MS data were acquired using the software Xcalibur (Thermo Fisher Scientific).

2.2.1.3 Ecoli datasets: Data preparation

The output of the LC-MS/MS experiments were converted from the proprietary RAW format into mzXML files using ProteoWizard [135]. It led to files of 11.4 GB (Ecoli-DIA) and of 10.2 GB (Ecoli-FMS), containing several pieces of information: discretized spectra under the form of coupled lists of m/z and intensity values; as well as metadata about the experiment (number of spectra, retention time range, etc).

In the case of the Ecoli-FMS dataset, all the spectra are peptide mass spectra, also termed MS1. However, the Ecoli-DIA datasets contains two types of spectra: precursor spectra (MS1) and fragmentation spectra (MS2). Thus, to work on the elution profiles, we have extracted the MS1 signals from the Ecoli-DIA file. Then, for both files, we have reconstructed chromatographic signals from MS1 spectrum intensities. As the proposed method aims to work on data as raw as possible (*i.e.* without preliminary denoising, smoothing and so on), we converted each mzXML file into an intensity matrix such as the ones of Figure 2.1

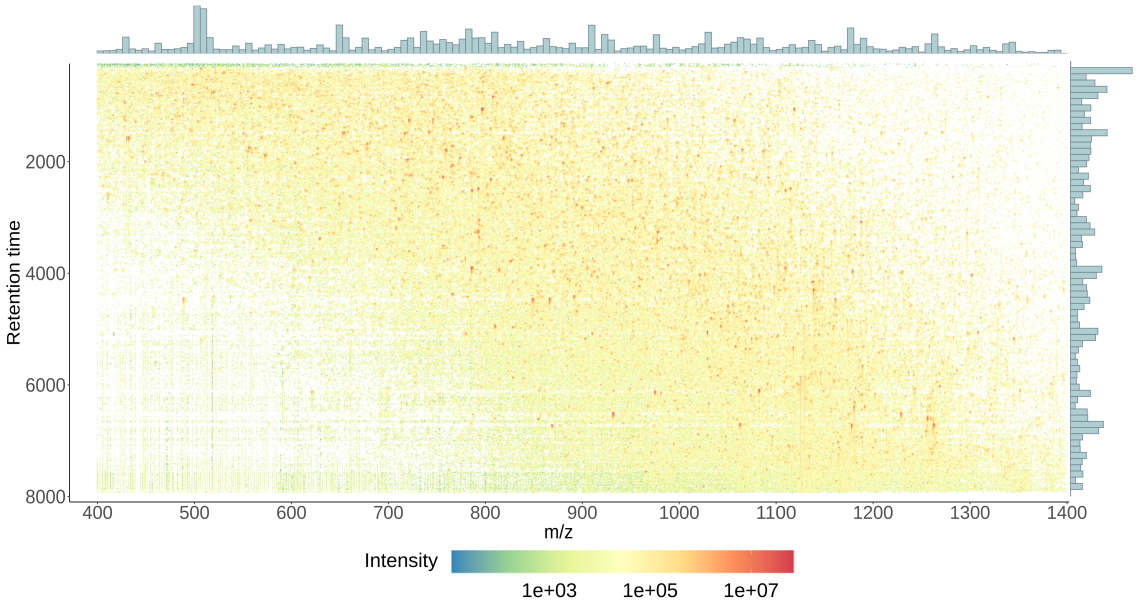


Figure 2.1: Ecoli-DIA data matrix. Each matrix column corresponds to a chromatographic profile for a fixed m/z value. Maximum Intensity for columns and for rows is depicted in bar plots.

(Ecoli-DIA) and of Figure 2.2 (Ecoli-FMS), where each row corresponds to a spectrum and each column to an elution profile (despite possible m/z fluctuations that may hamper the signal continuity). We concretely constructed each data matrix using the LC-MS analysis time-stamps and a non-uniform sampling of the m/z range (see Appendix A for a detailed description). Concretely, the resampled m/z values are given by the following recursive formula:

$$m_{i+1} - m_i = \frac{0.015}{Res_{_{EXP}}} m_i^{\frac{3}{2}}, \tag{2.1}$$

where m_i is the i^{th} sampled m/z value and $Res_{_{EXP}}$ is the instrument resolution used in the experiment ($Res_{_{FMS}} = 240,000$ and $Res_{_{DIA}} = 60,000$). Finally, we have linearly interpolated the intensity values at each node m_i of the grid:

$$I_i = I_{\text{left}} + (m_i - m_{\text{left}}) \cdot \frac{I_{\text{right}} - I_{\text{left}}}{m_{\text{right}} - m_{\text{left}}}, \tag{2.2}$$

where m and I pairs with sub-indexes "left", "right" refer the left and

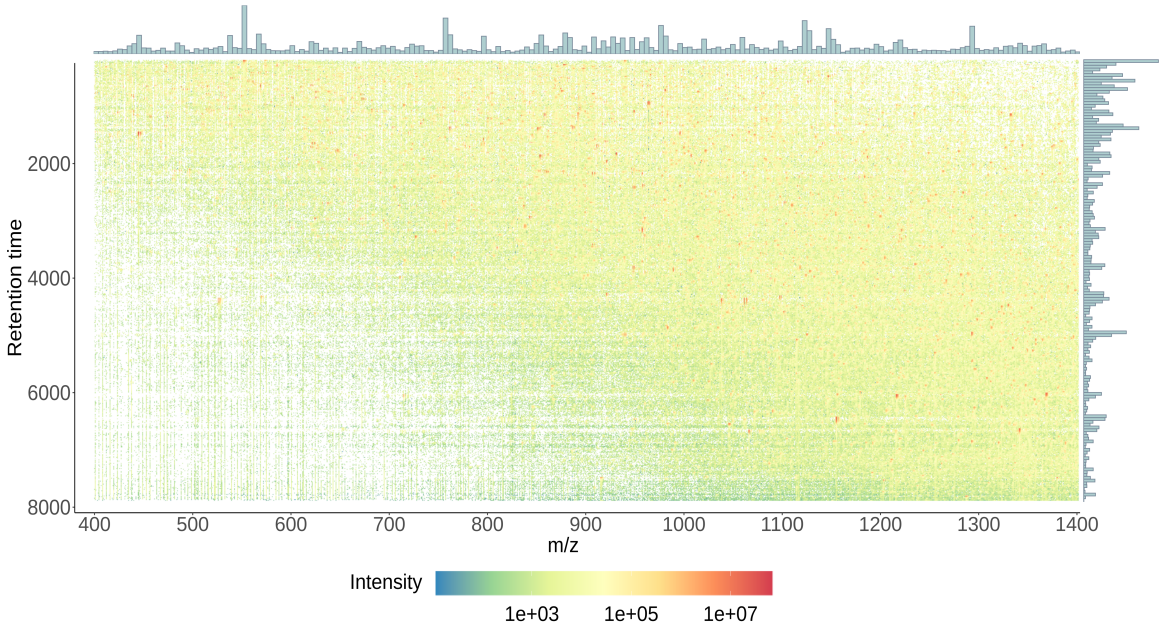


Figure 2.2: Ecoli-FMS data matrix. Each matrix column corresponds to a chromatographic profile for a fixed m/z value. Maximum Intensity for columns and for rows is depicted in bar plots.

right neighboring peaks. This is followed by the deletion of the few empty columns. The resulting Ecoli-DIA data matrix is depicted in Figure 2.1: it contains around 3,300 rows and 190,000 columns and it has a footprint of 4.8 GB. As expected, the Ecoli-FMS data matrix (Figure 2.2) is bigger: 14,000 rows, 700,000 columns and 82 GB. The bar plots in the margins of both figures represent the intensity distribution across the matrix columns and rows. They show that the Ecoli-FMS and Ecoli-DIA matrices have the same structure and intensity range, despite different size.

2.2.2 Methodology overview

The proposed methodology is composed of three consecutive parts, hereafter detailed:

1. **Profile similarity definition**: As frequently discussed in the literature [96, 98, 99, 104–106], the choice of a similarity measure that reflects the biochemical semantics of LC-MS data is essential to achieve efficient processing. In this article, we relied on *Wasserstein-*

1 distance [136–138] (or W1, detailed in Section 2.2.3.1) and we transformed it into a similarity by applying a negative exponential function: If x_i and x_j are two chromatograms (or columns from the data matrix), their similarity thus reads:

$$k(x_i, x_j) = e^{-\gamma \cdot \left[d_{W_1}(x_i, x_j)\right]^p} \tag{2.3}$$

where d_{W_1} is the W1 distance and where γ is a neighborhood parameter, which tuning authorizes up/down scaling the similarity values. The use of a similarity measure of the form of a negative exponential of a distance is convenient, since it makes it possible to apply the *kernel trick* [139] (see Section 2.2.3.2), *i.e.* to apply a machine learning algorithm as if it were operating in a so-called *feature space* (depicting a non-linear data transform which respects the semantic of the chosen similarity measure).

2. **Data compression**: Applying the kernel trick can be rather computationally demanding: For a dataset of size N, it requires the computation of a kernel (or similarity) matrix of size $N \times N$. Thus, with between 10^5 and 10^6 chromatograms in the Ecoli datasets, computing and storing the kernel matrix is simply not tractable. The purpose of *Nyström method* [140] (see Section 2.2.4.1) is to replace the kernel matrix by a low rank approximation, as illustrated in Figure 2.3. By relying only on the similarities between each data element and a randomly selected subset, it provides a dramatic reduction of the computational burden at the price of a small and controlled loss of accuracy. Even though Nyström approximation allows for an efficient computation of the kernel matrix, it does not accelerate the clustering algorithm itself, which requires multiple traversing of the entire dataset (*i.e.* N elements). To cope for this, it has recently been proposed in the compressive learning framework [80] to summarize the entire dataset by a relatively small vector of fixed size, referred to as *data sketch*, and to have the

algorithm operating on his sketch only, irrespective of the original data. Concretely, we built the data sketch as an average of *random Fourier features* of the chromatographic profiles in the feature space (see Section 2.2.4.2).

3. **Cluster and centroid definitions**: Lloyd algorithm [141] (*i.e.* the most classical algorithm to cluster data according to the k-means objective function) cannot directly be applied on sketched data. Fortunately, it is possible to rely on the *Compressive k-means* (CKM) algorithm proposed in [142] (see Section 2.2.5.1). However, CKM only returns a set of cluster centroids and does not cluster the data *per se*. Therefore, traversing the entire (original) dataset to perform the *assignment* of each chromatogram to its closest centroid (according to the W1 distance) is necessary (see Section 2.2.5.2). CKM complexity does not depend on the original data size (as it operates on the data sketch) which makes it well-scalable. However, its complexity grows rapidly with the number of clusters, which is an issue as thousands of clusters can be sought in LC-MS data. To cope for this, we implemented a *hierarchical clustering scheme*, where each cluster is recursively divided into a small number of sub-clusters until the desired number of clusters is obtained (see Section 2.2.5.2). This procedure provides a set of clusters with centroids only defined in the feature space. To recover the corresponding consensus chromatograms, one has to solve a *pre-image problem*. We practically did so by computing the mean of the elution profiles neighboring each centroid (see Section 2.2.5.3).

To the best of the authors' knowledge, this work is the first one to combine Nyström method and compressive learning with random Fourier features on a problem as difficult as the clustering of LC-MS data, which combines high-dimensionality and a very large number of potential clusters in addition to the traditional difficulties of raw biological data (non-linearities, low signal-to-noise ratio, *etc.*). From this point on, we refer to the proposed method as CHICKN (standing

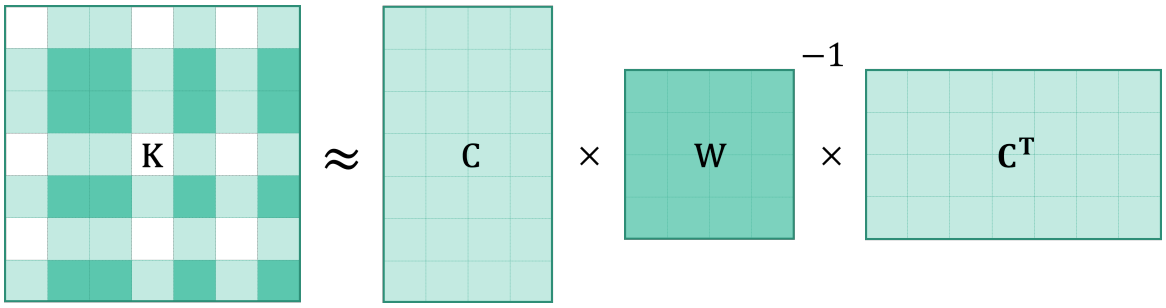


Figure 2.3: Nyström kernel approximation. The matrix C represents the similarity between each data point and the random sample. The matrix W corresponds to the pairwise similarity evaluation between selected data points.

for Chromatogram HIerarchical Compressive K-means with Nyström approximation).

2.2.3 Profile similarity definition

2.2.3.1 Metric choice

Originally, the Wasserstein-1 (W1) metric was defined to compute optimal transport strategies, which explains why it is also referred to as the *earth mover's distance*. It has witnessed a recent gain of interest in machine learning as an efficient way to measure a distance between two probability distributions [143, 144]: Essentially, if one sees probability distributions as earth heaps, the most energy efficient way to move one earth heap in place of the other makes an interesting distance estimate. In this work, we leveraged a similar analogy between an earth heap and a chromatographic elution profile. Concretely, this approach is insightful since it accounts for two distinct components of what makes chromatographic elution profiles similar or not: their time separation as well as their difference of shape. Let us also note that this distance has recently been applied to LC-MS data, yet, to spectra rather than to chromatograms [138].

In general, the W1 distance between distributions $\mathcal{P}$ and $\mathcal{Q}$ is com-

puted by solving Kantorovitch minimization problem, namely:

$$d_{W_1}(\mathcal{P}, \mathcal{Q}) = \inf_{\xi \in \mathcal{J}(\mathcal{P}, \mathcal{Q})} \int \|x - y\| d\xi(x, y) \tag{2.4}$$

where $\mathcal{J}(\mathcal{P}, \mathcal{Q})$ denotes all joint distributions $\xi(x, y)$ that have marginals $\mathcal{P}$, $\mathcal{Q}$. However, in the 1-dimensional discrete setting where distributions $\mathcal{P}$ and $\mathcal{Q}$ are replaced by chromatograms $x = (x^1, \ldots, x^n)$ and $y = (y^1, \ldots, y^n) \in \mathbb{R}^n$, the W1 distance boils down to a difference between empirical cumulative functions:

$$d_{W_1}(x, y) = \sum_{j=1}^{n} |F_x(j) - F_y(j)|, \tag{2.5}$$

where $F_x(j) = \sum_{i \leq j} \frac{x^i}{\sum_{k=1}^{n} x^k}$ is the j^{th} component of the cumulative distribution function of chromatogram x.

2.2.3.2 Kernel trick

Converting distances between data vectors into similarities by means of a negative exponential function is a good way to derive a similarity measure endowed with the positive semi-definite (or PSD) property[3]. This property is essential to the application of the kernel trick [145], which notably explains why kernels of the form $k(x_i, x_j) = e^{-\gamma \cdot [d_2(x_i, x_j)]^p}$, with $p = 1$ (the Laplacian kernel) or $p = 2$ (the Gaussian kernel) and with d_2 depicting the Euclidean distance are classically used.

Concretely, let $X = [x_1, \ldots, x_N] \in \mathbb{R}^{n \times N}$ be the data matrix composed of N chromatograms. The kernel trick actually consists in using the similarity measure to implicitly map the data onto a feature space that better represents them. The mapping is deemed "implicit" as it does not require the computation of coordinates of the data point images $\Phi = [\phi(x_1), \ldots, \phi(x_N)]$, where ϕ denotes the mapping function. Two

[3]Positive semi-definiteness or PSD-ness, means the resulting similarity matrix will have only non-negative eigenvalues (if the eigenvalues are positive, the matrix is called positive definite or PD, see Appendix B, Section B.1).

conditions must be met for this trick to work: First, the algorithm must rely on similarity measures only (*i.e.* once the similarities are computed, the values of the x_i's are not used any more). Second, the similarity measure reproduces the inner product of the feature space: $k(x, y) = \langle \phi(x), \phi(y) \rangle$. According to Mercer's theorem [146], any PSD similarity measure satisfies the second condition. From that point on, we refer to $K = \Phi^T \Phi = [k(x_i, x_j)]_{i,j=1,\ldots,N}$ as the *kernel matrix*.

However, when using a distance like d_{W_1}, which does not derive from a norm inducing an inner product on the data space (like for instance d_2), then the PSD-ness is not guaranteed [147]. In this work, we have investigated both the Laplacian W1 and the Gaussian W1 kernels: While we exhibit a formal proof of the Laplacian W1 kernel PD-ness (see Appendix B, Section B.3), we only have empirical evidence in the Gaussian case (see Appendix B, Section B.2). As in practice, both kernels lead to similar ranks in pairwise similarities, the resulting clusters only marginally differ. Owing to its popularity in life science applications, as well as to its easier tuning (interpretation and stability of the hyperparameter) the article thus focuses on the Gaussian case. Notably, as computational costs are necessarily higher with $p = 2$ than $p = 1$, the displayed runtimes are an upper bound for both cases. However, for qualitative analysis, results with $p = 1$ are also depicted in various figures (see below).

2.2.4 Data compression

2.2.4.1 Nyström approximation

Brute force computation of a kernel matrix has a quadratic complexity, so that it does not easily scale-up. To cope for this, a classical solution is to apply Nyström approximation. This approach relies on the fast decaying property of the kernel spectrum (the set of kernel matrix eigenvalues): the smallest eigenvalues of the kernel matrix can safely be removed (intuitively, alike principal component analysis). Concretely,

one approximates the kernel matrix $K \in \mathbb{R}^{N \times N}$ as following:

$$K \approx CW^{-1}C^{\top}, \tag{2.6}$$

with $C = KP \in \mathbb{R}^{N \times l}$ and $W = P^{\top}KP \in \mathbb{R}^{l \times l}$, where $P \in \mathbb{R}^{N \times l}$ is constructed from an $N \times N$ identity matrix where $(N - l)$ randomly selected columns are removed. The larger l, the better the approximation, but the heavier the computations. Finally, according to [140], an additional rank-s truncated singular value decomposition (SVD) is of interest to increase numerical stability. This leads to Algorithm 1, which complexity[4] is $\mathcal{O}(N \cdot n \cdot l + N \cdot l^2)$.

Algorithm 1 The rank restricted Nyström kernel approximation from [140]

1: **Input:** data set $X \in \mathbb{R}^{n \times N}$, similarity measure $k(\cdot, \cdot)$, Nyström sample size l, intermediate rank r, target rank s.
2: Construct a random sample: $\{x_{p_1}, \ldots, x_{p_l}\} \in \mathbb{R}^{n \times l}$
3: Compute matrix C and W: $C = \{k(x_q, x_{p_j})\}_{\substack{q=1,\ldots,N \\ j=1,\ldots,l}}$, $W = \{k(x_{p_i}, x_{p_j})\}_{i,j=1,\ldots,l}$.
4: Perform r-truncated SVD of W: $W_r = U_r D_r U_r^{\top}$.
5: Approximate matrix as $K \approx CW_r^{-1}C^{\top} = CU_r D_r^{-1} U_r^{\top} C^{\top} = RR^{\top}$, where $R \in \mathbb{R}^{N \times r}$.
6: Perform s-truncated SVD of R: $R = U_s \Sigma_s V_s^{\top}$.
7: **Output:** Matrix approximation $K \approx \widetilde{\Phi}^{\top}\widetilde{\Phi} = U_s \Sigma_s^2 U_s^{\top}$.

It provides the following approximation of the kernel matrix: $K \approx \widetilde{\Phi}^{\top}\widetilde{\Phi}$ where the matrix $\widetilde{\Phi} = \left[\tilde{\phi}(x_1), \ldots, \tilde{\phi}(x_N)\right]$ is obtained by applying the feature mapping $\tilde{\phi}(x_i) = (\lambda_1 u_{1i}, \ldots, \lambda_s u_{si})$, where λ_j and $u_{ji}, j = 1, \ldots, s$ and $i = 1, \ldots, N$ are the s highest eigenvalues and eigenvectors (columns of matrix U_s) of K (see Algorithm 1). Moreover, it is demonstrated in [148] that the approximation accuracy is guaranteed when Nyström sample size l is on the order of $\sqrt{N}$. It was also shown in [140] that the target dimension s scales to $\mathcal{O}(\sqrt{l \cdot k})$, where k is the number of clusters, and the intermediate rank r is equal to $\frac{l}{2}$.

[4] As a recall, $\mathcal{O}(f(n))$ indicates that with an input data of size n, the running time will not exceed $C.f(n)$ where C is a constant factor (*i.e.* independent of n).

2.2.4.2 Random Fourier feature sketching

The sketching procedure of [80] is closely related to random Fourier features [81], which seminal idea is to rely on Bochner's theorem [149] to approximate any shift-invariant (*i.e.* $k'(x,y) = \kappa(x-y)$) PD kernel (by leveraging the fact it is a Fourier transform of some non-negative measure μ):

$$k'(x,y) = \mathbb{E}_{w\sim\mu}\left(e^{-iw^\top(x-y)}\right). \tag{2.7}$$

Elaborating on this, [80] proposed to apply a similar random Fourier map

$$\varphi(x) = \frac{1}{\sqrt{m}}\left[e^{-iw_j^\top x}\right]_{j=1}^{m}, \tag{2.8}$$

(where Fourier frequencies $w_1,\ldots,w_m$ are randomly sampled from some distribution Ω) and to average it over all data points to approximate the data distribution itself, instead of the kernel. Concretely, applying $\varphi(\cdot)$ onto the Nyström extended data $\widetilde{\Phi}$ (that is $Z = [\varphi(\tilde{\phi}(x_1)),\ldots,\varphi(\tilde{\phi}(x_N))] \in \mathbb{C}^{m\times N}$), led us to computing the data sketch as:

$$SK(\widetilde{\Phi}) = \frac{1}{N\sqrt{m}}\left[\sum_{i=1}^{N} e^{-iw_j^\top \tilde{\phi}(x_i)}\right]_{j=1}^{m} \in \mathbb{C}^m \tag{2.9}$$

The critical step of this data compression method lies in the frequency distribution estimation. It has been empirically shown in [80] that $\Omega = \mathcal{N}(0, \frac{1}{\sigma^2}\mathbf{I})$ is a suitable choice for it mimicks well the fast decaying property of real life signals. Then, σ^2 can be estimated from a small data fraction using nonlinear regression. Applying this frequency distribution law allows to promote more informative sketch components and to eliminate small sketch values, which are usually related to noise. The key computational benefit of the compression is the independence between the data sketch length m and the data size N: m should be of the order of $k \cdot s$ [80], where s is the target dimension in Nyström approximation and k is the number of clusters.

2.2.5 Cluster and centroid definitions

2.2.5.1 Cluster computations

CKM (the compressive implementation of the k-means clustering presented in [142]) can be used to compute the cluster centroids from the data sketch $SK(\widetilde{\Phi})$ introduced in Eq. (2.9). Briefly, and in contrast with classical Lloyd's algorithm, it is a greedy heuristic based on orthogonal matching pursuit, which searches for a data representation as a weighted sum of cluster centroids by minimizing the difference between corresponding sketches:

$$\|SK(\widetilde{\Phi}) - \sum_{i=1}^{k} \alpha_i SK(c_i)\|_2^2 \tag{2.10}$$

The CKM involves two main steps summarized in Algorithm 2 (where $\Re$ denotes the real part of the complex number). First, across several iterations, it alternates between expanding the cluster centroid set with a new element, whose sketch is the most correlated to the residue; and recomputing the centroid weights using non-negative least-squares minimization. The second step consists in the global minimization of (2.10) with respect to cluster centroids and their weights.

2.2.5.2 Cluster assignment

The CKM algorithm only provides the cluster centroids and does not assign data points to clusters. Nevertheless, this can be achieved afterwards by finding the centroid which has the highest similarity value to each data point. Concretely, a cluster centroid c in the feature space can be defined using Nyström extension as follows:

$$c \approx \tilde{\phi}(y) = \Sigma_s^{-1} U_s^T k_c \tag{2.11}$$

where y is a cluster centroid in the input (chromatograms) space, and where $k_c = [k(x_1, y), \ldots, k(x_N, y)]$ is an unknown vector of similarities

Algorithm 2 Compressive k-means from [142]

Input: data sketch $SK(\widetilde{\Phi})$, frequency set $w_1, \ldots, w_m$, the number of centroids k, lower and upper bounds lb, ub of data $\widetilde{\Phi}$.
Initialization: $r = SK(\widetilde{\Phi})$, $C = \emptyset$
for t $\leftarrow$ 1 to $2k$ **do**
    Find new centroid: $c = \arg\max_{lb \leq c \leq ub} \Re \left\langle r, \frac{SK(c)}{\|SK(c)\|} \right\rangle$
    Expand centroid set: $C = \{C, c\}$
    if $t > k$ **then**
        $\beta = \arg\min_{\beta \geq 0} \|SK(\widetilde{\Phi}) - \sum_{i=1}^{|C|} \beta_i \frac{SK(c_i)}{\|SK(c_i)\|}\|^2$
        Choose centroids with k largest weights $C = \{c_{\beta_{i_1}}, \ldots, c_{\beta_{i_k}}\}$
    end if
    Project to find weights: $\alpha = \arg\min_{\alpha \geq 0} \|SK(\widetilde{\Phi}) - \sum_{i=1}^{|C|} \alpha_i SK(c_i)\|^2$
    Global optimization: $C, \alpha = \arg\min_{\substack{lb \leq c_i \leq ub \\ \alpha \geq 0}} \|SK(\widetilde{\Phi}) - \sum_{i=1}^{|C|} \alpha_i SK(c_i)\|^2$
    Update residue: $r = SK(\widetilde{\Phi}) - \sum_{i=1}^{|C|} \alpha_i SK(c_i)$
end for
Output: $C \in \mathbb{R}^{s \times k}$ and $\alpha_1, \ldots, \alpha_k$.

between y and all given chromatograms. The columns of matrix U_s contain s eigenvectors of K corresponding to its s highest eigenvalues (the diagonal matrix Σ_s). The estimation of k_c can be achieved by minimizing the difference between c and $\tilde{\phi}(y)$:

$$\min_{y \in \mathbb{R}^n} \left\| \Sigma_s^{-1} U_s^T k_c - \frac{c}{\|c\|} \right\|^2 \tag{2.12}$$

The importance of the normalization term in (2.12) has been highlighted in [150] as an energy-preserving term to balance Nyström approximation. The solution of (2.12) can be found using the Moore-Penrose pseudo-inverse:

$$k_c \approx U_s \Sigma_s \frac{c}{\|c\|} \approx \widetilde{\Phi}^T \frac{c}{\|c\|}. \tag{2.13}$$

Finally, the chromatographic profile $x_i,\ i = 1, \ldots, N$ is associated to cluster j if

$$c_j = \arg \max_{c \in \{c_1, \ldots, c_k\}} \left\langle \tilde{\phi}(x_i), \frac{c}{\|c\|} \right\rangle \tag{2.14}$$

The most important CKM feature is its constant execution time regardless of the data size. However, its computational complexity grows cubically with the number of clusters, so that it is not realistic to process LC-MS data where tens of thousands of clusters are classically expected. To cope for this, a divisive hierarchical scheme can be instrumental: Starting from a small number of clusters, one iteratively splits each cluster into k sub-clusters until a sufficiently large number of clusters k_{total} is achieved. However, this strategy requires, for each independent call of the clustering algorithm, an update of the data sketch as well as a complete assignment to clusters. Thus, to practically improve its computational efficiency, we leveraged the expected decrease of the cluster size at each iteration to optimize the code, and we decided to compute all the data sketches from the same frequency samples, either on the entire dataset (at first step) or on the cluster to be re-clustered (at the following iterations). Finally, it appeared these repetitive computations of the cluster sketches and assignments did not hamper the efficiency of the whole process.

2.2.5.3 Pre-image computation

The combination of Nyström approximation and of random Fourier features leads to an additional difficulty: To recover the signal of each consensus elution profile, it is necessary to compute its reverse mapping from the feature space back to the input space. This is referred to as a *pre-image* problem and it is ill-posed: only an approximation of the cluster centroids in the input space can be obtained. The conventional fixed point iteration method [151] cannot be applied due to the use of the W1 distance. Similarly, the reconstruction of a consensus chromatogram as the mean of the cluster chromatograms is not adapted, due to large

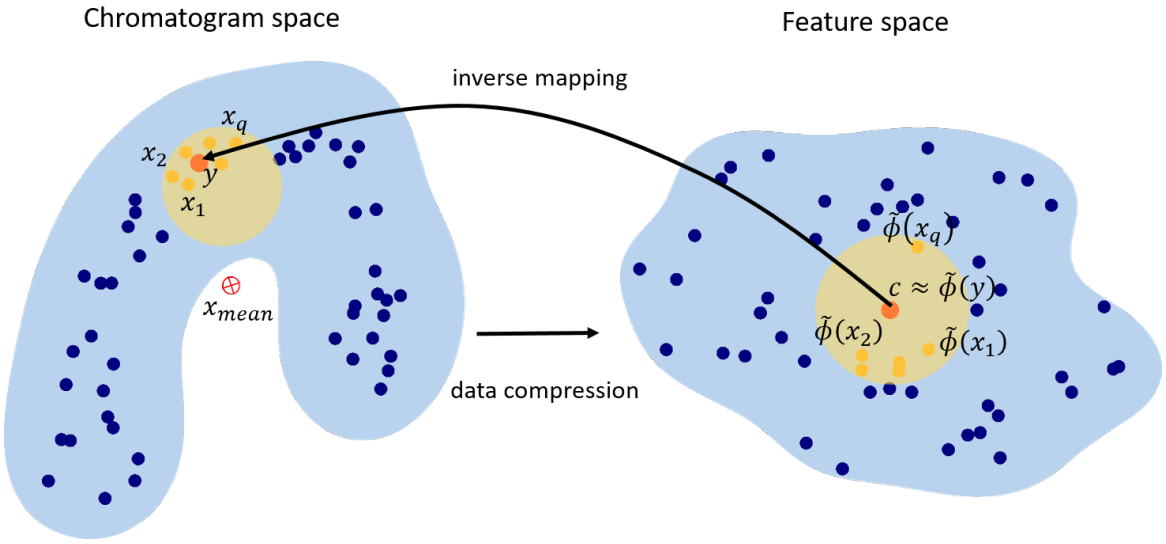


Figure 2.4: Pre-image problem illustration. Consensus chromatogram construction amounts to solve a pre-image problem, *i.e.* to map the feature space (right) back to the space of chromatograms (left). Blue points depict the elution profiles (left) and their images in the feature space (right). The red points are the cluster centroid (right) and the corresponding consensus chromatogram (left). The yellow circles represent the cluster centroid and consensus chromatogram neighborhoods. Due to the mapping non-linearity, the mean chromatogram may lie outside the cluster, while the correct consensus chromatogram should belong to it.

scale non-linearities between the input and feature spaces, as illustrated in Figure 2.4.

To correct for this, we decided to compute a local (*i.e.* small-scale) mean by considering only a subset of the closest chromatograms. To determine the cluster centroid neighbourhood $\mathcal{N}(c)$, we proceeded similarly to the cluster assignment step, by choosing the chromatograms in the cluster $\mathcal{J}(c)$ with the highest similarities to the cluster centroid:

$$\mathcal{N}(c) = \{x_1, \ldots, x_q\} \subset \mathcal{J}(c) \mid k(c, x) > k(c, y) \quad \forall x \in \mathcal{N}(c), y \in \mathcal{J}(c) \setminus \mathcal{N}(c), \quad (2.15)$$

where similarities $k(c, \cdot)$ were estimated using Eq. (2.13). Concretely, $\mathcal{N}(c)$ was defined by selecting the q closest neighbors (so that $q = |\mathcal{N}(c)|$). The tuning of parameter q is discussed with that of other parameters in Section 2.3.3.

2.2.6 Performance metrics

For experiments annotated with a ground truth (like UPS2GT dataset), clustering accuracy can be evaluated with the Rand index (RI). The Rand index measures the percentage of correctly clustered pairs of signals over the total number of pairs. Let us denote as $U = \{U_1, \ldots, U_k\}$ the obtained clusters and as $V = \{V_1, \ldots, V_q\}$ the ground truth clusters. A pair of signals is considered as correctly clustered: *true positive (TP)* or *true negative (TN)*, if signals are assigned to the same cluster in U and V or on the contrary, to different clusters in U and V. A pair of signals is called *false positive (FP)* (resp. *false negative (FN)*), if signals are grouped in U (resp. V) but not in V (resp. U). Then, the Rand index is given by:

$$\mathrm{RI} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.16}$$

The maximum value of the Rand index is 1 (perfect match with the ground truth). Additionally, it is possible to evaluate how often different chromatograms are grouped in the same cluster; and how often similar chromatograms were assigned to different clusters. To do so, one classically relies on the Precision and Recall metrics, respectively:

$$\mathrm{Precision} = \frac{TP}{TP + FP} \qquad \mathrm{Recall} = \frac{TP}{TP + FN} \tag{2.17}$$

For datasets without ground truth annotation (like both Ecoli datasets), it is possible to rely on the Davies - Bouldin (DB) index. Let us denote as $\mathcal{J}(c_j)$ the j^{th} cluster with the cluster centroid c_j, and as $\{\mathcal{J}(c_1), \ldots, \mathcal{J}(c_k)\}$ the set of obtained clusters. The within cluster distance reads:

$$S_j = \frac{1}{|\mathcal{J}(c_j)|} \sum_{x_i \in \mathcal{J}(c_j)} d_{W_1}(x_i, c_j) \tag{2.18}$$

The DB index is defined through the ratio of the within cluster distances

to the between cluster distance $d_{W_1}(c_i, c_j)$:

$$DB = \frac{1}{k} \sum_{i=1}^{k} \max_{i \neq j} \frac{S_i + S_j}{d_{W_1}(c_i, c_j)}, \tag{2.19}$$

It should be noted that the distance metric in the DB index and in the clustering algorithm must be the same, in our case the W1 distance in the original space. Moreover, the smaller the DB index, the better the clustering (as a good clustering minimizes cluster overlaps).

Finally, the computational load can easily be approximated by the recorded execution time, *i.e.* the difference between the end and start times, both of which being accessible in `R` with the `Sys.time()` function. For sake of brievety, execution times are reported for the Gaussian W2 kernel only, as Laplacian similarities are necessarily faster to compute (no squared distance to evaluate).

