Unsupervised statistical learning with latent block models on dynamic discrete data

Giulia Marchello

Apprentissage statistique non supervisé avec des modèles à blocs latents sur des données discrètes dynamiques

Giulia MARCHELLO

Centre INRIA d'Université Côte d'Azur,
équipe MAASAI

Présentée en vue de l'obtention du grade de docteur en Mathématiques
d'Université Côte d'Azur
Dirigée par : Charles BOUVEYRON
Co-encadrée par : Marco CORNELI
Soutenue le : 05/12/2023

Devant le jury, composé de :
Christine KERIBIN, Maîtresse de conférences, Université Paris-Saclay
Julien CHIQUET, Directeur de recherche, Université Paris-Saclay, AgroParisTech, INRAE
Julie JOSSE, Directrice de recherche, INRIA Montpellier
Luca SCRUCCA, Associate Professor, Università degli Studi di Perugia
Marco LORENZI, Chargé de recherche, Université Côte d'Azur, Inria
Charlotte LACLAU, Associate Professor, Télécom Paris

Unsupervised statistical learning with latent block models on dynamic discrete data

Jury:

Rapporteurs:
Christine KERIBIN, Maîtresse de conférences, Université Paris-Saclay
Julien CHIQUET, Directeur de recherche, Université Paris-Saclay, AgroParisTech, INRAE

Examinateurs:
Julie JOSSE, Directrice de recherche, INRIA Montpellier
Luca SCRUCCA, Associate Professor, Università degli Studi di Perugia
Marco LORENZI, Chargé de recherche, Université Côte d'Azur, Inria
Charlotte LACLAU, Associate Professor, Télécom Paris

Directeur de thèse:
Charles BOUVEYRON, Professeur, Université Côte d'Azur & Inria

Co-encadrant de thèse:
Marco CORNELI, Associate Professor, Université Côte d'Azur

Résumé

Dans un monde de plus en plus connecté, nous générons dans nos vies quotidiennes une quantité croissante d'informations de natures diverses, ce qui nécessite des méthodes automatiques et adaptées pour la détection, la synthèse et la compréhension des phénomènes qu'elles décrivent.

Au fil des années, l'apprentissage statistique a continuellement évolué, offrant diverses techniques pour relever les défis posés par cette énorme quantité de données. En particulier, l'apprentissage non supervisé joue un rôle essentiel dans la découverte de motifs dans les données sans étiquettes préalablement définies. Une méthode efficace pour résumer et visualiser des données de grande dimension est le clustering, qui rassemble des observations similaires en fonction de caractéristiques communes. En étendant le clustering pour grouper simultanément les observations et les caractéristiques, approche connue sous le nom de co-clustering, il est possible de mettre en évidence des relations complexes entre observations ou variables.

Cette thèse s'intéresse au co-clustering de données discrètes dépendantes du temps, une tâche essentielle pour résumer les énormes quantités de données générées dans divers domaines. Nous proposons ici trois modèles conçus pour relever ce défis. Tout d'abord, nous présentons le modèle dynamique à blocs latents (dLBM), une approche d'analyse rétrospective qui détecte les ruptures temporelles au sein d'une séquence de matrices de données. En s'appuyant sur des processus de Poisson non homogènes pour modéliser les interactions entre les lignes et les colonnes, dLBM segmente des intervalles de temps continus, générant une segmentation en lignes et colonnes pour chaque intervalle de temps.

En nous appuyant sur dLBM, nous présentons ensuite une extension qui offre une plus grande une flexibilité dans la manière dont les groupes en ligne et en colonne évoluent dans le temps et qui tient compte de la grande parcimonie des données. Ce modèle étendu intègre des systèmes d'équations différentielles ordinaires pour modéliser les changements abrupts à la fois dans l'appartenance aux groupes et modéliser la parcimonie des données au cours du temps.

Enfin, nous proposons une évolution naturelle des modèles proposés, les rendant aptes à fonctionner de manière dynamique sur flux de données. Cette nouvelle méthode d'inférence du modèle ne requiert pas la connaissance a priori de l'ensemble des données, rendant l'analyse en temps réel réalisable. En exploitant les réseaux neuronaux LSTM (Long Short-Term Memory), nous modélisons dynamiquement l'évolution des paramètres du modèle. Nous introduisons également une méthode de détection de points de ruptures en ligne qui fournit des alertes pour les changements brusques dans l'évolution des données.

Les trois méthodes proposées ont été appliquées à la pharmacovigilance, un discipline de santé publique dédiée à la surveillance et à l'évaluation de la sécurité des produits médicaux. En pharmacovigilance, il est essentiel de détecter en temps réel les changements dans les appartenances aux groupes de médicaments et des effets indésirables afin d'assurer une identification et une réponse rapides en matière de santé publique.

Mots clés: Co-clustering, modèle à blocs latents, distributions avec excès de zéros, inférence variationnelle, données dépendant du temps.

Abstract

In such a connected world, we generate an increasing quantity of information of various kinds in our daily lives, which requires automatic and adapted methods for the detection, synthesis and understanding of the phenomena they describe.

Over the years, statistical learning has continually evolved, offering various techniques to address the challenges posed by this enormous amount of data. In particular, unsupervised learning plays a critical role in discovering patterns in data without previously defined labels. An effective method for summarizing and visualizing high-dimensional data is clustering, which brings together similar observations based on common characteristics. By extending clustering to simultaneously group observations and features, an approach known as co-clustering, it is possible to highlight complex relationships between observations or variables.

This thesis focuses on the co-clustering of discrete time-dependent data, an essential task for summarizing the enormous amounts of data generated in various domains. Here we offer three models designed to meet this challenge. First, we present the dynamic latent block model (dLBM), a retrospective analysis approach that detects temporal breaks within a sequence of data matrices. Relying on non-homogeneous Poisson processes to model interactions between rows and columns, dLBM segments continuous time intervals, generating a row and column segmentation for each time interval.

Building on dLBM, we then present an extension that provides greater flexibility in how row and column groups evolve over time and accounts for high data sparsity. This extended model incorporates systems of ordinary differential equations to model abrupt changes in both group membership and model data sparsity over time.

Finally, we propose a natural evolution of the proposed models, making them able to operate dynamically on data flow. This new method of model inference does not require a priori knowledge of all the data, making real-time analysis feasible. By exploiting LSTM (Long Short-Term Memory) neural networks, we dynamically model the evolution of model parameters. We also introduce an online breakpoint detection method that provides alerts for abrupt changes in data evolution.

The three proposed methods were applied to pharmacovigilance, a public health discipline dedicated to monitoring and evaluating the safety of medical

products. In pharmacovigilance, it is essential to detect changes in drug group memberships and adverse effects in real time to ensure rapid identification and response in public health matters.

Keywords: Co-clustering, latent block model, zero-inflated distributions, variational inference, time-dependent data.

Acknowledgment

In this section, I would like to express my gratitude to everyone has stood by, accompanied, and supported me throughout my doctoral journey.
I would like to express my heartfelt gratitude to my two PhD supervisors, Charles and Marco, who welcomed me into the team when I was an Erasmus student and introduced me to the beauty of research. My infinite thanks for believing and guiding me throughout this journey, instilling confidence and assurance at every step. Your invaluable contribution has enriched me personally and professionally over the years. You both are a great source of inspiration for me, and I hope that our collaborations will continue over time.

My gratitude extends to the remarkable colleagues of the Maasai research team, who have turned this journey into something truly exceptional. Thank you for the countless coffee breaks and the Friday aperitifs that have brought us together. Your openness to share and care from the beginning has created a supportive and enriching environment that I am deeply grateful for.

A special thanks to my friends of the Valrose PhD group. You all are like a second family to me, the shared experiences we have had together have created a sense of belonging that goes beyond the academic sphere. Thank you for the everlasting memories collected in these years and for the unwavering support, providing a true sense of belonging. Your encouragement and cheering me up during challenging times have meant the world to me.

Infine, vorrei ringraziare di cuore la mia famiglia per la fiducia incondizionata che ha riposto in me. Papà e mamma, grazie di cuore per essere sempre stati il mio incoraggiamento costante, spingendomi a dare sempre il massimo e facendo sacrifici incredibili per il mio percorso. La vostra presenza nel giorno della mia discussione, nonostante la distanza di oltre mille chilometri, ha reso quel momento ancora più speciale. Spero di avervi resi orgogliosi di me.
Desidero dedicare un ultimo riconoscimento speciale a mio fratello, un'immensa fonte di ispirazione e una presenza straordinaria che ha significato molto per me lungo tutto il percorso. Grazie perché con la tua costante positività, supporto e motivazione, hai reso ogni sfida più gestibile e ogni successo più significativo. La tua influenza ha lasciato un'impronta indelebile nel mio percorso, e sono grata di poter contare su di te come pilastro fondamentale nella mia vita.

Contents

List of Symbols

Variables

X	Observed data
N	Number of rows of the tensor X
M	Number of columns of the tensor X
Q	Row clusters
L	Column clusters
C	Time clusters
T	Number of time intervals (i.e. 3rd dimension of the tensor X)
W	Latent variable indicating the cluster of columns
Z	Latent variable indicating the cluster of rows
S	Latent variable indicating the cluster of time clusters

Distributions

$\mathcal{P}(\cdot)$	Poisson distribution
$\mathcal{M}(\cdot)$	Multinomial distribution
$\Gamma(\cdot)$	Gamma distribution
$\mathcal{B}(\cdot)$	Bernoulli distribution
$\mathcal{N}(\cdot)$	Normal distribution

Chapter 1

Introduction

The growing complexity and abundance of data have sparked the need for advanced techniques to both uncover meaningful patterns and structures and summarize the available information, for instance via unsupervised machine learning. Throughout this section, we will delve into the challenges and methodologies involved in the clustering of high-dimensional data, a particular kind of data whose feature dimension is of the same order of magnitude or even exceeds the number of observations. In this context, traditional approaches may struggle to produce meaningful results. In particular, we will introduce a powerful extension of traditional clustering approaches known as co-clustering. This technique proves particularly valuable making high-dimensional data more interpretable, as it has the unique ability to simultaneously group both rows and columns of a data matrix. Furthermore, this section explores time-dependent data, focusing on the challenge of identifying patterns that evolve over time. The ability to effectively cluster and analyze time-dependent data holds great promise in numerous applications, from e-commerce companies to healthcare and beyond.

1.1 Context

In the contemporary landscape of data, we are surrounded by a lot of information that keeps growing. This big amount of data comes from many different sources, and we need autonomous ways to detect important patterns, summarize and visualize them.
To illustrate this idea, we can examine some real data examples. First, we can think about online stores, like Amazon. Every second, lots of people visit the website, look at products, and buy things. Every time they do something on the site, new data are created. These data include what people like, what they click on, what they buy, and when they buy it. The challenge here is figuring out how to make sense of all this data to make better decisions for businesses, based on the customers preferences. For example, based on how people shop and what is popular, businesses can orient their selling policies and propose ad-hoc products to the users.

As another example, consider the bike-sharing systems you might see in cities. These systems let people rent bicycles using a phone app. Every time someone takes a bike for a ride, data are generated. The data tell us where the ride starts, where it ends, and how long it lasts. This information arrives quickly and it is incredibly helpful for city planners who want to make transportation more efficient for everyone.

Another example can be found in healthcare, particularly in the field of pharmacovigilance. This central medical discipline is dedicated to ensuring the safety of medical products by monitoring adverse effects both before and after they enter the market. This process involves the detection, assessment, understanding, and prevention of adverse effects or other drug-related problems. Pharmacovigilance centers receive numerous reports of adverse drug reactions daily, originating from diverse sources such as their websites, family doctors, and patients themselves. Within this vast amount of data, there might be subtle patterns or signals that indicate a medication is causing unexpected adverse reactions. These signals could be early warnings of potential health risks associated with a particular drug. Early detection is crucial because it allows healthcare authorities and pharmaceutical companies to take prompt actions. If a medical product is found to have unforeseen risks, it can be withdrawn from the market or its usage can be better controlled, protecting the health and safety of patients. However, expert-driven identification is resource-intensive, time consuming, and susceptible to human error.

1.2 The challenges of high-dimensional data

The three examples we have previously explored, e-commerce platforms, bike-sharing systems, and pharmacovigilance, are illustrative of a fundamental concept: high-dimensional data. In statistics, high-dimensional data is a term used to describe datasets where the number of variables or features (columns) is of the same order of magnitude or even exceeds the number of observations (rows). This phenomenon has become increasingly prevalent in our data-driven age, and it presents unique challenges and opportunities for statistical analysis. Also, high-dimensional data naturally exhibit sparsity, characterized by the presence of numerous empty cells in the dataset. This phenomenon also poses significant challenges for data analysis and it can be seen as one facet of the the curse of dimensionality. This concept was initially introduced by Bellman (1957) within the field of dynamic programming and it continues to be a challenge in nowadays statistical methods. The curse of dimensionality refers to the exponential increase in data volume as the number of dimensions grows. In high-dimensional spaces, data points become increasingly distant from one another, that may lead to a loss of information. Traditional clustering methods are not naturally suited to address the complexities of sparse high-dimensional data. Distinguishing meaningful patterns from random noise becomes especially challenging in this context, leading to a deterioration in the performance of clustering models. This issue arises because in high-dimensional spaces, the concept of distance loses its conventional meaning, which can result in suboptimal outcomes (Steinbach et al., 2004; Giraud, 2021). Moreover, working with high-dimensional data can lead to computational challenges, as analyzing and processing such data demands more significant computational resources and time. In general, extracting meaningful insights and identifying patterns becomes more complex. We can consider our earlier examples to have a better understanding of this concept.