2.3 Results

2.3.1 Objectives of the experimental assessment

Many independent elements deserve evaluations: The first one is the practical interest of W1 distance in the context of LC-MS data. The second one is the computational load of our complete algorithm in function of the parameter tuning (on the one hand, an efficient compression technique is used; on the other hand, one targets the clustering of raw data into a high number of clusters, making its efficiency a challenge). The third one is the clustering result itself. However, a classical evaluation of the clustering performances will be of little interest: In fact, all k-means related algorithms (including their kernelized versions) have been extensively studied [131], so that their strengths and weaknesses are now well-documented. For instance, k-means optimizers can easily be trapped into local minima and cannot naturally deal with outliers, which are both significant drawbacks; however, they scale up well to

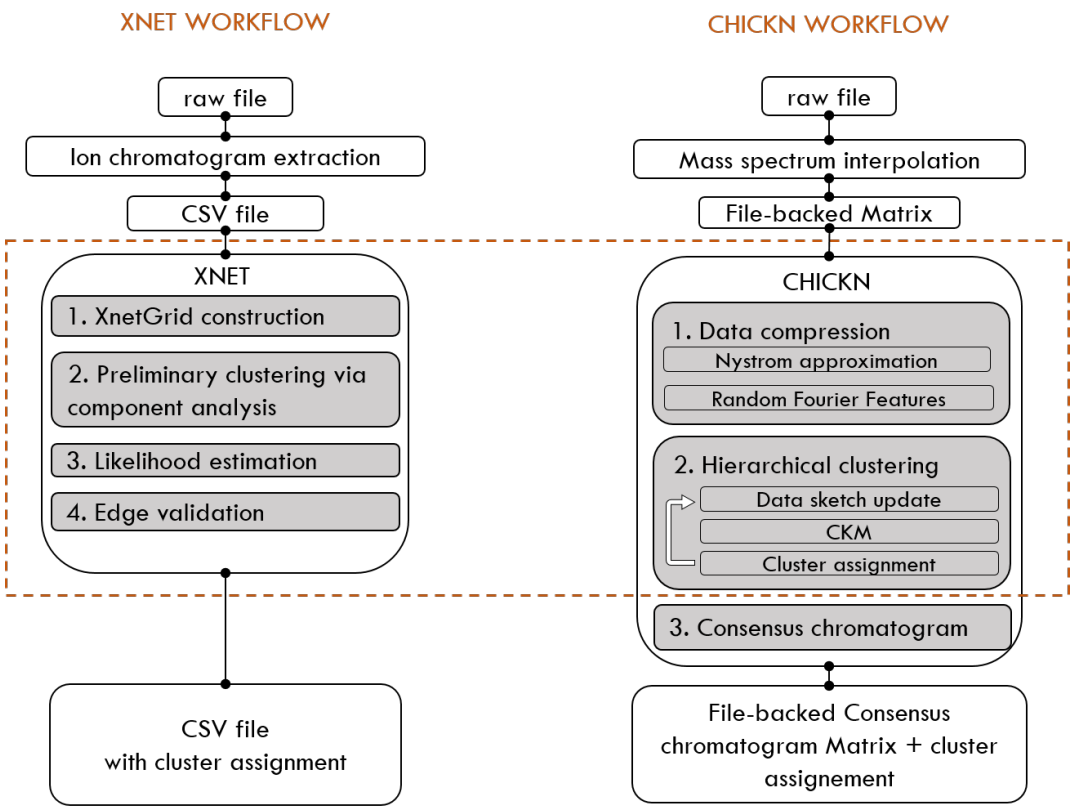


Figure 2.5: Xnet and CHICKN worflow comparison. To allow for fair comparisons, we have focused on the core algorithms, depicted within the dotted rectangle.

very-high dimensional data, which definitely is an asset for LC-MS applications. In contrast, highlighting the differences of our approach with respect to linkage-based agglomerative clustering and showing that despite noticeable differences, one obtains clusters which are meaningful, is of real practical interest to computational mass spectrometry experts.

As reported in Section 2.1, comparisons with Xnet is mandatory. However, considering the reported specificities (trace extraction preprocessing, envelope assumption simplification, *etc.*), comparing Xnet and CHICKN workflows may appear as somewhat arbitrary.

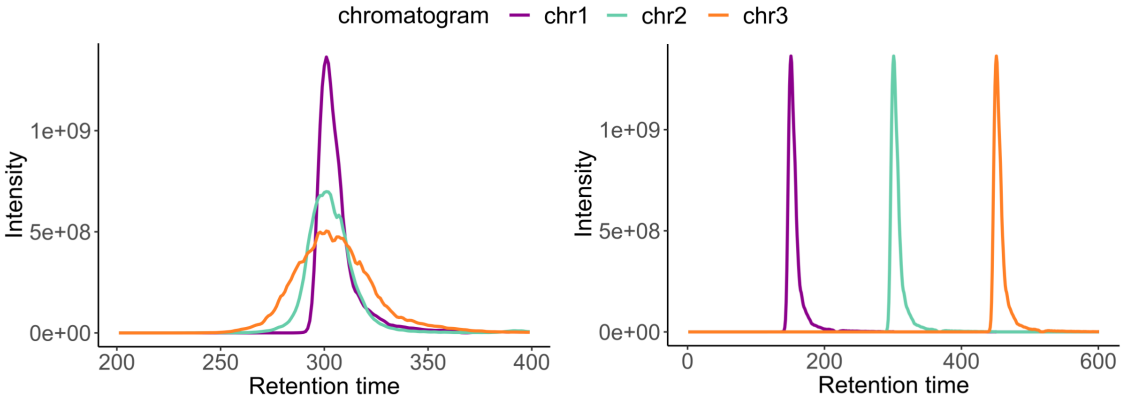
To cope for this, we have made the following choices: First, we have focused on the core of each algorithm, as represented in Figure 2.5. Second, we have adapted the UPS2GT and Ecoli datasets to be processed by each algorithm: The UPS2GT data are already formatted into a CSV file meeting Xnet requirements.

To construct a data matrix suitable to CHICKN from the UPS2GT

data, we simply loaded the data points according to their retention time and trace labels in the matrix columns (similarly to Xnet, we excluded point with trace indices -1 and 0, as assumed to be noise). This led to a data matrix containing 57,140 columns and 6,616 rows. Conversely, to build the CSV files from Ecoli datasets, we stored any non-zero entry of the data matrix in a row, the column index being used in place of the trace labels.

2.3.2 Wasserstein distance validation

W1 distance was proposed to discriminate between signals that represent different elution profiles. To assess this choice, we compared it with two distances amongst the most widely used in mass spectrometry signal processing: The first one is the classical Euclidean distance. The second one is the peak retention time difference (or ΔRT): It corresponds to the difference between the time stamps at which each signal reaches its highest intensity value. Based on the Ecoli-FMS dataset (which provides the finest temporal sampling), we examined two situations presented in Figure 2.6: In the first one, we selected 3 signals with different shapes, that we precisely aligned so that their pairwise ΔRT was zero; in other words, only the shape difference makes it possible to discriminate them. Conversely, in the second situation, an elution profile was translated to mimic a case where only the ΔRT was meaningful. In both situations, the second chromatogram (chr2) stands as an in-between the first (chr1) and the third chromatogram (chr3). As illustrated by the distance ratios given in the tables embedded in Figure 2.6, both the Euclidean and the ΔRT distances are meaningful in one case: The Euclidean distance captures the shape information, while ΔRT captures the time translation effect. However, none of these classically used distances is able to capture both the shape and the translation simultaneously. On the contrary, W1 distance is efficient on both situations, making it a suitable distance to construct a similarity measure adapted to LC-MS data.



Distance type	Distance value	Reflect the shape difference
Wasserstein-1	$\frac{d_{W_1}(\text{chr1},\text{chr2})}{\text{d}_{W_1}(\text{chr1},\text{chr3})} = 0.4$ $\frac{d_{W_1}(\text{chr2},\text{chr3})}{\text{d}_{W_1}(\text{chr1},\text{chr3})} = 0.86$	☑
Euclidean	$\frac{d_E(\text{chr1},\text{chr2})}{\text{d}_E(\text{chr1},\text{chr3})} = 0.63$ $\frac{d_E(\text{chr2},\text{chr3})}{\text{d}_E(\text{chr1},\text{chr3})} = 0.47$	☑
ΔRT	$d_{\Delta RT}(\text{chr1},\text{chr2}) = 0$ $d_{\Delta RT}(\text{chr2},\text{chr3}) = 0$	☒

Distance type	Distance value	Reflect the time separation
Wasserstein-1	$\frac{d_{W_1}(\text{chr1},\text{chr2})}{\text{d}_{W_1}(\text{chr1},\text{chr3})} = 0.5$ $\frac{d_{W_1}(\text{chr2},\text{chr3})}{\text{d}_{W_1}(\text{chr1},\text{chr3})} = 0.5$	☑
Euclidean	$\frac{d_E(\text{chr1},\text{chr2})}{\text{d}_E(\text{chr1},\text{chr3})} = 1$ $\frac{d_E(\text{chr2},\text{chr3})}{\text{d}_E(\text{chr1},\text{chr3})} = 1$	☒
ΔRT	$\frac{d_{\Delta RT}(\text{chr1},\text{chr2})}{\text{d}_{\Delta RT}(\text{chr1},\text{chr3})} = 0.5$ $\frac{d_{\Delta RT}(\text{chr2},\text{chr3})}{\text{d}_{\Delta RT}(\text{chr1},\text{chr3})} = 0.5$	☑

Figure 2.6: Distance metrics for chromatographic data analysis. Comparison of Wasserstein-1, Euclidean and RT difference distances on real chromatographic profiles from the Ecoli-FMS dataset.

2.3.3 Parameter tuning

Unlike Xnet, CHICKN is governed by eight parameters. Four of them are involved in the data compression: Nyström sample size (l), target rank (s), kernel parameter (γ) and sketch size (m). Three parameters are involved in the hierarchical clustering: number of clusters at each iteration of the hierarchical clustering (k), upper bound of the total number of expected clusters (k_{total}) and maximum number of levels in the hierarchy (T). The remaining parameter is the neighbourhood size in the consensus chromatogram computation (q). However, all parameters except γ and q are interrelated (see Section 2.2.4 as well as [140,148]) and

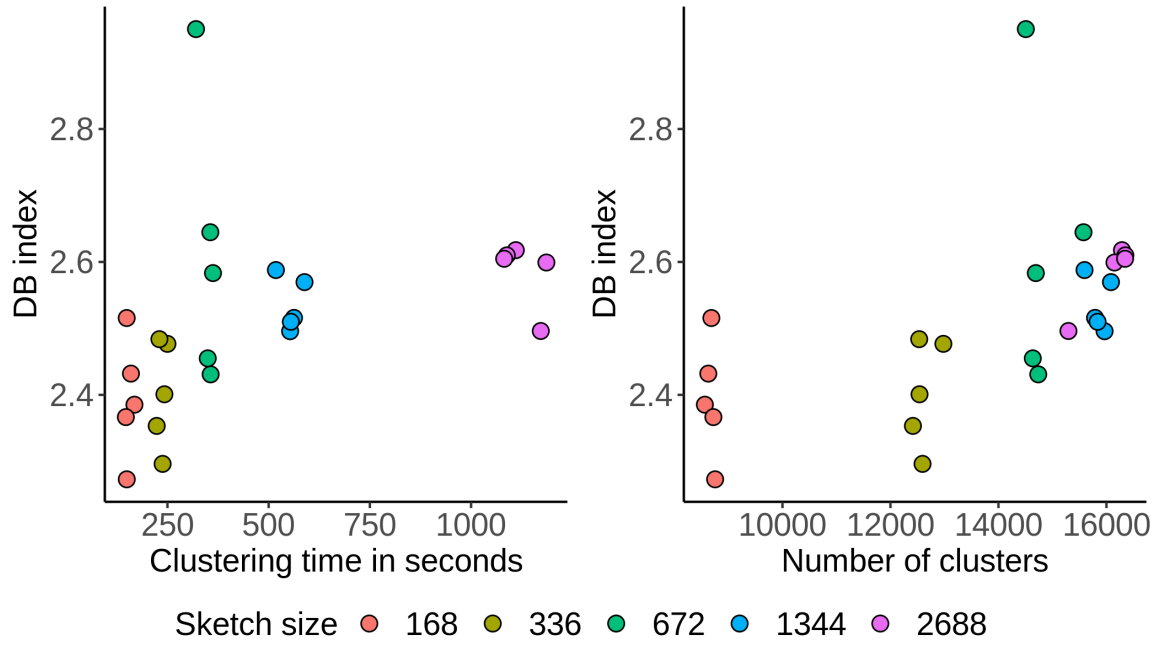


Figure 2.7: Influence of the sketch size on performances clustering of the Ecoli-DIA dataset, in function of the computational cost and the number of clusters.

can be expressed through k, k_{total} and N (the dataset size) as follows:

$$\begin{aligned} l &\approx \sqrt{N}, \\ s &\approx \sqrt{k} \cdot N^{1/4}, \\ m &\approx k^{3/2} \cdot N^{1/4}, \\ T &= \lfloor \log(k_{\text{total}}, k) \rfloor. \end{aligned} \tag{2.20}$$

These theoretical results can nonetheless be discussed. Notably, tuning the sketch size m to a larger value may be of interest if contrarily to our case, the computational efficiency is not the only targeted goal. Thus, we have performed complementary investigation to relate the clustering performance (in terms of DB index) to the sketch size (see Figure 2.7, leftmost figure). Oddly enough, it appears the DB index increases (*i.e.* the performances deteriorates) when the sketch size increases (leading to a more refined representation of the data). However, it appears to be an indirect consequence: when increasing m, more differences between the signals are represented, making it possible to define a larger number of smaller clusters (see Figure 2.7, rightmost figure).

Dataset	ν	γ	$\sigma = \sqrt{\frac{1}{2\cdot\gamma}}$	$\log_{10}\sigma$
Ecoli-DIA	8	1.78e-05	167.63	2.22
	16	1.32e-05	194.32	2.29
	32	9.35r-06	231.29	2.36
	64	6.13e-06	285.48	2.46
	128	3.50e-06	377.82	2.58
Ecoli-FMS	8	1.77e-06	532.10	2.73
	16	1.36e-06	606.41	2.78
	32	9.35e-06	231.29	2.85
	64	6.13e-06	285.48	2.92
	128	4.69e-07	1032.33	3.01

Table 2.1: Gassian W1 kernel hyperparameter γ stability with respect to the neighborhood maximum size ν.

Finally, four parameters remain (γ, q, k and k_{total}). Concretely, we tuned the kernel parameter γ as an average of the power of p distances to the ν nearest neighbors for all chromatographic profiles:

$$\gamma = \frac{1}{N \cdot \nu} \sum_{i=1}^{N} \sum_{j=1}^{\nu} [d_{W_1}(x_i, x_{i_j})]^p, \tag{2.21}$$

where $x_{i_1}, \ldots, x_{i_\nu}$ are ν neighbors of x_i (selected among the l points of the Nyström sample) and $p \in \{1, 2\}$ depending on the kernel type.

Practically, we observed that tuning ν to 32 guaranteed each data point to be sufficiently connected to the rest of the dataset, as advised in [125]. Moreover, we observed that γ was rather stable with respect to ν, for both Laplacian W1 and Gaussian W1 kernels. However, as expected, the stability is higher with the latter than with the former (see Table 2.1 and Table 2.2).

For q (in the consensus chromatogram computation) we observed that the shape cluster problem (see Section 2.2.5.3) could only occur with

Dataset	ν	γ	$\sigma = \frac{1}{\gamma}$	$\log_{10}\sigma$
Ecoli-DIA	8	3.21e-03	311.67	2.49
	16	3.76e-03	266.06	2.42
	32	2.31e-03	433.62	2.64
	64	1.85e-03	539.49	2.73
	128	1.39e-03	719.38	2.86

Table 2.2: Laplacian W1 kernel hyperparameter γ stability with respect to the neighborhood maximum size ν.

significantly large clusters (few tenth of elements). Thus, as preliminary stability analysis indicated us that the consensus chromatogram shapes were preserved across various values of ν (see Figure 2.8), we decided to bound q with ν and to set $q = \min(\nu, \text{cluster size})$.

A known drawback of k-means objective function is the requirement to set the maximum number of expected clusters (knowing some clusters can remain empty). In our case, this is achieved by tuning k and k_{total}. Yet, it should be noted that increasing k leads to decreasing T for a fixed value of k_{total} so that a trade-off between T and k must be sought. With this respect, we have evaluated different scenarios with $k = 2$, 4, 8 and 16. CHICKN execution times (excluding the data compression step, which remains constant whatever the various scenario) on the smallest (UPS2GT) and largest (Ecoli-FMS) datasets are depicted in Figure 2.9. This experiment pointed out the importance of tuning k to a small enough value, which is coherent with the observation that the original CKM algorithm does not scale up well with the number of clusters. Practically, working with $k = 2$ or 4 appeared to be the most efficient.

In the case of UPS2GT, the expected number of isotopic envelopes is known (*i.e.* 14,076). Thus, it is easy to tune k_{total} accordingly (*i.e.* $2^{14} = 4^7 = 16,384$). However, knowing that CHICKN does not rely on the envelope assumption simplification, it can be expected to find

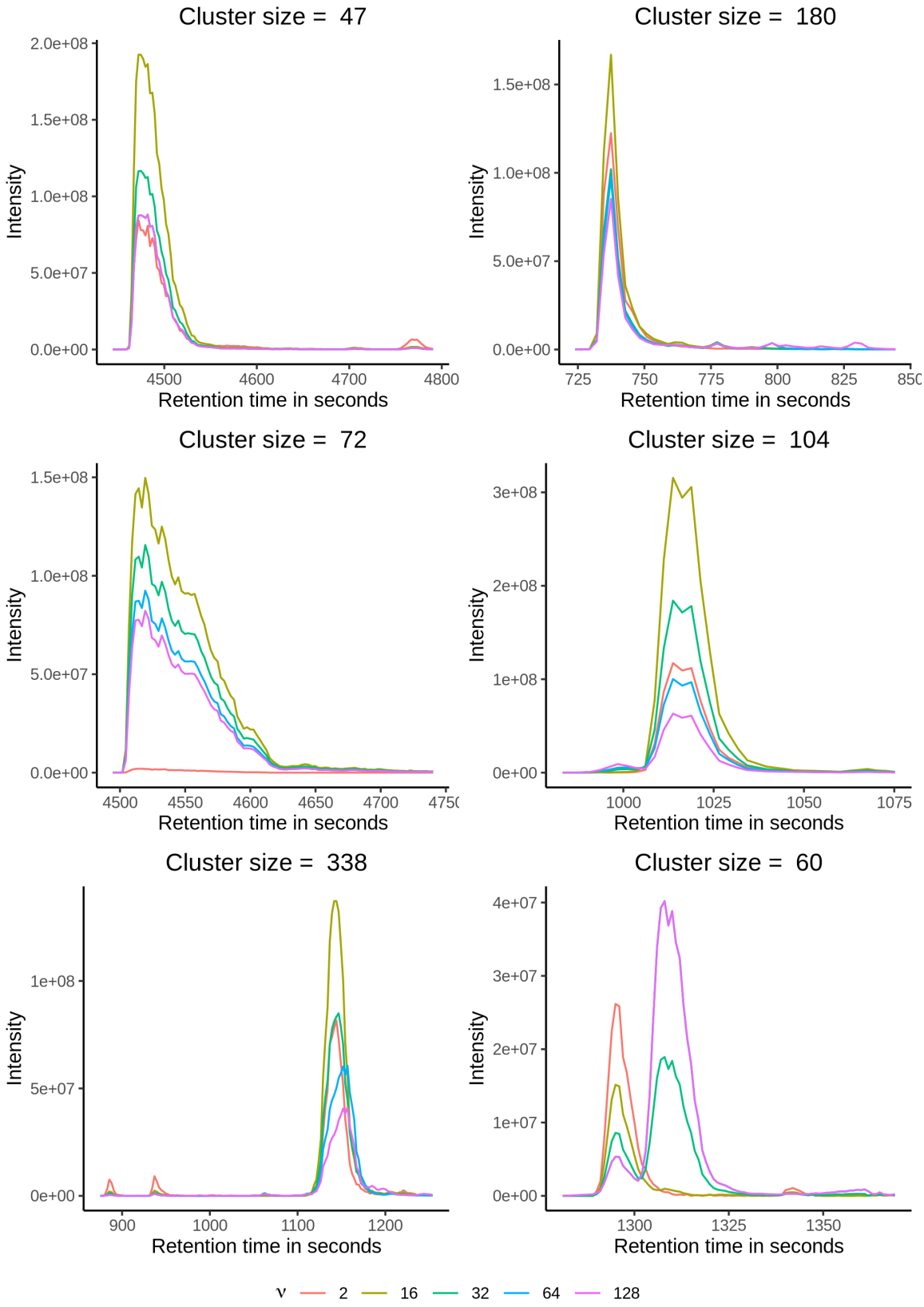


Figure 2.8: Consensus chromatogram stability. A set of 6 figures illustrating the stability of the pre-image computation through the averaging of a neighborhood of varying size ν.

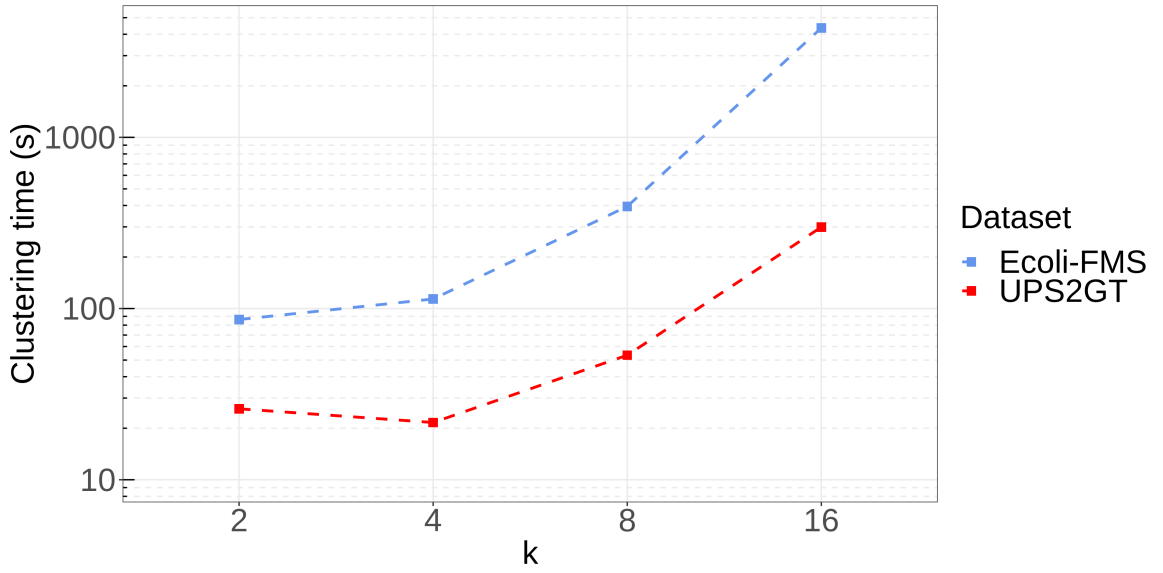


Figure 2.9: CHICKN execution time as a function of k, the number of clusters at each iteration, for both UPS2GT (blue) and Ecoli-FMS (red) datasets.

a much lower number of clusters: broadly, all the isotopic envelopes corresponding to different charge states of a same peptide can be expected to cluster together. Therefore, it also makes sense to tune k_{total} to $4^5 = 1,024$; *i.e.* close enough from the expected number of identifiable peptides in the sample (around 700, according to [26]).

Tuning k_{total} for any real life data (*i.e.* unlabeled) is much more complicated. However, the *Escherichia Coli* sample is well studied, and based on prior biological/analytical knowledge, 15,000 different peptides can be expected, broadly. Consequently, for both Ecoli datasets, $k_{\text{total}} = 16,384$ seems reasonable. Finally, even though it is not as sensible from a biological viewpoint, we have decided to also consider $k_{\text{total}} = 4^6 = 4,096$, which provides an even ground for computational load comparisons (see next section for details).

To summarize, three different ways to tune k_{total} are insightful: 1,024 for the UPS2GT dataset only (as it matches the number of expected peptides); 4,096 on all datasets (for computational benchmarcks); and 16,384 on all datasets (number of isotopic envelopes in UPS2GT and number of expected peptides in Ecoli datasets).

Finally, we fixed the remaining parameter values using the formulas

Parameter \ Dataset		UPS2GT		Ecoli-DIA		Ecoli-FMS	
γ		6e-06	7e-06	9.1e-06	9.3e-06	7.1e-07	7e-07
l		240	240	432	432	863	863
s		22	31	30	42	42	59
m		44	124	60	168	84	236
k		2	4	2	4	2	4
T	$k_{\text{total}} = 1,024$	10	5	-	-	-	-
	$k_{\text{total}} = 4,096$	12	6	12	6	12	6
	$k_{\text{total}} = 16,384$	14	7	14	7	14	7

Table 2.3: Summary of the different combinations of parameter tuning.

in Eq. (2.20), as summarized in Table 2.3.

2.3.4 Computational load

We have compared the execution times of CHICKN and Xnet cores (see Figure 2.5). Previously reported comparisons showed us that CHICKN execution time largely depends on k. However, it only has a sub-linear complexity with respect to k_{total}: As illustrated in Figure 2.10, multiplying k_{total} by 4 only results in a threefold (resp. twofold) increase in the CHICKN run-time for the Ecoli-FMS (resp. UPS2GT) dataset. As reducing k_{total} to limit the execution time will therefore be of little interest, experiments hereafter reported only focused on the influence of k. Despite CHICKN being more efficient when run with $k = 2$ and 4 (see Section 2.3.3), we also included comparisons with $k = 8$ and 16 to investigate the consequences of sub-optimal parameter tuning. The corresponding tests are referred to as CHICKN2, CHICKN4, CHICKN8 and CHICKN16. Therefore, to rely on an even basis for comparisons, we focused on $k_{\text{total}} = 4,096$: it is a power of 16, contrarily to 1,024 and

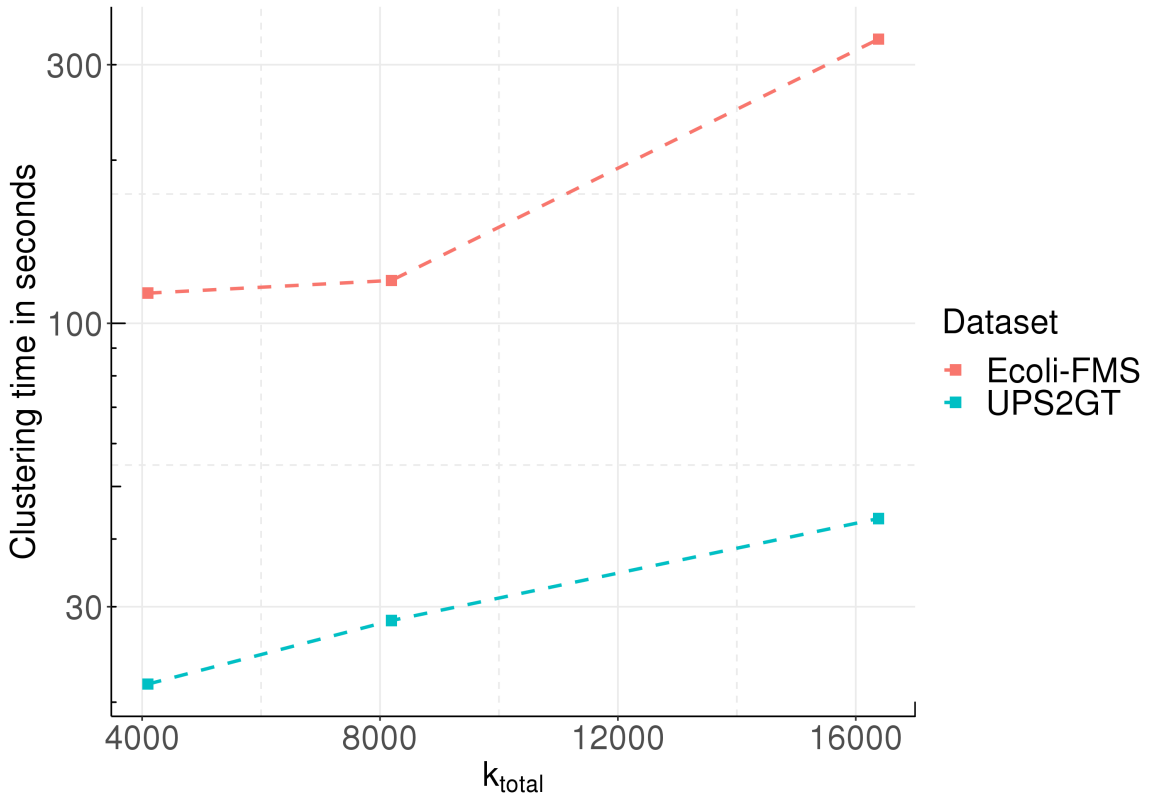


Figure 2.10: CHICKN execution time as a function of k_{total}, the maximum number of clusters, for both UPS2GT (blue) and Ecoli-FMS (red) datasets.

16,384 (which are even not a power of 8).

Since CHICKN algorithm embeds a Compressive k-means algorithm which may converge towards different local minima depending on the stochasticity of several steps, each scenario was repeated 10 times and the average execution time was reported. In contrast, Xnet being deterministic, it was executed once. In [53], Xnet exhibits impressive computational times on pre-processed and adequately formatted data. However, raw LC-MS data stored in a matrix format are more cumbersome. Thus, our first experiment was to compare the efficiency of Xnet and of CHICKN on the Ecoli-DIA dataset, using a laptop machine with the following characteristics: HP Pavilion g6 Notebook PC with Intel(R) Core(TM) i5-3230M CPU @ 2.60GHz, 8 Gb of RAM, 4 cores, running under Ubuntu 18.04.4 LTS OS. Xnet produced an "out-of-memory" error when trying to cluster more than 10,000 columns (*i.e.* 5% of the Ecoli-DIA dataset) in a single batch. This is why Figure 2.11A compares

the computational time of CHICKN2, CHICKN4, CHICKN8 and of CHICKN16 on the entire Ecoli-DIA dataset to that of Xnet on only 5% of the same dataset. On this figure, different colors are used to discriminate between the clustering step *per se* and CHICKN preliminary data compression step. Let us note that the compression step is time consuming, however, it also includes the computations of all the W1 similarities. This as-a-matter-of-factly illustrates the computational cost of relying on more elaborated metrics to capture the semantics of data as complex as LC-MS ones. Except for CHICKN16, which has already been pointed as suboptimal, CHICKN is always faster for a dataset 20 times larger.

This first experiment clearly showed CHICKN could be used on a simple laptop, even with large datasets, in long but acceptable times (half an hour to two hours, broadly). Then, to reduce the execution times of our multiple experiments, but also to allow Xnet working on a larger dataset, we moved to a larger station using 10 cores of an Intel Xeon CPU E5-2470 v2 @ 2.40GHz, 94 GB of RAM and running with CentOS Linux release 7.4.1708. As depicted in Figure 2.11B, on such a machine, CHICKN was able to process Ecoli-FMS within 5h30 (most of them being necessary to perform the preliminary compression), despite its huge size. On the contrary, with the same machine, Xnet only processed 10% of it in a comparable time (almost 8 hours). Moreover, larger fractions of the dataset were not processable, as leading to memory failure.

To explain this discrepancy, we noticed that Xnet spent a considerable time to construct the preliminary network. The nature of Ecoli data (raw data without any trace pre-processing and recorded with the highly resoluted *profile* mode, see Section 2.2.1) contrasts with that of UPS2GT, on which Xnet is really efficient. As it appears on Figure 2.11C, CHICKN is clearly not as fast as Xnet to process UPS2GT: The Xnet analysis took less then 40 seconds, while CHICKN computation times varied from 2 to 7 minutes depending on values of parameter k (from 2 to 16).

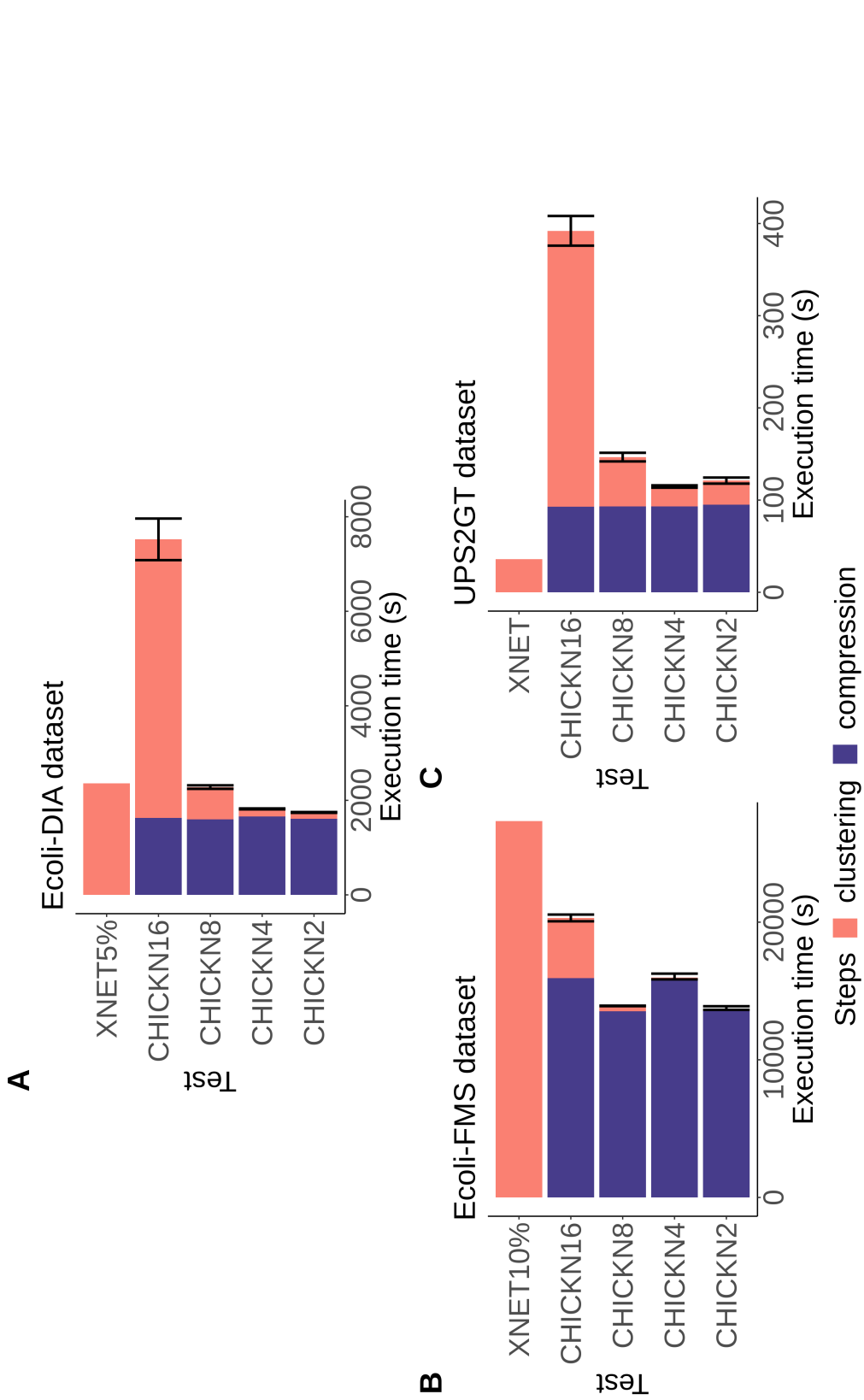


Figure 2.11: The execution time comparison for Ecoli (A-B) and for the UPS2GT (C) datasets. The CHICKN execution time is decomposed into the data compression time (blue) and the clustering time (pink). Note that XNet had to be run on 5% of the Ecoli-DIA dataset and 10% of the Ecoli-FMS dataset only, to avoid "out of memory" issues. The experiments on Ecoli-DIA were performed on a laptop, while other datasets were processed with a multi-core machine.

As a whole, these experiments illustrate the utmost importance of prior preprocessing methods when studying LC-MS data. In this context, algorithms working on raw data, such as CHICKN, are real assets.

2.3.5 Cluster evaluation

Figure 2.12 reports the Rand index, Precision and Recall (UPS2GT dataset) as well as the DB index (Ecoli datasets) with different clustering strategies: CHICKN2 and CHICKN4 (with $k_{\text{total}} \in \{1,024\,;4,096\,;16,384\}$ and with $p = 2$), as well as Xnet (on UPS2GT only, for computational reasons). A similar statistical result analysis for $p = 1$ is available in Figure 2.13.

First, it can be noted that the Rand index is hardly informative (Figure 2.12A): All clustering methods exhibit an index of almost 1, and it is necessary to go three (and sometimes four) decimals to notice a difference. Such high values are a direct consequence of the huge number of expected clusters in UPS2GT datasets, which comes with an excessively large number of true negative pairs (almost 99 % of all possible pairs). In this context, the Rand index obtained with "only" 1,024 expected clusters is particularly highlighting: Despite 16 times less clusters, it achieves an equivalent index. This indicates that, relatively, the provided clustering is probably of better quality.

However, contrarily to the Rand index, Precision and Recall are informative to compare with Xnet, as the true negative pair count does not level the scores. With this regard, it clearly appears on Figures 2.12B that the Precision is incomparably better with Xnet. Although foreseeable (ground truth with 14,076 envelopes whereas CHICKN sought a thousand of peptides), this requires a deeper analysis: Concretely, Xnet tends to over-cluster (which artificially improves the Precision index), as it provided 17,153 clusters covering 93% of the dataset (7% of the elution profiles are excluded by Xnet) where the ground truth labels proposed only 14,076 of them (on 100% of the dataset). In addition, Xnet priors were trained on the same UPS2GT dataset as for evaluation, so that

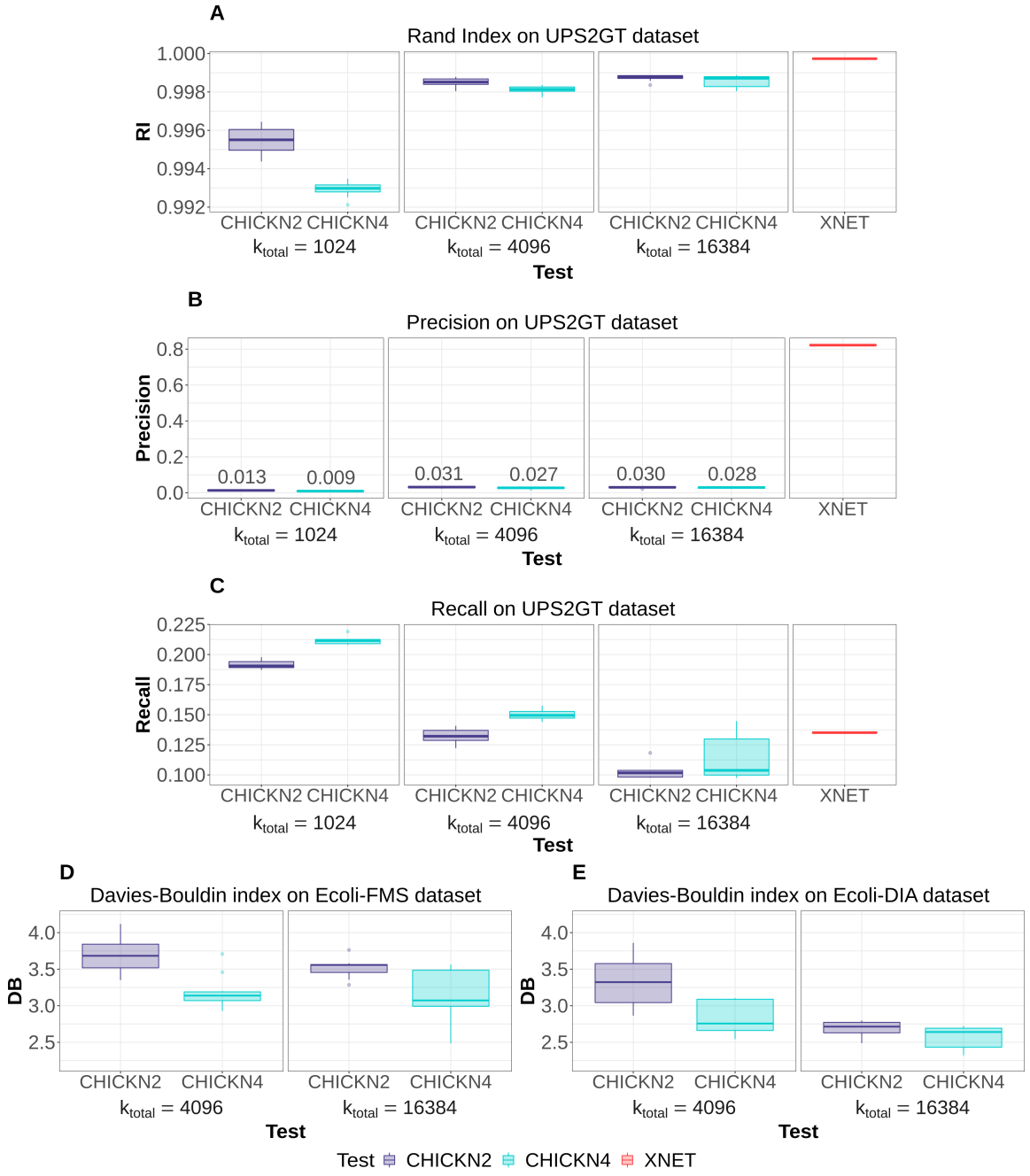


Figure 2.12: Statistical result analysis. CHICKN tests are performed with the Gaussian W1 kernel. (A) Rand index, (B) Precision, (C) Recall and (D-E) DB index depending on the k and k_{total} parameters; CHICKN2 and CHICKN4 tests are depicted in purple and light blue respectively; For the UPS2GT dataset, additional comparisons with Xnet (in red) are provided.

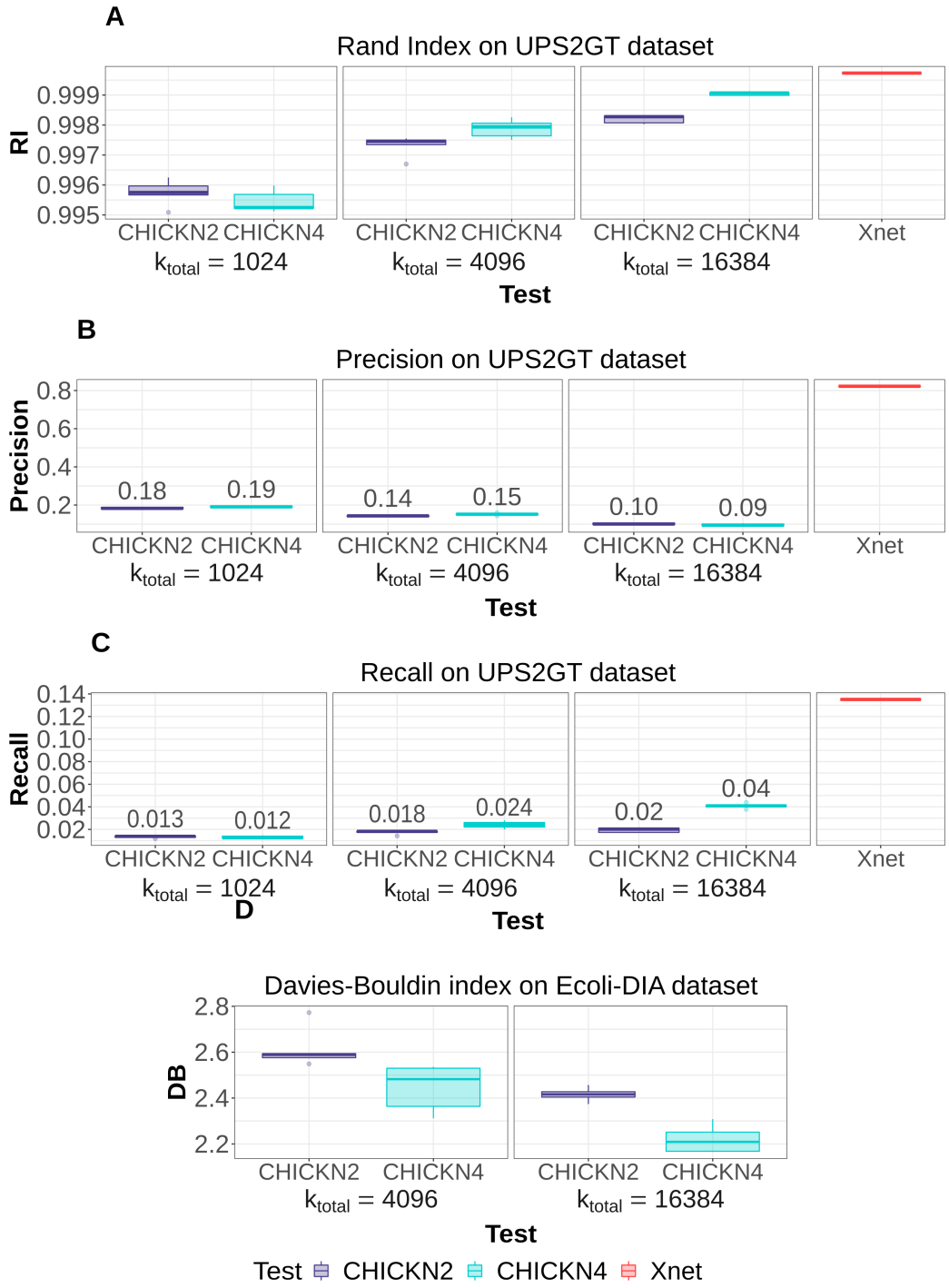


Figure 2.13: Statistical result analysis. CHICKN tests are performed with the Laplacian W1 kernel. (A) Rand index, (B) Precision, (C) Recall and (D) DB index depending on the k and k_{total} parameters; CHICKN2 and CHICKN4 tests are depicted in purple and light blue respectively; For the UPS2GT dataset, additional comparisons with Xnet (in red) are provided. The performance on the UPS2GT dataset are a bit lower than with the Gaussian W1 kernel (equivalent Rand index, better precision, lower recall), making it unable to compete with Xnet. However, on raw data such as Ecoli-DIA (*i.e.* on data CHICKN should work with), the Laplacian W1 kernel exhibit slightly better DB index than its Gaussian counterpart; however, this is hardly significant, making us conclude that strict performance should not be the criterion to chose the kernel.

high performance are expectable. With this regard, it is particularly noteworthy that the Recall (Figures 2.12C) varies the other way around. Concretely, it is best for CHICKN4 with $k_{\text{total}} = 1,024$ despite this number being completely different from the one derived from the ground truth. In addition to be in line with our observations on the Rand index, this concurs with the peptide-level knowledge of the dataset: CHICKN was supposed to group together differently charged peptides, which it did (see Figures 2.14 and 2.15 as well as Section 2.4 below), as it provided only 510 (CHICKN4)/ 740 (CHICKN2) clusters on the entire UPS2GT dataset, hereby leaving 300 to 500 empty clusters[5]; and leading to a number of clusters in line with the expected number of peptides in the sample. Overall, the differences between Xnet and CHICKN on UPS2GT seem to be more related to the difference of objectives (finding isotopics envelopes *vs.* finding peptide-related clusters), as already discussed. Interestingly, this interpretation is confirmed by the Ecoli dataset experiments.

In absence of ground truth for both Ecoli datasets, we chose the tuning minimizing the DB index (see Figure 2.12D and 2.12E): $k_{\text{total}} = 16,384$ for Ecoli-FMS and for Ecoli-DIA. With such a tuning, we obtained around 11,600 (resp. around 9,400) non-empty clusters for Ecoli-FMS (resp. Ecoli-DIA). This number is obviously lower than the expected number of identifiable peptides (between 15 and 20 thousands), however under-clustering was clearly supported by empirical observations (see above, as well as Figure 2.7, rightmost figure). This clearly means that CHICKN could not separate too many peptides with too similar elution profiles. However, this can be easily explained by the difference of complexity between the UPS2GT and the Ecoli samples: while the former is fairly simple (a handful of spiked proteins), the latter ones are complex real life samples for which the discriminative power of the liquid chromatography is clearly challenged (as illustrated in the next section). This is notably why fragmentation spectra are classically used

[5] More generally, the capability of CHICKN to adapt the cluster sizes to the data distribution is illustrated on Figure 2.16.

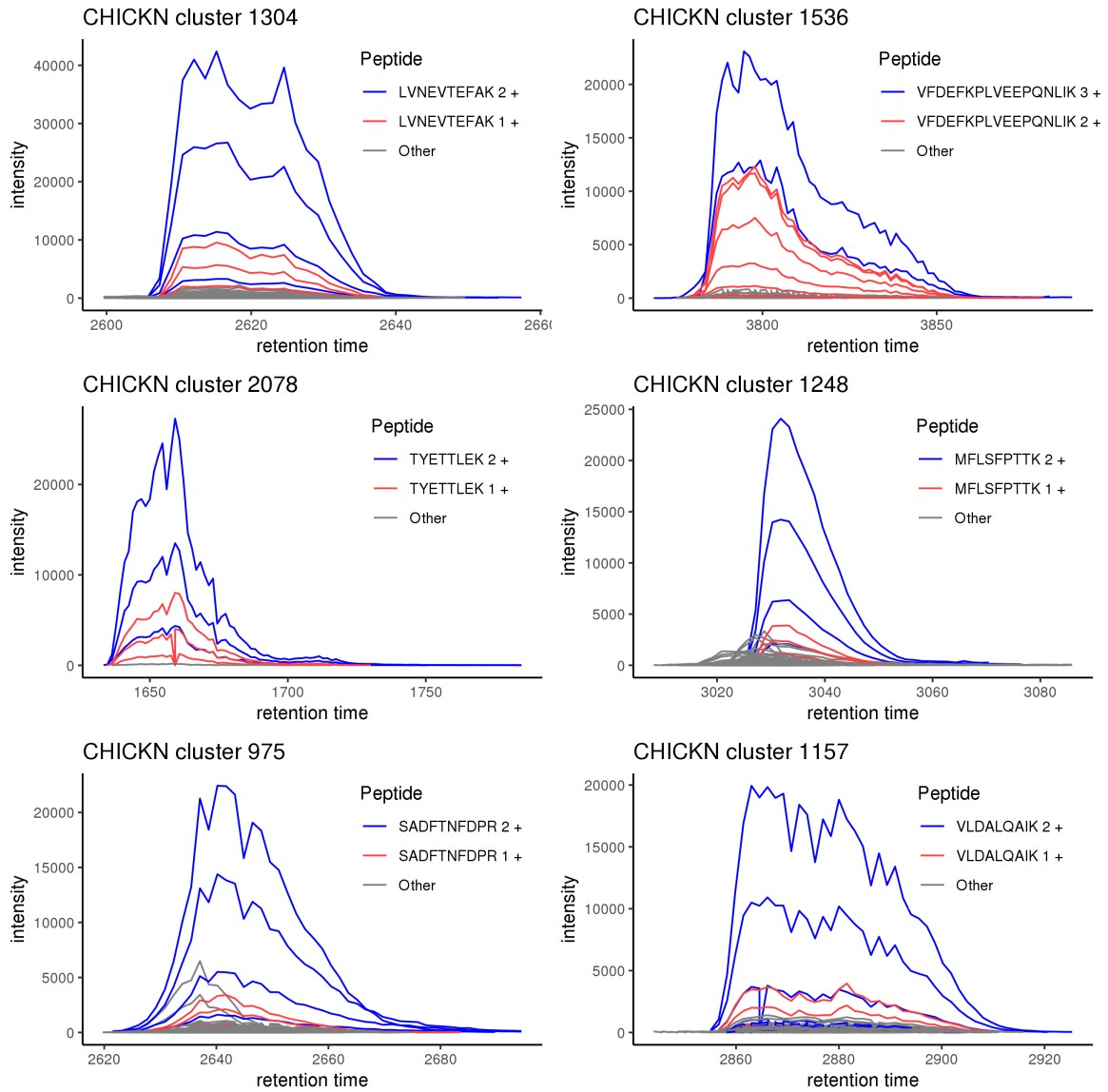


Figure 2.14: Differently charged ions of a same peptide clustered together. A subset of clusters were manually inspected so as to label as many profiles with the corresponding identified ion. Although this labelling cannot be exhaustively conducted due to the largely incomplete coverage of MS/MS analysis, it could be established that ions of a same peptide cluster together in many cases.

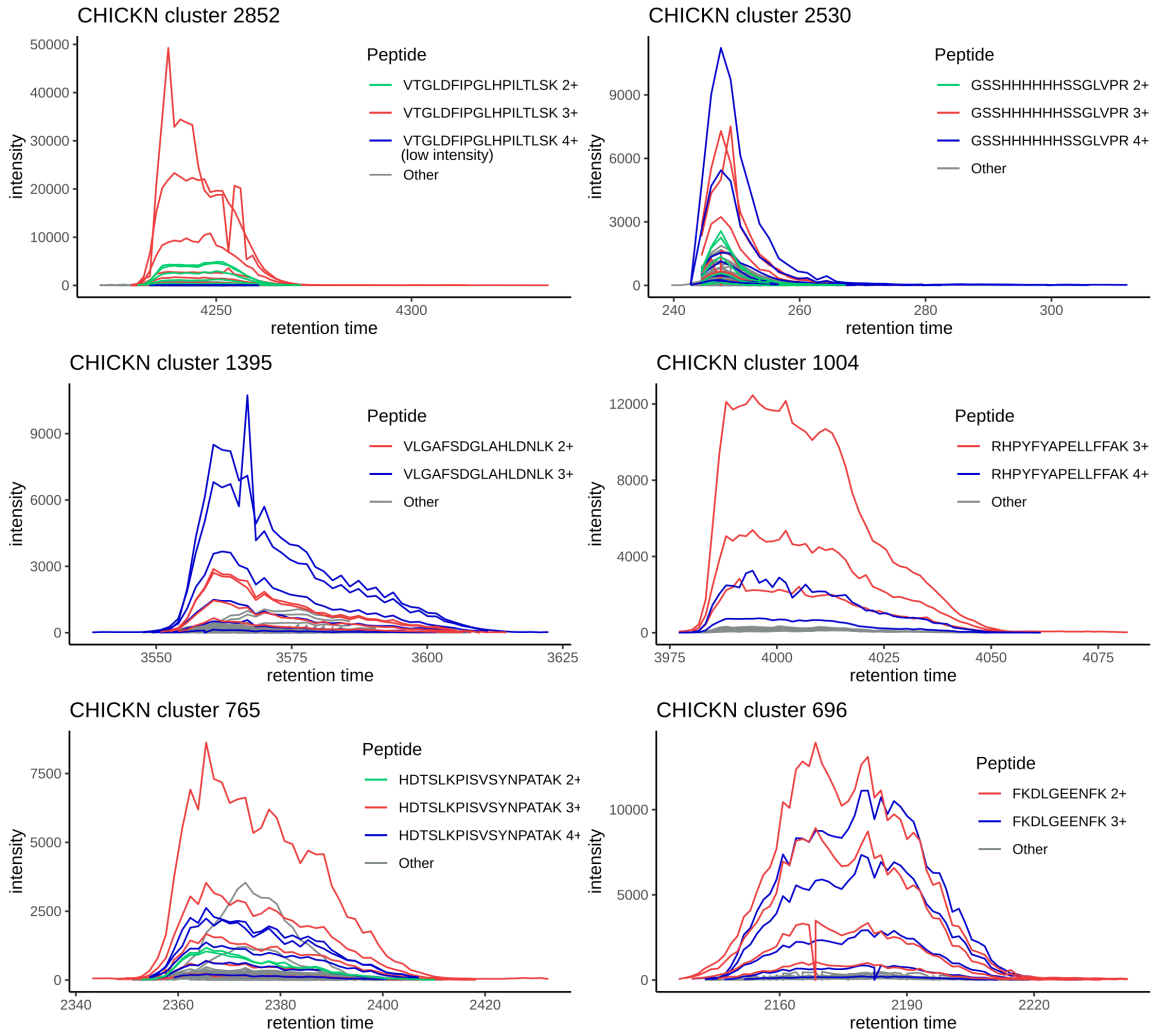


Figure 2.15: Differently charged ions of a same peptide clustered together. Figure similar to Figure 2.14. It depicts another subset of CHICKN clusters with chromatographic profiles manually annotated with the corresponding peptide ion.

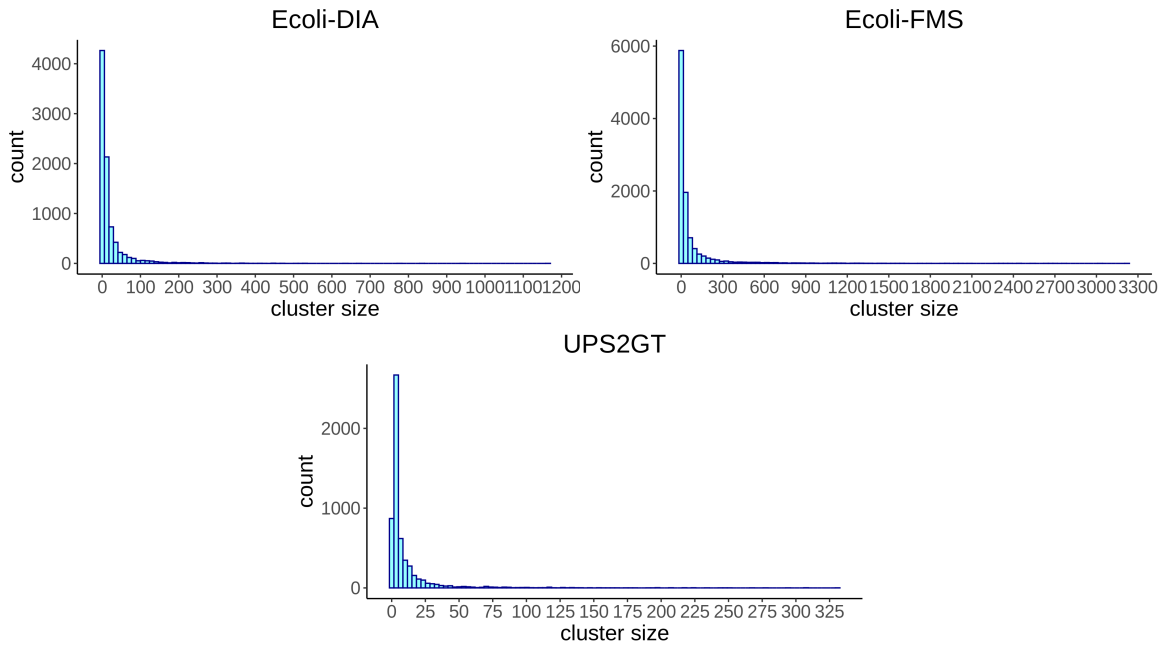


Figure 2.16: Histograms of the cluster size distribution resulting from the application of CHICKN on each of the three datasets.

to identify as many as 15 to 20 thousand peptides. However, achieving to discriminate half of this number of peptides with MS1 processing only is noticeable.

Finally, let us note, that, in general, relying on $k = 4$ provided slightly better scores. We assume that $k = 4$ was a trade-off between cluster diversity ($k > 4$) and computational efficiency ($k = 2$), as discussed above.

2.4 Discussions

2.4.1 Cluster interpretability

Beyond evaluation metrics, it is insightful to compare algorithms according to the interpretability of the clusters they can provide. Figure 2.17 represents different elution profiles from UPS2GT (their shape as well as their m/z position) in the context of the clusters they fall into, according to CHICKN and Xnet. The envelope assumption simplification clearly appears: As expected, Xnet splits into different clusters

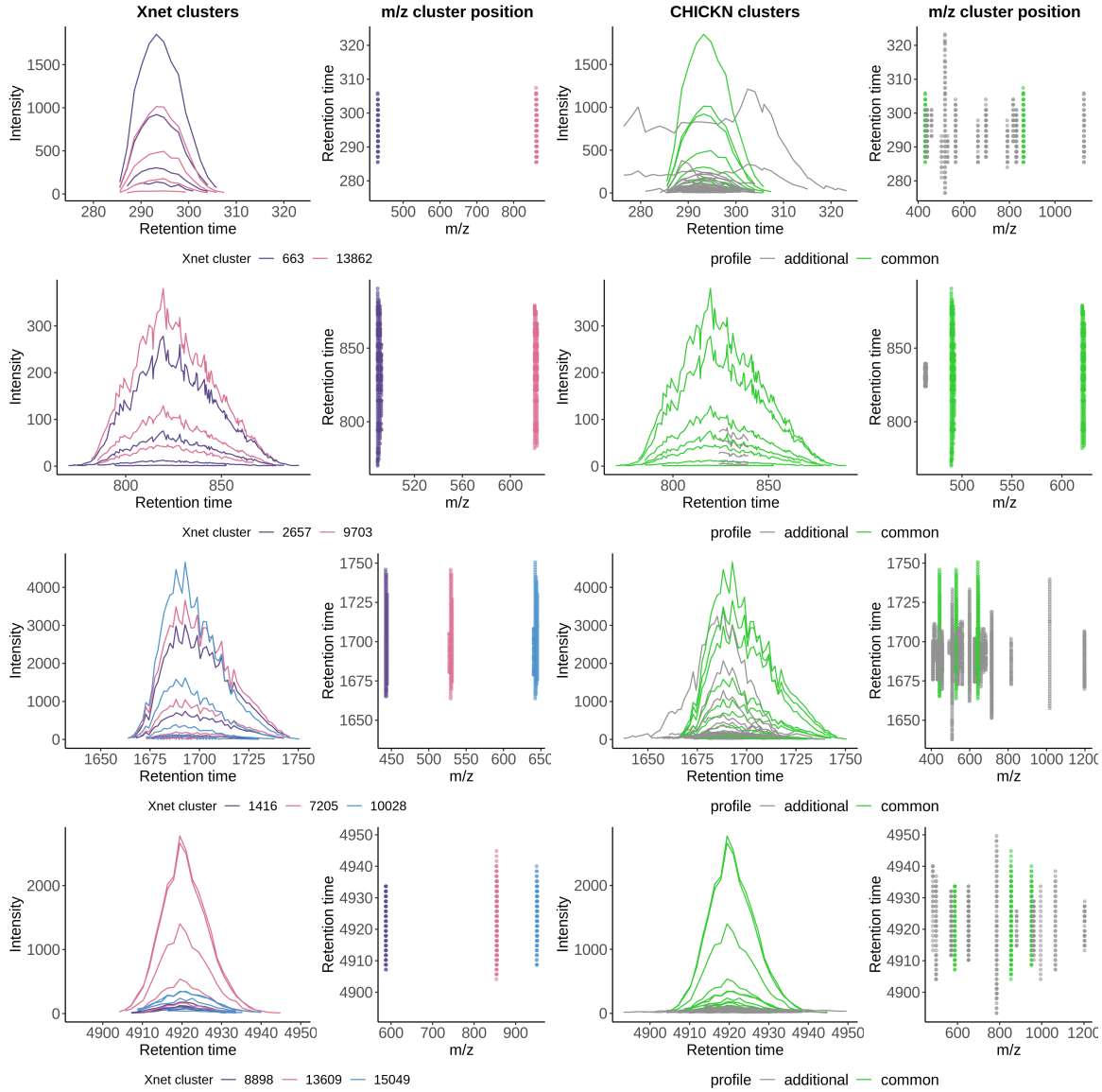


Figure 2.17: Xnet and CHICKN clusters for UPS2GT dataset. Each of the four lines represent a series of chromatograms in the context of their Xnet and CHICKN Cluster. On the plot of the leftmost column, a series of chromatograms with similar shapes are represented in different colors (2 or 3) according to the distinct Xnet clusters they belong to. In the second column, each elution profile is represented with the same color, according to its m/z position, hereby illustrating that Xnet clusters similar signals in different clusters because of a too large m/z difference. The plot of the third column represents the CHICKN cluster which encompasses all the Xnets cluster profiles of the leftmost column (in green), as well as other signals (in gray) falling in the same CHICKN cluster, hereby illustrating CHICK builds meaningful patterns irrespective of the m/z information that is essential to isotopic envelope construction. In the rightmost column, the m/z positions of the signals of the third columns, depicted with the same color code.

elution profiles that are arguably similar for the reason they have too different m/z values. In contrast, CHICKN promotes the inner coherency of clusters as it aggregates related Xnet clusters together. Notably, Figures 2.14 and 2.15 show a subset of 12 clusters provided by CHICKN, each gathering at least 2 differently charged ions from a same peptide (all of them being identified and manually validated with the associated MS2 spectra). Interestingly, the multiple isotopes of each ion also appear to be grouped, as illustrated by the manifold of profile co-clustered with each ion. Morevoer, a refine analysis of CHICKN clusters shows that, globally, they contain similar chromatograms, which is coherent both with the clustering metrics provided above, and with the expected behavior of the W1 kernel. However, some clusters also contain noise signals, as for examples, the first two lines of Figure 2.17. Although undesirable, this is a direct consequence of (i) the grouping capabilities of CHICKN, which captures similarities between slightly different but largely overlapping signals (third line); and (ii) the possibility to run CHICKN on raw data, which also contains many spurious signals that need be spread across various meaningful clusters.

Similar conclusions regarding CHICKN behavior can be derived from the Ecoli datasets (let us focus on the Ecoli-FMS one, as it displays elution profile signals with higher sampling resolution, due to the Full-MS acquisition). The majority of clusters (Figure 2.18 for the Gaussian W1 kernel and Figure 2.19, for the Laplacian W1 one) containing high intensity signals depicts meaningful consensus chromatograms, as well as similar profiles even though corresponding to different m/z values.

However, we observed that some clusters could be separated into several sub-clusters to improve readability (see Figure 2.20). It could intuitively be interpreted as the necessity to increase k_{total}. However, two observations goes against this: First, from a signal viewpoint, as the phenomenon mainly impacts lower intensity profiles, it also highlights the difficulty of finding consensus patterns near the noise level, which equally affects most of the clustering algorithms. In this context, over-clustering

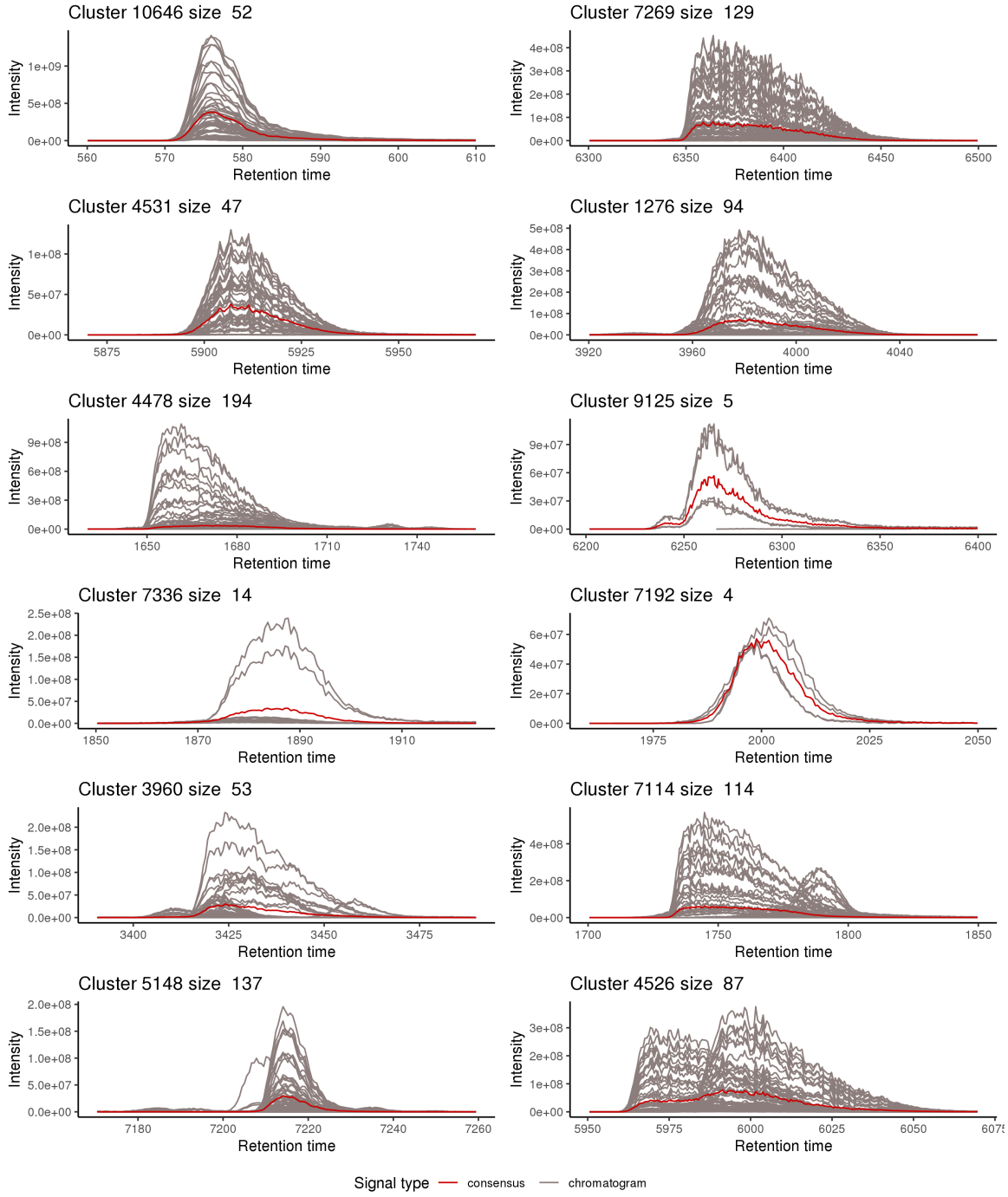


Figure 2.18: Examples of well-formed clusters for the Ecoli-FMS dataset. 12 clusters proposed by CHICKN with Gaussian W1 kernel (represented as time series), where each chromatogram is represented in gray, and where the consensus chromatogram is represented in red. The numbers above each example indicate the cluster ID and the number of chromatograms it encompasses.