- **Amazon**: In the context of Amazon e-commerce platform, each data point may represent a user, and the attributes could include various aspects of users behavior (e.g., products viewed, products purchased, time spent on the platform, etc.). With millions of users and an extensive array of products, the dataset can quickly become high-dimensional. An example can be seen in Figure 1.1a representing the raw data of Amazon[1], where each colored

[1]The dataset used in this example can be found at https://snap.stanford.edu/data/web-FineFoods.html.

point represents an interaction between users (on the y-axis) and products (on the x-axis).

- **Bike sharing systems**: Here, every bike ride represents a data point, and attributes could includes information such as the starting station, ending station, duration, and time of day. In a bustling city like London with numerous stations and extensive riders, this dataset can easily become high-dimensional. An example can be seen in Figure 1.1b, illustrating interactions between departure stations (on the y-axis) and arrival stations (on the x-axis) of the London bike sharing system[2].

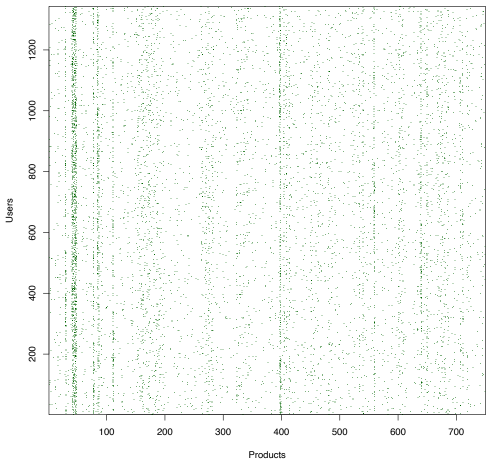
- **Pharmacovigilance**: With a multitude of medications and a wide range of possible adverse effects, this dataset also qualifies as high-dimensional. Figure 1.1c presents a pharmacovigilance dataset, where the y-axis represents medical products, the x-axis denotes adverse effects and each data point corresponds to a reported adverse drug reaction. The dataset used in this example was obtained thanks to a fruitful collaboration with the Regional Pharmacovigilance Center located in Nice (France) and will be analyzed further in the next chapters.

In all of these situations, there is a common need: we need to understand and summarize a massive amount of data. The data come in quickly, and we need advanced statistical methods to help us summarizing them, finding important patterns and make better decisions.

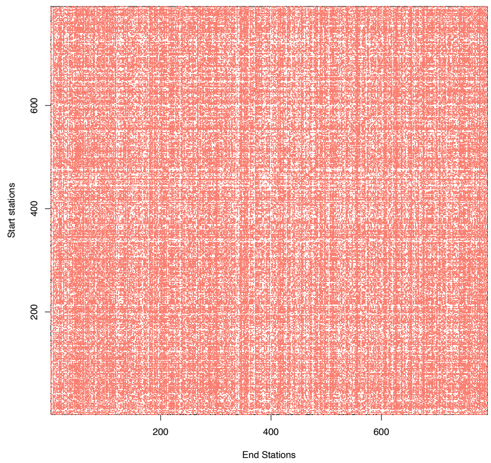
1.3 The role of unsupervised learning

Over the decades, statistical learning continuously evolved, presenting different methods to address the challenges posed by high-dimensional data for understanding and extracting insights from complex datasets. Among the techniques that have attracted significant attention are those falling under the umbrella of unsupervised learning. Unsupervised learning is a branch of machine learning, whose primary goal is to identify underlying patterns, structures, or relationships within data, without the guidance of predefined outcomes or labels. Typically, in this scenario, one common approach for tackling the challenges posed by high-dimensional data is by simplifying and condensing complex datasets while

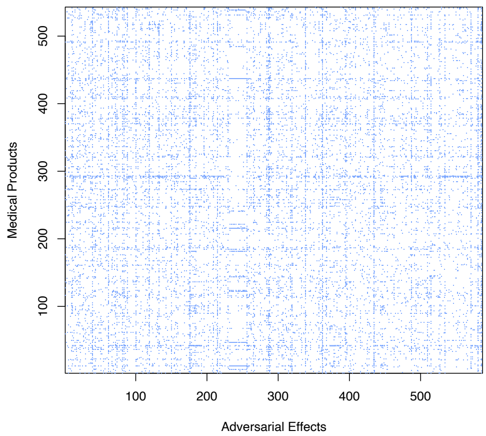
[2]The dataset used in this example can be found at https://cycling.data.tfl.gov.uk and it will be analyzed further in the next chapters.



(a) Raw Amazon data.



(b) Raw London bike sharing data.



(c) Raw pharmacovigilance data.

Figure 1.1: Examples of raw high-dimensional data.

retaining the most essential information.
One effective way to achieve this task is through clustering, a technique that involves grouping similar observations based on shared characteristics. Clustering allows for a summarization and interpretation of the data because it helps identifying natural groupings or patterns within the dataset.
When we extend the concept of clustering to simultaneously group both the observations (rows) and features (columns) in a dataset, this approach is known as co-clustering. Co-clustering is particularly valuable in situations where we aim to uncover complex relationships between rows and columns, allowing for a more comprehensive understanding of the dataset. Indeed, it helps revealing how certain observations relate to specific variables and vice versa.

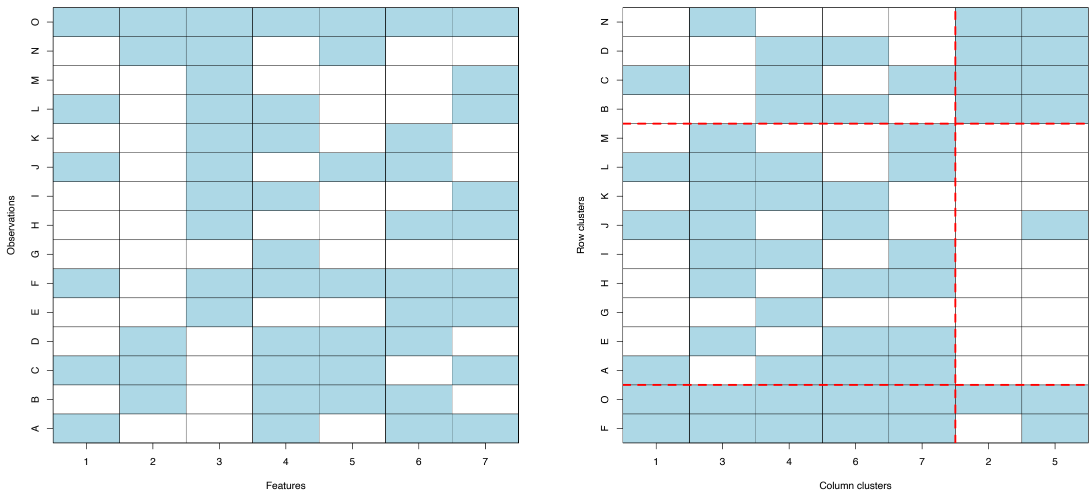
1.4 Analysis of discrete data

The three datasets in Figure 1.1 are examples of discrete data matrices. Discrete data is characterized by values that are distinct, countable, and typically restricted to integers or a finite set of specific values. To illustrate this concept, consider binary data as a straightforward example, consisting of only two possible values encoded as 0 or 1. Also, another example are count data, which involve integers representing the frequency of events occurring within a defined interval. Count data will be extensively used in the following chapters.
Given that this thesis primarily focuses on co-clustering of discrete data, we can now introduce the incidence matrix as a foundational tool for this task. The incidence matrix serves as a concise representation of the relationships between observations and features. In the case of binary data, the incidence matrix contains 0s and 1s, where 1 indicates the presence of a specific feature for a given observation.
For example, Table 1.1 presents an incidence matrix of simulated binary data, where rows and columns represent different populations (letters and numbers). To enhance clarity, Figure 1.2a visually represents the same incidence matrix, using colored squares to denote 1s and white squares to denote 0s. Furthermore, Figure 1.2b showcases the results of applying co-clustering to this data. In this figure, we observe a reorganized incidence matrix, where rows and columns have been rearranged to group nearby rows and columns into clusters. These clusters are delimited by red dashed lines, illustrating the outcome of the co-clustering process. Also, the incidence matrix is a common tool in graph theory and network

	1	2	3	4	5	6	7
A	1	0	0	1	0	1	1
B	0	1	0	1	1	1	0
C	1	1	0	1	1	0	1
D	0	1	0	1	1	1	0
E	0	0	1	0	0	1	1
F	1	0	1	1	1	1	1
G	0	0	0	1	0	0	0
H	0	0	1	0	0	1	1
I	0	0	1	1	0	0	1
J	1	0	1	0	1	1	0
K	0	0	1	1	0	1	0
L	1	0	1	1	0	0	1
M	0	0	1	0	0	0	1
N	0	1	1	0	1	0	0
O	1	1	1	1	1	1	1

Table 1.1: Incidence matrix of simulated binary data.

analysis, as it can be seen as a representation of a bipartite network. A bipartite network is a type of network where the nodes can be divided into two distinct sets, and edges only connect nodes from one set to nodes from the other set. Hence, there are no connections within the same set of nodes. Considering the example depicted in Figure 1.2, we can effectively illustrate the relationships between the two distinct sets (letters and numbers) through a bipartite graph, as represented in Figure 1.3. Here the left side corresponds to the set of numbers, while the right side represents the set of letters. The connecting edges between them depict the connections between letters and numbers, as defined by the entries within the incidence matrix. To illustrate this further, consider the edge connecting the letter "M" to both numbers 3 and 7. This connection is established because the incidence matrix exhibits a value of 1 in the row corresponding to "M" and in the columns corresponding to 3 and 7, thus indicating these specific associations. The positions of letters and numbers in the bipartite network recalls the same clusters we had in Figure 1.2b.



(a) Incidence matrix (b) Reorganized incidence matrix

Figure 1.2: Example of binary data co-clustering.

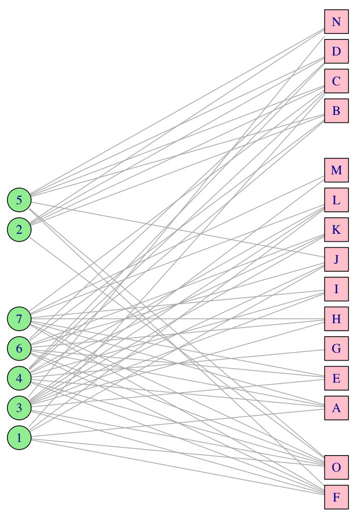
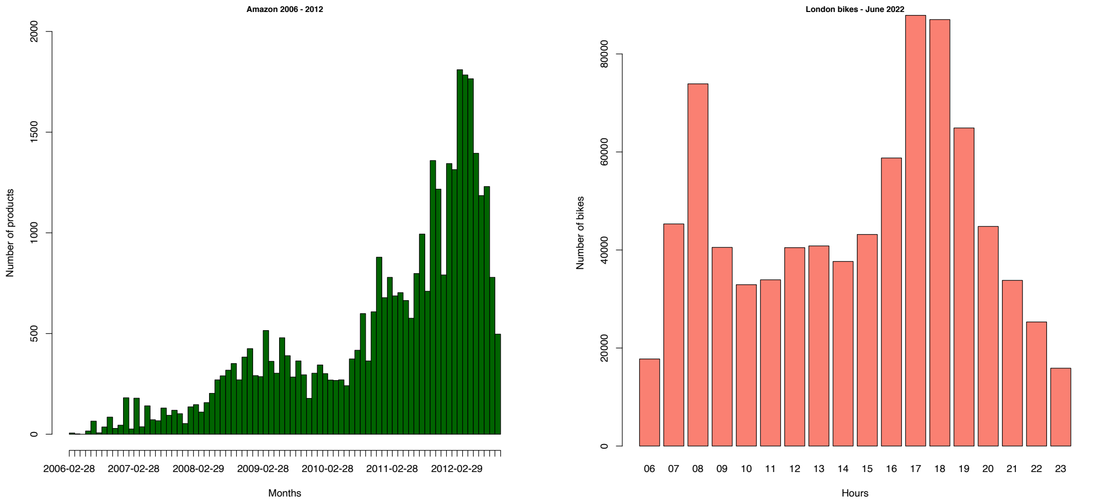


Figure 1.3: Bipartite network on binary simulated data.