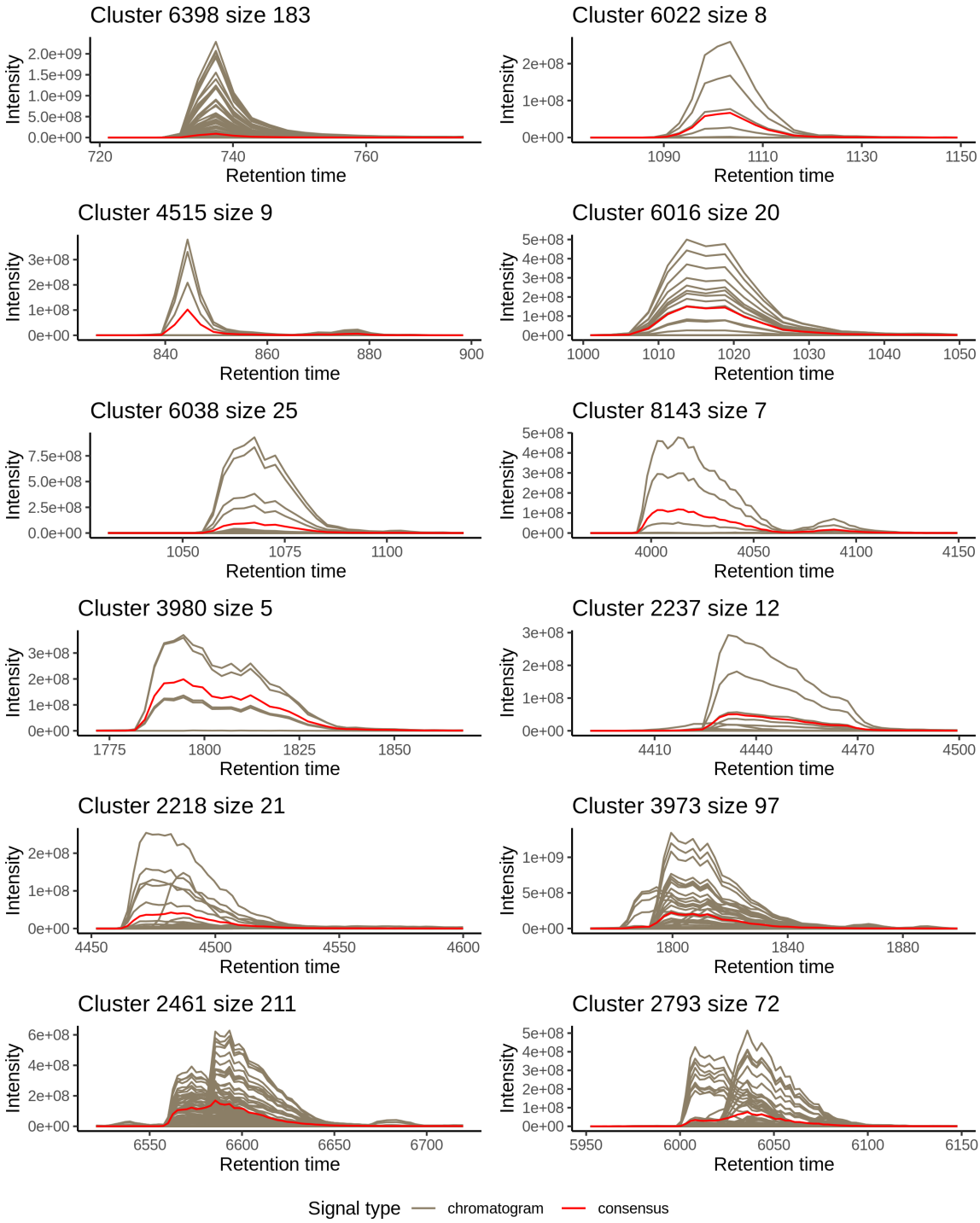


Figure 2.19: Examples of well-formed clusters for the Ecoli-FMS dataset obtained by CHICKN with Laplacian W1 kernel.

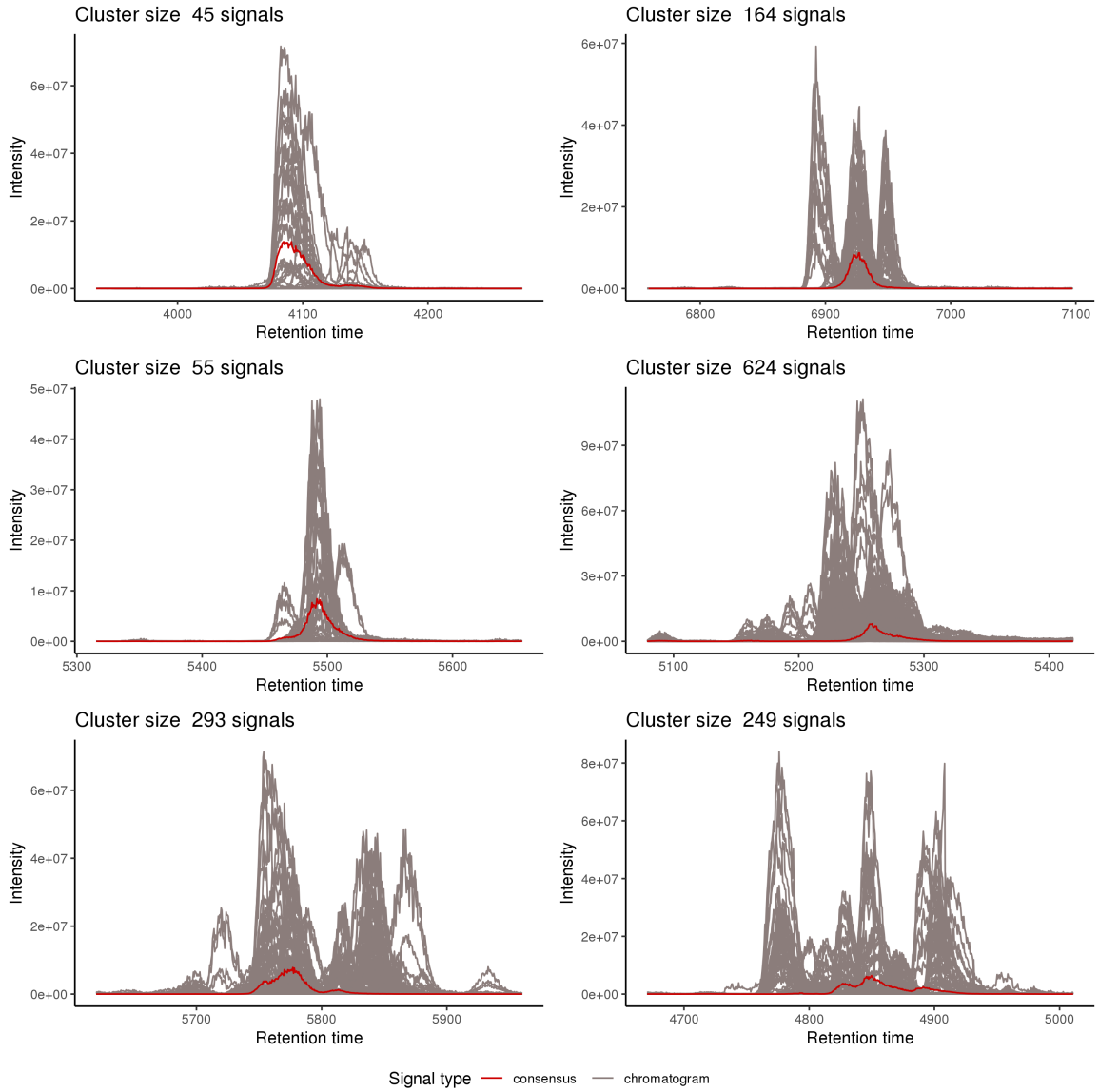


Figure 2.20: Examples of multiplexed clusters for the Ecoli-FMS dataset using CHICKN method. Figure illustrates that dividing multiplexed clusters into several sub-clusters would improve the elution profile interpretation. The real chromatograms and the consensus chromatograms are depicted in gray and in red, respectively.

is usually not considered a viable solution. Second, from an analytical viewpoint, the clustering algorithm cannot be expected to separate beyond the chromatographic capabilities (as in Figure 2.20, where few different profiles have too important overlap to expect discrimination).

Finally, it is worthy focusing on consensus chromatograms: interestingly enough, most of those observed in Figure 2.18 and in Figure 2.19 have meaningful shapes that are not deteriorated by the presence of noisy signals in the cluster, which can be interpreted as a positive consequence of our method to compute the cluster centroids pre-image based on a restricted neighborhood (see Section 2.2.5.3).

2.4.2 Implementation and code availability

CHICKN algorithm was implemented in `R`. The W1 distance computations and the gradient descent were accelerated using `C` and interfaced with `R` thanks to `Rcpp`. The data compression procedure and the hierarchical strategy were parallelized with `RcppParallel`, `foreach` and `doParallel`. To access and manipulate large data matrices, we relied on the File-backed Big Matrix class of the `bigstatsr` package [152]. A File-backed matrix allows to overcome the memory limitation by storing the data on the disk, using a binary memory-mapped file. However, `bigstatsr` is only available under Linux OS, leading to a similar restriction for CHICKN.

For practitioners, the proposed algorithm is available through an R package, available on Gitlab [153], as well as on the CRAN [154].

2.5 Conclusion

We have presented two complementary contributions to the cluster analysis of LC-MS data. First, we have proposed a unique combination of hierarchical strategy, of Nyström approximation and of random Fourier features based compression technique to scale up the kernel k-means clustering to the large size, the large dimensionality and the large number

of expected clusters of LC-MS data. Second, we have proposed to rely on the optimal transport framework (Wasserstein-1 distance) to define a similarity measure and we have shown it is insightful to capture the semantics of elution profiles in LC-MS data. On a more theoretical front, we have established the Wasserstein-1 distance could lead to a positive-definite Laplacian kernel, and exhibit a path for further investigations about a Gaussian one.

We have demonstrated these contributions could help extracting other structures than isotopic envelopes, even on multiplexed data acquired with Data Independent Acquisition protocol. However, the experimental assessment of these contributions is difficult to interpret. On the one hand, when compared to the canonical application of isotopic envelope extraction, CHICKN does not outperform the state-of-the-art algorithm (better Recall and worse Precision, as it tends to under-cluster rather than over-cluster). However, it provides an important advantage: it can be run on raw data and does not require costly preprocessing. As for an application-independent evaluation, it clearly appears that CHICKN is able to extract patterns from the data which are not accessible to linkage-based algorithms. Put together, we interpret this as following: Although cluster analysis has made important progresses in the theoretical front over the past 50 years, processing LC-MS data remains a challenge which requires research efforts. It is still necessary to propose complementary and differently principled algorithms that will help make LC-MS practitioners extract the best from their data. In this context, new kernels could be defined; and numerous state-of-the-art clustering algorithms recently developed in the machine learning community could advantageously be applied to LC-MS data.

Availability of data and materials

The UPS2GT dataset supporting the conclusions of this article is available in the Github, `https://github.com/optimusmoose/ups2GT` [134].

The Ecoli DIA and Ecoli FMS raw data (generated and analyzed for the current

study) are not publicly available due to their too large size. However they are available from the corresponding author upon reasonable request. To reproduce all experiments described in the article, the preposessed Ecoli datasets in the file-backed matrix format can also be provided upon request.

Funding

This work was supported by grants from the French National Research Agency: ProFI project (ANR-10- INBS-08), GRAL project (ANR-10-LABX-49-01), DATA@UGA and SYMER projects (ANR-15-IDEX-02) and MIAI @ Grenoble Alpes (ANR-19-P3IA-0003). Grants ANR-10-INBS-08 and ANR-10-LABX-49-01 contributed to wet-lab equipment, including mass spectrometer; Grant ANR-15-IDEX-02 contributed to human resources; Grant ANR-19-P3IA-0003 supported other expanses.

Author's contributions

OP designed the method, implemented the R package, carried out the computational experiments, analysed the results and drafted the manuscript. RG implemented Nyström approximation method and the Ecoli dataset construction routines. TF designed the preprocessing steps, performed preliminary implementations and advised on some mathematical issues. AK and AMH produced the Ecoli datasets (LC-MS analysis design and production, preliminary bioinformatics processing). AMH wrote the wet-lab analysis section. TB designed the method, directed the work, participated to the result analysis and drafted the manuscript. All authors proofread the manuscript and approved its final version.

Acknowledgements

The authors thank Virginie Brun, Yohann Couté and Christophe Bruley for supports and fruitful discussions.

List of abbreviations

- **CHICKN**, Chromatogram HIerarchical Compressive K-means with Nystrom approximation;
- **CKM**, Compressive k-means;
- **DB**, Davies-Bouldin index;
- **ΔRT**, peak Retention Time difference;

- **DIA**, Data Independent Acquisition;
- **Ecoli-DIA**, Dataset resulting from the analysis of an Escherichia coli sample in DIA mode;
- **Ecoli-FMS**, Dataset resulting from the analysis of an Escherichia coli sample in FMS mode;
- **FMS**, Full mass spectrum (only MS1s are recorded);
- **FN**, False Negative;
- **FP**, False Positive;
- **LC-MS**, liquid chromatography and mass spectrometry;
- **MS1**, peptide (or precursor) mass spectrum;
- **MS2**, peptide fragment mass spectrum;
- **m/z**, mass to charge ration;
- **PD**, positive definite;
- **PSD**, positive semi-definite;
- **RI**, Rand index;
- **RT**, Retention Rime;
- **SVD**, Singular Value Decomposition;
- **TN**, True Negative;
- **TP**, True Positive;
- **UPS2GT**, dataset resulting from the analysis of the Proteomics Dynamic Range Standard (UPS2) and manually annotated (ground truth);
- **W1**, Wasserstein-1 distance;

CHAPTER 3

Sketched Stochastic Dictionary Learning

Sketched Stochastic Dictionary Learning for large-scale data and application to large-scale mass spectrometry data[1]

Olga Permiakova[a], Thomas Burger[b]

[a] Univ. Grenoble Alpes, CEA, Inserm, BGE U1038, 38000 Grenoble, France

[b] Univ. Grenoble Alpes, CNRS, CEA, Inserm, BGE U1038, 38000 Grenoble, France

Abstract

Factorization of large data corpora has emerged as an essential technique to extract dictionaries (sets of patterns that are meaningful for sparse encoding). Following this line, we present a novel algorithm based on compressive learning theory. In this framework, the (arbitrary large) dataset of interest is replaced by a fixed-size sketch resulting from a random sampling of the data distribution characteristic function. We applied our algorithm to the extraction of chromatographic elution profiles in mass spectrometry data, where it demonstrates its efficiency and interest compared to other related algorithms.

Keywords: *Dictionary learning; Stochastic gradient descent; Compressive statistical*

[1] This article is submitted in *Statistical Analysis and Data Mining* journal

learning; Nesterov accelerated descent; Computational mass spectrometry; Matrix factorization

3.1 Introduction

Finding a linear decomposition of an observed signal $x \in \mathbb{R}^s$ is essential for many applications, as it provides a way to exhibit its elementary constitutive patterns, as well as to denoise it. Formally, this task amounts to finding a vector of coefficients $c = (c^1, \ldots, c^K) \in \mathbb{R}^K$, referred to as *code*, such that:

$$x = c^1 \cdot d_1 + \cdots + c^K \cdot d_K + \epsilon, \tag{3.1}$$

where $D = \{d_1, \ldots, d_K\} \in \mathbb{R}^{s \times K}$ is a matrix referred to as *dictionary*, which is composed of K s-dimensional column vectors (the *dictionary atoms*), and where ϵ represents the (hopefully small) part of x that is not explained by D. While many solutions to this problem are already available when D is known, the decomposition of a signal, which potential constitutive elementary patterns are unknown (referred to as blind source separation), is much more difficult. As a result, despite being almost 30 years old [155], this problem still focuses investigations.

According to compressive sensing theory [156], a good dictionary is such that any signal can be precisely approximated using few dictionary atoms only, *i.e.*, only a restricted number of c^i are non-null in Decomposition (3.1); and the fewer the better. As emphasized in [157], this type of representations, referred to as *sparse representations*, are widespread in many real-life applications: image denoising [158], super resolution [159], compression [160], etc.

The oldest strategies to decompose signal have used Riesz bases as dictionaries (*e.g.*, Fourier, wavelet or curvelet bases [161]). Their mathematical properties have made the decomposition straightforward, yet, for highly complex real-life signals, sparse representations have generally been achievable only at the price of an important unexplained

residue ϵ. In fact, it has since then been established [162] that sparse representations were easier to obtain when Decomposition (3.1) involved an overcomplete dictionary, *i.e.* a dictionary which size exceeds the signal dimensionality $K > s$. However, working with an overcomplete dictionary has two (related) drawbacks: First, the corresponding matrix D is not full-rank, which can potentially lead to numerical issues; Second, Decomposition (3.1) may not be unique, so that additional constraints are usually necessary to lead to a well-posed problem and a practically satisfactory solution. This is why as an alternative strategy, it has been proposed to extract elementary patterns from a set of signals akin to that for which a decomposition is sought, and to form an overcomplete dictionary with these patterns. This approach, referred to as *dictionary learning*, has been demonstrated to lead to dictionaries that are of real practical interest, for three reasons: First, they capture well the specificities of the data [163]. Second, they yield even sparser representations; Third, their atoms are easier to relate to physical signals and thus to interpret [164]. Concretely, learning a dictionary from a set of observed signals $X \in \mathbb{R}^{s \times N}$, is related to finding a decomposition of X into a product of two low rank matrices [165]. The effectiveness of this matrix factorization approach has been illustrated in many applications, such as medical signal modeling and analysis [164], natural image processing [166], audio and video processing [167]. In this article, we aim to apply dictionary learning to another type of data: those resulting from the high-throughput mass spectrometry analysis of complex biological samples.

Mass spectrometry (MS) coupled with liquid chromatography (LC) is a commonly used analytical chemistry technique, which has witnessed an increasing popularity in the last decade [168], due to its application to omics biology; as it is the method of choice for proteome, metabolome and lipidome investigations. Despite increasing resolution and cycle speed, the LC-MS pipeline is still challenged by the complexity of classical biological samples. Therefore, deep sample coverage requires

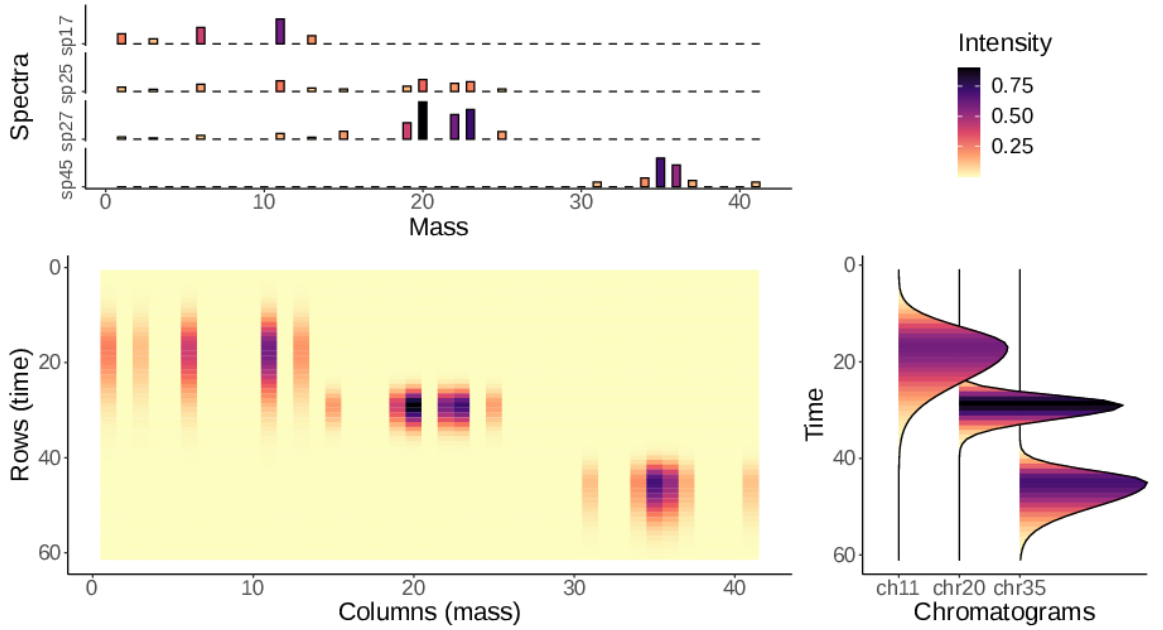


Figure 3.1: Toy illustration of a LC-MS data matrix with three analytes yielding three distinct chromatograms and four spectra.

multiplexed measurements [111], which demultiplexing intuitively translates into solving a blind source separation problem [28]. Following [169], which proposes to denoise simple LC-MS data by relying on a matrix factorization formulation, we consider learning a dictionary of chemical signals. Concretely, the LC-MS data can easily be formatted into a matrix: Broadly, the LC can be seen as a way to serialize the analytes into the MS, so as to avoid that too many of them are concomitantly analyzed. Thus, if the mass spectra produced over time are stored as high-dimensional vectors and stacked as matrix rows, the matrix columns can be interpreted as chromatograms (a.k.a. elution profiles), *i.e.* as vectors that represent each analyte's flowrate outputted from the LC toward the MS (see Figure 3.1). Notably, any analyte's flowrate being a physical signal, we have formerly established [170] it is insightful to rely on the chromatogram smoothness to extract meaningful chemical patterns through cluster analysis.

Learning a meaningful dictionary from LC-MS data comes along with numerous challenges. First, the dictionary atoms must be interpretable as chromatograms (*i.e.* smooth, non-negative, slightly heavy right-tailed

waveforms, see Figure 3.1). Second, owing to the number of analytes in a classical sample (up to tens of thousands), which largely exceeds the signal dimensionality, the dictionary must be highly overcomplete. Third, LC-MS data are already rather big and their size is ever-increasing due to the constant improvement of MS resolution and cycle speed. Therefore, processing them in a time compliant with the various constraints of a standard analytical platform is a computational challenge that requires scalable solutions.

We hereby describe a new approach meeting these constraints. From a methodological viewpoint, our contribution is twofold: First, following recent developments in compressive learning (see below), we give an original formulation to the dictionary learning objective function; Second, we rely on a stochastic gradient descent algorithm to efficiently minimize this objective function. Together, this yields a new method referred to as Sketched Stochastic Dictionary Learning (SSDL), which improve upon the state-of-the-art to extract with a small computational footprint a set of meaningful patterns from LC-MS data. The article is structured as follows: Section 3.2 gathers the related works, including standard dictionary learning formulation, presentation of state-of-the-art methods and summary of background knowledge that are instrumental to a clear exposure of SSDL. Then Section 3.3 presents SSDL. Finally, Section 3.4 is dedicated to experimental validations on LC-MS data.

3.2 Related works

3.2.1 Classical dictionary learning strategies

Let $X = \{x_1, \dots, x_N\} \in \mathbb{R}^{s \times N}$ be a data matrix. The classical formulation of dictionary learning reads as the following joint optimization problem:

$$\min_{\substack{C \in \mathbb{R}^{K \times N} \\ D \in S}} \frac{1}{N} \sum_{i=1}^{N} \|x_i - c_i \cdot D\|_2^2 + \lambda \cdot \|c_i\|_1 \tag{3.2}$$

where C is a matrix gathering all code vectors c_i; where λ is the regularization parameter of a LASSO penalty [171], which controls the representation sparsity; and where S is the convex set where dictionary atoms can be picked up. It has been proven in [172] that solving the minimization problem of Eq. (3.2) leads to sparse representations. Moreover, this problem being convex with respect to C and D separately, most methods to solve it rely on an alternative minimization scheme (*i.e.* minimization of Eq. (3.2) with respect to one variable while the other is fixed):

$$\min_{C \in \mathbb{R}^{K \times N}} \frac{1}{N} \sum_{i=1}^{N} \|x_i - c_i \cdot D\|_2^2 + \lambda \cdot \|c_i\|_1 \tag{3.3}$$

$$\min_{D \in S} \frac{1}{N} \sum_{i=1}^{N} \|x_i - c_i \cdot D\|_2^2 \tag{3.4}$$

The first sub-problem (Eq. 3.3), which consists in computing the code matrix for a given dictionary, is referred to as *sparse coding*. It can be solved using LASSO [171], LARS [70], iterative shrinkage thresholding [173]. However, the specificity of any dictionary learning approach essentially lies in the technique used to solve Eq. (3.4), *i.e.* the *dictionary update*. For example, Engan et al. [174] rely on an analytical solution to re-compute the dictionary at each iteration: $D = X \cdot C^{\dagger}$, where $C^{\dagger}$ is the pseudo-inverse of the code matrix. Alternatively, K-SVD [69] updates each dictionary atoms independently by performing singular value decomposition. Finally, there are many ways to numerically address Eq. (3.3) by relying on descent paradigms [175, 176].

3.2.2 Large-scale dictionary learning techniques

Stochastic or online learning [74, 177] is an efficient and broadly used technique to train a dictionary from a large dataset. It consists in updating the dictionary and calculating the code at each iteration of the minimization procedure, by relying on a randomly selected subset of signals only (possibly, a single one). As working on a small sub-

set drastically reduces the computational time of both sub-problems, multiple passes through the data (called *epochs*) can be used until convergence. We have singled out two online dictionary learning methods to benchmark against our approach: MODL [178,179] and IcTKM [180]. The former is the state-of-the-art approach, and it demonstrates excellent computational performance on extremely large datasets. The latter is a recently published method which underlying mathematics are conceptually related to those leveraged in the present work, making the comparison worth of interest. More precisely, MODL minimizes Objective Functions (3.3) and (3.4): Sparse coding is achieved using LARS algorithm, while coordinate gradient descent is applied to update dictionary atoms. Formally, at the tth iteration, the dictionary atoms are recomputed as follows:

$$d_k^t = d_k^{t-1} - \frac{1}{A_{kk}^t} \cdot \left(D^{t-1} \cdot A_{:,k}^t - B_{:,k}^t\right), \tag{3.5}$$

where auxiliary matrices A^t and B^t are defined as $A^t = \frac{1}{t}\sum_{i=1}^t c_i \cdot c_i^t \in \mathbb{R}^{K\times K}$ and $B^t = \frac{1}{t}\sum_{i=1}^t x_i \cdot c_i^t \in \mathbb{R}^{s\times K}$, and where $A_{:,k}^t$ and $B_{:,k}^t$ denote kth columns of matrices A^t and B^t respectively. Using these auxiliary matrices allows to gather the statistics about the signals $x_1, \ldots, x_t$ and codes $c_1, \ldots, c_t$ observed at previous iterations without explicitly storing them in memory. This strategy provides low memory consumption and computational cost, at least for small K. MODL is compliant with positiveness constraints on both the dictionary and code matrices, as required by LC-MS data (see Section 3.1). Finally, MODL embeds an optional dimensionality reduction step based on random projections [181].

Although IcTKM reformulates dictionary learning as a constrained minimization problem, it also proposes a solution based on alternate minimization. Instead of a regularization parameter, the sparsity level Λ is directly defined through a constraint on the number of non-zero elements in the columns of the codes. Sparse coding is solved using

Iterative Thresholding [182]. The dictionary update is carried out by computing the K-residual means (following the known equivalence between clustering and matrix factorization [67]). Concretely, each dictionary atom d_k is updated by averaging the data vectors x_i with non-null code coefficients C_{ki}. To speed up the computations, IcTKM relies on fast Johnson–Lindenstrauss transform [183].

3.2.3 Scaling-up by sketching

The method proposed in this article borrows two important features from the large-scale machine learning literature and adapt them to the dictionary learning context. The first one is the *compressive statistical learning* framework [80]. Its seminal idea is to summarize the data collection into a complex vector of fixed size, referred to as the *data sketch*, so that the algorithm complexity does not depend on the data size anymore. Concretely, the data sketch is constructed by sampling the characteristic function of the data distribution $P(X)$:

$$SK(X) = \left[\mathbb{E}_{x\sim P(X)}\left(e^{iw_j^T x}\right)\right]_{j=1}^{m}, \tag{3.6}$$

where $w_1, \ldots, w_m \in \mathbb{R}^s$ are frequency vectors, randomly sampled from some predefined distribution [80]. Starting from this theoretical ground, the main challenge is to adapt the machine learning method of interest so that it operates on the data sketch rather than on the original data. The authoring team demonstrated both the practical interest and the efficiency of this approach on various problems, including Gaussian mixture estimation [80] and data clustering [56]. However, to the best of the authors' knowledge, there is to date no dictionary learning method based on this framework.

Our second cornerstone is Nesterov accelerated gradient descend method (NAGD, [78]). Conceptually, it is akin to classical gradient descent, however, it includes an additional term, the momentum, denoted as η (a weighted average of the gradient vectors computed in the previous

iterations). Adding this momentum term makes quadratic convergence possible in the deterministic cases [78]. The NAGD update rule reads:

$$\begin{aligned} D^{t+\frac{1}{2}} &= D^t + \alpha \cdot v^t \\ \eta^{t+1} &= \alpha \cdot \eta^t - \gamma \cdot \nabla_D f\left(D^{t+\frac{1}{2}}\right) \\ D^{t+1} &= D^t + \eta^{t+1} \end{aligned} \tag{3.7}$$

where α is the momentum weight; where γ defines the length of each gradient step (the *learning rate*); and where $D^{t+\frac{1}{2}}$ is referred to as the *ahead*. Recently, NAGD scheme has attracted great interest for stochastic optimization: Its convergence under convex and smooth optimization has been heavily documented [184, 185], but scarce results are so far available for non-convex cases. Despite, it is of practical interest, as when correctly tuned, it outperforms the classical stochastic method [74].

3.3 SSDL method

3.3.1 Objective function

As SSDL follows a classical alternate minimization scheme, the dictionary update and the code computation objective functions can be separated. The former differs from the classical dictionary update (Eq. 3.4), as we propose to include the sketching operator of Eq. (3.6). As for the code computations, we have modified Eq. (3.3) to fit the stochastic learning framework. Concretely, at each iteration, the code matrix is computed for a randomly selected data subset only, denoted $\hat{X} = \{\hat{x}_1, \ldots, \hat{x}_n\} \in \mathbb{R}^{s \times n}$, where $n < N$:

$$C^* = \arg\min_{C \in \mathbb{R}^{K \times n}} \frac{1}{n} \sum_{i=1}^{n} \|\hat{x}_i - D \cdot c_i\|_2^2 + \lambda \cdot \|c_i\|_1, \tag{3.8}$$

Then SSDL looks for a dictionary D such that the data sketch $SK(X)$ is as close as possible to the sketch of the decomposition $SK(D \cdot C^*)$:

$$\min_{D \in S} F(D, C^*), \text{ with } F(D, C^*) = \|SK(X) - SK(D \cdot C^*)\|_2^2 \tag{3.9}$$

Since the sketching procedure amounts to sampling an empirical characteristic function, D does not only represent the observed data, but to some extent, its underlying distribution. Therefore, the dictionary resulting from this procedure can be expected to generalize well to other data with similar distribution. This behavior should moreover be strengthened by an adequate tuning of the frequency generation procedure [80]; as it aims to capture only the most relevant features and eliminate noise. Finally, it is also of interest in a stochastic learning context: despite the use of a random sampling procedure on the training set, the optimizer can also rely on the complete data summary provided by the data sketch.

These changes in the objective functions lead to several advantages: First, the stochastic minimization is fully efficient, as the additional computations it requires are immaterial on small data batches. Notably, the decomposition sketch $SK(D \cdot C^*)$ only involves $\mathcal{O}(K \cdot s \cdot n + m \cdot s \cdot n)$ operations, and even with small data batch $\hat{X}$, one ends up with sufficiently accurate approximations (even though the larger the data batch, the more accurate the approximation). Second, the gradient of Eq. (3.9) does not contain the term $D \cdot A$, where the matrix $A = \sum_{i=1}^{n} c_i \cdot c_i^\top \in \mathbb{R}^{K \times K}$, which computational footprint could be important for highly overcomplete dictionary. Finally, the data sketch $SK(X)$ is computed once, at the algorithm initialization (afterwards the data sketch remains unchanged) and this initial computation parallelizes well in case of extremely large datasets.

3.3.2 Minimization

At each iteration of SSDL, the dictionary is updated by performing one gradient step as in Nesterov scheme (3.7), with the gradient of Objective function (3.9) reading:

$$\begin{aligned} \nabla_{d_l} \|SK(X) - SK(D \cdot C^*)\|^2 &= -2 \left(\nabla_{d_l} SK(D \cdot C^*)\right)^\top \cdot r \\ \frac{\delta}{\delta d_l} SK^j(D \cdot C^*) &= 1i \cdot \left(\frac{1}{n} \sum_{i=1}^{n} C^*_{li} \cdot \prod_{k=1}^{K} SK^j(C^*_{ki} \cdot d_k) \right) \cdot w_j^\top \end{aligned} \tag{3.10}$$

where $r = SK(X) - SK(D \cdot C^*)$ and $SK^j(\cdot)$ is the j^{th} coordinate of the sketch vector. However, the dictionary update of SSDL differs from the general Nesterov scheme. First, to ensure the positivity of the dictionary atoms, the ahead dictionary is projected on $\mathbb{R}_+^{s \times K}$ (see Alg. 3, Line 6). Second, SSDL uses a decaying learning rate (see Alg. 3, Line 7):

$$\gamma_t = \frac{\gamma_0}{(1 + (t-1) \cdot \nu)}. \tag{3.11}$$

This is motivated by the following fact: using a large γ far away from the minimum and progressively decreasing it allows to accelerate the convergence. As a drawback, it requires to tune an additional parameter, ν (the decay speed). The last difference relies in the Euclidean projection on $S = \{d \in \mathbb{R}_+^K \mid \|d\|_2^2 \leq 1\}$, which is included after the dictionary update (see Alg. 3, Line 9) to avoid too small scalars in the code matrix C. The complete pseudocode of SSDL is presented in Alg. 3.

3.4 Experimental validation

3.4.1 Implementation

SSDL is implemented in R language. The gradient vector calculation is implemented in C++ and it is wrapped to R using the Rcpp package [186] and parallelized using the RcppParallel package [187]. The sparse coding problem is addressed by the GLMNET R function from the

Algorithm 3 Sketched Stochastic Dictionary Learning

Input: Data matrix $X \in \mathbb{R}^{s \times N}$; Sketch $SK(X)$; Initial dictionary D^0; Initial learning rate γ_0; Decay parameter ν; Momentum weight α; Batch size n; Regularization weight λ.

Initialization: $t = 0$, $\eta^0 = 0$

Repeat until convergence:

Construct a data batch $\hat{X}^t \in \mathbb{R}^{s \times n}$.

Sparse coding: $C^t = \underset{C \in \mathbb{R}^{K \times n}}{\min} \frac{1}{n} \sum_{i=1}^{n} \|\hat{x_i}^t - D^t \cdot c_i\|_2^2 + \lambda \cdot \|c_i\|_1$

Dictionary update:

$D^{t+\frac{1}{2}} = \max(D^t + \alpha \cdot \eta^t, 0)$

$\gamma_t = \frac{\gamma_0}{(1+(t-1)\cdot\nu)}$

$\eta^{t+1} = \alpha \cdot \eta^t - \gamma_t \cdot \nabla_D F(D^{t+\frac{1}{2}}, C^t)$

$D^{t+1} = \mathcal{P}_S\left(D^t + \eta^{t+1}\right)$, where $\mathcal{P}_S(\cdot)$ is a Euclidean projection on convex set $S = \{d \in \mathbb{R}_+^K \mid \|d\|_2 \leq 1\}$

glmnet package [188], which is parallelized using the mclapply R function from the parallel package [189]. The File-backed Big Matrix (FBM) class of the bigstatsr [190] package is used to store and to manipulate the matrices that are too large to be memory allocated. The SSDL code is available on gitlab `https://gitlab.com/Olga.Permiakova/ssdl`. The tests of SSDL and MODL were performed on a laptop machine with the following characteristics: HP Pavilion g6 Notebook PC with Intel (R) Core (TM) i5-3230M CPU @ 2.60GHz, 8 Gb of RAM, 4 cores, running under Ubuntu 04/18/4 LTS OS. The IcTKM method being distributed as Matlab code, it was tested on the same machine, but for license issues, under another OS (Windows 8). To measure the execution time of SSDL, we computed the difference between the times at the beginning and at the end of the learning using Sys.time R function. For MODL, we used the same procedure with the tic/toc python functions. Finally, IcTKM matlab code provides the execution duration in terms of CPU time.

3.4.2 Data description and data preprocessing

To validate SSDL, we relied on a proteomic dataset obtained with the LC-MS analysis of a sample of Escherichia coli bacteria (E.coli). A full description of the data acquisition pipeline as well as of the dataset is available in [170]. An important feature of this pipeline is that the basic elements that are analyzed are peptides (*i.e.* protein fragments). As described in Section 3.1, the matrix columns contain discrete chromatographic profiles obtained during the elution of the sample's peptides in liquid chromatography, while the rows represent mass spectra acquired at different time stamps. The chemical properties of LCs are so that peptides with low masses are usually eluted at the beginning of the analysis, and heavy ones at the end. This leads to a specific matrix structure with high intensity peaks distributed along the diagonal, and with many zeros in the corners. This eases the data processing as it makes it possible to split the data matrix horizontally into several slightly overlapping slices, for which dictionaries can be independently trained. Then, since each slice contains a different set of peptides, the entire dictionary can be formed by concatenating the dictionaries from all the slices, which significantly reduces the computational cost.

The learning procedure is the same for each slice, so we present the experimental results for a single slice. We focus on a data slice of 718 rows acquired between 10 and 30 minutes (amongst the two hours that lasted the complete LC-MS analysis, that is a quarter of the entire dataset). We have chosen this specific slice as for chemical reasons, it contains the highest density of MS peaks; making it the hardest part to extract a dictionary from. The resulting data matrix contains 74,193 chromatographic profiles (matrix columns). To further reduce the data dimension, the matrix rows were randomly sub-sampled (from 718 to 256 rows).

3.4.3 Parameter tuning

SSDL capabilities to extract a dictionary from the Ecoli dataset [170] are compared to those of MODL and IcTKM. Each method having its own set of parameters, we hereafter survey their tuning.

SSDL method has eight parameters: the dictionary size K, the regularization parameter λ, the sketch size m, the batch size n, the initial learning rate γ_0, the decay parameter ν, the momentum weight α and the number of epochs T. In addition, SSDL requires an initial dictionary D^0, but it can easily be defined by a random selection from the data. We observed that 3 epochs ($T = 3$) were practically sufficient to reach convergence.

Among these parameters, a number of them should be tuned according to the specificities of the LC-MS pipeline and to the data it has delivered. Notably, the dictionary size must be consistent with the number of distinct peptides that are expected to be found in the sample. In our case, E.coli being well studied, the number of peptides identified by a conventional mass spectrometry analysis is known to lie somewhere between 12,000 and 15,000, depending on the instrument and its tuning. Thus, for a single slice, a suitable dictionary size should be 3000 to 3750. To investigate the effect of the dictionary size on the SSDL execution time, we also considered smaller values of K. Notably, we considered various scenarios with $K = \{384; 768; 1,536; 2,304; 3,072; 3,712\}$.

The regularization parameter λ should be tuned so that the number of non-zero elements in the codes broadly amounts to the multiplexing level of the MS acquisitions. With these regards, it is commonly assumed that on data such as those produced, chromatograms with more than 20 peaks are not sufficiently resoluted: they either correspond to noise, or to chemical species which disentanglement stands beyond the analytical power of the instrument. We thus tried various values of λ to find the one that would lead to the desired sparsity level. Concretely, we considered $\lambda \in \{0, 0.001, 0.01\}$, and for each of these values, we computed the code C^0 using the initial dictionary D^0. Figure 3.2a depicts the distribution

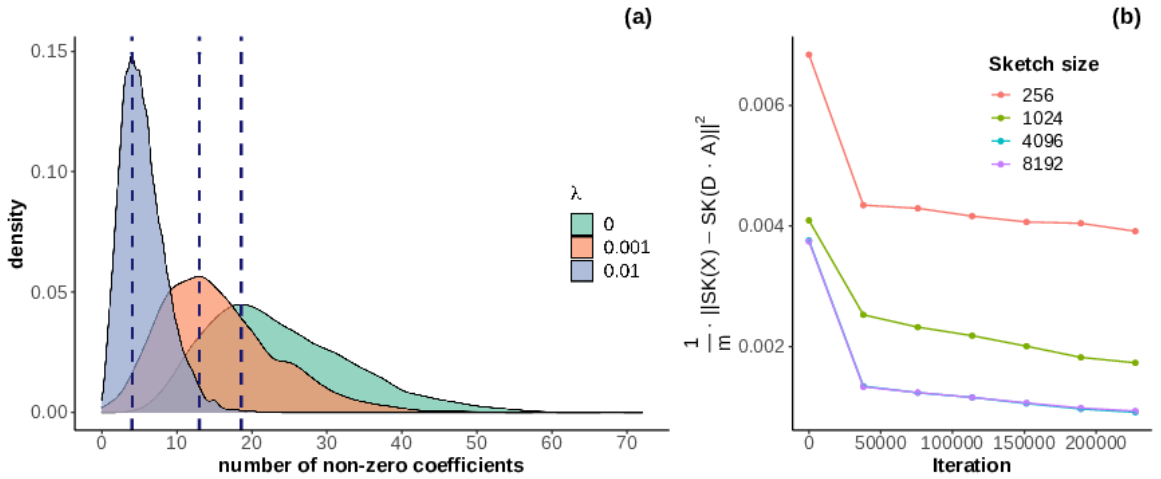


Figure 3.2: (a) Tuning of the regularization parameter λ. The distribution of the number of non-zero coefficients for different values of the regularization parameter λ. The coefficients are computed using the initial dictionary D^0. Vertical dotted lines indicate the mode of each distribution. (b) Value of the objective function of the dictionary update with respect to the number of iterations, with different sketch sizes.

of the number of non-zero elements in each column of C^0 for each λ: According to our expectations, we chose $\lambda = 0.001$.

The tuning of the other five parameters (sketch size, batch size, initial learning rate, decay parameter, and momentum weight) was carried out by means of a classical greed search.

For the sketch size, we focused on how it impacted the objective function value. Notably, increasing the sketch size allows revealing more details about the data distribution, but it increases the execution time. We had tested increasing values of sketch size $m = \{256; 1,024; 4,096; 8,192\}$, and we observed that at some point, this no longer reduced the objective function value (on Figure 3.2b, the lines depicting $m = 4,096$ and $m = 8,192$ superimpose), so that the additional computational time did not worth it: We finally fixed m to 4,096.

Naturally, using large batches leads to a better approximation of the decomposition sketch $SK(D \cdot C^*)$ in Eq. (3.9); yet, from a computational viewpoint the trade-off is not obvious: with larger batches, fewer iterations are required, but each iteration is more computationally demanding. We have considered various powers of 2 as batch sizes:

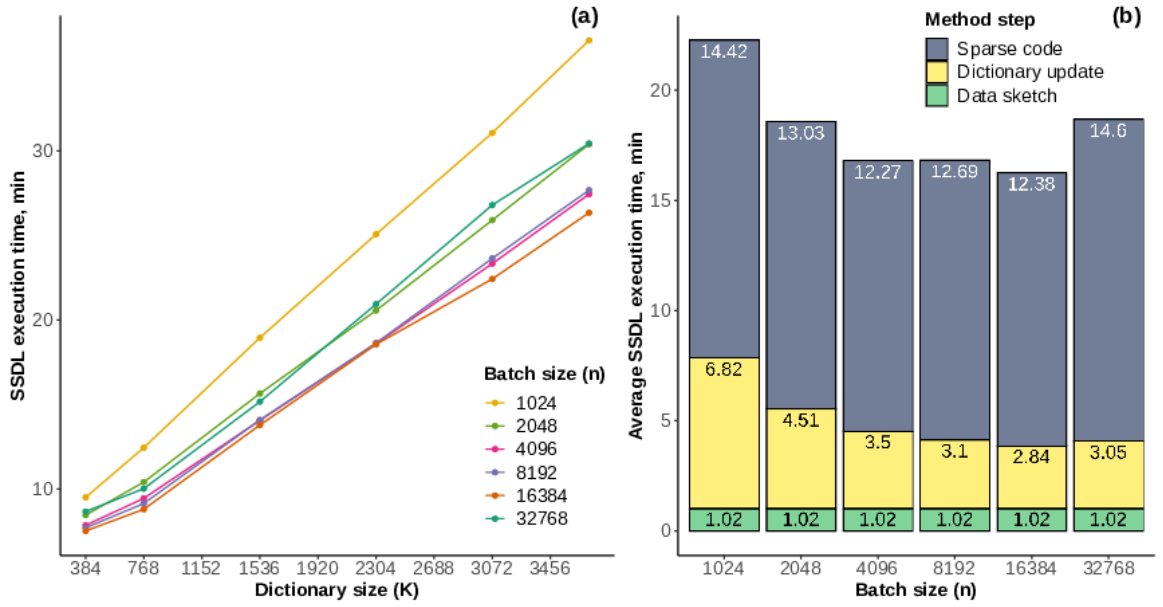


Figure 3.3: SSDL execution time for different batch size values. (a) The SSDL execution time as a function of the dictionary size and of the batch size. (b) Total average execution time as a function of the batch size as well as average execution time for each step among: data sketch computation (green); dictionary update (yellow); and sparse coding (blue).

$2^{10} = 1,024$, $2^{11} = 2,048$, $2^{12} = 4,096$, $2^{13} = 8,192$, $2^{14} = 16,384$. In practice, we observed the batch size did not affect the method convergence, so we focused on the execution time. Figure 3.3a depicts SSDL execution time as a function of the dictionary size K for different batch size tests. Figure 3.3b illustrates SSDL execution time, averaged across the different possible values of K. Overall, SSDL execution time mostly amounts to that of sparse coding, and thanks to our parallelized implementation (see Section 3.4.1), the dictionary update time hardly depends on n for $n \geq 4,096$.

The remaining three parameters (initial learning rate γ_0, decay parameter ν and momentum weight α) influence the convergence speed. We have compared the 27 combinations resulting from the following tuning: $\gamma_0 = \{0.05; 0.1; 0.2\}$, $\nu = \{0; 0.01; 0.1\}$ and $\alpha = \{0; 0.5; 0.9\}$. The tests with $\nu = 0$ correspond to the case of constant learning rate. Three momentum weight values $\alpha = \{0, 0.5, 0.9\}$ represent three scenarios, respectively: (1) the momentum is not involved in the dictionary update (*i.e.*, the classical stochastic mini-batch method), (2) the gradient and

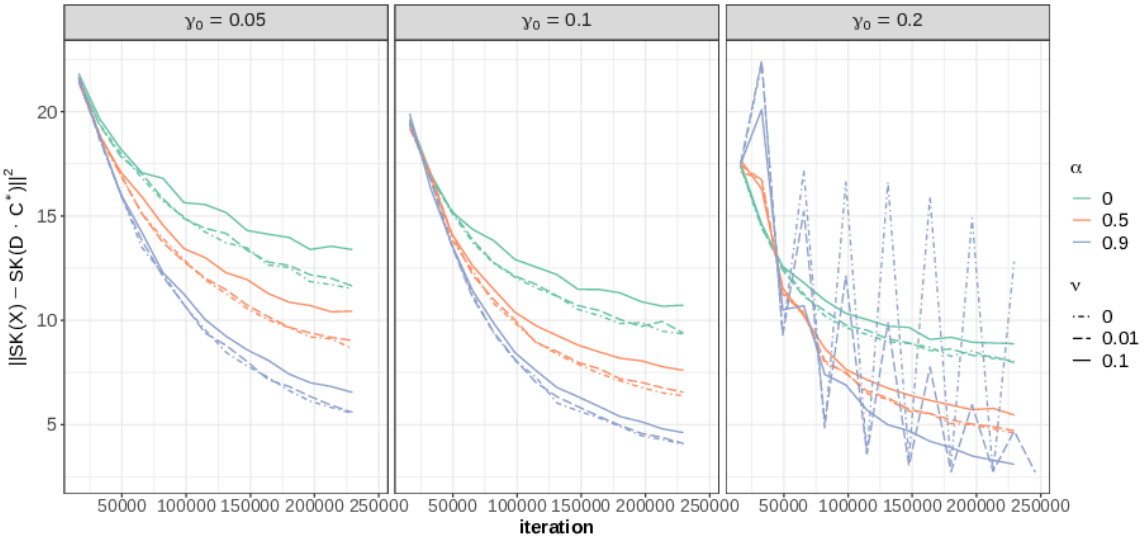


Figure 3.4: SSDL method convergence depending on the learning rate, the momentum weight, and the decay parameter. Subfigures correspond to different initial learning rate tests (the smallest on the left and the biggest on the right). Each subfigure depicts the dictionary update objective function $F(D, C^*)$ as a function of the number of iterations for different parameter value combinations. Different colors depict the three momentum weight scenarios, and different line types illustrate different values of the decay parameter. Batch size is fixed at 16,384. The regularization parameter λ is equal to 0.001.

the momentum have equivalent weights; and (3) the momentum has the majority impact. The results are presented in Figure 3.4.

At first glance, irrespective of the other parameters, the fastest convergence is given by $\alpha = 0.9$. Moreover, there is an important gap in the convergence rate of the classical stochastic mini-batch scheme and the momentum based Nesterov scheme. Naturally, we have considered $\alpha = 0.9$. Furthermore, the higher the initial learning rate γ_0, the lower the final value of the objective function. However, setting the initial learning rate to 0.2 led to too large fluctuations in the objective function, as illustrated on the rightmost part of Figure 3.4. In this case, a large decay parameter $\nu = 0.1$ can correct for this. We also observed that using a decaying learning rate when the initial learning rate is smaller than 0.2 did not improve the convergence rate, and even slowed it down sometimes. Finally, we selected the following tuning: $\alpha = 0.9$, $\gamma_0 = 0.2$ and $\nu = 0.1$.

MODL is driven by five parameters: the reduction parameter r, the

batch size n_{MODL}, the dictionary size K, the regularization parameter λ_{MODL} and the number epochs T_{MODL}. The last tree parameters are the same as for SSDL and they are set to the same values. As for the batch size, it is fixed to a value equal to that of the dictionary size K, as recommended in [178]. Finally, we have tested two scenarios for the reduction level r: (1) We have applied MODL to the original slice of 718 rows with $r = 3$; and (2) we have applied MODL to the sub-sampled slice of 256 rows with $r = 1$. These tests are denoted as $MODL_{718,r=3}$ and $MODL_{256,r=1}$ respectively.

To select the parameters of IcTKM, we simply followed the recommendations of [180]: Random projector based on discrete Fourier transform, sparsity level Λ_{IcTKM} set to the same value as expected for SSDL (see Sup. Mat. 1) and reduction parameter $r_{IcTKM} = 5$ (the highest value according to [180], Table 1, based on the sparsity level and data dimension). As preliminary tests highlighted the important computational load of IcTKM, we decided to combine this dimensionality reduction with our subsampling to 256 rows, and the associated results are denoted as those of $IcTKM_{256,r=5}$. As for algorithm termination, instead of a number of epochs, IcTKM requires to fix the number of iterations. We observed that on our data, 32 of them were sufficient to near the convergence plateau. Finally, concerning the initialization required for all the considered methods, dictionary D^0 is defined by a random selection from the data.

3.4.4 Results

Our comparisons focus on the execution time as well as on the quality of the resulting dictionaries with respect to the expectations listed in Section 3.1. Figure 3.5a depicts the execution time of $\text{MODL}_{256,r=1}$, $\text{MODL}_{718,r=3}$, $\text{IcTKM}_{256,r=5}$ and SSDL (which for the symmetry is denoted as SSDL_{256}) depending on the dictionary size $K = \{384; 768; 1,152; 1,536; 1,920; 2,304; 2,688; 3,072; 3,712\}$ see Section 3.4.3.

Despite the combination of both dimensionality reduction methods,

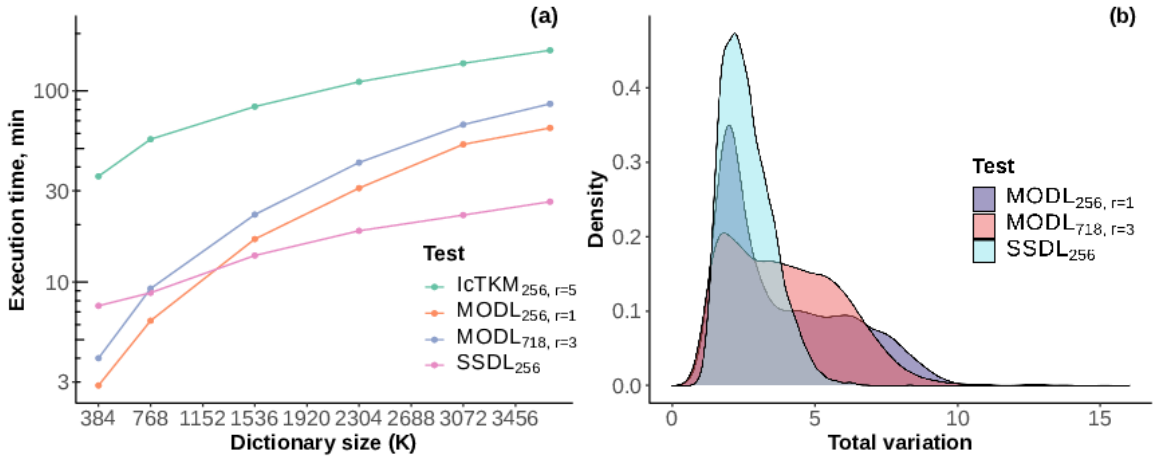


Figure 3.5: (a) The execution time (logarithmic scale) of SSDL, MODL, IcTKM tests as a function of the dictionary size K. The execution time of all methods consists in the execution time spent on the sparse coding and the dictionary update. However, for SSDL (resp. IcTKM) it also includes the data sketch computation time (resp. the random projection operator construction time). (b) The distribution of the dictionary atom total variation for $\text{MODL}_{256,r=1}$ (purple), $\text{MODL}_{718,r=3}$ (red) and SSDL_{256} (light blue) resulting dictionaries.

$\text{IcTKM}_{256,r=5}$ is the slowest approach. MODL is more computationally efficient than SSDL for small dictionaries (broadly, less than 1,000-1,500 atoms for $\text{MODL}_{256,r=1}$ and less than 768 for $\text{MODL}_{718,r=3}$), but it does not easily scale up to too large dictionaries. As a result, for datasets resulting from the LC-MS analysis of highly complex biological samples, SSDL is more efficient: Concretely, for a single slice of E.coli dataset, for which $K \in [3000, 3750]$), SSDL_{256} test is two times faster than $\text{MODL}_{256,r=1}$, three times than $\text{MODL}_{718,r=3}$ and four times than $\text{IcTKM}_{256,r=5}$.

Concerning the dictionary quality, as discussed in Section 3.1, the dictionary atoms must have shapes akin to that of real chromatographic profiles: positive and smooth signals with a Gaussian like, yet slightly asymmetric, shape. Figure 3.6 illustrates with several examples, the type of dictionary atoms obtained by SSDL_{256} (first row), $\text{MODL}_{256,r=1}$ (second row) and $\text{IcTKM}_{256,r=5}$ (third row). Since IcTKM method does not allow to impose any positiveness or smoothness constraints, the obtained dictionary atoms cannot be interpreted as peptides chromatograms, hereby hampering their use for processing multiplexed

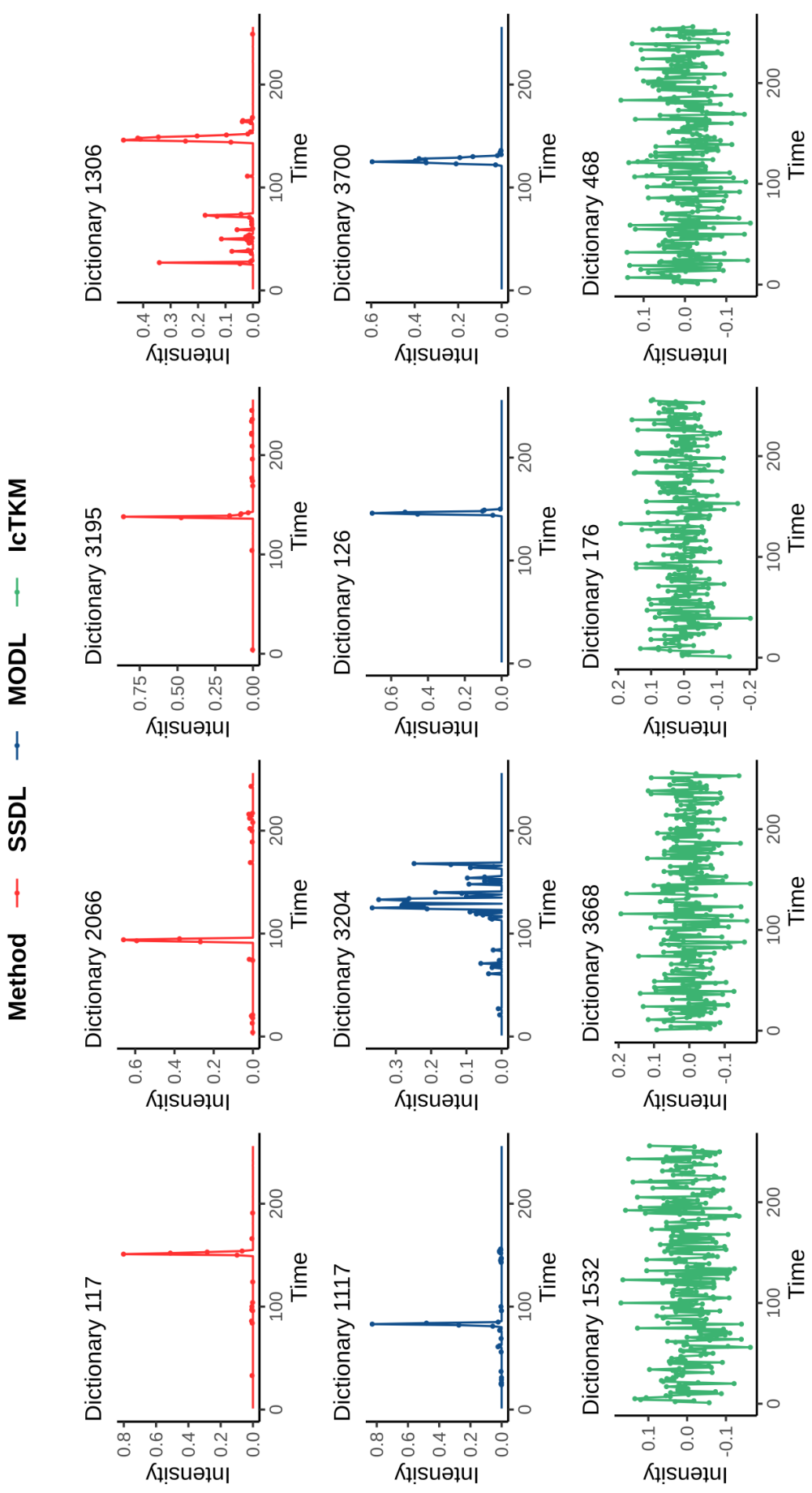


Figure 3.6: The examples of the resulting dictionary atoms obtained by SSDL_{256} (first row), $\mathrm{MODL}_{256,r=1}$ (second row) and $\mathrm{IcTKM}_{256,r=5}$ (third row).

acquisitions. In contrast, both SSDL and MODL provide dictionary atoms with the expected chromatogram shape. Of course, both SSDL and MODL also provides atoms that cannot be interpreted as single chromatogram: for instance, the fourth dictionary atom in the first row of Figure 3.6 contains many well-separated peaks; and the second one in the second row of Figure 3.6 displays what appears to be an overlap of various chromatograms. However, this is simply a consequence of the biological complexity of the analyzed sample, which may require more resoluted instruments as well as, possibly, further improvements in blind source separation.

To obtain a more precise and more exhaustive comparison of MODL and SSDL dictionaries, displaying the total variation distributions is insightful (see Figure 3.5b): Since the dictionary atoms have a unitary norm, chromatogram-like atoms should have a total variation smaller than 2. Both $\text{MODL}_{256,r=1}$ and $\text{MODL}_{718,r=3}$ dictionaries contain many atoms with a total variation norm around 2, but the distributions are also heavy-tailed, with a significant proportion of less smooth atoms (total variation norm lying between 3 and 10). In contrast, the total variation norm distribution for SSDL_{256} does not exceed 6, and indicates that a larger proportion of the atoms are smoother than their MODL counterparts. Overall, SSDL provides with a smaller computational time, dictionaries that are more suited to LC-MS data than those produced with MODL and IcTKM.

3.5 Conclusions

Extracting meaningful patterns from LC-MS data is challenging, because of the multiple constraints attached to their production method: First, the corresponding matrix can be of very large size, especially when resulting from last generation high resolution instruments, hereby requiring scalable approaches. Second, the extracted patterns must have the physical interpretation of a chromatogram and their number must

be consistent with the number of analytes potentially detected in the analyzed sample (up to several thousands). In this article, we have introduced a new dictionary learning method, referred to as Sketched Stochastic Dictionary Learning (SSDL), which combines the latest trends of compressive statistical learning, as well as of online learning and of stochastic optimization, while being compliant with all the aforementioned constraints. This is notably the reason why, compared to state-of-the-art methods, it provides more meaningful dictionaries at a smaller computational cost. Beyond the specificities of LC-MS data, SSDL is also of interest from a more fundamental viewpoint, as to the best of the authors' knowledge, it is the first dictionary learning method that can directly operate on a data sketch (a controlled-sized proxy of the data distribution in the Fourier domain). As future work, we will consider embedding random projection based dimensionality reduction techniques, hereby enabling the processing of entire datasets in a single batch; as well as eventually, the processing of datasets acquired on longer time frames with longer elution columns. From a more applicative viewpoint, SSDL will unleash an efficient and convenient handling of highly multiplexed data acquisitions. Such acquisitions are already an important research direction in proteomics for the depth of analysis they potentially enable, however to date the associated data processing challenges have prevented their widespread use; a hurdle that SSDL will help to overcome.

Acknowledgments

The authors would like to thank Anne-Marie Hesse and Alexandra Kraut for carrying out the mass spectrometry experiment that provided the data on which this paper is based, and also Virginie Brun, Yohann Couté and Christophe Bruley for supports and fruitful discussions.

Data Accessibility

The data that support the findings of this study are openly available in figshare public repository at http://doi.org/10.6084/m9.figshare.13589621.

Author contributions

Olga Permiakova designed the method, implemented the R package, carried out the computational experiments, analysed the results and drafted the manuscript. Thomas Burger designed the method, directed the work, participated to the result analysis and drafted the manuscript. All authors proofread the manuscript and approved its final version.

Financial disclosure

This work was supported by grants from the French National Research Agency: ProFI project (ANR-10- INBS-08), GRAL project (ANR-10-LABX-49-01), DATA@UGA and SYMER projects (ANR-15-IDEX-02) and MIAI @ Grenoble Alpes (ANR-19-P3IA-0003).

CHAPTER 4

General conclusions and perspectives

Bottom-up and label-free approaches using LC-MS acquisition pipelines have considerably increased the throughput of protein identification and quantification. However, despite these progresses, the in-depth proteomics analysis of a complex biological sample remains a challenge. To cope for this, multiplexed acquisitions (based on the simultaneous fragmentation of several peptides) are insightful, despite the multiple issues they raised for subsequent data processing. Recently, it has been demonstrated that relying on a chromatogram library (a collection of chromatographic elution profiles, hopefully the post representative ones) could help with the handling of multiplexed acquisitions, hereby leading to an increment in the number of identified proteins (about 20%). However, to date, these chromatogram libraries can only be generated by additional (and costly) experiments.

In this context, our objective was to propose a computational methodology to construct chromatogram libraries without any additional wet-lab experiments, by exclusively leveraging the structure of the data at hand. This concretely, constitutes a challenge for conceptual and applicative reasons: First, the extracted patterns must look like real and well-formed chromatographic profiles, while they must meaningfully represent all

those of the peptides within the considered sample, even if it is of high biological complexity. Second, the developed workflow must have a computational footprint that is compatible with mass spectrometry platforms constraints, which notably implies to rely on machine learning methods that are scalable.

Overall, we have proposed two solutions to this challenge. The first one, referred to as CHICKN, relies on cluster analysis (as the set of cluster centroids forms a meaningful chromatogram library); while the second solution (termed SSDL) elaborates on the conceptual links between chromatogram libraries and dictionaries resulting from data factorization approaches. The corresponding workflows are schematically presented in Figure 4.1, which emphasizes their common embedding of a data sketching procedure capable of compressing the large-scale input data into a small size vector for subsequent processing[1]. However, beside this common feature, the methods are conceptually different.

The novelty of CHICKN clustering method essentially relies in the combination of the sketching procedure with Nystrom kernel approximation to achieve unprecedented compression level. Moreover, as CHICKN is compliant with the kernel trick, we have proposed two new kernels based on the Wasserstein-1 distance (W1), which both have an interesting semantic for chromatographic elution profiles. Notably, compared to the classically used distances (the Euclidean one as well as the time difference between the retention time apexes) the Wasserstein-1 distance is efficient in capturing simultaneously the shape and time differences. By means of the exponentiation of this distance, we derived the Gaussian W1 and Laplacian W1 kernel functions. The positive (semi-)definiteness of the kernels was formally demonstrated in the Laplacian case and supported by empirical evidence in the Gaussian one.

SSDL is a unique dictionary learning approach, which can learn the dictionary by operating on the data sketch. SSDL updates the dictionary

[1] As mentioned in Chapter 1, there is also a methodological link between these approaches [67]: Both can be formulated as the matrix factorization problem with the sparsity constraint. In the case of the clustering the sparsity level is fixed at 1.

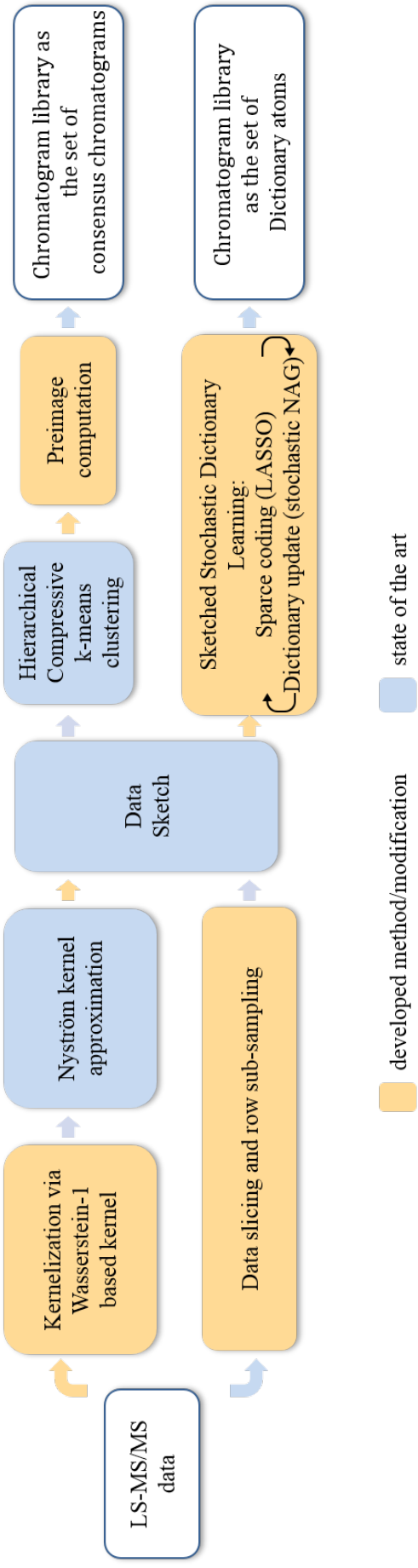


Figure 4.1: The workflows of CHICKN (top pipeline) and SSDL (bottom pipeline) methods. White boxes depict inputs/ outputs. The state-of-the art methods are encompassed in blue boxes, while the developed approaches are depicted by orange boxes and arrows. The acronym NAG stands for Nesterov Accelerated Gradient descent.

using a stochastic version of the Nesterov accelerated gradient descent (NAG) scheme. Combining these machine learning trends allowed SSDL gaining the scalability over both data size and dictionary size and respecting all requirements subsequent to LC-MS data production. To further scale up the algorithm to complex biological samples (containing thousands of peptides), we have also proposed to slice the dataset into overlapping timeframe, and to process these chucks independently. As a result, on LC-MS data, SSDL outperforms the state-of-the-art dictionary learning methods regarding the computational time as well as the quality of the constructed dictionary. Unfortunately, we did not find any way to integrate a Wassertein-1 kernel into the dictionary learning procedure, and thus to benefit from the semantic it would induce. This appoints opens a possible future work direction of real interest (see below).

From a more applicative viewpoint, we observed both methods produced interesting and meaningful chromatogram libraries: Most of the extracted patterns have shapes that meet the expectations of a chromatogram library: First, they have one well-distinguished narrow peak, and their shapes look like a Gaussian bell, yet with a heavier right tail. However, the libraries resulting from both approaches also contain irregular shapes or have multiple peaks, that most likely corresponds to repetitive patterns in the noise, without obvious biological or physical interpretation. However, in both cases, some of them appeared to correspond indeed to real chromatograms with multiple elution profiles. This notably highlights the high level of signal multiplexing in LC-MS data, as well as the room for future improvements. As detailed in Chapter 2, the mass spectrometry engineers from the lab platform were able to gather identification evidence about various precursor ions, hereby validating that CHICKN correctly clusters isotopes of the same peptide. Unfortunately, owing to the abstract nature of the patterns extracted by SSDL (the elution profiles constituting the library are not endowed with an m/z value), a similar validation is to date not possible. It would require further mathematical developments (notably

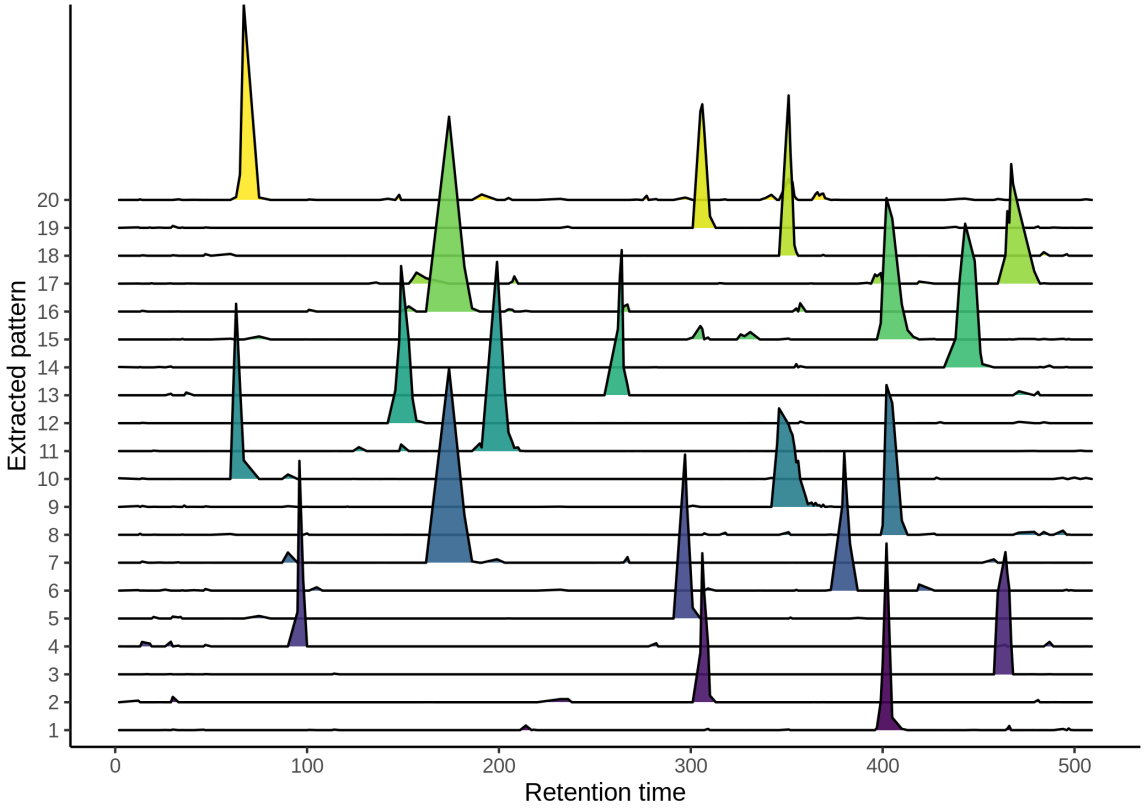


Figure 4.2: 20 examples of SSDL patterns with high intensity.

an efficient sparse projector on the cone spans by the library elements), which despite their obvious interest, stand beyond the scope of this work.