(a) Number of purchases in the Amazon dataset, from 2006 to 2012, sorted by month.

(b) Number of bikes taken from the London sharing system corresponding to June 2022, by hour.

Figure 1.4: Histograms of time dependent data.

1.5 Time dependent data

The exponential growth of technology and the increasingly frequent use of various web platforms permeate every aspect of our daily lives, continuously generating a massive stream of data. This gives rise to the need of extracting information dynamically and automatically. Such analyses are not only crucial for informed decision-making but also offer a deep understanding of the underlying processes governing various real-world phenomena. As a result, the ability to summarize vast amounts of time-dependent data has become of paramount significance.

To illustrate this further, we can recall the examples we explored earlier regarding e-commerce platforms, bike-sharing systems, and pharmacovigilance. For instance, Figure 1.4a provides a representation of the frequency of purchases on Amazon between 2006 and 2012, categorized by months, revealing clear patterns over time. Additionally, Figure 1.4b represents the number of bikes used in the London sharing system at every hour, in a cumulative day, over the whole month of June 2022. These histograms demonstrate the existence of patterns that are time-dependent, highlighting the critical role of time-sensitive analysis in extracting meaningful insights. Regarding pharmacovigilance data, we can refer to Figure 1.5. This figure represents the data structure of a large-scale dataset made of the notifications of adverse drug reactions (ADRs) gathered between 2015 to 2022 by the Regional Center of Pharmacovigilance (RCPV) of Nice (France). Each upper panel shows the interactions between few adverse drug reactions (ADRs), on the x-axis, and a sample of molecules, on the y-axis, for three selected periods (highlighted in dark blue in the lower panel). The histograms shows the distribution of ADR notifications received by the RCPV during the whole time period. In such a context, the need for an automated, scalable, and efficient approach to analyze the data becomes apparent. This is precisely where dynamic co-clustering methods could come into play. The idea is to apply these techniques to ease the transition from labor-intensive manual signal detection to an automated, data-driven approach. Dynamic co-clustering could be a valuable tool in identifying clusters of drugs and adversarial effects, as well as temporal patterns, within the datasets helping medical authorities discovering possible atypical patterns or abrupt changes in the progress of declarations. Looking at Figure 1.5, it can easily be noticed that, during the years 2017 and 2021, there is an extremely uncommon behavior in the progress of notifications to the pharmacovigilance center. This departure from the usual patterns highlights the challenges of dealing with data that display signals of very different magnitudes. Indeed, behind those very visible effects, there might be other less visible signals that need to be detected for obvious public health reasons.

1.6 Motivations and contributions of the thesis

This thesis is structured to provide a systematic exploration of the topic of dynamic co-clustering. Our primary goal is, in fact, the development of dynamic co-clustering methods capable of handling high-dimensional sparse data. The motivation behind this research arises from the growing prevalence of such complex datasets in various fields, often characterized by time-dependency, sparsity and high dimensionality. In this Chapter 1, we introduced the overall scope of the thesis, delineating the central theme, challenges to be addressed, methodologies to be employed and some examples of the broad application domains. This chapter lays the foundation for the subsequent detailed discussions. Chapter 2 presents a more formal exploration of the related state of the art. Specifically, the chapter focus is on the details of co-clustering problems in a general context and then highlights the specifics of Latent Block Models (LBM) and various inference methods. This chapter sets the stage for the original contributions of this thesis, which will be detailed in the following chapters.

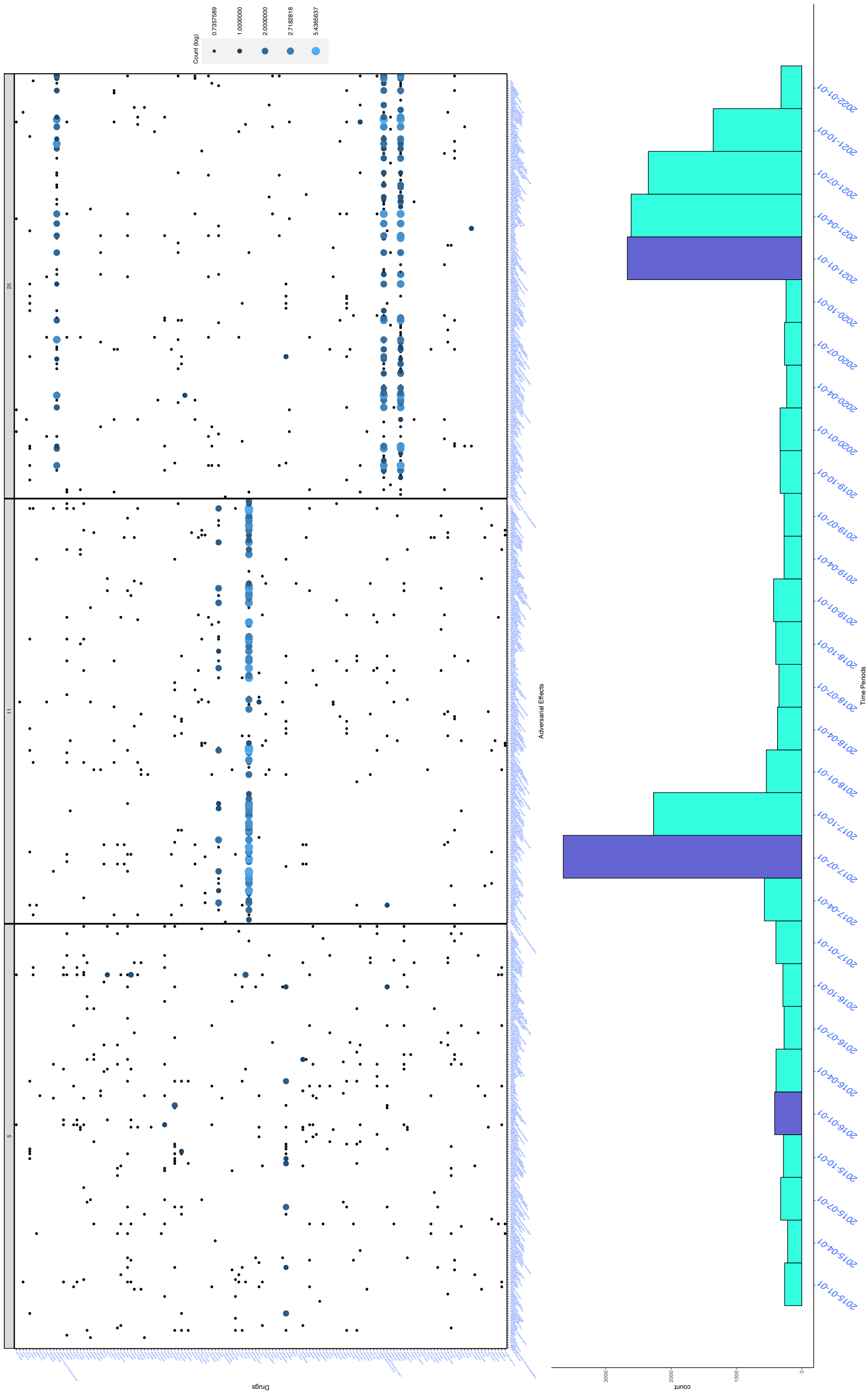


Figure 1.5: Notifications of adverse drug reactions (ADRs) received by Regional Center of Pharmacovigilance (RCPV) of Nice (France), in 3 different trimesters, highlighted in dark blue.

- Chapter 3 introduces a first contribution of the thesis – the dynamic Latent Block Model (dLBM). Here, we presents the formalization of this novel method, by leveraging the non-homogeneous Poisson process (NHPP) for the data modeling and the Stochastic EM-Gibbs algorithm for the inference process.
- Chapter 4 extends the previous model by incorporating sparsity modeling via Zero-Inflated distributions. Additionally, the integration of dynamic systems for cluster membership modeling with neural networks further enriches the model functionality.
- Chapter 5, the model scope expands further by introducing online capabilities. The logical progression is to allow the model to run in real-time, by enabling it to continually operate as new observations come, by keeping it computationally efficient.

Chapter 6 first explores the ongoing work within this thesis, providing an overview of the current research landscape. Then, we delve into potential future directions, offering insights for further studies in this domain. Finally, the chapter ends by drawing comprehensive conclusions based on the work accomplished throughout the thesis.

Chapter 2

Backgrounds

In this chapter, we recall the theoretical foundations for the methodologies that will be presented in the following chapters. Section 2.1 introduces clustering for high-dimensional data, then Section 2.2 focuses on problem the co-clustering problem, considering both metric-based and model-based approaches. Then, Section 2.3 gives an insight of Latent Block Models (LBMs). Dynamic models emerge as a central theme in Section 2.4, with a special focus of their application to both SBM and LBM. Shifting towards inference methods, the chapter introduces three pivotal algorithms: the EM algorithm, the Stochastic EM algorithm, and the Variational EM algorithm (Section 2.5). Deep learning extensions of the models and of the inference process are then presented in Section 2.6.

2.1 Clustering of high-dimensional data

In the contemporary landscape of data, we are surrounded by a lot of information that keeps growing. This big amount of data comes from many different sources, and we really need autonomous ways to find important patterns and to summarize them. That is where clustering comes into play. Operating in an unsupervised context, clustering algorithm involve grouping similar data points together based on shared attributes. This unsupervised nature allows us to explore the natural structure within the data without relying on predefined labels or classes. Also, when the dimension of a dataset is big, as it usually happens, the analysis of a restricted number of clusters is easier than exploring the whole dataset.

One notable distinction within clustering algorithms lies in their underlying approach: discriminative and generative. Discriminative methods are based on aiming to assign data points to clusters by identifying decision boundaries between them. Hence, these methods seek to discern observations that share similarities through quantifying the proximity or likeness of their attributes.
One popular way to accomplish this task is the k-means algorithm (MacQueen et al., 1967). Most commonly, the k-means algorithm aims to identify compact and well-separated clusters of data points relying to the Euclidean distance between them. However, any other distance can be employed depending on the geometry of the space where the observations live in. Due to the highly complex and non-convex nature of the related optimization problem, the k-means algorithm employs an iterative, heuristic approach that explores potential partitions. The process begins with an initial partition of the data into K clusters and iteratively executes two steps until convergence: first, it computes the K barycenters corresponding to the current partition, and subsequently, it updates the partition by assigning each data point to the cluster represented by its nearest barycenter. Other prevalent clustering methodologies construct a weighted affinity graph based on similarity scores between the observations, thereby reflecting local connections between objects. Spectral clustering (Von Luxburg, 2007) is a branch of unsupervised machine learning that focuses on how to build affinity (or similarity) graphs from the data and exploit eigen-decompositionn in order to obtain expressive clusters. Those methods are useful for uncovering heterogeneous clusters, possibly non linearly separable configurations.
On the other hand, probabilistic approaches directly model the probability distribution the data are supposed to come from. Since a generative model is usually

employed, they can be referred as model-based approaches. In the clustering context and built upon statistical principles, these methods propose a probabilistic model that is assumed to have generated the data based on a hidden partition. Typically, this is a discrete latent random variable and the goal is to find the partition that best labels one observation membership to a cluster. A well-acknowledged example of this approach can be seen in finite mixture models (McLachlan et al., 2019). Here, every cluster is defined by a specific parametric distribution known as a mixture component. In this setup, it is assumed that objects in the same cluster come from the same component and they are assumed independent and identically distributed (i.i.d). The advantage of these methods it that, being the partition treated as a random variable, one can compute (or estimate) the probabilities of observations belonging to each cluster. Furthermore, mixture models serve as the foundation for various other model-based techniques, including block modeling for co-clustering and graph clustering (Govaert and Nadif, 2003; Daudin et al., 2008). Specifically considering the scope of this thesis, it is important to emphasize that we do indeed utilize model-based co-clustering as a fundamental building block. Model-based approaches facilitate the measurement of uncertainty in the outcomes through a probabilistic understanding of the partition. Furthermore, numerous criteria rooted in similarity-based methods can be viewed as particular cases of model-based approaches, with the latter presenting increased adaptability and transparency in modeling. For instance, the k-means algorithm corresponds to a particular Gaussian mixture model, assuming identical and proportionally scaled covariance matrices across clusters, and equal mixture proportions.

However, traditional clustering techniques might be not sufficient to provide a synthetic view of the data in situations where the number of covariates (M), is higher than the count of observations (N): $N < M$ or $N \ll M$. This phenomenon generally characterizes the concept of high-dimensional data (Bouveyron and Brunet-Saumard, 2014). In this scenario, a common approach to tackle the challenge is by employing data reduction techniques. To be more precise, the original dataset X, sized $N \times M$, can be effectively summarized into a new dataset sized $Q \times L$. In this context, $L(\ll M)$ clusters efficiently capture the essence of the M variables, mirroring the idea that $K(\ll N)$ clusters encapsulate the N individuals (Biernacki et al., 2023). The simultaneous clustering of both individuals and variables is referred to as co-clustering and it constitutes the central focus of this thesis.