A refined comparison between CHICKN and SSDL libraries is nonetheless insightful: Notably, we have observed that the SSDL library usually contains a greater number of correct shape patterns than the CHICKN library. However, we have also noticed that the SSDL patterns have simpler and smoother forms than CHICKN ones, which is probably more a consequence of the preprocessing choices (SSDL data sampling leading to simplified profiles) than a property strictly attached to the methods. On the other hand, with raw data, CHICKN also captures imperfections that sufficiently repeats across the LC-MS matrix. Finally, both CHICKN and SSDL chromatogram libraries contain duplicated elements, which indicate that in both cases, it would be interesting to further refine the library size according to the data complexity.

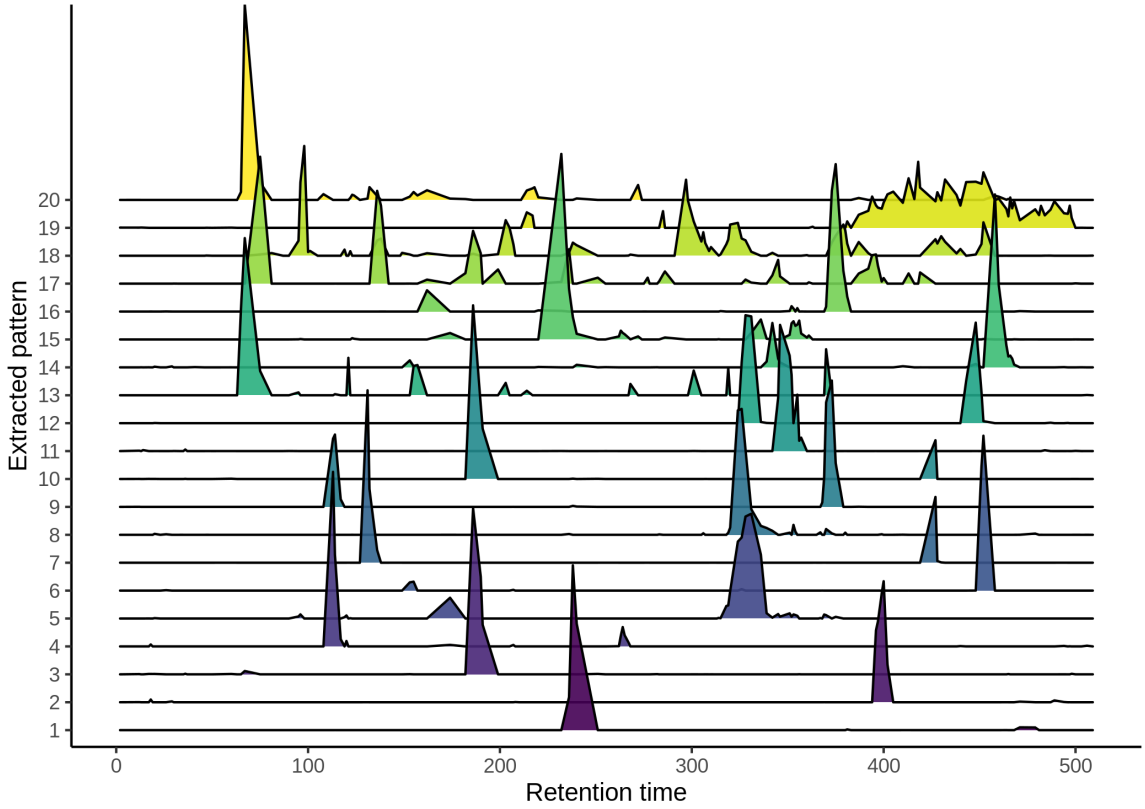


Figure 4.3: 20 examples of SSDL patterns with low intensity.

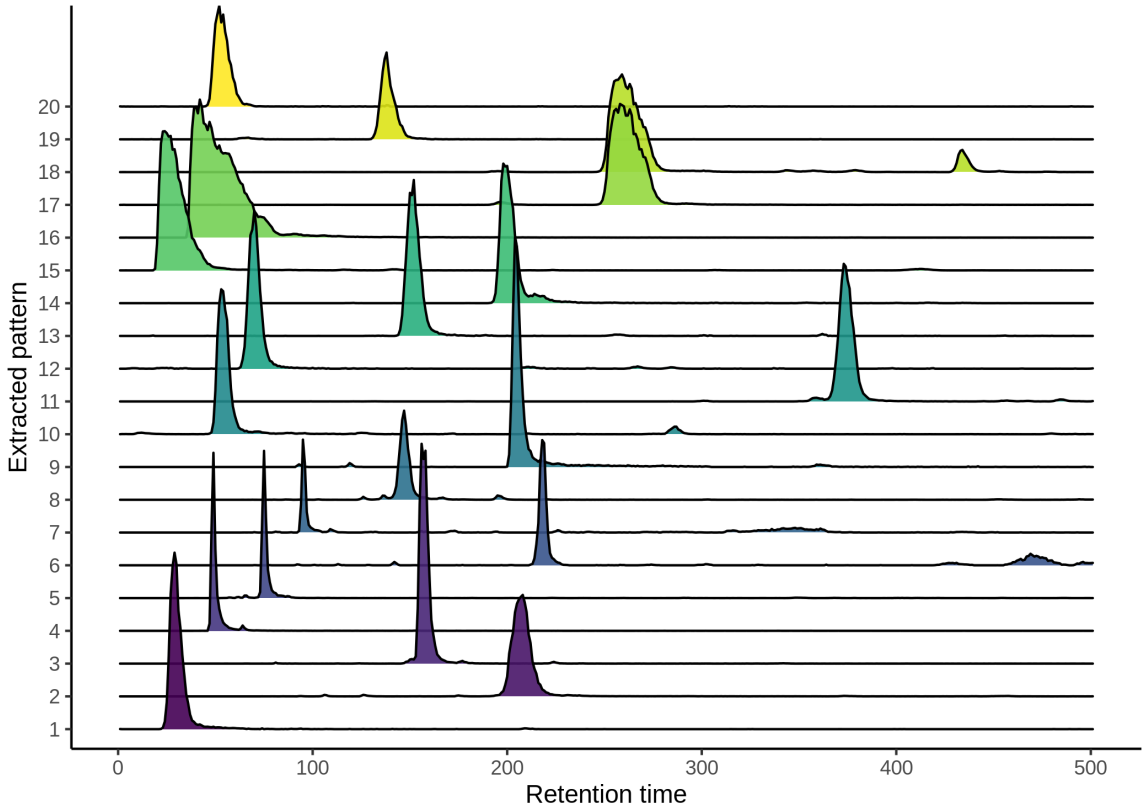


Figure 4.4: 20 examples of CHICKN patterns with high intensity.

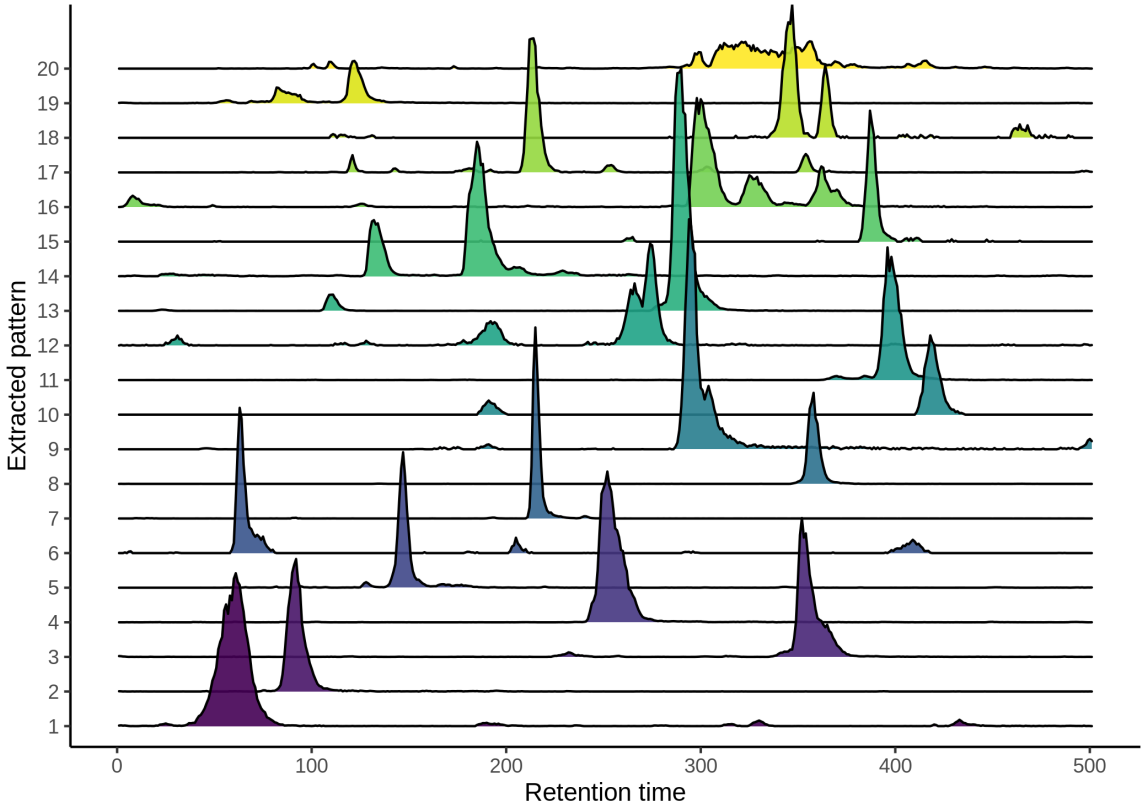


Figure 4.5: 20 examples of CHICKN patterns with low intensity.

From a computational load viewpoint, CHICKN is faster than SSDL. On a four-core computer, the execution time of the entire CHICKN workflow on Ecoli-DIA dataset is around 45 minutes, including compression (*i.e.*, Nyström approximation and sketching), clustering, and preimage computations. With the same machine, SSDL takes around 20 minutes to process 25% of the same dataset (thus, the total execution time would be around 1 hour 20 minutes on the entire dataset). However, considering the volume of the processed data, the computational times of both workflows are acceptable. Moreover, as both approaches are already parallelized, a proteomics platform producing many exceptionally large datasets can reduce the computational time by relying on a highly multicore machine.

As already hinted when summarizing the main results, many directions would be worth investigating, even though they stand beyond the scope of this thesis. One possible work direction is to focus on the improvement of the quality of the chromatographic profiles. Denois-

ing and filtering the LC-MS data before starting the learning process would result in improved dictionary patterns, both for CHICKN and SSDL. It could potentially solve the aforementioned problem of too highly multiplexed patterns. It would also allow to reduce the necessary number of chromatograms in the library, because there would be less noisy structure to encode. However, this pre-processing should be carefully performed. Excessive signal smoothing may result in loss of meaningful peaks. However, this is challenging as the noise in LC-MS data is multiplicative (rather than addition) and not uniform. This is why, I think approaches based on the Wavelet transforms could be effective, as already hinted in the literature [191–193].

As kindly suggested by one of our reviewers, it could be interesting to improve the stopping criterion of CHICKN (decision rule to go further through the clustering hierarchy). For the moment, the number of levels is fixed: one stops when the cluster size is smaller than $2 \cdot k$. Alternatively, more sophisticated criteria can be advantageous to solve the problem of multiplexed clusters for small intensity chromatograms. However, this would potentially lead to an additional parameter to tune, which can be a hurdle from the user viewpoint.

Another enhancement of CHICKN is to use a data-dependent sampling scheme in the Nyström kernel approximation instead of the currently used random sampling. I suggest choosing the recursive sampling presented in [194]. It has been proven to yield a more accurate kernel matrix approximation. This recursive sampling can evaluate the contribution of each data point in the span of the kernel matrix relying on ridge leverage scores, but unlike other related approaches, it has a linear computational time complexity over the data size. This is an important point for CHICKN as compression is its bottleneck.

The important theoretical work on CHICKN is to supplement our empirical demonstration of the positive semi-definiteness of the Gaussian W1 kernel by a theoretical proof. The corresponding investigations can be grounded on the works presented in [147, 195, 196]. More pre-

cisely, according to their theorems, we need to demonstrate either that the squared Wasserstein-1 distance is a conditionally negative definite function; or that the underlying metric space is flat. Both questions are not trivial and require deeper theoretical analysis. However, our experimental observations, together with the fact that the Laplacian W1 kernel already possesses this property make us confident that one of these paths should be successful.

It is worth noting that CHICKN has already been placed on CRAN website, whereas for SSDL, this work has not been finished yet. This will not take a lot of time, as the work is almost finished: the code has already been packaged, and it only remains to complete the documentation, add the examples, and make it compatible with Windows[2]. Once it is done, I will submit SSDL on CRAN website. This technical work requires fewer efforts and less time than other methodological works, but it does not diminish its importance. It will make the SSDL method more accessible and visible in both proteomics and machine learning communities.

As it has already been mentioned in the conclusions of Chapter 3, the performances of SSDL can be increased by introducing more elaborated preprocessing, and notably (preliminary dimensionality reduction, for example by exploring different random projection operators. The choice of possible random projectors is vast, ranging from the classical one, which elements are sampled from the normal Gaussian distribution to the fast Johnson-Lindenstrauss transform, which is generated by randomizing the discrete Fourier unitary matrix. The latter is more time-consuming to construct, but it guarantees lower error rate. Using random projection would lead to both speeding up the algorithm and improving the smoothness of the obtained dictionary elements. However, it may deteriorate the convergence rate since it introduces extra randomness into the learning process (both rows and columns of data matrix being

[2]It is one of the CRAN requirements. The current version of SSDL can be executed only under Linux-like OS. I know exactly how to carry out this work, because I had to do the same modification for CHICKN package.

subsampled).

Another strategy to improve SSDL is to incorporate the kernel trick of CHICKN. It would be of prime interest, from a theoretical viewpoint as well as from an applicative one. On one hand, we would be able to use Nyström method to reduce the signal dimensionality[3]. On other hand, the kernelization would allow to benefit from the appeal of Wasserstein-1 distance. However, doing so it not trivial, and such direction has many caveats: First, it will require to modify both objective functions, sparse coding, and dictionary update. However, it is not entirely clear how to properly incorporate the feature mapping $\Phi(\cdot)$ in the dictionary update objective function, so that the equivalence of the data sketch and the decomposition sketch is ensured. Moreover, this function must have an explicit gradient and its calculation must have a small computational cost. Second, there is no theoretical evidence that the stochastic scheme can work in combination with kernelization and sketching. Third, we need to assess whether the speed-up is significant so that the computationally demanding compression does not degrade the total runtime. Finally, computing the dictionary preimage would also be challenging, as we could not use the same procedure as in CHICKN (where the consensus chromatograms were approximated by averaging chromatograms in the corresponding clusters).

On a more practical side, CHICKN and SSDL can be extended to other applications. Contrary to SSDL, which was developed with a very precise objective (*i.e.*, to learn elementary patterns from multiplex data), CHICKN can be used with more general aims, as it only helps exploring the data structure to unveil hidden patterns. For example, CHICKN could be used align retention times across different LC-MS runs, hereby helping to cope for the poor reproducibility of LC output. This problem often arises in clinical proteomics studies, where it is required to compare the abundance of each peptide through different samples,

[3]It should be noted that Nyström method relies on the matrix column sampling to map the data into the feature space, therefore combining it with the stochastic learning would not deteriorate the convergence rate

generally split into groups of healthy and diseased patients. In this context, the Wasserstein-1 distance-based kernel functions could be of interest, for they simultaneously capture the shape differences and time shift. In addition, with its high-level compression procedure, CHICKN can efficiently process a large number of LC-MS runs simultaneously. Moreover, CHICKN could be dedicated to isotopic envelope construction (this problem has been described in Chapter 2), by incorporating the dependence on $\Delta m/z$ between chromatograms into the kernel function.

However, the appealing application of both methods is the demultiplexing of M2 spectra in DIA data. Achieving this mathematically amounts to factorizing the LC-MS data matrix so that the chromatogram library forms the first matrix of the product (with the strong prejudice that MS1 spectra and MS2 spectra display coherent elution profiles, which is already empirically observed). Doing so would require projecting the MS2 spectra onto the libraries trained from MS1 data to obtain "pseudo- DDA" MS2 spectra. As above, an appropriate projection method among those available in the literature must be chosen. It is possible to use a classical LARS method, which is simple to implement and easy to parallelize for large-scale data. A more elaborate method is Fast Iterative Shrinkage-Thresholding (FISTA), which is naturally adapted to large-scale problem, thanks to its quadratic convergence rate. Another possibility is Nesterov gradient descent method, which has the same global convergence rate as FISTA. It should be mentioned that FISTA is more popular, so that many variants and extensions have been proposed, such as adaptive restart [197] or robust step size search [198], which allow to overcome some its drawbacks. For that reason, I would suggest FISTA in our context.

An interesting aspect of this research direction is that it will connect by the other tail the practical and theoretical sides of my work. More precisely, it will provide an opportunity to concretely evaluate the chemical meaningfulness of the dictionary elements, notably the abstract ones derived from SSDL: In fact, once the projection is performed, it is

possible to reconstruct pseudo-DDA spectra that can be processed with classical proteomics workflows and to use classical proteomics metrics (identification FDR, proteome coverage, etc.) to monitor and compare various library construction strategies.

Aside these methodological research directions, it would be useful to integrate both CHICK and SSDL into a software tool provided with graphical user interfaces. This would allow proteomics experts who are not familiar with programming or scripting to easily use them, to visualize the results, and fine-tune their parameters to best fit their data. Implementing the entire workflow, from building the data matrix from raw LC-MS outputs to the final post-processing (such as automatic pattern validation using peptide identification) is an important engineering workload, which overrides the scope of my PhD studies. Notably, along most of my doctoral studies, my will was to put the focus on the theoretical and algorithmic aspects of my research subject, and I did not attach much importance to the engineering work. However, my preferences and vision of scientific project management have significantly evolved along these few years. In addition of enjoying both parts of the work, and I have become convinced that theory and engineering are in fact are equally important. If I had more time, I would handle this engineering tasks to make my approaches more visible and useful to the proteomics community.

This work experience taught me a lot: I have significantly improved my programming skills as well as my knowledge in machine learning and mass spectrometry. I realized the importance of non-scientific works (communications, publication writing and networking). Notably, by working with the mass spectrometry experts of the lab I learned how to work in an interdisciplinary team. This has led me to improve my pedagogical skills (to present my work to researchers from other fields of study) as well as my curiosity towards other knowledge fields.

APPENDIX A

Non-uniform grid construction from RAW mass spectrometry data

A.1 Motivation for grid-interpolation of LC-MS data

LC-MS data are acquired progressively, one mass spectrum after the other, which from a matrix viewpoint, means row-wise (see Figure 2.1 in Chapter 2). However, CHICKN requires to process elution profiles, *i.e.* accessing the data matrix column-wise. Unfortunately, direct access to elution profiles is not possible: The spectra are acquired and discretized independently from one another, with a non-uniform sampling of the m/z scale, which depends on (i) the m/z values (finer for smaller m/z values, coarser for larger ones); and (ii) the density of peaks (to avoid storing too many zeros between peaks). To cope for these, it is thus necessary to align the raw data, and to interpolate all spectra on a common grid. Moreover, to optimaly store and exploit the LC-MS data, the grid must respect the fluctuation of the m/z precision in function of the m/z value (as aforementioned, finer for smaller m/z values, coarser for larger ones).

A.2 Non-uniform grid construction

To unveil how the grid step should adapt across the m/z range, we computed the differences between consecutive m/z values (referred hereafter to as $\Delta m/z$) within each raw spectrum and analyzed their distribution in function of m/z over all spectra. However, some $\Delta m/z$ reach very high values, as over large m/z intervals devoid of peaks, the signal does not need to be sampled/stored. To cope for this, we have gotten rid of the $\Delta m/z$ for which at least one of the two intensity values was lower than $5 \cdot 10^4$. Although this strategy led to some information loss, the remaining one was largely sufficient to compute statistics and to derive conclusions on how the signal is discretized in the MS constructor data format. In fact, despite this reduction, the amount of obtained $([m/z]_i, [\Delta m/z]_i)$ pairs was still too large to be visualized on a single plot. This is why Figure A.1 (respectively, Figure A.2) depicts only a random 10% (respectively 2%) of them for Ecoli-DIA dataset (respectively, Ecoli-FMS dataset).

It can be observed on both figures that most of the $([m/z]_i, [\Delta m/z]_i)$ aggregate along a curve, which appears continuous and black (while it is in fact a collection of partly transparent crosses) because of the point density (the others, in proportion, rather few of them, being outliers). We empirically found that a power function of degree $\frac{3}{2}$ (dashed red line) led to a perfect fit on the data.

In addition, we noticed that the magnitudes of $\Delta m/z$ in the Ecoli-DIA dataset were 4 times larger than those in the Ecoli-FMS one. As the ratio of instrument resolutions is also exactly 4, a simple regression provided us with the missing coefficient $\frac{0.015}{Res_{EXP}}$, with Res_{EXP} being the resolution tuning of the instrument. Finally, we obtain:

$$[\Delta m/z]_i = \frac{0.015}{Res_{EXP}} [m/z]_i^{\frac{3}{2}}, \quad \forall\, i = 0, \ldots, N-1, \tag{A.1}$$

where N is the last grid index of the grid. Using a more convenient

Dataset	**m/z in Th**	**Δm/z in mTh**
Ecoli-DIA	400	2.0
	800	5.66
	1,000	7.9
	1,200	10.39
	1,400	13.09
Ecoli-FMS	400	0.5
	800	1.41
	1,000	1.98
	1,200	2.6
	1,400	3.27

Table A.1: Some of $\Delta m/z$ values computed at different m/z values (in mili-thomsons) using Eq. (A.2) for Ecoli-DIA ($Res_{EXP} = 60,000$) and Ecoli-FMS ($Res_{EXP} = 240,000$) datasets.

notation where m_i stands for $[m/z]_i$, one obtains Equation (2.1) from Chapter 2:

$$m_{i+1} - m_i = \frac{0.015}{Res_{EXP}} m_i^{\frac{3}{2}} \tag{A.2}$$

Table A.1 illustrates the change of the grid step ($\Delta m/z$, expressed in mili-thomsons) obtained using Eq. (A.2) depending on the m/z values for both considered datasets.

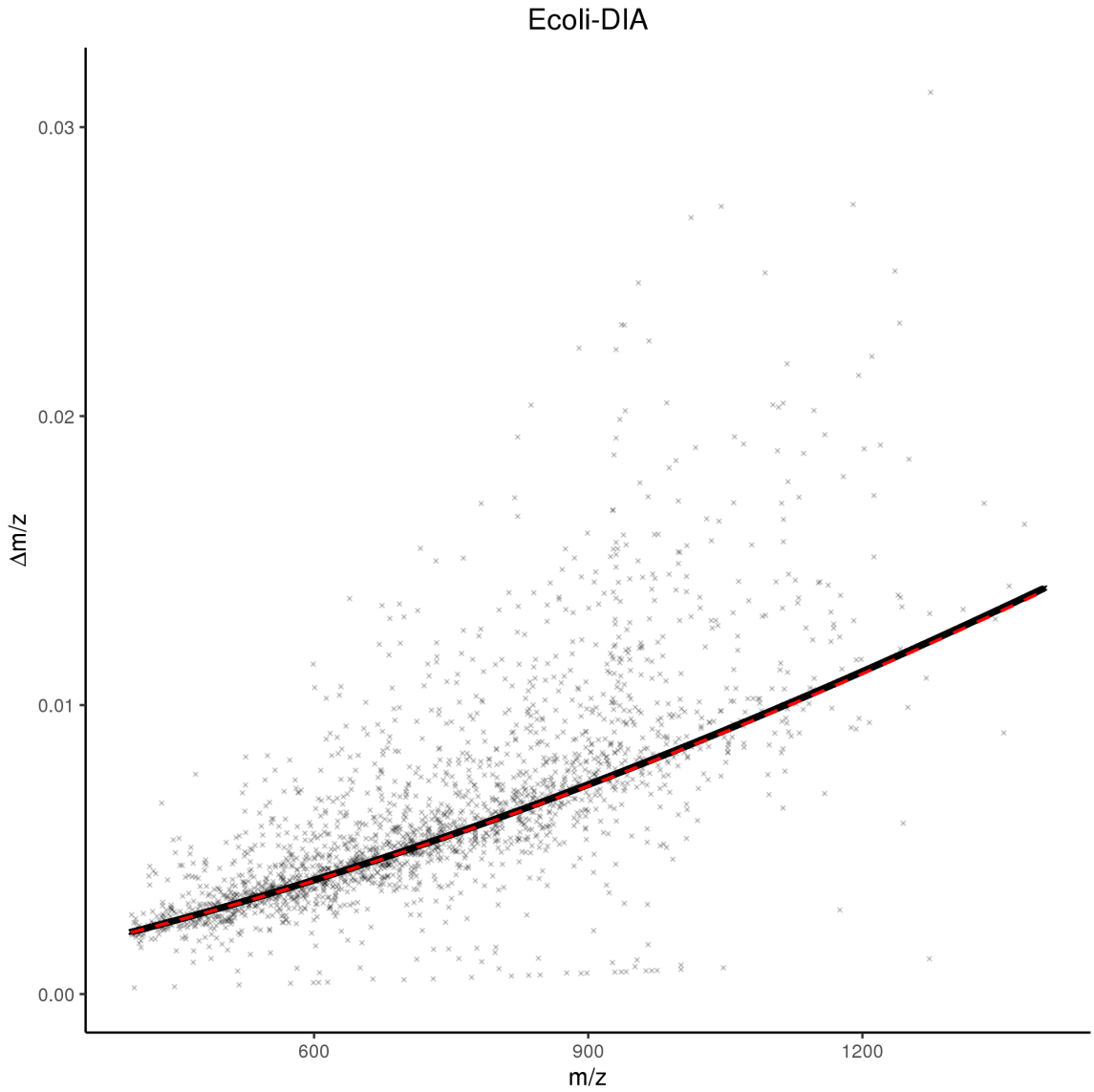


Figure A.1: Gray crosses (in fact, black transparent crosses) depict $(m/z, \Delta m/z)$ pairs computed from raw Ecoli-DIA dataset. Red dashed line depicts Equation (2.1).

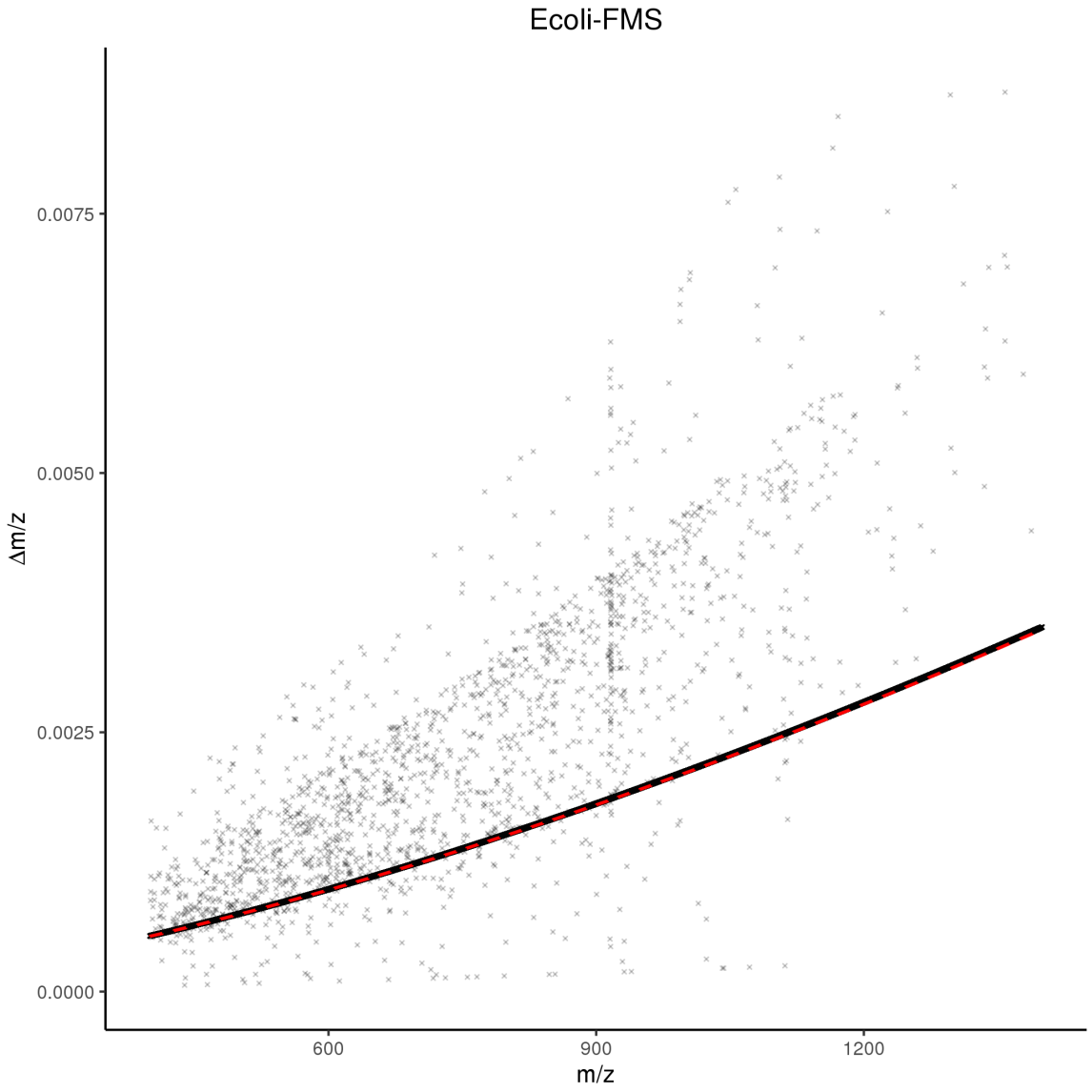


Figure A.2: Gray crosses (in fact, black transparent crosses) depict $(m/z, \Delta m/z)$ pairs computed from raw Ecoli-FMS dataset. Red dashed line depicts Equation (2.1).

APPENDIX B

Positive (semi-)definiteness of Wasserstein-1 based kernels for real-valued signals

B.1 Notations and definitions

Definition 1 (Real-valued series)**.** *This work focuses on real-valued signals, discretized on t time stamps, such as for instance $x := [x^1, \ldots, x^t]$ which can simply be referred to as vector $x \in \mathbb{R}^t$.*

Definition 2 (Wasserstein-1 distance on real-valued series)**.** *Let $x, y \in \mathbb{R}^t$. The W1 distance between x and y reads:*

$$d_{W1}(x, y) = \sum_{k=1}^{t} |F_x(k) - F_y(k)| = \|F_x - F_y\|_{\ell_1}$$

where F_x and F_y are the empirical cumulative functions of signals x and y, respectively:

$$F_x(k) = \sum_{i=1}^{k(k\leq t)} \frac{x^i}{\|x\|_{\ell_1}}$$

and

$$F_y(k) = \sum_{i=1}^{k(k\leq t)} \frac{y^i}{\|y\|_{\ell_1}}$$

Definition 3 (Positive definite and positive semi-definite kernel)**.** *A*

kernel $k(\cdot,\cdot)$ is positive semi-definite (PSD) (respectively, positive definite (PD)) if and only if it is symmetric and for any choice of n distinct $x_1,\dots,x_n \in \mathbb{R}^t$ (respectively, $\in \mathbb{R}^t \setminus 0$) and of $c_1,\dots,c_n \in \mathbb{R}$:

$$\sum_{i,j=1}^{n} c_i c_j k(x_i,x_j) \geq 0 \quad (\textit{respectively, } > 0). \tag{B.1}$$

Property 1. *A kernel k is PSD if and only if for any set of n distinct $x_1,\dots,x_n \in \mathbb{R}^t$, the **kernel matrix** $K \in \mathbb{R}^{n\times n}$ defined by $K_{ij} = k(x_i,x_j)$ has only non-negative eigenvalues.*

Proof. [199] (Theorem 4.1.10 p.231) □

Definition 4 (Gaussian W1 kernel)**.** *$\forall\, x,y \in \mathbb{R}^t, \forall\, \gamma \in \mathbb{R}_+^*$, the Gaussian W1 kernel reads:*

$$k_{GW1}^{\gamma}(x,y) = e^{-\gamma \cdot d_{W_1}(x,y)^2} \tag{B.2}$$

Definition 5 (Laplacian W1 kernel)**.** *$\forall\, x,y \in \mathbb{R}^t, \forall\, \gamma \in \mathbb{R}_+^*$, the Laplacian W1 kernel reads:*

$$k_{LW1}^{\gamma}(x,y) = e^{-\gamma \cdot d_{W_1}(x,y)} \tag{B.3}$$

Definition 6 (Exponential 1D kernel [200])**.** *$\forall\, x,y \in \mathbb{R}, \forall \gamma \in \mathbb{R}_+^*$, the Exponential 1D kernel reads:*

$$k_{E1D}^{\gamma}(x,y) = e^{-\gamma \cdot |x-y|} \tag{B.4}$$

Property 2. *$\forall\, x,y \in \mathbb{R}$, $\gamma \in \mathbb{R}_+^*$, the Exponential 1D kernel $k_{E1D}^{\gamma}(x,y)$ is positive definite.*

Proof. [201] (Corollary 2.10. p. 78 and Theorem 2.2 p. 74) □

B.2 Gaussian W1 kernel

Conjecture 1. $\forall x, y \in \mathbb{R}^t, \forall\ \gamma \in \mathbb{R}_+^*$ *the kernel* $k_1^\gamma(x,y) = e^{-\gamma \cdot \|x-y\|_{\ell_1}^2}$ *is positive definite.*

If **Conjecture 1** holds, then, demonstrating the positive definiteness of k_{GW1}^γ is possible by following a line akin to the one used in the k_{LW1}^γ case (see Section B.3).

Nevertheless, we provide here empirical supports for the PSD-ness of k_{GW1}^γ (which is sufficient to apply the kernel trick): For each dataset, we performed 5 Nyström approximations of the Gaussian W1 kernel matrix, as described in Algorithm 1 (main article) with different random subsampling, and we verified that all the eigenvalues were non-negative (leading to a PSD kernel, according to **Property 1**). The results are reported on Figures B.1, B.2 and B.3, which display the 5 series of eigenvalues (for datasets Ecoli-DIA, Ecoli-FMS and UPS2GT, respectively), sorted by decreasing order, together with the largest (λ_{max}) and smallest (λ_{min}) eigenvalues across all the 5 tests. In addition, we observed that for raw data like Ecoli ones, for which CHICKN was designed, the λ_{min} is clearly positive (contrarily to datasets such as UPS2GT, which by construction may not lead to full rank data matrices). This makes us optimistic about **Conjecture 1**.