In addition, traditional clustering methods may suffer from computational inefficiency when dealing with high-dimensional data, as memory requirements can become prohibitive. Several strategies have been put forth to address these challenges. One method, proposed by Guyon and Elisseeff (2003), involves variable selection. This approach seeks to select and retain only the most pertinent variables within the dataset, reducing its dimensionality. In the context of clustering, Witten and Tibshirani (2010) introduced a novel framework allowing for data sparsity, it relies on similarities among data points within a sparse context, offering an alternative perspective for clustering. Additionally, Raftery and Dean (2006); Maugis et al. (2009) have proposed model-based approaches to feature selection. An alternative way of addressing the curse of dimensionality is through dimension reduction, where the data is assumed to exist within lower-dimensional subspaces. Principal Component Analysis (PCA), proposed by Jolliffe (2002) identifies these lower-dimensional subspaces, capturing the data variance while simplifying the dataset representation. Moreover, probabilistic variations of PCA have been introduced by Tipping and Bishop (1999) and Chiquet et al. (2018).

2.2 The co-clustering problem

Back in 1965, Good (1965) introduced an extension of a clustering algorithm proposing to achieve a joint classification of rows and columns in a table. The aim of this analysis is to uncover any underlying structure consisting of homogeneous blocks that might exist between the two sets; this is referred to as co-clustering. Since then, other significant contributions have laid the groundwork for the development of co-clustering (Bock, 1979; Govaert, 1983; Dhillon et al., 2003b). Subsequently, co-clustering has evolved into various forms, and its significance has grown considerably in recent years with the emergence of diverse applications. These include gene expression analysis (Cheng et al., 2008; Hanisch et al., 2002; Jagalur et al., 2007), text mining and topic modeling (Bergé et al., 2019; Dhillon et al., 2003a; Wang et al., 2009), recommending systems and collaborative filtering (George and Merugu, 2005; Deodhar and Ghosh, 2010; Xu et al., 2012; Shan and Banerjee, 2008) and political data, having been studied to identify groups of politicians and political issues (Wyse and Friel, 2012a; Hartigan, 1975).
This section sets the stage for the co-clustering problem, offering an overview of its formulation. Furthermore, it explores the distinction between the metric-based approach and the model-based approach, describing in detail the two main

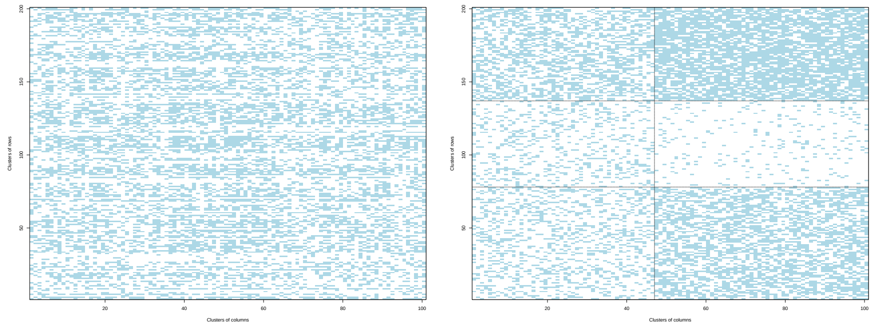


Figure 2.1: Example of co-clustering of binary simulated data. We see on the left side the raw interaction pattern and on the right side the reorganized incidence matrix. The black dashed lines identify the block-clusters.

conceptual paths.

2.2.1 Definition of co-clustering

Block-wise co-clustering involves the simultaneous clustering of rows and columns, using two partitions. The goal is to identify the structure comprising Q homogeneous row clusters and L homogeneous column clusters, as depicted in Figure 2.1. This illustration shows a co-clustering algorithm applied to some simulated data, with the incidence matrix of raw data on the left and the reorganized incidence matrix on the right. This matrix is built such that rows (columns) are permuted in a way that nearby rows (columns) belong to the same cluster. The dashed lines identify the block-clusters. This type of co-clustering can be applied to binary data, continuous data, or contingency tables. Drawing from the earlier notations, the identification of homogeneous blocks in matrix X can be accomplished by partitioning rows into Q classes and columns into L classes. The objective is to uncover the partitions for N observations into Q classes, indicated by the latent variable Z and the partitions for M variables into L classes, indicated by the latent variable W. As mentioned earlier, to accomplish this task different methods have been proposed over the years. In the following some major approaches are formalized.

2.2.2 Co-clustering of discrete data

Co-clustering stands out as a powerful technique in machine learning, with a distinct approach for uncovering hidden patterns within complex datasets. As we mentioned in the previous section, unlike traditional clustering methods that focus solely on grouping rows (columns), co-clustering "simultaneously" clusters both dimensions. With this respect, it is important to emphasize the conceptual distinction between co-clustering and bi-clustering. Bi-clustering (Madeira and Oliveira, 2004) methods permit the existence of clusters that overlap and also allow for some observations to remain unclustered. In contrast, in co-clustering each individual or feature entry belongs exclusively to a single cluster; the option of leaving entries unassigned is not feasible within the co-clustering framework.

Similar to traditional clustering, co-clustering methods can be classified into two main categories: discriminative and generative. Discriminative methods determine boundaries that differentiate clusters, while generative methods employ probabilistic models to explain data generation within clusters. Further details on these methods can be found in Section 2.2.3 and 2.2.4.

In this thesis, we mainly apply co-clustering to discrete data. These data consists of categorical or binary attributes, like presence or absence of features. By applying co-clustering to such data we aim at finding consistent associations between subsets of items, revealing connections that may not be readily apparent through traditional analysis methods. In the specific context of this thesis, the key focus is on count data.
Count data refers to a specific type of discrete data where observations are represented as non-negative integer numbers, indicating the frequency or occurrences of certain events. Count data possesses peculiar characteristics as their unique nature, often resulting in sparse matrices with numerous zeros, poses challenges for traditional analysis methods.

However, co-clustering techniques proved to be a valuable tool also in summarizing these data by detecting intricate patterns within sparse matrices (Ailem et al., 2017a). Indeed, despite count data frequently results in many infrequent occurrences of certain events, co-clustering identifies latent relationships by considering both rows and columns simultaneously, that might be neglected through singular row or column clustering. Also, in scenarios involving large interaction datasets, such as online user engagement or network activities, co-clustering efficiently summarizes complex patterns. By grouping similar entities based on their interaction profiles, it simplifies the interpretation of intricate interactions, making the data more manageable and insightful.

2.2.3 Metric-based approaches

Among the co-clustering methods not involving generative models, we mention the following metric based approches such as: non-negative matrix tri-factorization (NMTF) (Labiod and Nadif, 2011b; Ding et al., 2006), spectral co-clustering (Dhillon, 2001) and information theory (Dhillon et al., 2003b). In the following we give an insight of these approaches.

Non-negative matrix tri-factorization The non-negative matrix tri-factorization (NMF, Lee and Seung, 2000) was first introduced for dimensionality reduction and feature extraction purposes and then extended as an algorithm for co-clustering (Labiod and Nadif, 2011b; Ding et al., 2006). When presented with a non-negative data matrix X, NMF-based co-clustering algorithms aim to achieve a three-factor decomposition ZSW^T of X, where Z and W can be seen as latent variables accounting for the rows and columns clustering and S is a non-negative reduced form of X due to co-clustering. The task of minimizing the distance between the data matrix X and the non-negative decomposition, is achieved by solving:

$$\min_{Z>0,S>0,W>0} ||X - ZSW^T||^2$$

where $||\cdot||$ is the Frobenius norm. The minimization can be addressed through an iterative process of alternating least-squares optimization. In cases where the matrix X is not necessarily non negative, various algorithms based on dual k-means approach have been proposed to minimize the criterion (Rocci and Vichi, 2008).

Spectral co-clustering One of the most popular papers introducing the spectral co-clustering is the one from Dhillon (2001). The algorithm presented in that paper focuses on a bipartite graph model for a document-word collection. The authors proposed to find an optimal partitioning of the document-word bipartite graph, dividing it into Q row and column clusters, the first representing documents the second representing words. This partitioning is achieved through a spectral relaxation approach of the minimum cut graph partitioning problem. In the context of bipartite graph, each row and column within a dataset are represented as a distinct vertex. These vertices are partitioned into two separate sets, each connected by a collection of undirected edges. Given the data matrix X, the adjacency matrix of the related bipartite graph assumes the structure:

$$A = \begin{pmatrix} 0 & X \\ X^T & 0 \end{pmatrix}. \tag{2.1}$$

Built on this graph representation, co-clustering strives to identify Q clusters composed of closely interconnected nodes. This is achieved by minimizing the cumulative edge weights between clusters and simultaneously maximizing the cumulative edge weights within clusters. The objective is to uncover coherent substructures in the dataset by emphasizing strong interconnectivity within clusters while minimizing the connections between them.
Then, Labiod and Nadif (2011a) introduced a normalized generalized version of the modularity measure, frequently employed in network analysis. Modularity in a graph quantifies the quality of the community structure, evaluating how well nodes are grouped into clusters. They further developed a spectral co-clustering algorithm that maximizes this proposed criterion, aimed at simultaneously clustering binary and categorical data. Their research highlighted the algorithm effectiveness in text document clustering and explored determining the number of co-clusters using their criterion. A more recent study by Ailem et al. (2015) presented a co-clustering algorithm designed to maximize the graph modularity. Notably, the authors devised an efficient k-means-like alternating optimization scheme that directly optimizes graph modularity, eliminating the computationally intensive eigenvector computation. Also, in that paper, the authors demonstrated how the modularity measure can be leveraged for assessing the number of clusters.

Information theory Dhillon et al. (2003b) employ concepts from information theory to formulate the co-clustering problem theoretically. The paper proposes to perform the co-clustering via the maximization of the mutual information between the clustered random variables. Hence, the optimal co-clustering is essentially the arrangement of clusters that results in the highest mutual information between the grouped random variables. Alternatively, the authors suggest that the optimal co-clustering also minimizes the difference (referred to as "loss") between the mutual information among the original random variables $I(Z, W)$ and the mutual information among the clustered random variables $I(\hat{Z}, \hat{W})$, such that the

clustering should retain as much information as possible.
The quality of the resulting co-clustering is evaluated based on the loss of mutual information, a lower loss indicates a higher-quality co-clustering.

2.2.4 Model-based approaches

As mentioned in the previous section, model-based co-clustering methods assume that the data matrix is generated by an underlying probabilistic model. This models captures the latent structure within the data, allowing for the identification of clusters that exhibit shared characteristics. Among the advantages of model-based co-clustering, we mention its sound statistical foundations and its flexibility in terms of data complexities such as missing values, heterogeneous dimensions and sparsity, enhancing their robustness and accuracy. A first model was introduced by Rooth (1995) who proposed a probabilistic model for block-wise classification of contingency data with a distinctive diagonal structure. The model includes an algorithm that employed formulas similar to the Baum-Welch estimation equations used in hidden Markov models. In a separate study, Hartigan (2000) investigated the co-clustering of votes within the United States Congress. In this context, senators were divided into distinct blocks, and legislative measures were categorized into various types. To accomplish this, Hartigan proposed a probabilistic model aimed at performing co-clustering of binary data. Notably, the probabilities associated with each block and type in the final partition were estimated using a Monte Carlo method leveraging Markov chains. The much popular Latent Block Model (LBM) (Govaert and Nadif, 2003) was introduced for the co-clustering of binary data matrices, based on the assumption that rows and columns are grouped in hidden clusters and that observations within a block (intersection of a row cluster and a column cluster) are independent and identically distributed. This model is the milestone of this thesis and it will be presented in detail in Section 2.3.
In the context of Bayesian model-based co-clustering, Kemp et al. (2006) introduced an Infinite Relational Model (IRM) aimed at revealing stochastic structures in relational data, particularly binary observations. Also Banerjee et al. (2004) advanced the field with the generalized Bregman co-clustering algorithm, tackling the co-clustering problem as a matrix approximation task. Building on this foundation, Shan and Banerjee (2008) introduced the Bayesian Co-Clustering (BCC) models. These innovative models introduced the concept of mixed membership

within both row and column clusters, allowing observations and features to belong to more then one cluster. The BCC models use distinct Dirichlet priors for rows and columns. Here, each observation is assumed to be generated through an exponential family distribution that corresponds to its respective row and column clusters. The model design allows for sparse matrices, as its inference process operates exclusively based on non-missing entries. Beyond the discovery of co-cluster structures within observations, the BCC model yields a low-dimensional co-embedding.