B.3 Laplacian W1 kernel

Lemma 1. *Let* $(X_i)_{i=1}^m$ *is a sequence of non empty sets,* $\forall i \in \{1,\dots,m\}$ $x^i, y^i \in X_i$ *and* $(k_i)_{i=1}^m$ *is a sequence of positive definite kernels such that* $k_i : X_i \times X_i \to \mathbb{R}$*, then a kernel defined as:*

$$K((x^1,\dots,x^m),(y^1,\dots,y^m)) = \prod_{i=1}^m k(x_i,y_i) \tag{B.5}$$

is positive definite on $X_1 \times \cdots \times X_m$*.*

Proof. [201] (Corollary 1.13 p. 70). □

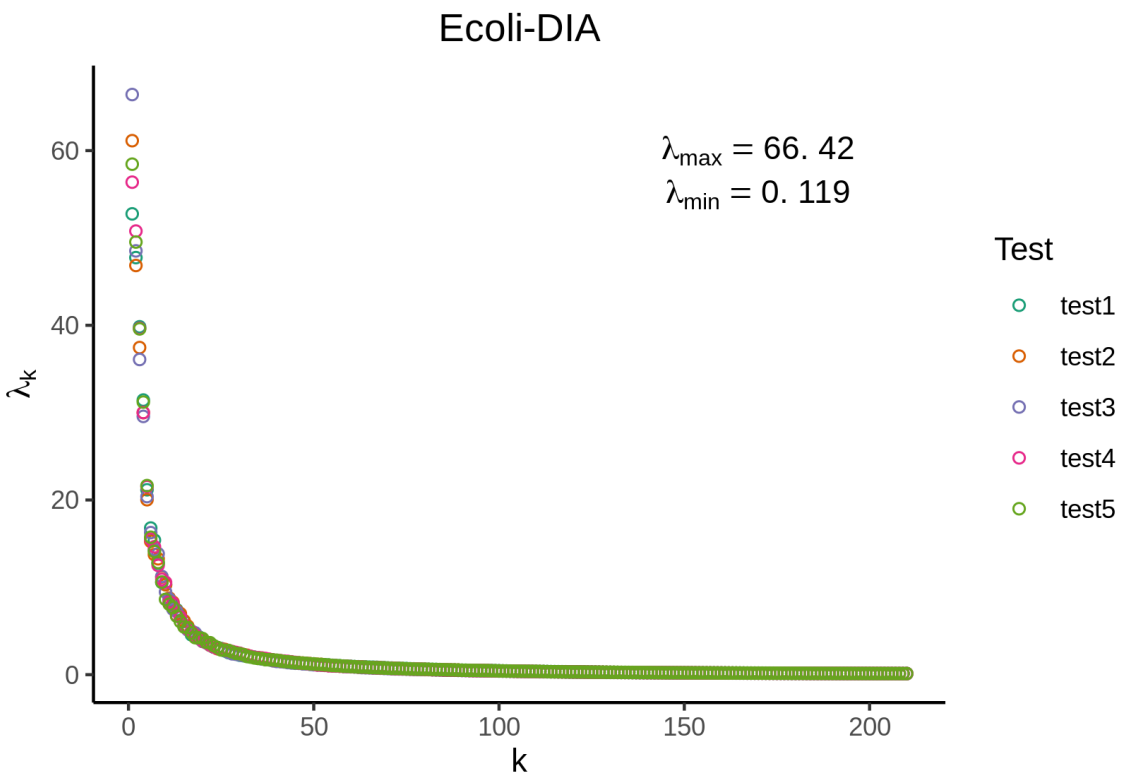


Figure B.1: Matrix spectrum for the 5 repetitions (each with a specific color) of Nyström approximation resulting from Ecoli-DIA dataset. The minimal and maximum values (λ_{min} and λ_{max}, respectively) over these 5 tests are indicated in the upper right corner.

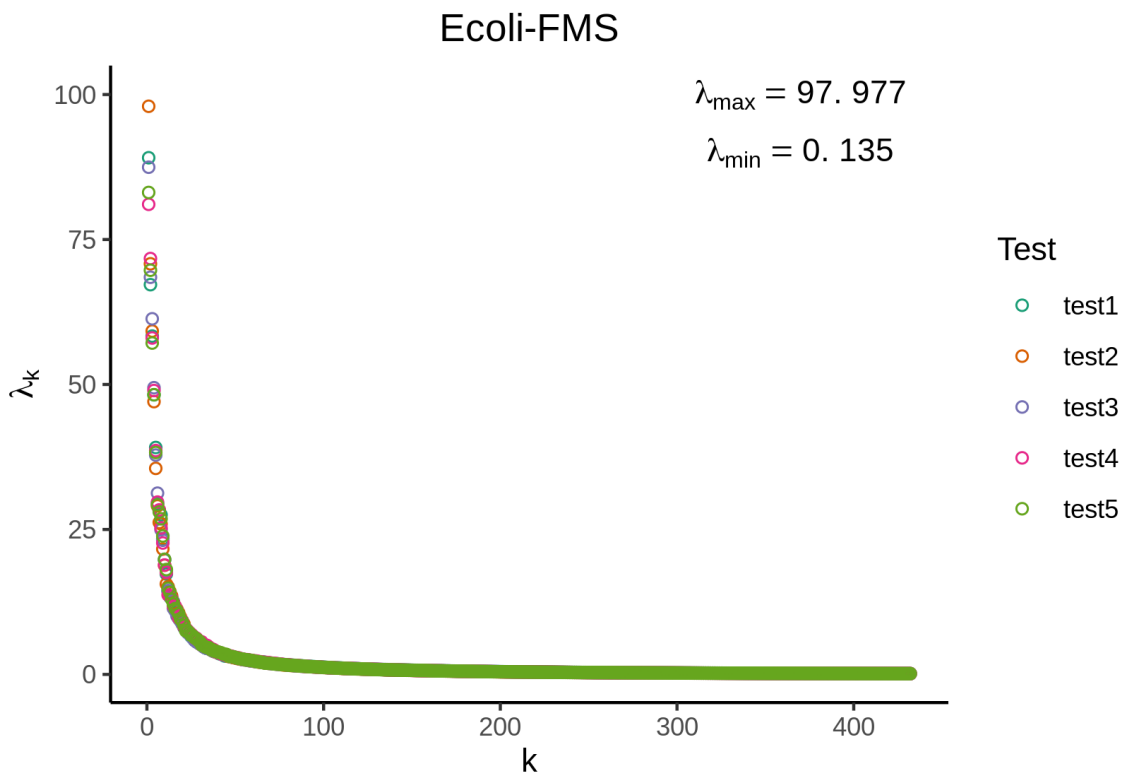


Figure B.2: Matrix spectrum for the 5 repetitions (each with a specific color) of Nyström approximation resulting from Ecoli-FMS dataset. The minimal and maximum values (λ_{min} and λ_{max}, respectively) over these 5 tests are indicated in the upper right corner.

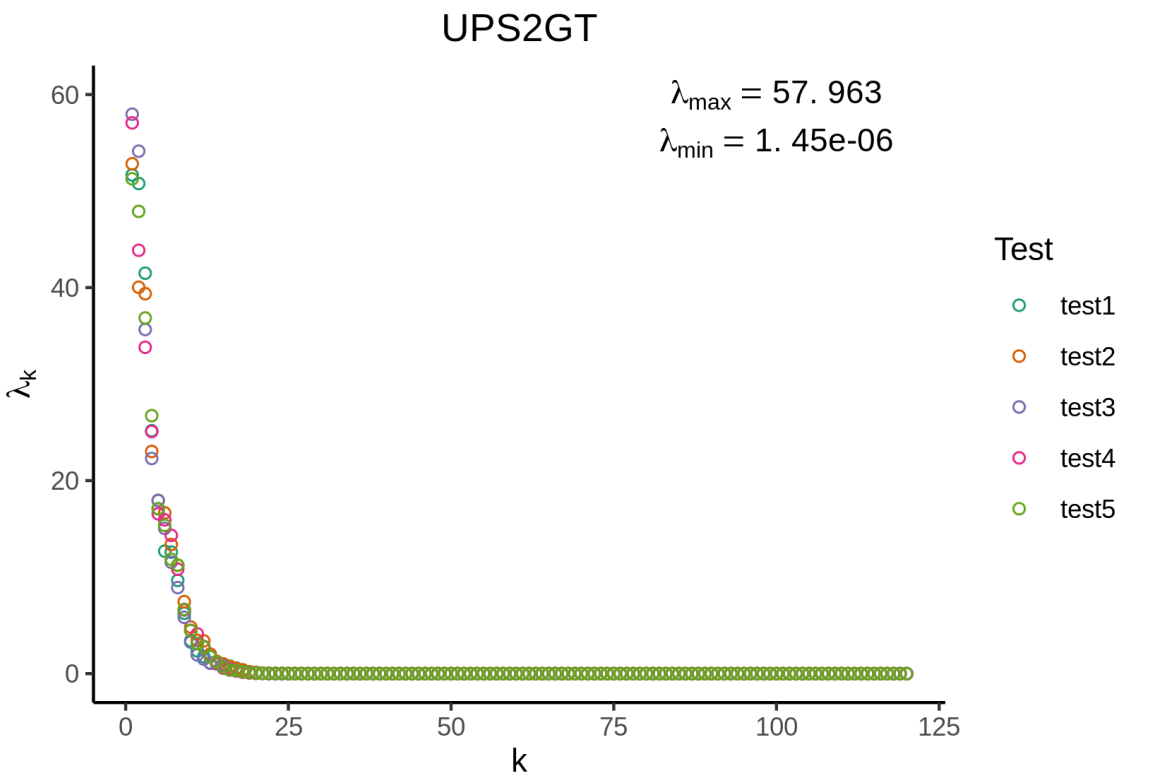


Figure B.3: Matrix spectrum for the 5 repetitions (each with a specific color) of Nyström approximation resulting from UPS2GT dataset. The minimal and maximum values (λ_{min} and λ_{max}, respectively) over these 5 tests are indicated in the upper right corner.

Lemma 2. *$\forall x, y \in \mathbb{R}^t$ and $\gamma \in \mathbb{R}_+^*$ the kernel $k_2^\gamma(x,y) = e^{-\gamma \cdot \|x-y\|_{\ell_1}}$ is positive definite.*

Proof. The ℓ_1 norm of a vector x reads

$$\|x\|_{\ell_1} = \sum_{i=1}^{t} |x^i|,$$

where x^i is i^{th} coordinate of x. The kernel $k_2^\gamma(x,y)$ can be rewritten as follows:

$$k_2^\gamma(x,y) = \prod_{i=1}^{t} e^{-\gamma \cdot |x^i - y^i|}$$

where $(e^{-\gamma \cdot |x^i - y^i|})_{i=1}^t$ is a sequence of Exponential 1D kernels, which are positive definite on $\mathbb{R}$ (**Property 2**). Thus, according to **Lemma 1**, $k_2^\gamma(x,y)$ is also positive definite. □

Corollary 1. *The Laplacian W1 kernel (see **Definition 5**) is positive definite.*

Proof. It is sufficient to notice that according the **Definition 2**, the Wasserstein-1 distance $d_{W1}(x,y)$ reads $\|F_x - F_y\|_{\ell_1}$, where F_x and F_y are the empirical cumulative functions, *i.e.* vectors $\in \mathbb{R}^t$. As the set of the empirical cumulative function $X_F = \{F \in \mathbb{R}^t \mid F^1 \leq, \cdots \leq F_t, \Sigma_{i=1}^t F^i = 1\}$ is a subset of $\mathbb{R}^t$, the positive definiteness of k_{LW1}^γ derives directly from the positive definiteness of k_2^γ (**Lemma 2**). □

APPENDIX C

Spectral vs. spectrum clustering

Distinguishing between spectral clustering and cluster analysis of mass spectra[1]

Hélène Borges [2][a], Romain Guibert[2][a], Olga Permiakova[2][a], Thomas Burger[a,b]

[a] Univ. Grenoble Alpes, CEA, INSERM, BIG-BGE, 38000 Grenoble, France

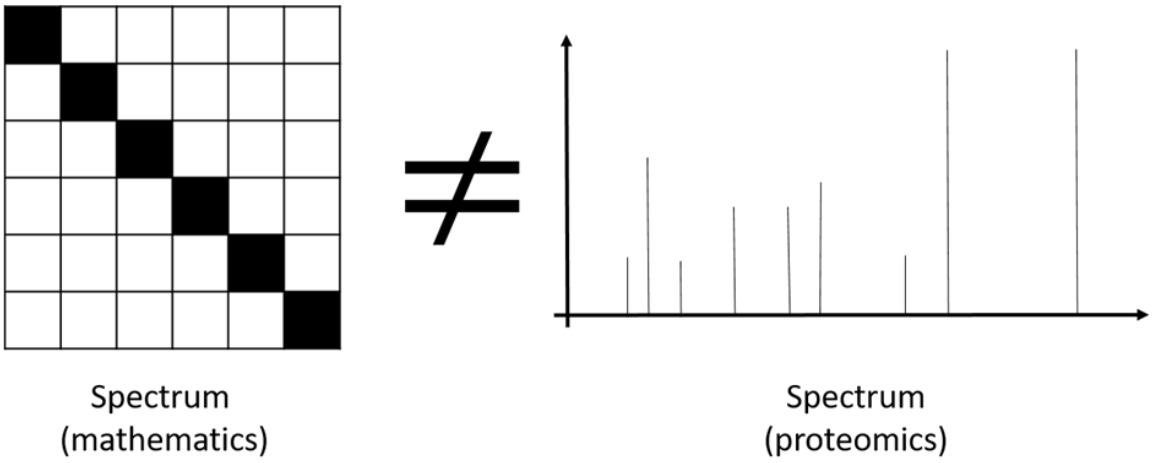
[b] CNRS, BIG-BGE, F-38000 Grenoble, France

Abstract

The term "spectral clustering" is sometimes used to refer to the clustering of mass spectrometry data. However, it also classically refers to a family of popular clustering algorithms. To avoid confusion, a more specific term could advantageously be coined.

[1] This letter article is published in *Journal of Proteome Research*

[2] Equal contribution, listed in alphabetical order



Keywords: Cluster analysis; Spectral clustering; Mass spectrometry; Proteomics

Introduction

In proteomics literature, "spectral clustering" refers to performing a cluster analysis on a dataset resulting from mass spectrometry (MS) acquisitions, with the objective to answer a wide variety of analytical questions (which have recently been surveyed by Perez-Riverol et al. [109]). However, this term also names a widespread family of algorithms for cluster analysis. This ambiguity deteriorates the keyword indexing quality of any work focusing on cluster analysis of MS spectra, and thus complicates the inevitable state-of-the-art review of new research in computational proteomics[3]. We believe there are advantages to adjusting the naming convention. Therefore, we propose to refer to the cluster analysis of MS data as "spectrum clustering" (its original name, see below); or to avoid spelling similarities and improve indexing, as "mass spectrum clustering".

Cluster analysis refers to a wide family of unsupervised statistical learning and multivariate analysis techniques. Roughly speaking, its goal is to aggregate similar observations into clusters, so that the resulting

[3] An anonymous reviewer kindly remarked this question has been debated during the 2015 Mid-Winter Proteomics Informatics meeting at Semmering, (https://coreforlife.eu/events/2014/midwinter-proteomics-bioinformatics-seminar) where the consensus was that of the status quo. The goal of this Letter is to extend and enlarge this discussion

clusters are as dissimilar as possible. Cluster analysis has numerous applications in a variety of scientific domains, including omics biology. Thus, its use on large-scale proteomics data is bound to develop [109].

To the best of the authors' knowledge, the idea of applying cluster analysis to MS based proteomics data [97, 98, 104, 202] goes back to the mid-2000s. Interestingly, the first proposal (in 2004) [98] referred to such clustering task as "spectrum clustering", before the term "spectral clustering" was coined in 2005 by Tabb et al. [97] The latter term was then used from time to time (for instance Bonanza algorithm [104] in 2008, or PRIDE Cluster [102] in 2013), before witnessing a recent regain of interest [109, 203] notably through a scholarly discussion on the subject relayed by Journal of Proteome Research [107, 203].

The term "spectral clustering" also designates a specific family of clustering algorithms, with theoretical foundations that are nearly twenty years old [123, 204]. Its name roots in algebra vocabulary, where the set of eigenvalues of a matrix is commonly referred to as its "spectrum". From an analytical chemistry viewpoint, this naming convention is surprising, while remaining compliant with the general meaning of "spectrum", *i.e.* a decomposition into elementary constituents (light spectrum, mass spectrum, etc.): Conceptually, eigenvalues amount to the atomic elements of a matrix.

Spectral clustering first developed in the machine learning community, and for several years, it has been almost exclusively applied to computer vision problems. This largely explains why, (i) the term was independently coined to refer to the cluster analysis of mass spectra; (ii) no cluster analysis of spectra reported in the proteomics literature has been conducted with it so far (on the contrary, other cluster analysis techniques, *e.g.* hierarchical clustering, are regularly applied to MS-based proteomics data) [102, 106]. However, since its original development, spectral clustering techniques have stepped out of computer vision applications, and have been demonstrated to be extremely powerful on various application domains, so that their popularity is now unparalleled.

Authors & year	Reference	# citations
Ng, Jordan & Weiss (2002)	13	6,967
Von Luxburg (2007)	14	6,087
Zelnik-Manor & Perona (2005)	15	1,793
Dhillon, Guan & Kulis (2004)	16	999
Bengio et al. (2004)	17	1,050

Table C.1: List of the five first articles proposed by Google Scholar when searching «Spectral Clustering», accompanied with the number of citations of these articles (on October 23, 2018).

To date, Shi and Malik's seminal article [123] has gathered more than 15,000 citations according to Google Scholar, and other articles explicitly referring to the term "spectral clustering" in their title gather as many of them (see Table C.1).

Briefly, the principle of spectral clustering is the following: First, the dataset is endowed with a graph structure. Then, one performs a dimensionality reduction guided by the eigenvalues of a specific matrix, referred to as the graph Laplacian (in the algorithm name, "spectral" thus refers to the graph Laplacian eigenvalues). Lastly, k-means clustering is performed via a Lloyd-type algorithm [130]. Concretely, the graph Laplacian encodes the connectivity levels between the vertices of the data graph (a kind of "diffusion capability" for each data item towards its neighborhood). Therefore, working on this matrix makes sense, as good clusters supposedly correspond to sets of highly connected vertices with few inter-cluster connections. This explains why an accurate clustering is expectable, even for datasets with a complex structure that cannot be captured by ball-shape clusters (as with classical k-means). For instance, Figure C.1 represents a famous toy dataset with two intermingled spiraling clusters (classically referred to as Swiss-rolls), on which spectral clustering achieves good performance. Nowadays, applying spectral clustering algorithms to data of various types is rather straightforward thanks to very detailed and pedagogical tutorials [125]

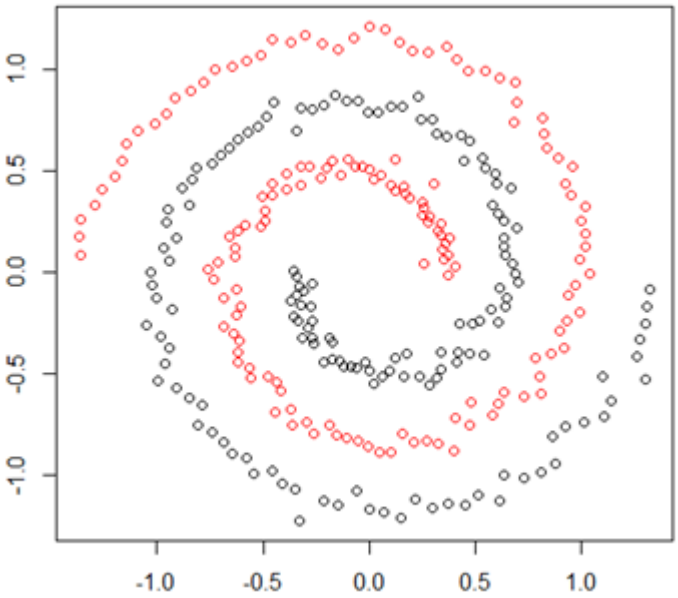


Figure C.1: A typical toy dataset with a complex non-linear structure (two intermingled Swiss-rolls) accurately clustered thanks to the default spectral clustering algorithm available in Kernlab [205].

as well as efficient toolboxes (e.g. Kernlab R package) [205].

Finally, one nowadays uses "spectral clustering" to name two different notions: First, it has been regularly used since 2000 to refer to a family of clustering algorithms. Second, it has been sometimes used since 2005 to refer to cluster analysis of spectral data. Among the large number of co-existing scientific domains, it is common to have the same names used to refer to different concepts. However, as the computational aspects of proteomics grow up, and as big data tools become pervasive in the processing of MS data, some confusion may appear. This is likely for the following reason: Spectral clustering popularity mainly comes from the underlying data embedding into a graph structure, which makes it particularly efficient for network-based data (ranging from social network [206] to interaction networks in biology) [207–209]. As this type of data becomes customary in interactomics studies [210], spectral clustering techniques are likely to become essential tools for proteomics data analysis. As a result, the proteomics community has much to gain in avoiding vocabulary confusion.

In the past, similar vocabulary confusions due to proteomics getting closer to data science were already witnessed. Notably, the concept

of *False Discovery Rate* [211] was confused with what are respectively called in the biostatistics literature, the *False Positive Rate* and the *False Discovery Proportion*, as opportunely pointed by Käll et al. [212] (for the former) and Serang & Käll [213] (for the latter). These two confusions did not help MS experts to get involved with the increasing use of statistics in proteomics. To avoid similar misunderstanding, it would make sense to return to the original naming convention [98] (*i.e.* "spectrum clustering") or to coin a more precise one, specific enough to be well visible and well indexed, such as "mass spectrum clustering".

Author Information

Corresponding author:
Email : thomas.burger@cea.fr
Tel: +33 4 38 78 22 72
ORCID: orcid.org/0000-0003-3539-3564
Notes: The authors declare no competing financial interest.

Acknowledgements

This work was supported by grants from the French National Research Agency: ProFI project (ANR-10-INBS-08), GRAL project (ANR-10-LABX-49-01) and LIFE project (ANR-15-IDEX-02).

List of Figures

List of Tables

Bibliography

[1] Carl Ivar Branden and John Tooze. *Introduction to protein structure*. Garland Science, 2012. (page 11).

[2] Engelbert Buxbaum. *Fundamentals of protein structure and function*, volume 31. Springer, 2007. (page 11).

[3] Srinivasan Damodaran. *Amino acids, peptides and proteins*, volume 4. CRC Press: Boca Raton, FL, 2008. (page 11).

[4] N Leigh Anderson and Norman G Anderson. Proteome and proteomics: new technologies, new concepts, and new words. *Electrophoresis*, 19(11):1853–1861, 1998. (page 11).

[5] Marc R Wilkins, Elisabeth Gasteiger, Andrew A Gooley, Ben R Herbert, Mark P Molloy, Pierre-Alain Binz, Keli Ou, Jean-Charles Sanchez, Amos Bairoch, Keith L Williams, et al. High-throughput mass spectrometric discovery of protein post-translational modifications. *Journal of molecular biology*, 289(3):645–657, 1999. (page 11).

[6] Ingvar Eidhammer, Kristian Flikka, Lennart Martens, and Svein-Ole Mikalsen. *Computational methods for mass spectrometry proteomics*. Wiley Online Library, 2007. (page 15).

[7] Chung-Hsuan Winston Chen. Review of a current role of mass spectrometry for proteome research. *Analytica chimica acta*, 624(1):16–36, 2008. (page 15).

[8] John R Yates, Cristian I Ruse, and Aleksey Nakorchevsky. Proteomics by mass spectrometry: approaches, advances, and applications. *Annual review of biomedical engineering*, 11:49–79, 2009. (page 15).

[9] Bilal Aslam, Madiha Basit, Muhammad Atif Nisar, Mohsin Khurshid, and Muhammad Hidayat Rasool. Proteomics: technologies and their applications. *Journal of chromatographic science*, 55(2):182–196, 2017. (page 15).

[10] Edmond De Hoffmann. Mass spectrometry. *Kirk-Othmer Encyclopedia of Chemical Technology*, 2000. (page 15).

[11] Ruedi Aebersold and Matthias Mann. Mass spectrometry-based proteomics. *Nature*, 422(6928):198–207, 2003. (page 15).

[12] Donald F Hunt, John R Yates, Jeffrey Shabanowitz, Scott Winston, and Charles R Hauer. Protein sequencing by tandem mass spectrometry. *Proceedings of the National Academy of Sciences*, 83(17):6233–6237, 1986. (page 15).

[13] Pedro R Cutillas and John F Timms. *LC-MS/MS in proteomics: methods and applications*. Springer, 2010. (page 15).

[14] Alex Hu, William S Noble, and Alejandro Wolf-Yadlin. Technical advances in proteomics: new developments in data-independent acquisition. *F1000Research*, 5, 2016. (page 15).

[15] David N Perkins, Darryl JC Pappin, David M Creasy, and John S Cottrell. Probability-based protein identification by searching sequence databases using mass spectrometry data. *ELECTROPHORESIS: An International Journal*, 20(18):3551–3567, 1999. (page 24).

[16] David Fenyö and Ronald C Beavis. A method for assessing the statistical significance of mass spectrometry-based protein identifications using general scoring schemes. *Analytical chemistry*, 75(4):768–774, 2003. (page 24).

[17] Jimmy K Eng, Ashley L McCormack, and John R Yates. An approach to correlate tandem mass spectral data of peptides with amino acid sequences in a protein database. *Journal of the american society for mass spectrometry*, 5(11):976–989, 1994. (page 24).

[18] Lewis Y Geer, Sanford P Markey, Jeffrey A Kowalak, Lukas Wagner, Ming Xu, Dawn M Maynard, Xiaoyu Yang, Wenyao Shi, and Stephen H Bryant. Open mass spectrometry search algorithm. *Journal of proteome research*, 3(5):958–964, 2004. (page 24).

[19] Jurgen Cox, Nadin Neuhauser, Annette Michalski, Richard A Scheltema, Jesper V Olsen, and Matthias Mann. Andromeda: a peptide search engine integrated into the maxquant environment. *Journal of proteome research*, 10(4):1794–1805, 2011. (page 24).

[20] Jing Zhang, Lei Xin, Baozhen Shan, Weiwu Chen, Mingjie Xie, Denis Yuen, Weiming Zhang, Zefeng Zhang, Gilles A Lajoie, and Bin Ma. Peaks db: de novo sequencing assisted database search for sensitive and accurate peptide identification. *Molecular & cellular proteomics*, 11(4), 2012. (page 24).

[21] J. Novak, T. Sachsenberg, D. Hoksza, T. Skopal, and O. Kohlbacher. On comparison of simtandem with state-of-the-art peptide identification tools, efficiency

of precursor mass filter and dealing with variable modifications. *Journal of integrative bioinformatics*, 10(3):228, 2013. (page 24).

[22] Brendan MacLean, Daniela M Tomazela, Nicholas Shulman, Matthew Chambers, Gregory L Finney, Barbara Frewen, Randall Kern, David L Tabb, Daniel C Liebler, and Michael J MacCoss. Skyline: an open source document editor for creating and analyzing targeted proteomics experiments. *Bioinformatics*, 26(7):966–968, 2010. (page 25).

[23] Hannes L Röst, George Rosenberger, Pedro Navarro, Ludovic Gillet, Saša M Miladinović, Olga T Schubert, Witold Wolski, Ben C Collins, Johan Malmström, Lars Malmström, et al. Openswath enables automated, targeted analysis of data-independent acquisition ms data. *Nature biotechnology*, 32(3):219–223, 2014. (page 25).

[24] Oliver M Bernhardt, Nathalie Selevsek, Ludovic C Gillet, Oliver Rinner, Paola Picotti, Ruedi Aebersold, and Lukas Reiter. Spectronaut: A fast and efficient algorithm for mrm-like processing of data independent acquisition (swath-ms) data. *Biognosys. ch*, 2012. (page 25).

[25] Vadim Demichev, Christoph B Messner, Spyros I Vernardis, Kathryn S Lilley, and Markus Ralser. Dia-nn: neural networks and interference correction enable deep proteome coverage in high throughput. *Nature methods*, 17(1):41–44, 2020. (page 25).

[26] Chih-Chiang Tsou, Dmitry Avtonomov, Brett Larsen, Monika Tucholska, Hyungwon Choi, Anne-Claude Gingras, and Alexey I Nesvizhskii. Dia-umpire: comprehensive computational framework for data-independent acquisition proteomics. *Nature methods*, 12(3):258–264, 2015. (pages 25, 51, 52, 54, 55, and 80).

[27] Yuanyue Li, Chuan-Qi Zhong, Xiaozheng Xu, Shaowei Cai, Xiurong Wu, Yingying Zhang, Jinan Chen, Jianghong Shi, Shengcai Lin, and Jiahuai Han. Group-dia: analyzing multiple data-independent acquisition mass spectrometry data files. *Nature methods*, 12(12):1105–1106, 2015. (page 25).

[28] Ryan Peckner, Samuel A Myers, Alvaro Sebastian Vaca Jacome, Jarrett D Egertson, Jennifer G Abelin, Michael J MacCoss, Steven A Carr, and Jacob D Jaffe. Specter: linear deconvolution for targeted analysis of data-independent acquisition mass spectrometry proteomics. *Nature methods*, 15(5):371, 2018. (pages 26, 47, 50, and 104).

[29] Ying S Ting, Jarrett D Egertson, James G Bollinger, Brian C Searle, Samuel H Payne, William Stafford Noble, and Michael J MacCoss. Pecan: library-free

peptide detection for data-independent acquisition tandem mass spectrometry data. *Nature methods*, 14(9):903–908, 2017. (page 26).

[30] Jan Eriksson and David Fenyö. Probity: a protein identification algorithm with accurate assignment of the statistical significance of the results. *Journal of proteome research*, 3(1):32–36, 2004. (page 26).

[31] Alexey I Nesvizhskii, Andrew Keller, Eugene Kolker, and Ruedi Aebersold. A statistical model for identifying proteins by tandem mass spectrometry. *Analytical chemistry*, 75(17):4646–4658, 2003. (pages 26 and 27).

[32] Oliver Serang, Michael J MacCoss, and William Stafford Noble. Efficient marginalization to compute protein posterior probabilities from shotgun mass spectrometry data. *Journal of proteome research*, 9(10):5346–5357, 2010. (page 26).

[33] David K Han, Jimmy Eng, Huilin Zhou, and Ruedi Aebersold. Quantitative profiling of differentiation-induced microsomal proteins using isotope-coded affinity tags and mass spectrometry. *Nature biotechnology*, 19(10):946–951, 2001. (page 27).

[34] David L Tabb, W Hayes McDonald, and John R Yates. Dtaselect and contrast: tools for assembling and comparing protein identifications from shotgun proteomics. *Journal of proteome research*, 1(1):21–26, 2002. (page 27).

[35] James S Eddes, Eugene A Kapp, David F Frecklington, Lisa M Connolly, Meredith J Layton, Robert L Moritz, and Richard J Simpson. Chomper: A bioinformatic tool for rapid validation of tandem mass spectrometry search results associated with high-throughput proteomic strategies. *PROTEOMICS: International Edition*, 2(9):1097–1103, 2002. (page 27).

[36] Roger E Moore, Mary K Young, and Terry D Lee. Qscore: an algorithm for evaluating sequest database search results. *Journal of the American Society for Mass Spectrometry*, 13(4):378–386, 2002. (page 27).

[37] Yoav Benjamini and Yosef Hochberg. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal statistical society: series B (Methodological)*, 57(1):289–300, 1995. (page 27).

[38] Yohann Couté, Christophe Bruley, and Thomas Burger. Beyond target-decoy competition: stable validation of peptide and protein identifications in mass spectrometry-based discovery proteomics. *bioRxiv*, 2019. (page 27).

[39] Lukas Reiter, Manfred Claassen, Sabine P Schrimpf, Marko Jovanovic, Alexander Schmidt, Joachim M Buhmann, Michael O Hengartner, and Ruedi Aebersold.

Protein identification false discovery rates for very large proteomics data sets generated by tandem mass spectrometry. *Molecular & Cellular Proteomics*, 8(11):2405–2417, 2009. (page 28).

[40] Shao-En Ong, Blagoy Blagoev, Irina Kratchmarova, Dan Bach Kristensen, Hanno Steen, Akhilesh Pandey, and Matthias Mann. Stable isotope labeling by amino acids in cell culture, silac, as a simple and accurate approach to expression proteomics. *Molecular & cellular proteomics*, 1(5):376–386, 2002. (page 28).

[41] Steven P Gygi, Beate Rist, Scott A Gerber, Frantisek Turecek, Michael H Gelb, and Ruedi Aebersold. Quantitative analysis of complex protein mixtures using isotope-coded affinity tags. *Nature biotechnology*, 17(10):994–999, 1999. (page 28).

[42] Loïc Dayon, Alexandre Hainard, Virginie Licker, Natacha Turck, Karsten Kuhn, Denis F Hochstrasser, Pierre R Burkhard, and Jean-Charles Sanchez. Relative quantification of proteins in human cerebrospinal fluids by ms/ms using 6-plex isobaric tags. *Analytical chemistry*, 80(8):2921–2931, 2008. (page 28).

[43] Philip L Ross, Yulin N Huang, Jason N Marchese, Brian Williamson, Kenneth Parker, Stephen Hattan, Nikita Khainovski, Sasi Pillai, Subhakar Dey, Scott Daniels, et al. Multiplexed protein quantitation in saccharomyces cerevisiae using amine-reactive isobaric tagging reagents. *Molecular & cellular proteomics*, 3(12):1154–1169, 2004. (page 28).

[44] Bertrand Fabre, Thomas Lambour, David Bouyssié, Thomas Menneteau, Bernard Monsarrat, Odile Burlet-Schiltz, and Marie-Pierre Bousquet-Dubouch. Comparison of label-free quantification methods for the determination of protein complexes subunits stoichiometry. *EuPA Open Proteomics*, 4:82–86, 2014. (page 28).

[45] Matthew E Monroe, Nikola Tolić, Navdeep Jaitly, Jason L Shaw, Joshua N Adkins, and Richard D Smith. Viper: an advanced software package to support high-throughput lc-ms peptide identification. *Bioinformatics*, 23(15):2021–2023, 2007. (page 28).

[46] Jürgen Cox and Matthias Mann. Maxquant enables high peptide identification rates, individualized ppb-range mass accuracies and proteome-wide protein quantification. *Nature biotechnology*, 26(12):1367–1372, 2008. (pages 28, 51, and 52).

[47] David Bouyssié, Anne-Marie Hesse, Emmanuelle Mouton-Barbosa, Magali Rompais, Charlotte Macron, Christine Carapito, Anne Gonzalez de Peredo,

Yohann Couté, Véronique Dupierris, Alexandre Burel, et al. Proline: an efficient and user-friendly software suite for large-scale proteomics. *Bioinformatics*, 36(10):3148–3155, 2020. (page 28).

[48] Xiaojin Zhu and Andrew B Goldberg. Introduction to semi-supervised learning. *Synthesis lectures on artificial intelligence and machine learning*, 3(1):1–130, 2009. (page 32).