2.3 The Latent Block Models

This section offers an introduction to the Latent Block Model (LBM), a seminal framework in the domain of model-based co-clustering. The first part revisits the most important aspects of the original binary LBM, followed by an exploration of the extensions that have emerged over the decades.
Within this framework, the dataset takes the form of a random matrix $X = \{X_{ij}\}_{i=1,\ldots,N,j=1,\ldots,M}$ with dimensions $N \times M$. This structure assumes that both the rows and columns of X are partitioned into distinct clusters, with Q clusters for rows and L clusters for columns. This clustering design assumes that data belonging to the same block are independent and identically distributed.

2.3.1 The binary Latent Block Model

The original binary Latent Block Model that we recall in this section was first introduced by Govaert and Nadif (2003). The random binary matrix X, represents the data and the entry (i, j) can be $X_{ij} = 1$, signifying a connection between row i and column j, or $X_{ij} = 0$ that implies the absence of a connection. Here, the latent structure of rows and columns of X is identified by the following latent variables:

- $Z := \{z_{iq}\}_{i\in 1,\ldots,N,q\in 1,\ldots,Q}$ represents the clustering of rows into Q groups: $\mathcal{A}_1, \ldots, \mathcal{A}_Q$. The row i belongs to cluster $\mathcal{A}_q$ iff $z_{iq} = 1$;

- $W := \{w_{j\ell}\}_{j\in 1,\ldots,M,\ell\in 1,\ldots,L}$ represents the clustering of columns into L groups: $\mathcal{B}_1, \ldots, \mathcal{B}_L$. The column j belongs to cluster B_ℓ iff $w_{j\ell} = 1$.

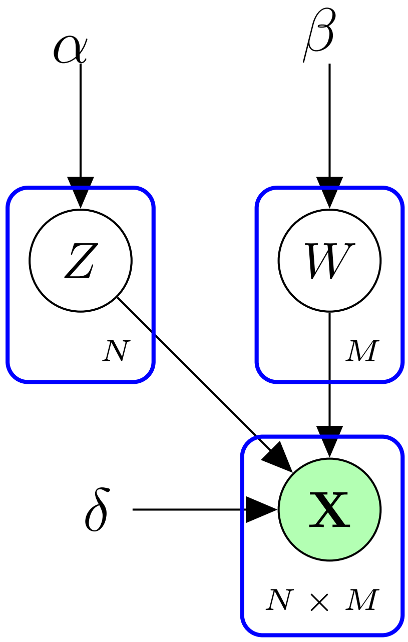


Figure 2.2: Graphical representation of LBM.

In other words, the elements z_{iq} (resp. $w_{j\ell}$) of this matrix are equal to 1 if row i (resp. j) belongs to cluster q (resp. ℓ) and 0 otherwise. Moreover, Z and W are assumed to be independent and distributed according to multinomial distributions:

$$p(Z|\alpha) = \prod_{q=1}^{Q} \alpha_q^{|\mathcal{A}_q|}, \qquad p(W|\beta) = \prod_{\ell=1}^{L} \rho_\ell^{|\mathcal{B}_\ell|},$$

where $\alpha_q = \mathbb{P}(z_{iq} = 1)$, $\rho_\ell = \mathbb{P}(w_{j\ell} = 1)$, $\sum_{q=1}^{Q} \gamma_q = 1$, $\sum_{\ell=1}^{L} \rho_\ell = 1$, and $|\mathcal{A}_q|$ and $|\mathcal{B}_\ell|$ respectively represent the number of rows in cluster $\mathcal{A}_q$ and the number of columns in cluster $\mathcal{B}_\ell$. A graphical representation of LBM can be seen in Figure 2.2, where the data matrix X is displayed in green because it is the only observed variable and the two latent variables, Z and W are identified by the circles. Also, the number of independent samples of X, Z and W are specified in the respective plates, as long as the dependence from the related parameters. The LBM model further assumes the entries X_{ij} are independent, conditionally to Z and W, and their distribution $\varphi(., \delta)$ belongs to the same parametric family, where the parameter δ only depends on the given block. In particular, since here we are considering the original binary formulation, we assume that the data are distributed following a Bernoulli distribution, hence:

$$X_{ij} \mid z_{ik} w_{j\ell} = 1 \sim \mathcal{B}(X_{ij}, \delta_{q\ell}). \tag{2.2}$$

Where $\delta_{q\ell}$ is the block-dependent parameter of the Bernoulli distribution, thus: $\delta_{q\ell} = \mathbb{P}(X_{ij} = 1 | z_{iq} w_{j\ell} = 1)$. With these assumptions the complete data likelihood can be written as:

$$\begin{aligned} p(X, Z, W|\theta) &= p(Z|\alpha)p(W|\beta)p(X|Z, W, \delta) \\ &= \prod_{q=1}^{Q} \gamma_q^{|\mathcal{A}_q|} \prod_{\ell=1}^{L} \rho_\ell^{|\mathcal{B}_\ell|} \prod_{i=1}^{N} \prod_{j=1}^{M} \delta_{Z_i W_j}^{X_{ij}} \cdot (1 - \delta_{Z_i W_j})^{(1-X_{ij})}. \end{aligned} \tag{2.3}$$

2.3.2 Extensions of the Latent Block model

Whereas the original formulation of the model dealt with binary data, in the last two decades the model has been extended to handle several data types (Biernacki et al., 2023).
Govaert and Nadif (2010) extended the Latent Block Model to handle count data, where the data can take non-negative integer values, $X_{ij} \in \mathbb{N}$. In this extension, Poisson-distributed latent variables allow the LBM to model the counts within co-clustered groups. Thus, Eq. (2.4) becomes:

$$X_{ij} | z_{iq} w_{j\ell} = 1 \sim \mathcal{P}(X_{ij}, \mu_i v_j \lambda_{q\ell}). \tag{2.4}$$

In this paper, the Poisson parameter is deconstructed into three components: μ_i for the size effect of row i, v_j for the size effect of column j, and $\lambda_{q\ell}$ for the effect of block (q, ℓ). However, this parameterization lacks identifiability, making the simultaneous estimation of μ_i, v_j, and $\lambda_{q\ell}$ unfeasible without additional constraints. To address this, the authors introduced a set of constraints for all q and ℓ: $\sum_i \mu_i = \sum_j v_j = \left(\sum_q \alpha_q \lambda_{q\ell}\right)^{-1} = \left(\sum_\ell \beta_\ell \lambda_{q\ell}\right)^{-1} = \sum_{ij} E(X_{ij})$. This constraint ensures that $E(\sum_j X_{ij}) = \mu_i$ and $E(\sum_i X_{ij}) = v_j$.

Also, Lomet (2012) and Nadif and Govaert (2010) proposed an extension for continuous data, with the data matrix having values $X_{ij} \in \mathbb{N}$. In this context, $\varphi(\cdot; \delta_{q\ell})$ is assumed to be the probability density function of the Gaussian distribution $\mathcal{N}(\mu_{q\ell}, \sigma^2_{q\ell})$ with parameter $\delta_{q\ell} = (\mu_{q\ell}, \sigma^2_{q\ell})$, denoting the mean and the variance respectively.

In the case of categorical data, for $h = 1, \ldots, k$, with k levels, the extension has been proposed by Keribin et al. (2015). In this context, the generic data element $X_{ij} = \left(X_{ij}^h\right)_h \in \{0, 1\}^k$, with $\sum_{h=1}^k X_{ij}^h = 1$. Here, the probability distribution used to model the data is the multinomial distribution: $\mathcal{M}(1, \delta_{q\ell})$, with $\delta_{q\ell} = (\delta^1_{q\ell}, \ldots, \delta^k_{q\ell})$.

Other extensions include LBM for ordinal data: Jacques and Biernacki (2018) use a recent distribution for ordinal data called Binary Ordinal Search (BOS) by Biernacki and Jacques (2016). Also, Corneli et al. (2020) proposed to take advantage of the binary formulation of LBM to manage missing data possibly not missing at random, solving the scalability issues related with the inference of the first paper.

LBM has also been extended to deal with textual data, in particular a recent work by Bergé et al. (2019) introduces the Latent Topic Block Model (LTBM). This model is designed to capture the inherent structures within textual interaction data, which involves simultaneously clustering of the non-null entries of the incidence matrix, that represent a corpora of documents. The primary objective of this approach is to discern and estimate the latent topics governing the textual interactions encapsulated within the incidence matrix, thereby facilitating a more profound comprehension of the co-clustering dynamics. Instead, Wang et al. (2009) use a Bayesian framework to simultaneously cluster documents and words of a document-term matrix, in such a way that topic proportions are no longer specific to each document but to each cluster of documents. More recently, Kumar et al. (2016) generalized this approach introducing statistical dependence between row and column clusters of a document-term matrix, enriching the model capability to capture intricate relationships between documents and words.

In contrast to the frequentist perspective, the Bayesian approach operates under the premise that the model parameters are treated as random variables, governed by prior distributions. To simplify computational aspects, many researchers opt for conjugate priors (Raiffa et al., 1961) for the parameter set θ (Shan and Banerjee, 2008; van Dijk et al., 2009; Wyse and Friel, 2012a; Keribin et al., 2015). For instance, in the work of Brault and Mariadassou (2015), this translates into adopting a Dirichlet distribution for the cluster proportions α and β. For simplicity, Dirichlet priors are often centered around uniform proportions with dispersion parameters denoted as a_1 and a_2 (frequently set to $a_1 = a_2 = a$). Wyse and Friel (2012a) employ $a = 1$, whereas Keribin et al. (2015) set $a = 4$ to mitigate the occurrence of empty classes. Both authors model all Z_i (and likewise W_j) to be sampled from $\mathcal{M}(1; \alpha)$ (and $\mathcal{M}(1; \beta)$) distributions. However, divergent formulations also emerge among different authors. For instance, Shan and Banerjee (2008) introduce an another layer for sampling Z and W, where distinct mixing parameters α_i are drawn for each row, hence they use these values to sample Z_i from $\mathcal{M}(1; \alpha_i)$, the same procedure for columns. While this approach significantly expands the dimension of the sampling space, it fosters more intricate modeling capabilities. Furthermore, variations arise in the treatment of Q and L, representing the numbers of classes. Keribin et al. (2015) treat them as fixed model parameters, while Wyse and Friel (2012a); van Dijk et al. (2009) consider them as random variables and impose a truncated Poisson distribution, using as a prior the mean λ. These distinct approaches lead to differences in their selection methods for Q and L. The former method involves estimating the model parameters for each pair $(Q; L)$ and subsequently selecting the optimal pair using a model selection criterion. In contrast, the latter method employs a single run to estimate a posterior distribution encompassing all parameters, including $(Q; L)$. However, this singular run often requires a longer convergence time, and estimating the marginal posterior distribution for the pair $(Q; L)$ can pose challenges.

Concerning co-clustering applications to pharmacovigilance, a seminal article was proposed by Robert et al. (2015). In that paper, the authors introduced the Multiple Latent Block Model (MLBM) by extending the Latent Block Model (Govaert and Nadif, 2008) through the construction of one row partition and two columns partitions that respectively rely on two binary matrices containing the relation between the individual and the medical product and the relation between the individual and the ADR.

2.3.3 Identifiability of LBM

As most mixture models, the LBM parameters are only identifiable up to block relabeling. Addressing this challenge, Keribin et al. (2015) analyzed the identifiability of binary LBM for $N \geq 2L - 1$ and $M \geq 2Q - 1$, provided that two conditions are satisfied:

- C_1: For all $1 \leq q \leq Q$, $\alpha_q > 0$, and all the coordinates of vector $\delta\beta$ are distinct.
- C_2: For all $1 \leq \ell \leq L$, $\beta_\ell > 0$, and all the coordinates of vector $\delta^{'}\alpha$ are distinct.

Remarkably, these conditions are not overly restrictive. Conditions C_1 and C_2 dictate that the probabilities $\mathbb{P}(X_{ij} = 1|z_{iq} = 1)$ and $\mathbb{P}(X_{ij} = 1|w_{j\ell} = 1)$, indicating the likelihood of an event in a cell of a row of class q or a column of class ℓ, can be arranged in a strictly ascending order. Also, C_1 and C_2 can be seen as extensions of the conditions for the identifiability of Stochastic Block Model introduced by Celisse et al. (2012). Keribin et al. (2015) also extends the condition set to the categorical case with k levels:

- C_1: For all $1 \leq q \leq Q$, $\alpha_q > 0$, and all the coordinates of vector $\delta^h \beta$ are distinct,
- C_2: For all $1 \leq \ell \leq L$, $\beta_\ell > 0$, and all the coordinates of vector $\delta'^h \alpha$ are distinct,

with $h = 1, \ldots, k-1$ and $\delta^h = \{\delta^h_{q\ell}\}_{q=1,\ldots,Q;\ell=1,\ldots,L}$.