[49] Lukas Käll, Jesse D Canterbury, Jason Weston, William Stafford Noble, and Michael J MacCoss. Semi-supervised learning for peptide identification from shotgun proteomics datasets. *Nature methods*, 4(11):923–925, 2007. (page 32).

[50] Bernd Fischer, Jonas Grossmann, Volker Roth, Wilhelm Gruissem, Sacha Baginsky, and Joachim M Buhmann. Semi-supervised lc/ms alignment for differential proteomics. *Bioinformatics*, 22(14):e132–e140, 2006. (pages 32 and 50).

[51] Hyungwon Choi and Alexey I Nesvizhskii. Semisupervised model-based validation of peptide identifications in mass spectrometry-based proteomics. *Journal of proteome research*, 7(01):254–265, 2008. (page 32).

[52] Robert Tibshirani. Regression shrinkage and selection via the lasso: a retrospective. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(3):273–282, 2011. (page 33).

[53] Mathew Gutierrez, Kyle Handy, and Rob Smith. Xnet: A bayesian approach to extracted ion chromatogram clustering for precursor mass spectrometry data. *Journal of proteome research*, 18(7):2771–2778, 2019. (pages 33, 50, 51, 52, 53, and 82).

[54] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. Technical report, Stanford, 2006. (page 35).

[55] Chris Ding and Xiaofeng He. K-means clustering via principal component analysis. In *Proceedings of the twenty-first international conference on Machine learning*, page 29, 2004. (page 35).

[56] Nicolas Keriven, Nicolas Tremblay, Yann Traonmilin, and Rémi Gribonval. Compressive k-means. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6369–6373. IEEE, 2017. (pages 35, 44, and 108).

[57] Hélène Borges, Romain Guibert, Olga Permiakova, and Thomas Burger. Distinguishing between spectral clustering and cluster analysis of mass spectra. *Journal of proteome research*, 18(1):571–573, 2018. (pages 35 and 52).

[58] Charlie Frogner, Chiyuan Zhang, Hossein Mobahi, Mauricio Araya, and Tomaso A Poggio. Learning with a wasserstein loss. In *Advances in neural information processing systems*, pages 2053–2061, 2015. (page 36).

[59] Cédric Villani. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008. (page 36).

[60] Justin Solomon, Fernando De Goes, Gabriel Peyré, Marco Cuturi, Adrian Butscher, Andy Nguyen, Tao Du, and Leonidas Guibas. Convolutional wasserstein distances: Efficient optimal transportation on geometric domains. *ACM Transactions on Graphics (TOG)*, 34(4):1–11, 2015. (page 36).

[61] Bjorn Engquist and Brittany D Froese. Application of the wasserstein metric to seismic signals. *arXiv preprint arXiv:1311.4581*, 2013. (page 36).

[62] Kangyu Ni, Xavier Bresson, Tony Chan, and Selim Esedoglu. Local histogram based segmentation using the wasserstein distance. *International journal of computer vision*, 84(1):97–111, 2009. (page 36).

[63] Leonid Vitalievich Kantorovich. On the translocation of masses. In *Dokl. Akad. Nauk. USSR (NS)*, volume 37, pages 199–201, 1942. (page 37).

[64] Elizaveta Levina and Peter Bickel. The earth mover's distance is the mallows distance: Some insights from statistics. In *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, volume 2, pages 251–256. IEEE, 2001. (page 37).

[65] J Mercer. Functions of positive and negative type, and their connection with the theory of integral equations. *RSPTA*, 209:415–446, 1909. (page 38).

[66] Daniel D Lee and H Sebastian Seung. Algorithms for non-negative matrix factorization. In *Advances in neural information processing systems*, pages 556–562, 2001. (page 39).

[67] Chris Ding, Xiaofeng He, and Horst D Simon. On the equivalence of nonnegative matrix factorization and spectral clustering. In *Proceedings of the 2005 SIAM international conference on data mining*, pages 606–610. SIAM, 2005. (pages 39, 108, and 125).

[68] Chris Ding, Tao Li, Wei Peng, and Haesun Park. Orthogonal nonnegative matrix t-factorizations for clustering. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 126–135, 2006. (page 39).

[69] Michal Aharon, Michael Elad, and Alfred Bruckstein. K-svd: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on signal processing*, 54(11):4311–4322, 2006. (pages 40 and 106).

[70] Bradley Efron, Trevor Hastie, Iain Johnstone, Robert Tibshirani, et al. Least angle regression. *The Annals of statistics*, 32(2):407–499, 2004. (pages 40 and 106).

[71] Thomas Blumensath and Mike E Davies. Iterative hard thresholding for compressed sensing. *Applied and computational harmonic analysis*, 27(3):265–274, 2009. (page 40).

[72] Stéphane G Mallat and Zhifeng Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on signal processing*, 41(12):3397–3415, 1993. (page 40).

[73] Tong Tong Wu, Kenneth Lange, et al. Coordinate descent algorithms for lasso penalized regression. *The Annals of Applied Statistics*, 2(1):224–244, 2008. (page 40).

[74] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010. (pages 40, 42, 106, and 109).

[75] Christopher KI Williams and Matthias Seeger. Using the nyström method to speed up kernel machines. In *Advances in neural information processing systems*, pages 682–688, 2001. (page 40).

[76] Shusen Wang, Alex Gittens, and Michael W Mahoney. Scalable kernel k-means clustering with nyström approximation: relative-error bounds. *The Journal of Machine Learning Research*, 20(1):431–479, 2019. (page 41).

[77] Boris T Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964. (page 42).

[78] Yurii E Nesterov. A method for solving the convex programming problem with convergence rate o (1/kˆ 2). In *Dokl. akad. nauk Sssr*, volume 269, pages 543–547, 1983. (pages 42, 108, and 109).

[79] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147, 2013. (page 42).

[80] Nicolas Keriven, Anthony Bourrier, Rémi Gribonval, and Patrick Pérez. Sketching for large-scale learning of mixture models. *Information and Inference: A Journal of the IMA*, 7(3):447–508, 2018. (pages 43, 60, 66, 108, and 110).

[81] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in neural information processing systems*, pages 1177–1184, 2008. (pages 44 and 66).

[82] Geoffrey Davis, Stephane Mallat, and Zhifeng Zhang. Adaptive time-frequency approximations with matching pursuits. In *Wavelet Analysis and Its Applications*, volume 5, pages 271–293. Elsevier, 1994. (page 45).

[83] Brian C Searle, Lindsay K Pino, Jarrett D Egertson, Ying S Ting, Robert T Lawrence, Brendan X MacLean, Judit Villén, and Michael J MacCoss. Chromatogram libraries improve peptide detection and quantification by data independent acquisition mass spectrometry. *Nature communications*, 9(1):1–12, 2018. (page 45).

[84] Alexandre Panchaud, Alexander Scherl, Scott A Shaffer, Priska D von Haller, Hemantha D Kulasekara, Samuel I Miller, and David R Goodlett. Precursor acquisition independent from ion count: how to dive deeper into the proteomics ocean. *Analytical chemistry*, 81(15):6481–6488, 2009. (page 46).

[85] Eric F Strittmatter, Lars J Kangas, Konstantinos Petritis, Heather M Mottaz, Gordon A Anderson, Yufeng Shen, Jon M Jacobs, David G Camp, and Richard D Smith. Application of peptide lc retention time information in a discriminant function for peptide identification by tandem mass spectrometry. *Journal of proteome research*, 3(4):760–769, 2004. (page 46).

[86] Wei-Jun Qian, Tao Liu, Matthew E Monroe, Eric F Strittmatter, Jon M Jacobs, Lars J Kangas, Konstantinos Petritis, David G Camp, and Richard D Smith. Probability-based evaluation of peptide and protein identifications from tandem mass spectrometry and sequest analysis: the human proteome. *Journal of proteome research*, 4(1):53–62, 2005. (page 46).

[87] Konstantinos Petritis, Lars J Kangas, Patrick L Ferguson, Gordon A Anderson, Ljiljana Paša-Tolić, Mary S Lipton, Kenneth J Auberry, Eric F Strittmatter, Yufeng Shen, Rui Zhao, et al. Use of artificial neural networks for the accurate prediction of peptide liquid chromatography elution times in proteome analyses. *Analytical chemistry*, 75(5):1039–1048, 2003. (page 46).

[88] Michael Woldegebriel and Eduard Derks. Artificial neural network for probabilistic feature recognition in liquid chromatography coupled to high-resolution mass spectrometry. *Analytical chemistry*, 89(2):1212–1221, 2017. (page 47).

[89] Michael Woldegebriel and Gabriel Vivó-Truyols. Probabilistic model for untargeted peak detection in lc–ms using bayesian statistics. *Analytical chemistry*, 87(14):7345–7355, 2015. (page 47).

[90] Michael Woldegebriel, Paul Zomer, Hans GJ Mol, and Gabriel Vivó-Truyols. Application of fragment ion information as further evidence in probabilistic compound screening using bayesian statistics and machine learning: A leap toward automation. *Analytical chemistry*, 88(15):7705–7714, 2016. (page 47).

[91] Pan Du, Warren A Kibbe, and Simon M Lin. Improved peak detection in mass spectrum by incorporating continuous wavelet transform-based pattern matching. *Bioinformatics*, 22(17):2059–2065, 2006. (page 47).

[92] Mikko Katajamaa, Jarkko Miettinen, and Matej Orešič. Mzmine: toolbox for processing and visualization of mass spectrometry based molecular profile data. *Bioinformatics*, 22(5):634–636, 2006. (page 47).

[93] Hai-Yan Fu, Jun-Wei Guo, Yong-Jie Yu, He-Dong Li, Hua-Peng Cui, Ping-Ping Liu, Bing Wang, Sheng Wang, and Peng Lu. A simple multi-scale gaussian smoothing-based strategy for automatic chromatographic peak extraction. *Journal of Chromatography A*, 1452:1–9, 2016. (page 47).

[94] Johan Teleman, Andrew W. Dowsey, Faviel F. Gonzalez-Galarza, Simon Perkins, Brian Pratt, Hannes L. Röst, Lars Malmström, Johan Malmström, Andrew R. Jones, Eric W. Deutsch, and Fredrik Levander. Numerical compression schemes for proteomics mass spectrometry data. *Molecular and Cellular Proteomics*, 13(6):1537–1542, 2014. (page 49).

[95] Bernd Klaus and Korbinian Strimmer. Signal identification for rare and weak features: Higher criticism or false discovery rates? *Biostatistics*, 14(1):129–143, 2013. (page 49).

[96] David L. Tabb, Michael J. MacCoss, Christine C. Wu, Scott D. Anderson, and John R. Yates. Similarity among tandem mass spectra from proteomic experiments: Detection, significance, and utility. *Analytical Chemistry*, 75(10):2470–2477, 2003. (pages 50, 54, and 59).

[97] David L Tabb, Melissa R Thompson, Gurusahai Khalsa-Moyers, Nathan C VerBerkmoes, and W Hayes McDonald. Ms2grouper: group assessment and synthetic replacement of duplicate proteomic tandem mass spectra. *Journal of the American Society for Mass Spectrometry*, 16(8):1250–1261, 2005. (pages 50 and 150).

[98] Ilan Beer, Eilon Barnea, Tamar Ziv, and Arie Admon. Improving large-scale proteomics by clustering of mass spectrometry data. *Proteomics*, 4(4):950–960, 2004. (pages 50, 54, 59, 150, and 153).

[99] Kristian Flikka, Jeroen Meukens, Kenny Helsens, Joël Vandekerckhove, Ingvar Eidhammer, Kris Gevaert, and Lennart Martens. Implementation and application of a versatile clustering tool for tandem mass spectrometry data. *Proteomics*, 7(18):3245–3258, 2007. (pages 50, 54, and 59).

[100] Ari M. Frank, Nuno Bandeira, Zhouxin Shen, Stephen Tanner, Steven P. Briggs, Richard D. Smith, and Pavel A. Pevzner. Clustering millions of tandem mass spectra. *Journal of Proteome Research*, 7(1):113–122, 2008. (pages 50 and 53).

[101] Ari M. Frank, Matthew E. Monroe, Anuj R. Shah, Jeremy J. Carver, Nuno Bandeira, Ronald J. Moore, Gordon A. Anderson, Richard D. Smith, and Pavel A. Pevzner. Spectral archives: Extending spectral libraries to analyze both identified and unidentified spectra. *Nature Methods*, 8(7):587–594, 2011. (page 50).

[102] Johannes Griss, Joseph M Foster, Henning Hermjakob, and Juan Antonio Vizcaíno. Pride cluster: building a consensus of proteomics data. *Nature methods*, 10(2):95–96, 2013. (pages 50 and 150).

[103] Johannes Griss, Yasset Perez-Riverol, Steve Lewis, David L. Tabb, José A. Dianes, Noemi Del-Toro, Marc Rurik, Mathias Walzer, Oliver Kohlbacher, Henning Hermjakob, Rui Wang, and Juan Antonio Vizcano. Recognizing millions of consistently unidentified spectra across hundreds of shotgun proteomics datasets. *Nature Methods*, 13(8):651–656, 2016. (page 50).

[104] Jayson A Falkner, Jarret W Falkner, Anastasia K Yocum, and Philip C Andrews. A spectral clustering approach to ms/ms identification of post-translational modifications. *Journal of proteome research*, 7(11):4614–4622, 2008. (pages 50, 54, 59, and 150).

[105] Fahad Saeed, Jason D. Hoffert, and Mark A. Knepper. CAMS-RS: Clustering algorithm for large-scale mass spectrometry data using restricted search space and intelligent random sampling. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 11(1):128–141, 2014. (pages 50, 54, and 59).

[106] Matthew The and Lukas Kll. Maracluster: A fragment rarity metric for clustering fragment spectra in shotgun proteomics. *Journal of proteome research*, 15(3):713–720, 2016. (pages 50, 54, 59, and 150).

[107] Vera Rieder, Karin U Schork, Laura Kerschke, Bernhard Blank-Landeshammer, Albert Sickmann, and Jórg Rahnenführer. Comparison and evaluation of clus-

tering algorithms for tandem mass spectra. *Journal of proteome research*, 16(11):4035–4044, 2017. (pages 50 and 150).

[108] Lei Wang, Sujun Li, and Haixu Tang. MsCRUSH: Fast Tandem Mass Spectral Clustering Using Locality Sensitive Hashing. *Journal of Proteome Research*, 18(1):147–158, 2019. (page 50).

[109] Yasset Perez-Riverol, Juan Antonio Vizcaíno, and Johannes Griss. Future prospects of spectral clustering approaches in proteomics. *Proteomics*, 18(14):1700454, 2018. (pages 50, 149, and 150).

[110] Stephane Houel, Robert Abernathy, Kutralanathan Renganathan, Karen Meyer-Arendt, Natalie G. Ahn, and William M. Old. Quantifying the impact of chimera MS/MS spectra on peptide identification in large-scale proteomics studies. *Journal of Proteome Research*, 9(8):4152–4160, 2010. (page 50).

[111] John D. Chapman, David R. Goodlett, and Christophe D. Masselon. Multiplexed and data-independent tandem mass spectrometry for global proteome profiling. *Mass Spectrometry Reviews*, 33(6):452–470, 2014. (pages 50 and 104).

[112] Alex Hu, Yang Young Lu, Jeff Bilmes, and William Stafford Noble. Joint Precursor Elution Profile Inference via Regression for Peptide Detection in Data-Independent Acquisition Mass Spectra. *Journal of Proteome Research*, 18(1):86–94, 2019. (page 50).

[113] Andreas Bertsch, Clemens Gröpl, Knut Reinert, and Oliver Kohlbacher. OpenMS and TOPP: open source software for LC-MS data analysis. In *Methods in molecular biology (Clifton, N.J.)*, volume 696, pages 353–367. Springer, 2011. (pages 51 and 52).

[114] Matthew Bellew, Marc Coram, Matthew Fitzgibbon, Mark Igra, Tim Randolph, Pei Wang, Damon May, Jimmy Eng, Ruihua Fang, Chen Wei Lin, Jinzhi Chen, David Goodlett, Jeffrey Whiteaker, Amanda Paulovich, and Martin McIntosh. A suite of algorithms for the comprehensive analysis of complex protein mixtures using high-resolution LC-MS. *Bioinformatics*, 22(15):1902–1909, 2006. (pages 51 and 52).

[115] Sugato Basu, Ian Davidson, and Kiri Wagstaff. *Constrained clustering: Advances in algorithms, theory, and applications*. CRC Press, 2008. (page 51).

[116] R. Sibson. SLINK: An optimally efficient algorithm for the single-link cluster method. *The Computer Journal*, 16(1):30–34, 1973. (page 51).

[117] D. Defays. An efficient algorithm for a complete link method. *The Computer Journal*, 20(4):364–366, 1977. (page 51).

[118] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*, volume 96, pages 226–231, 1996. (page 51).

[119] Sokal R C and Michener. A statistical method for evaluating systematic relationships. *Univ Kans Sci Bull*, 38:1409–1438, 1958. (page 51).

[120] Ulrike Von Luxburg, Robert C Williamson, and Isabelle Guyon. Clustering: Science or art? In *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, pages 65–79, 2012. (page 51).

[121] Andreas Adolfsson, Margareta Ackerman, and Naomi C. Brownstein. To cluster, or not to cluster: An analysis of clusterability methods. *Pattern Recognition*, 88:13–26, 2019. (page 52).

[122] Susmita Datta and Somnath Datta. Comparisons and validation of statistical clustering techniques for microarray gene expression data. *Bioinformatics*, 19(4):459–466, 2003. (page 52).

[123] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000. (pages 52, 150, and 151).

[124] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems*, pages 849–856, 2002. (page 52).

[125] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007. (pages 52, 77, and 151).

[126] Yizong Cheng. Mean shift, mode seeking, and clustering. *IEEE transactions on pattern analysis and machine intelligence*, 17(8):790–799, 1995. (page 52).

[127] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002. (page 52).

[128] Erich Schubert and Peter J Rousseeuw. Faster k-medoids clustering: improving the pam, clara, and clarans algorithms. In *International Conference on Similarity Search and Applications*, pages 171–187. Springer, 2019. (page 52).

[129] J Macqueen. Some methods for classification and analysis. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, volume 233, pages 281–297. Oakland, CA, USA, 1967. (page 52).

[130] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982. (pages 52 and 151).

[131] Anil K. Jain. Data clustering: 50 years beyond K-means. *Pattern Recognition Letters*, 31(8):651–666, 2010. (pages 53 and 72).

[132] Christopher K. I Williams. *Learning With Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*, volume 98. MIT press, 2003. (page 53).

[133] Bernhard Schölkopf, Alexander Smola, and Klaus Robert Müller. Nonlinear Component Analysis as a Kernel Eigenvalue Problem. *Neural Computation*, 10(5):1299–1319, 1998. (page 54).

[134] Jessica Henning, Annika Tostengard, and Rob Smith. A Peptide-Level Fully Annotated Data Set for Quantitative Evaluation of Precursor-Aware Mass Spectrometry Data Processing Algorithms. *Journal of Proteome Research*, 18(1):392–398, 2019. (pages 54, 55, and 98).

[135] Matthew C Chambers, Brendan Maclean, Robert Burke, Dario Amodei, Daniel L Ruderman, Steffen Neumann, Laurent Gatto, Bernd Fischer, Brian Pratt, Jarrett Egertson, et al. A cross-platform toolkit for mass spectrometry and proteomics. *Nature biotechnology*, 30(10):918–920, 2012. (page 57).

[136] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. The earth mover's distance as a metric for image retrieval. *International journal of computer vision*, 40(2):99–121, 2000. (page 60).

[137] Nicolas Courty, Rémi Flamary, and Devis Tuia. Domain adaptation with regularized optimal transport. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 274–289. Springer, 2014. (page 60).

[138] Szymon Majewski, Michal Aleksander Ciach, Michal Startek, Wanda Niemyska, Blazej Miasojedow, and Anna Gambin. The wasserstein distance as a dissimilarity measure for mass spectra with application to spectral deconvolution. In *18th International Workshop on Algorithms in Bioinformatics (WABI 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018. (pages 60 and 62).

[139] Bernhard Schölkopf. The kernel trick for distances. In *Advances in Neural Information Processing Systems*, pages 301–307, 2001. (page 60).

[140] Shusen Wang, Alex Gittens, and Michael W Mahoney. Scalable kernel k-means clustering with nyström approximation: relative-error bounds. *The Journal of Machine Learning Research*, 20(1):431–479, 2019. (pages 60, 65, and 75).

[141] J. A. Hartigan and M. A. Wong. Algorithm AS 136: A K-Means Clustering Algorithm. *Applied Statistics*, 28(1):100, 1979. (page 61).

[142] Nicolas Keriven, Nicolas Tremblay, Yann Traonmilin, and Rémi Gribonval. Compressive K-means. In *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, pages 6369–6373. Institute of Electrical and Electronics Engineers Inc., jun 2017. (pages 61, 67, and 68).

[143] Clark R. Givens and Rae Michael Shortt. A class of Wasserstein metrics for probability distributions. *The Michigan Mathematical Journal*, 31(2):231–240, 1984. (page 62).

[144] Alison L. Gibbs and Francis Edward Su. On choosing and bounding probability metrics. *International Statistical Review*, 70(3):419–435, 2002. (page 62).

[145] Thomas Hofmann, Bernhard Schölkopf, and Alexander J. Smola. Kernel methods in machine learning. *Annals of Statistics*, 36(3):1171–1220, 2008. (page 63).

[146] Alain Berlinet and Christine Thomas-Agnan. *Reproducing Kernel Hilbert Spaces in Probability and Statistics*. Springer Science & Business Media, 2004. (page 64).

[147] Aasa Feragen, François Lauze, and Soren Hauberg. Geodesic exponential kernels: When curvature and linearity conflict. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3032–3042, 2015. (pages 64 and 131).

[148] Daniele Calandriello and Lorenzo Rosasco. Statistical and computational trade-offs in kernel K-means. In *Advances in Neural Information Processing Systems*, volume 2018-Decem, pages 9357–9367, 2018. (pages 65 and 75).

[149] S. E. Puckette and Walter Rudin. *Fourier Analysis on Groups.*, volume 72. Wiley Online Library, 1965. (page 66).

[150] Pablo Arias, Gregory Randall, and Guillermo Sapiro. Connecting the out-of-sample and pre-image problems in Kernel methods. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007. (page 68).

[151] Sebastian Mika, Bernhard Schölkopf, Alex Smola, Klaus Robert Müller, Matthias Scholz, and Gunnar Rätsch. Kernel PCA and de-noising in feature spaces. In *Advances in Neural Information Processing Systems*, pages 536–542, 1999. (page 69).

[152] Florian Prive, Hugues Aschard, Andrey Ziyatdinov, and Michael G.B. Blum. Efficient analysis of large-scale genome-wide data with two R packages: Bigstatsr and bigsnpr. *Bioinformatics*, 34(16):2781–2787, aug 2018. (page 97).

[153] Olga Permiakova and Thomas Burger. Gitlab of CHICKN (Chromatogram HIerarchical Compressive K-means with Nystrom approximation) R package, 2020. (page 97).

[154] Olga Permiakova and Thomas Burger. CRAN repository of CHICKN (Chromatogram HIerarchical Compressive K-means with Nystrom approximation) R package, 2020. (page 97).

[155] Christian Jutten and Jeanny Herault. Blind separation of sources, part i: An adaptive algorithm based on neuromimetic architecture. *Signal processing*, 24(1):1–10, 1991. (page 102).

[156] D. L. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4):1289–1306, 2006. (page 102).

[157] Michael Elad. *Sparse and redundant representations: from theory to applications in signal and image processing.* Springer Science & Business Media, 2010. (page 102).

[158] Shutao Li, Haitao Yin, and Leyuan Fang. Group-sparse representation with dictionary learning for medical image denoising and fusion. *IEEE Transactions on biomedical engineering*, 59(12):3450–3459, 2012. (page 102).

[159] Jianchao Yang, John Wright, Thomas S Huang, and Yi Ma. Image super-resolution via sparse representation. *IEEE transactions on image processing*, 19(11):2861–2873, 2010. (page 102).

[160] Michael Lustig, David Donoho, and John M Pauly. Sparse mri: The application of compressed sensing for rapid mr imaging. *Magnetic Resonance in Medicine: An Official Journal of the International Society for Magnetic Resonance in Medicine*, 58(6):1182–1195, 2007. (page 102).

[161] S. G. Mallat. A theory for multiresolution signal decomposition: the wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7):674–693, 1989. (page 102).

[162] David L Donoho, Michael Elad, and Vladimir N Temlyakov. Stable recovery of sparse overcomplete representations in the presence of noise. *IEEE Transactions on information theory*, 52(1):6–18, 2005. (page 103).

[163] Bruno A Olshausen and David J Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609, 1996. (page 103).

[164] Jinglei Lv, Xi Jiang, Xiang Li, Dajiang Zhu, Hanbo Chen, Tuo Zhang, Shu Zhang, Xintao Hu, Junwei Han, Heng Huang, et al. Sparse representation of whole-brain fmri signals for identification of functional networks. *Medical image analysis*, 20(1):112–134, 2015. (page 103).

[165] Patrik O Hoyer. Non-negative matrix factorization with sparseness constraints. *Journal of machine learning research*, 5(Nov):1457–1469, 2004. (page 103).

[166] M. Elad and M. Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Transactions on Image Processing*, 15(12):3736–3745, 2006. (page 103).

[167] Mark D Plumbley, Thomas Blumensath, Laurent Daudet, Rémi Gribonval, and Mike E Davies. Sparse representations in audio and music: from coding to source separation. *Proceedings of the IEEE*, 98(6):995–1005, 2009. (page 103).

[168] Thomas E Angel, Uma K Aryal, Shawna M Hengel, Erin S Baker, Ryan T Kelly, Errol W Robinson, and Richard D Smith. Mass spectrometry-based proteomics: existing capabilities and future directions. *Chemical Society Reviews*, 41(10):3912–3928, 2012. (page 103).

[169] Jérémy Rapin, Antoine Souloumiac, Jérôme Bobin, Anthony Larue, Chistophe Junot, Minale Ouethrani, and Jean-Luc Starck. Application of non-negative matrix factorization to lc/ms data. *Signal Processing*, 123:75–83, 2016. (page 104).

[170] Olga Permiakova, Romain Guibert, Alexandra Kraut, Thomas Fortin, Anne-Marie Hesse, and Thomas Burger. Chickn: Extraction of peptide chromatographic elution profiles from large scale mass spectrometry data by means of wasserstein compressive hierarchical cluster analysis. *BMC Bioinformatics (under revision)*, 2020. (pages 104, 113, and 114).

[171] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996. (page 106).

[172] David L Donoho and Michael Elad. Optimally sparse representation in general (nonorthogonal) dictionaries via l1 minimization. *Proceedings of the National Academy of Sciences*, 100(5):2197–2202, 2003. (page 106).

[173] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1):183–202, 2009. (page 106).

[174] Kjersti Engan, Sven Ole Aase, and J Hakon Husoy. Method of optimal directions for frame design. In *1999 IEEE International Conference on Acoustics, Speech,*

and Signal Processing. Proceedings. ICASSP99 (Cat. No. 99CH36258), volume 5, pages 2443–2446. IEEE, 1999. (page 106).

[175] Dar Gilboa, Sam Buchanan, and John Wright. Efficient dictionary learning with gradient descent. In *International Conference on Machine Learning*, pages 2252–2259, 2019. (page 106).

[176] Bao-Di Liu, Yu-Xiong Wang, Bin Shen, Xue Li, Yu-Jin Zhang, and Yan-Jiang Wang. Blockwise coordinate descent schemes for efficient and effective dictionary learning. *Neurocomputing*, 178:25–35, 2016. (page 106).

[177] Konstantinos Slavakis and Georgios B Giannakis. Online dictionary learning from big data using accelerated stochastic approximation algorithms. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 16–20. IEEE, 2014. (page 106).

[178] Arthur Mensch, Julien Mairal, Bertrand Thirion, and Gaël Varoquaux. Stochastic subsampling for factorizing huge matrices. *IEEE Transactions on Signal Processing*, 66(1):113–128, 2017. (pages 107 and 118).

[179] Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online learning for matrix factorization and sparse coding. *Journal of Machine Learning Research*, 11(Jan):19–60, 2010. (page 107).

[180] Karin Schnass and Flavio Teixeira. Compressed dictionary learning. *Journal of Fourier Analysis and Applications*, 26(2):1–37, 2020. (pages 107 and 118).

[181] Ella Bingham and Heikki Mannila. Random projection in dimensionality reduction: applications to image and text data. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 245–250, 2001. (page 107).

[182] Thomas Blumensath and Mike E Davies. Iterative thresholding for sparse approximations. *Journal of Fourier analysis and Applications*, 14(5-6):629–654, 2008. (page 108).

[183] Nir Ailon and Bernard Chazelle. The fast johnson–lindenstrauss transform and approximate nearest neighbors. *SIAM Journal on computing*, 39(1):302–322, 2009. (page 108).

[184] Mahmoud Assran and Michael Rabbat. On the convergence of nesterov's accelerated gradient method in stochastic settings. *arXiv preprint arXiv:2002.12414*, 2020. (page 109).

[185] Andrei Kulunchakov and Julien Mairal. A generic acceleration framework for stochastic composite optimization. In *Advances in Neural Information Processing Systems*, pages 12556–12567, 2019. (page 109).

[186] Dirk Eddelbuettel. *Seamless R and C++ Integration with Rcpp*. Springer, New York, 2013. ISBN 978-1-4614-6867-7. (page 111).

[187] JJ Allaire, R Francois, K Ushey, G Vandenbrouck, and M Geelnard. Rcppparallel: Parallel programming tools for "rcpp". *R package version*, 4:20, 2016. (page 111).

[188] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22, 2010. (page 112).

[189] Markus Schmidberger, Martin Morgan, Dirk Eddelbuettel, Hao Yu, Luke Tierney, and Ulrich Mansmann. State-of-the-art in parallel computing with r. *Journal of Statistical Software*, 47(1), 2009. (page 112).

[190] Florian Privé, Hugues Aschard, Andrey Ziyatdinov, and Michael G.B. Blum. Efficient analysis of large-scale genome-wide data with two R packages: bigstatsr and bigsnpr. *Bioinformatics*, 34(16):2781–2787, 2018. (page 112).

[191] Salvatore Cappadona, Fredrik Levander, Maria Jansson, Peter James, Sergio Cerutti, and Linda Pattini. Wavelet-based method for noise characterization and rejection in high-performance liquid chromatography coupled to mass spectrometry. *Analytical Chemistry*, 80(13):4960–4968, 2008. (page 131).

[192] CR Mittermayr, SG Nikolov, H Hutter, and M Grasserbauer. Wavelet denoising of gaussian peaks: a comparative study. *Chemometrics and Intelligent Laboratory Systems*, 34(2):187–202, 1996. (page 131).

[193] Xueguang Shao, Wensheng Cai, and Zhongxiao Pan. Wavelet transform and its applications in high performance liquid chromatography (hplc) analysis. *Chemometrics and intelligent laboratory systems*, 45(1-2):249–256, 1999. (page 131).

[194] Cameron Musco and Christopher Musco. Recursive sampling for the nystrom method. In *Advances in Neural Information Processing Systems*, pages 3833–3845, 2017. (page 131).

[195] Henri De Plaen, Michaël Fanuel, and Johan AK Suykens. Wasserstein exponential kernels. *arXiv preprint arXiv:2002.01878*, 2020. (page 131).

[196] Sadeep Jayasumana, Richard Hartley, Mathieu Salzmann, Hongdong Li, and Mehrtash Harandi. Kernel methods on the riemannian manifold of symmetric positive definite matrices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 73–80, 2013. (page 131).

[197] Brendan O'donoghue and Emmanuel Candes. Adaptive restart for accelerated gradient schemes. *Foundations of computational mathematics*, 15(3):715–732, 2015. (page 134).

[198] Mihai I Florea and Sergiy A Vorobyov. A robust fista-like algorithm. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4521–4525. IEEE, 2017. (page 134).

[199] Roger A Horn and Charles R Johnson. *Matrix analysis*. Cambridge university press, 2012. (page 142).

[200] Marc G Genton. Classes of kernels for machine learning: a statistics perspective. *Journal of machine learning research*, 2(Dec):299–312, 2001. (page 142).

[201] Christian Berg, Jens Peter Reus Christensen, and Paul Ressel. *Harmonic analysis on semigroups: theory of positive definite and related functions*, volume 100. Springer, 1984. (pages 142 and 143).

[202] Ari M Frank, Nuno Bandeira, Zhouxin Shen, Stephen Tanner, Steven P Briggs, Richard D Smith, and Pavel A Pevzner. Clustering millions of tandem mass spectra. *Journal of proteome research*, 7(01):113–122, 2008. (page 150).

[203] Johannes Griss, Yasset Perez-Riverol, Matthew The, Lukas Käll, and Juan Antonio Vizcaíno. Response to "comparison and evaluation of clustering algorithms for tandem mass spectra". *Journal of proteome research*, 17(5):1993–1996, 2018. (page 150).

[204] Ravi Kannan, Santosh Vempala, and Adrian Vetta. On clusterings: Good, bad and spectral. *Journal of the ACM (JACM)*, 51(3):497–515, 2004. (page 150).

[205] Alexandros Karatzoglou, Alex Smola, Kurt Hornik, and Achim Zeileis. kernlab-an s4 package for kernel methods in r. *Journal of statistical software*, 11(9):1–20, 2004. (pages 152 and 159).

[206] Jure Leskovec, Kevin J Lang, Anirban Dasgupta, and Michael W Mahoney. Statistical properties of community structure in large social and information networks. In *Proceedings of the 17th international conference on World Wide Web*, pages 695–704, 2008. (page 152).

[207] Marcilio CP de Souto, Ivan G Costa, Daniel SA de Araujo, Teresa B Ludermir, and Alexander Schliep. Clustering cancer gene expression data: a comparative study. *BMC bioinformatics*, 9(1):497, 2008. (page 152).

[208] Johanna K Thurlow, Claudia L Peña Murillo, Keith D Hunter, Francesca M Buffa, Shalini Patiar, Guy Betts, CM West, Adrian L Harris, Eric K Parkinson,

Paul R Harrison, et al. Spectral clustering of microarray data elucidates the roles of microenvironment remodeling and immune responses in survival of head and neck squamous cell carcinoma. *J Clin Oncol*, 28(17):2881–8, 2010. (page 152).

[209] Guimin Qin and Lin Gao. Spectral clustering for detecting protein complexes in protein–protein interaction (ppi) networks. *Mathematical and Computer Modelling*, 52(11-12):2066–2074, 2010. (page 152).

[210] Ariel Bensimon, Albert JR Heck, and Ruedi Aebersold. Mass spectrometry–based proteomics and network biology. *Annual review of biochemistry*, 81:379–405, 2012. (page 152).

[211] Thomas Burger. Gentle introduction to the statistical foundations of false discovery rate in quantitative proteomics. *Journal of proteome research*, 17(1):12–22, 2018. (page 153).

[212] Lukas Käll, John D Storey, Michael J MacCoss, and William Stafford Noble. Assigning significance to peptides identified by tandem mass spectrometry using decoy databases. *Journal of proteome research*, 7(01):29–34, 2008. (page 153).

[213] Oliver Serang and Lukas Käll. Solution to statistical challenges in proteomics is more statistics, not less. *Journal of proteome research*, 14(10):4099–4103, 2015. (page 153).