2.3.4 Model selection

In the domain of clustering, the quest to find the most suitable number of clusters is a core concern. Likewise, within the framework of co-clustering, determining the optimal values for the number of row clusters (Q) and column clusters (L) takes center stage. The model-based approach allows us to see this decision as a model selection task. However, using conventional criteria in the co-clustering context may not be directly applicable. Indeed, the use of asymptotic criteria, such as BIC (Schwarz, 1978), can be precarious given the dual asymptotic nature introduced by both quantities, N and M. Also, the BIC criterion relies on the integrated likelihood, however, when used for selecting a mixture model in a model-based clustering context, the BIC criterion may overestimate the appropriate model size. Hence, to avoid those issues the exact Integrated Complete Likelihood (ICL) has been proposed by Biernacki et al. (2000). Then, Keribin et al. (2015) proposed an approximation to a allow for a closed-form expression. For co-clustering methods, as the dimensions N and M go to infinity, we can write this ICL approximation, as follows:

$$ICL(Q, L) = \log p(x, \hat{Z}, \hat{W}|\theta) - \frac{Q-1}{2}\log N - \frac{L-1}{2}\log M - \frac{QL(r-1)}{NM}\log(NM), \tag{2.5}$$

where r is the number of free parameter of the model. The pair $(\hat{Q}, \hat{L})$ that leads to the highest value of the ICL is considered as the most meaningful number of clusters for those data. Keribin et al. (2015) also suggest that BIC and ICL showed asymptotic equivalence, sharing the same asymptotic behavior. If BIC is consistent for the Latent Block Model as in the simple mixture case, it is plausible that ICL also demonstrates consistency in selecting both Q and L for co-clustering. This contrasts with simple clustering, where ICL consistency holds only for well-separated clusters (Baudry, 2015).

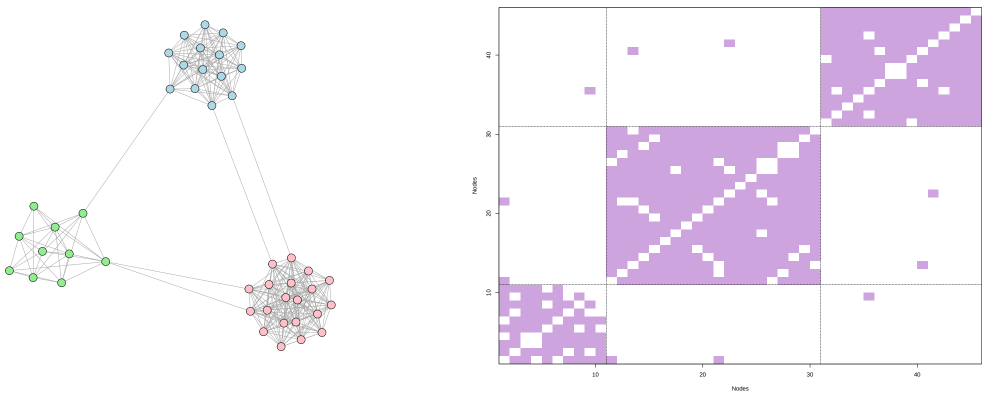
Also, exploring the entire set of potential values for (Q, L) in co-clustering is a more complex task than in traditional clustering because it involves considering two dimensions. With $Q \in [0, Q_{max}]$ and $L \in [0, L_{max}]$, the total number of possible models, $Q_{max}L_{max}$ can become quite large. Robert (2017) proposes to use greedy search, which focuses on exploring a pertinent subset of combinations. In this method, the algorithm begins by calculating the model selection criterion for models created by adding one cluster at a time, in rows or columns. This process continues until the model selection criterion no longer improves, the solution with the highest criterion is retained. In a Bayesian variation of the Latent Block Model, Wyse and Friel (2012b) simultaneously estimate both partitions and the number of clusters using a Markov Chain Monte Carlo (MCMC) algorithm. Wyse et al. (2017) replace the MCMC approach with a greedy search, optimizing the ICL criterion directly. This alternative approach offers scalability benefits, particularly in larger settings.

2.3.5 Links with the Stochastic Block Model

This section is divided into two parts. In the first part, we provide an overview of the Stochastic Block Model (SBM), a fundamental tool for analyzing network structures. The second part delves into the connections and similarities that exist between the Stochastic Block Model and the Latent Block Model.

The Stochastic Block Model

The Stochastic Block Model (SBM), introduced by Nowicki and Snijders (2001); Holland et al. (1983); Wang and Wong (1987) is a probabilistic clustering method mainly used to model community structures within networks. In SBM, nodes are partitioned into different blocks, and the goal is to describe how nodes within the same block are connected. More formally, let us consider a graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ where $\mathcal{N}$ is the node set of size $N := |\mathcal{N}|$ and $\mathcal{E}$ the list of edges of size $E := |\mathcal{E}|$. We call a pair of nodes a dyad, and consider the existence or absence of an edge for the dyad (i, j), using the $N \times N$ adjacency matrix, denoted by $Y = \{Y_{ij}\} \in \mathbb{R}^{N \times N}$. If $\mathcal{G}$ is undirected, $Y_{ij} = Y_{ji} = 1$ if i and j connect, 0 otherwise. By construction, for an undirected graph, Y. If $\mathcal{G}$ is directed, $Y_{ij} = 1(0)$ represents an edge (non-edge)



(a) Graph with 3 communities. (b) Reorganized adjacency matrix.

Figure 2.3: Example of node clustering using SBM on simulated binary data with 3 communities.

and is independent of Y_{ji}, hence Y will not be symmetric. Also, here we assume $Y_{ii} = 0$, that is, self loops are not allowed.

Figure 2.3 gives an example on how SBM works. Here, we simulate a binary symmetric adjacency matrix, representing a network with 45 nodes and 315 edges. As we see from Figure 2.3a, the nodes are divided into 3 groups depicted with different colors, representing the communities, with groups 1, 2 and 3 containing 10, 20 and 15 nodes, respectively. The nodes within the same group are more closely connected to each other, than with nodes in another group. Also, an example of adjacency matrix is shown in Figure 2.3b where the colored and white squares represent 1 and 0, respectively, indicating the presence or the absence of a connection.

In SBM, each node belongs to one of the $Q(< N)$ groups, $Q = 3$ in the example. Each node i is associated with a random variable Z_i, such that $Z_i = q$ if and only if $i \in q$. More formally, the cluster memberships of nodes are identified by the latent vector Z, following a multinomial distribution, parameterized by α:

$$Z_i \overset{iid}{\sim} \mathcal{M}(1, \alpha := (\alpha_1, \dots, \alpha_Q)), \tag{2.6}$$

where Z_i is the i-th row of Z and $\mathcal{M}(1, \cdot)$ denotes the multinomial probability mass function and $\alpha_q = \mathbb{P}\{z_{iq} = 1\}$, with $\sum_{q=1}^{Q} \alpha_q = 1$.

Thus $Z := \{z_{iq}\}_{i \in 1,\dots,N; q \in 1,\dots,Q}$ represents the clustering of N rows into Q clusters. Then, in the original model formulation, conditionally on Z, Y_{ij} is assumed to be independently drawn from a Bernoulli distribution:

$$Y_{ij}|z_{iq}z_{j\ell} = 1 \sim \mathcal{B}(Y_{ij}; \pi_{q\ell}), \qquad \forall q, \ell \in \{1, \dots, Q\} \tag{2.7}$$

whose parameter π only depends on the block memberships of (i, j), respectively. This implies that if node i belongs to cluster A_q and node j to cluster A_ℓ, the probability that one edge between them occurs is $\pi_{q\ell}$. A possible variant of the SBM focuses on weighted graphs in which Y_{ij} represents the number of interactions between i and j. In this context conditionally on Z, Y_{ij} follows a Poisson distribution, $\mathcal{P}(\cdot)$,with:

$$Y_{ij}|z_{iq}z_{j\ell} = 1 \sim \mathcal{P}(Y_{ij}; \lambda_{q\ell}), \qquad \forall q, \ell \in \{1, \dots, Q\} \tag{2.8}$$

where $\Lambda = \{\lambda_{q\ell}\}_{q,\ell \in 1,\dots,Q}$ denotes the parameters of the Poisson distributions, representing the expected number of interactions between any node in cluster q and any node in cluster ℓ. Therefore, following Eq. (2.6) and Eq. (2.8), we can finally write the joint probability distribution of one edge as follows:

$$p(Y_{ij}, Z_i, Z_j|\Lambda, \alpha) = \prod_{q,\ell=1}^{Q} \left(\mathcal{P}(Y_{ij}; \lambda_{q\ell})\alpha_q\alpha_\ell\right)^{Z_{iq}Z_{j\ell}}. \tag{2.9}$$

A graphical representation of SBM can be seen in Figure 2.4, where the adjacency matrix Y is displayed in green because is the only observed variable and the latent variables Z_i and Z_j are identified by the circles. Also, the dimensions of the variables (latent and observed) are specified in the respective plates, as well as the dependence from the related parameters.

Links between LBM and SBM

The Latent Block Model (LBM) and the Stochastic Block Model (SBM), are fundamental tools in co-clustering and node clustering in graphs, respectively. SBM is typically used for community detection and it can be seen as a special case of LBM, which does not need the data matrices to be square and/or symmetrical. Although they exhibit distinct characteristics that make each model suitable for different scenarios, they share also relevant similarities. Both LBM and SBM operate based on the core principle of clustering. They aim to group entities, such as rows and columns in a matrix, into coherent blocks or communities. These blocks capture latent structures within the data that facilitate meaningful insights. From the perspective of matrix decomposition, both models can be understood

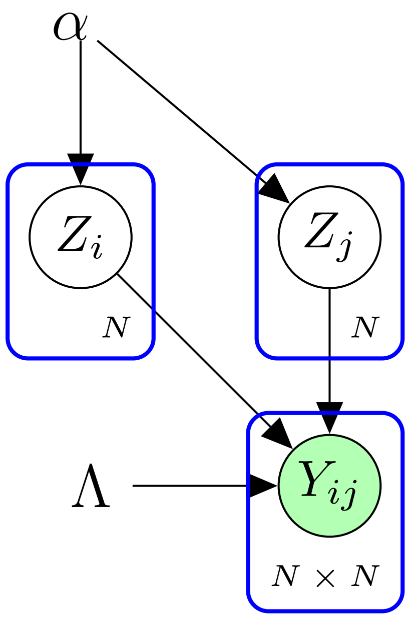


Figure 2.4: Graphical representation of SBM.

as performing matrix factorization and dimension reduction. For instance, SBM factorizes $E(Y) = Z^T \Lambda Z$, while LBM factorizes $E(X) = Z^T \Lambda W$. Moreover, both LBM and SBM involve a generative process that defines how the data are generated from underlying latent structures, where the type of generated data depends on the model assumptions.

2.4 Dynamic extensions of Latent Block Models

Whereas there is a decade-long literature about static model-based clustering and co-clustering methods, dynamic models are more recent. It is worth noting that much more work has been made in the context of networks, in particular, for the Stochastic Block Model (SBM, Nowicki and Snijders, 2001) than for LBM.

2.4.1 Dynamic models for SBM

In the context of model-based dynamic network analysis, various extensions of the Stochastic Block Model (SBM) have been proposed to capture evolving patterns in connectivity and cluster memberships. In the context of discrete time modeling, Yang et al. (2011) introduced a dynamic variant of the SBM, which introduces a temporal aspect by enabling nodes to switch clusters at time $t + 1$ based on their current state at time t. This dynamic behavior is framed within a Markovian framework, where the transition probabilities are organized into a transition matrix.

Then, in a seminal paper, Matias and Miele (2017) highlighted a crucial limitation in dynamic SBMs: it's not feasible to simultaneously vary both the connectivity parameters and cluster memberships over time without encountering identifiability issues. This observation highlights the intricate challenges involved in modeling evolving networks. Also, Matias and Miele (2017) introduced a dynamic Stochastic Block Model that addresses label switching concerns, guaranteeing consistent labels for the same cluster throughout all time instances. This model is rooted in the Stochastic Block Model (SBM), with each time point being treated as an independent SBM. Here, the connection probabilities depends only on the latent clusters at a specific time point. The connections among time points are established by making the cluster probabilities for each time point dependent on the cluster memberships from the preceding time point. To effectively address label-switching challenges, the authors adopt a strategy that looks for groups exhibiting consistent within-group connectivity behavior. This is achieved through constraining the connectivity parameters of the diagonal blocks to remain constant across different time points. For continuous-time dynamic network modeling, an innovative approach involving the non-homogeneous Poisson processes (NHPPs) was put forth by several authors (DuBois et al., 2013; Corneli et al., 2016; Matias et al., 2018).

To define a non-homogeneous Poisson process, let us consider $\{V(t)\}_{t\geq 0}$ being an increasing, right continuous integer-valued process starting from 0. Let $\lambda(\cdot)$ be a strictly positive integrable function. Then $\{V(t)\}_{t\geq 0}$ is a non homogeneous Poisson process (NHPP) if it has independent increments and for all $s \leq t$:

$$V(t) - V(s) \sim \mathcal{P}\Big(\int_s^t \lambda(u)du\Big). \tag{2.10}$$

When $\lambda(u)$ is not a time-dependent function, $\{V(t)\}_{t\geq 0}$ becomes an homogeneous Poisson process, with stationary increments (Norris, 1998). For further details on NHPPs see Corneli (2017).

In particular, Corneli et al. (2016) focus on the clustering of nodes and time intervals in dynamic networks. In this model, the nodes remain fixed, and directed interactions between these nodes occur at specific instances. This results in examining a directed multigraph, allowing multiple edges between nodes, with each edge carrying a time label to denote its occurrence. The count of interactions between any two nodes is modeled using a non-homogeneous Poisson counting process. Therefore, the temporal aspect is introduced through the non-homogeneity property of the counting processes, where the intensity parameter only depends on the cluster memberships of the nodes. Then, to make the inference process tractable, a time interval segmentation strategy is introduced within the study period. This segmentation facilitates the aggregation of interactions within subintervals of the partition, allowing for more manageable analysis. Also Matias et al. (2018) proposed a closely related framework where an extension is proposed for recurrent interaction events in continuous time. Here, every node belongs to a latent group and conditional interactions between two individuals follow an non-homogeneous Poisson process with intensity driven by the respective latent groups of the individuals. In this setup, the nodes are assumed to retain their cluster memberships throughout time, offering an alternative perspective on the temporal dimension of the network evolution.

2.4.2 Dynamic models for LBM

In recent years, co-clustering techniques have seen advancements that encompass dynamic and multi-dimensional data structures. One of the first methods considering the temporal aspect of a co-clustering model is Green et al. (2011). This paper introduced the Evolutionary Spectral Co-Clustering (ESCC) method, which offers two distinct strategies to incorporate historical relationships between instances and features: Respect to Current (RTC) and Respect to Historical (RTH). RTC emphasizes the present by considering only the previous time-step, whereas RTH incorporates information from all previous time-steps. The dynamic extensions of LBMs are, however, more recent. The exploration of probabilistic co-clustering techniques for time-dependent data has been pursued by Bouveyron et al. (2018), who have extended the application of co-clustering to functional data settings. In their work, the primary objective is to handle data that evolves over time, such as functional data represented by curves. To achieve this, they employ a novel parametric approach by mapping the original curves into a transformed space, which is spanned by the coefficients of a basis expansion. This transformation allows for the representation of functional data in a structured manner that is amenable to co-clustering analysis. This approach involves a dimension reduction stage, where functional PCA projections (Ramsay and Silverman, 2005) of the time series are employed. This paper has been recently extended in Bouveyron et al. (2022) to deal also with multivariate functional curves and by Goffinet et al.

(2022) who proposed a non-parametric approach to tackle the problem. In this paper, the authors introduced a novel Bayesian non-parametric approach for latent block modeling in the context of multivariate time series. Called the Functional Non-Parametric LBM (FunNPLBM), this methodology simultaneously partitions both observations and temporal variables. This is achieved by utilizing latent multivariate Gaussian block distributions, wherein a bi-dimensional Dirichlet Process serves as the prior for the block distribution parameters and block proportions. An interesting feature of this approach, owing to its Bayesian non-parametric nature, is its incorporation of model selection throughout the entire process. Recently, Casa et al. (2021) proposed a notable contribution by extending the Latent Block Model to accommodate longitudinal data. They employed the Shape Invariant Model (Lindstrom, 1995) as a foundation to handle temporal complexities in evolving datasets. Unlike traditional methods that rely on basis expansion coefficients, such that their approach directly models the observed data, enhancing the natural representation of temporal evolution and cluster interpretation. The central idea is that individual curves within a cluster can be understood as transformations of a common shape function. This approach handles both functional and longitudinal data, irrespective of their dimensions. The model's flexibility allows for diverse cluster definitions, catering to subject-specific considerations. By integrating temporal dynamics into co-clustering, this work enhances the analysis of complex time-dependent datasets, providing valuable insights into dynamic co-occurrence patterns. Another extension has been proposed by Boutalbi et al. (2020) who introduced the Tensor Latent Block Model (TLBM) for co-clustering, whose aim is to simultaneously cluster rows and columns of a 3D matrix, where covariates represent the third dimension. TLBM was also implemented for different types of datasets: continuous data (Gaussian TLBM), binary data (Bernoulli TLBM) and contingency tables (Poisson TLBM). Then, this method has been extended by the same authors in Boutalbi et al. (2021). In this paper the authors developed a model-based co-clustering method for sparse three-way data, where the third dimension can be seen as a discrete temporal one. Here, the sparsity is handled following the same assumption as in Ailem et al. (2017b) that all blocks outside the main diagonal share the same parameter.

2.5 Inference methods for latent and dynamic models

In the context of model-based clustering and co-clustering, one of the significant challenges is parameter estimation. However, it's worth noting that numerous intricate inference problems are often approached by framing exact inference as an optimization problem. As a result, approximate inference algorithms come into play, offering a means to effectively approximate these optimization tasks (LeCun et al., 2015, Chap. 19). The Expectation-Maximization algorithm (EM) is widely recognized as one of the most popular methods for latent model inference (Dempster et al., 1977). However, despite its foundational role in the algorithms discussed in the subsequent chapters, the original formulation of EM cannot be directly applied for parameter estimation in Latent Block Models (LBMs). The reason lies in the algorithm's inability to compute the joint conditional distribution of the latent variables, $p(Z, W|X, \theta)$, due to the complex interdependent double missing structure. As a consequence, various alternative inference procedures have been introduced to address the challenges posed by latent block models. These include likelihood-based methods (Govaert and Nadif, 2008), variational inference (Keribin et al., 2012), Bayesian inference (Keribin et al., 2012; Wyse and Friel, 2012a), and greedy search approaches (Wyse et al., 2017). In the subsequent section, we will provide an overview of the traditional EM algorithm, followed by an exploration of two extensions that will be employed in the upcoming chapters: the Stochastic EM (SEM) algorithm and the Variational EM (VEM) algorithm.

2.5.1 The EM algorithm

In the context of parametric statistics, maximum likelihood estimation is the general way to go for the estimation of model parameters in the frequentist approach. MLE seeks to find the most likely parameters configuration that aligns with the observed data points, in order to capture their underlying distributional characteristics. This process involves finding the parameter values that maximize the log-likelihood function. Given a parameter set Θ, the maximum likelihood estimator (MLE) is defined as:

$$\hat{\theta} := \underset{\theta \in \Theta}{\operatorname{argmax}} \log p(X|\theta) \tag{2.11}$$

where $\log p(X|\theta)$ represents the data log-likelihood. However, solving this problem is not an easy task when dealing with latent variable models. For example, in finite mixture models there is no closed form solution for the MLE. Even though in this case the maximum likelihood estimate (MLE) of the mixture parameters can be derived using standard optimization techniques in simple cases, the complexity of the required calculation grows rapidly with the dimension of variables and the number of components. This makes the numerical optimization techniques (i.e. Newton-Raphson) computationally expensive and they do not guarantee an improved likelihood is obtained at each iteration (Fruhwirth-Schnatter et al., 2019, Chap. 2). In contrast, the EM (Expectation-Maximization) algorithm, introduced by Dempster et al. (1977) is widely regarded as the most popular approach for obtaining the ML estimates in latent variable models. The basic intuition is that, when the computation of the log-likelihood, $\log p(X|\theta)$, is not feasible, we can compute a lower bound $\mathcal{L}(q;\theta)$ on that quantity. More formally, let us consider a latent variable Z and an arbitrary probability distribution, $q(\cdot)$ over Z, we can then write[1]

$$\begin{aligned} \log p(X|\theta) &= \log \sum_Z p(X, Z|\theta) \\ &= \log \sum_Z p(X, Z|\theta) \frac{q(Z)}{q(Z)} \\ &= \log E_{q(Z)}\Big[\frac{p(X, Z|\theta)}{q(Z)}\Big]. \end{aligned} \tag{2.12}$$

Then, by making use of the Jensen inequality we can write:

$$\log E_{q(Z)}\Big[\frac{p(X, Z|\theta)}{q(Z)}\Big] \geq E_{q(Z)}\Big[\log \frac{p(X, Z|\theta)}{q(Z)}\Big]. \tag{2.13}$$

This allows us to define the lower bound as:

$$\begin{aligned} \mathcal{L}(q;\theta) &:= E_{q(Z)}\Big[\log p(X, Z|\theta)\Big] - E_{q(Z)}[\log q(Z)] \\ &= E_{q(Z)}\Big[\log p(X, Z|\theta)\Big] + H(q), \end{aligned} \tag{2.14}$$

where $H(q)$ is the entropy term. $\mathcal{L}(q;\theta)$ is called evidence lower bound (ELBO) or it can be sometimes referred to as negative variational free energy. Also, the difference between the log-likelihood and the ELBO can be shown to be equal to the Kullback-Leibler (KL) divergence. We can then write:

$$\log p(X|\theta) = \mathcal{L}(q;\theta) + KL(q(Z|X)||p(Z|X,\theta)). \tag{2.15}$$

[1]In this thesis, our primary emphasis is on discrete variables, leading us to employ summations instead of integrals. Nevertheless, the analysis remains unchanged in the case of continuous variables, by substituting summations with integrals.

The KL divergence is always non-negative, hence the lower bound $\mathcal{L}(q;\theta)$ can be at most equal to the log-likelihood $\log p(X|\theta)$. This derivation has an essential implication, the better $q(Z)$ approximate $p(Z|X)$, the tighter the lower bound $\mathcal{L}(q;\theta)$ will be to the log-likelihood. Hence, when $q(Z) = p(Z|X,\theta)$, $\mathcal{L}(q;\theta)$ perfectly approximates $\log p(X|\theta)$. The EM is an iterative algorithm that starts with some initial values for the parameters $\theta^{(0)}$ and, then, each iteration involves the execution of two steps, expectation (E) and maximization (M). Hence, a generic iteration h of the algorithm is composed by:

- E-Step: the algorithm maximizes the lower bound $\mathcal{L}(q;\theta)$ (e.g. minimize the KL divergence) with respect to $q(\cdot)$ for a given value of $\theta^{(h)}$, obtaining $q^{(h+1)} = q^*$.

- M-Step: the algorithm maximizes $\mathcal{L}(q;\theta)$ over θ, keeping q^* fixed, in order to obtain an updated version of the parameter, $\theta^{(h+1)}$. The update here depends on the chosen generative model.

The E-Step requires the posterior distribution $p(Z|X,\theta)$ to be tractable. In fact, when $q = p(Z|X,\theta)$, the KL divergence is equal to zero.
Interestingly, the work conducted by Wu (1983) delved into the convergence characteristics of the EM algorithm. Wu identified specific conditions, applicable to exponential family distributions, which ensure the convergence of $\{\theta^{(h)}\}_h$ towards either stationary points or local maxima of the likelihood function. In practical applications, most implementations of the EM algorithm conclude either when a user-defined iteration limit is attained or if the absolute difference between consecutive ELBO values falls below a predetermined threshold set by the user. Furthermore, the performance of the EM algorithm is notably influenced by its initial values. It is in fact recommended to experiment with various starting choices for $\theta^{(0)}$ and select the outcome that yields to the highest value of the lower bound.
In the context of LBMs, the complete data log-likelihood can be derived from Eq. (2.9) and it writes as follows:

$$\begin{aligned}\log p(X,Z,W|\theta) = &\sum_{i=1}^{N}\sum_{q=1}^{Q} z_{iq}\log\alpha_q + \sum_{j=1}^{M}\sum_{\ell=1}^{L} w_{j\ell}\log\beta_\ell + \\ &+\sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{q=1}^{Q}\sum_{\ell=1}^{L} z_{iq}w_{j\ell}\log\varphi(X_{ij};\delta_{q\ell}).\end{aligned} \tag{2.16}$$

So, at the h-th interation, during the E-step, the expected value of the complete data log-likelihood, according to the posterior distribution with respect to Z and W, writes as follows:

$$E[\log p(X, Z, W|\theta)] = \sum_i \sum_q \mathbb{P}(z_{iq} = 1|X, \theta^{(h)}) \log \alpha_q + \sum_j \sum_\ell \mathbb{P}(w_{j\ell} = 1|X, \theta^{(h)}) \log \beta_\ell + \\ + \sum_i \sum_j \sum_q \sum_\ell \mathbb{P}(z_{iq} w_{j\ell} = 1|X, \theta^{(h)}) \log \varphi(X_{ij}; \delta_{q\ell}), \tag{2.17}$$

where $\sum_i = \sum_{i=1}^N$, $\sum_j = \sum_{j=1}^M$, $\sum_q = \sum_{q=1}^Q$, $\sum_\ell = \sum_{\ell=1}^L$.
Looking at this E-Step we can easily notice that challenges emerge owing to the intricate interdependencies within the model, rendering the computation of terms like $\mathbb{P}(z_{iq} w_{j\ell} = 1|X, \theta^{(h)})$ not tractable. In response, various extensions of the EM algorithm have been introduced over the years, aimed at mitigating this issue. The subsequent sections delve into some of these extensions, shedding light on their specifics.

2.5.2 The Stochastic EM algorithm

Celeux (1985) introduced a modification to tackle the sensitivity of the EM algorithm to its initialization, denoted as $\theta^{(0)}$. The deterministic nature of EM often results in convergence towards suboptimal local maxima within the likelihood landscape. To address this, the Stochastic EM (SEM) was proposed, injecting randomness into the process. SEM incorporates an S-step after the E-step, where a partition $\hat{Z}^{(h)}$ is drawn by sampling from the posterior distribution $q^*(Z) = p(Z \mid X, \theta)$, which is approximated by making use of a Gibbs sampler. This shifts the sequence of updates $\{\theta^{(h)}\}_h$ to rely on complete data likelihood rather than its expectation.

It's important to note that while point-wise convergence is not guaranteed, under general assumptions, the sequence of parameters $\{\theta^{(h)}\}$ generated by SEM forms an ergodic Markov chain, thus converging to a unique stationary distribution (for more details, see Celeux and Diebolt (1992)). Hence, SEM update $\theta^{(h+1)}$ even when $\mathcal{L}(q, \theta^{(h+1)}) < \mathcal{L}(q, \theta^{(h)})$. This ability allows SEM to circumvent insignificant local optima and even saddle points. This characteristic often allows SEM to provide superior solutions compared to EM.
In the context of LBMs, in the S-step, random pairs (Z, W) are drawn based on the posterior distributions of labels, conditional on the given data X: although these

distributions are intractable for certain models, they can be efficiently sampled using a two-step Gibbs algorithm: first simulating $Z|X, W; \theta$ and then simulating $W|X, Z; \theta$. This particular version of SEM, named SEM-Gibbs Keribin et al. (2012, 2010), employs a Gibbs S-step, ensuring exact distributions without resorting to numerical approximations. While SEM-Gibbs does not guarantee an increase in likelihood at each iteration as well, it generates an irreducible Markov chain with a unique stationary distribution, centered around the maximum likelihood parameter estimate.
The estimator $\hat{\theta}$ is derived from averaging a sequence of $\theta^{(h)}$ values obtained after a burn-in period. In SEM-Gibbs, once $\hat{\theta}$ is obtained, a new Gibbs algorithm should be utilized to simulate pairs $(Z, W)|X; \hat{\theta}$. The final partitions $(\hat{Z}, \hat{W})$ are then estimated using the empirical mode from the sample data. It is worth noting that the SEM-Gibbs algorithm might encounter the label switching problem, particularly pronounced in co-clustering scenarios, as it might manifest in both row and column partitions Stephens (2000). Given infinite resources this method produce exact results, however in the context of high dimensions, the resampling can be highly computationally demanding.

2.5.3 The Variational EM algorithm

As discussed in Section 2.5.1, the conventional EM algorithm faces limitations when applied to parameter estimation in LBMs due to their unique structural characteristics. As a remedy, the variational EM algorithm (VEM) emerges as a valid alternative. The fundamental concept behind VEM involves the optimization of a lower bound or the minimization of the Kullback-Liebler divergence within a defined family of distributions denoted as $q(\cdot)$.
This concept mirrors the formalization of the EM algorithm, extending to the domain of VEM. Building upon the equation given in Eq. (2.15), the likelihood decomposition for LBMs is expressed as follows:

$$\log p(X|\theta) = \mathcal{L}(q; \theta) + KL(q(Z, W|X)||p(Z, W|X, \theta)). \tag{2.18}$$

Here, $\mathcal{L}$ represents a lower bound defined by:

$$\begin{aligned} \mathcal{L}(q, \theta) &= \sum_{Z} \sum_{W} q(Z, W) \log \frac{p(X, Z, W|\theta)}{q(Z, W)} \\ &= E_{q(Z,W)}[\log(p(X, Z, W|\theta)] - E_{q(Z,W)}[\log(q(Z, W))], \end{aligned} \tag{2.19}$$

and KL denotes the Kullback-Liebler divergence, between the actual and approximated posterior:

$$KL(q(.)||p(.|X,\theta)) = -\sum_{Z}\sum_{W} q(Z,W)\log\frac{p(Z,W|X,\theta)}{q(Z,W)}.$$

Our objective concerns the search for a distribution $q(\cdot)$ that optimizes the lower bound $\mathcal{L}(q,\theta)$. To make the effective optimization of $\mathcal{L}(q,\theta)$ feasible, we introduce the assumption that the joint distribution $q(Z,W)$ factorizes as follows:

$$q(Z,W) = q(Z)q(W) = \prod_{i=1}^{N}\prod_{j=1}^{M} q(Z_i)q(W_j).$$

This factorized form of variational inference corresponds to an approximation commonly known in physics as the mean field approximation (Parisi and Shankar, 1988). With this factorization in place, the VEM algorithm unfolds as an iterative process, alternating between maximizing the lower bound in Eq. (2.19) with respect to the variational distribution $q(\cdot)$ and updating the model parameters θ, continuing until convergence is achieved. The mean field approximation allows us to rewrite the lower bound in Eq. (2.19) in terms of the variational parameters, with respect to $q(\cdot)$:

$$\begin{aligned}\mathcal{L}(q,\theta) = &\sum_{i}\sum_{q}\tau_{iq}\log\alpha_q + \sum_{j}\sum_{\ell}\eta_{j\ell}\log\beta_\ell + \\ &+\sum_{i,q}\sum_{j,\ell}\tau_{iq}\eta_{j\ell}[X_{ij}\log\delta_{q\ell} + (1-X_{ij})\log(1-\delta_{q\ell})] - \sum_{i,q}\tau_{iq}\log\tau_{iq} - \sum_{j,\ell}\eta_{j\ell}\log\eta_{j\ell},\end{aligned} \tag{2.20}$$

where the quantities $\tau_{iq} = E(z_{iq})$ and $\eta_{j\ell} = E(w_{j\ell})$ represents the variational parameter and the last two member refers to the entropy terms.

2.5.4 Other EM-like extensions

Empirically, the VEM algorithm has demonstrated its ability to provide accurate estimates. Furthermore, there have been notable studies on the convergence of the VEM, as in works Celisse et al. (2012) and Mariadassou and Matias (2015), these estimates heavily rely on an appropriate initialization and tend to generate empty clusters frequently (Brault, 2014; Biernacki et al., 2023). To address the issue of class degeneracy, alternative algorithms based on the EM algorithm have been proposed for estimating the parameters of the latent block models. One such approach involves a Bayesian regularization of the VEM algorithm known as VBayes, as proposed by (Brault, 2014; Keribin et al., 2012, 2015) to handle binary and contingency data. The VBayes technique employs a Dirichlet prior on the mixture parameters, effectively mitigating the problem of empty clusters. In the original papers, this algorithm adopts an initialization strategy involving a Gibbs sampler to estimate the mode of the posterior probability.

Additionally, supplementary algorithms like CEM (Classification Expectation Maximization) have been developed by Celeux and Govaert (1992) with the aim of maximizing the classification likelihood. Notably, CEM inserts an additional C-step between the E and M-steps, wherein a partition $\hat{Z}^{(h)}$ is computed based on the current MAP estimate $\tau^{(h)}$. Subsequently, the conditional expectation in the M-step is substituted with the classification log-likelihood $\log p(X, \hat{Z}^{(h)}|\theta)$. Computationally, this modification involves straightforwardly replacing $\tau_{iq}^{(h)}$ with $\hat{z}_{iq}^{(h)}$ in the M-step.
The CEM algorithm is sometimes characterized as a hard-clustering estimate of the conditional expectation, allocating all weight to the maximum posterior (MAP) estimate, in contrast to the soft-clustering version characteristic of the conventional EM algorithm. In practice, this version of the algorithm converges more rapidly than standard EM.

2.6 Extending the modeling with deep learning approaches

In the last few decades, deep learning emerged as a successful subfield of machine learning able to tackle intricate challenges of very different nature. This section delves into two main topics: the use of deep learning techniques for mixture models and dynamic systems.

2.6.1 Deep learning for mixture models

The use of deep neural networks in the context of model-based clustering has become a subject only very recently. With a broaden look, the first method who combine deep learning with latent variable models was proposed by Tang et al. (2012) for multiple factor analysis (MFA).
Certainly the introduction of variational auto encoders (VAE) from Kingma and Welling (2013) opened the way to the use of deep neural networks (DNN) for

clustering. VAEs can be viewed as parametric functions to encode data into a latent representation. In the article, the authors introduce a reparameterization for the variational lower bound, yielding a differentiable stochastic gradient variational bayes estimator. Particularly, the Auto-Encoding VB (AEVB) algorithm is proposed for cases with i.i.d. datasets and continuous latent variables per data point. This approach efficiently learns model parameters without costly iterative inference methods. The learned inference model holds potential for various tasks, and when coupled with a neural network, it forms the basis of the variational auto-encoder (VAE). In practice, VAE can be seen as non-linear extension of the well-known probabilistic PCA (Tipping and Bishop, 1999). The two neural networks used in the VAE are denoted as encoder and decoder, respectively. The role of the encoder, is to project the input variable to a latent space. This enables the encoder to generate distinct samples, from the same distribution.Conversely, the decoder performs the inverse operation, mapping from the latent space back to the input space. Since their introduction, the VAEs have been greatly used and extended, in particular, recently, Mehta et al. (2019) presented an extension of the overlapping Stochastic Block Model (Latouche et al., 2011), encoding the node embeddings with a neural networks. Then, Van den Oord and Schrauwen (2014) and Viroli and McLachlan (2019) defined Deep Gaussian Mixture Models (DGMMs). The architecture in the latter includes dimensionality reduction at each layer of the network and provides an EM algorithm for the inference of the parameters. Then, Selosse et al. (2020) showed that is a challenging task to fit those kind of models because some clusters tend to be under represented and the model converge to very different local maxima at every run, leading to different partitions. Recently, Kock et al. (2022) proposed to ease the task with a Bayesian approach accounting for sparsity and variational inference. However, the implementation of both co-clustering and deep learning techniques remains largely under explored in the literature, with only one instance standing out (to the best of our knowledge). Specifically, the work of Xu et al. (2019) pioneers this territory by introducing the concept of deep co-clustering (DeepCC). The core of DeepCC concerns the utilization of a deep autoencoder for effective dimension reduction, coupled with the application of a specialized variant of the Gaussian Mixture Model (GMM) for clustering. Notably, this integration is helped by the incorporation of a mutual information loss mechanism, facilitating the bridge between the optimization of autoencoder and mixture model parameters. This dual-optimization strategy plays a pivotal role in enhancing the convergence of the autoencoder and allows the mixture model to overcome the limitations of the

Expectation-Maximization (EM) algorithm.

It is worth acknowledging that despite the efforts made by Xu et al. (2019), there remains a notable gap in the literature concerning dynamic co-clustering models that exploits the potential of deep learning techniques. Recognizing this gap in the existing scientific literature opens up an exciting avenue for future exploration. In this thesis, we try to fill this gap by using deep learning techniques in our proposed co-clustering approach. Our main goal is to create a model that not only handles dynamic co-clustering but also captures how mixing parameter change over time. Chapters 4 and 5 will dive into the details of this approach.

2.6.2 Deep learning for dynamic systems

The task of estimating hidden states in dynamic systems from real-time data, has been tackled by Kalman thanks to the introduction of the Kalman filter (KF) Kalman (1960). This method introduced a minimum mean-squared error estimator tailored for systems with linear state space models, which has become a cornerstone in discrete-time state estimation. However, the hardest challenge lies in the model-based filters reliance on precise domain comprehension and model assumptions, especially in scenarios marked by uncertainty or non-linearity.
In the context of latent variable models, the state space models, along with the extended Kalman filter (EKF) for parameter estimation was proposed by Xu and Hero (2014) and Zreik et al. (2017). The former introduced a dynamic extension of the Stochastic Block Model (SBM) for social network analysis, considering time-varying edge probabilities. The latter introduced the dynamic random subgraph model (DRSM) building upon the random subgraph model (RSM) Jernite et al. (2014), uncovering clusters within pre-defined subgraphs.

To deal with the challenges coming from long and irregular time series, the state space models have been recently extended using deep learning tools. For instance, Rangapuram et al. (2018) proposed a probabilistic time series forecasting method that jointly make use of a linear state space model and recurrent neural networks and Krishnan et al. (2017) proposed the Deep Markov Model (DMM) where in the Gaussian transition dynamics the mean and covariance matrix are modeled through fully connected neural networks.

In a departure from traditional methods, the rise of deep neural networks introduces a shift towards data-driven approaches for state estimation (LeCun

et al., 2015, Chap. 10). In fact, it has been shown that architectures such as recurrent neural networks (RNNs), and in particular long short-term memory (LSTM, Hochreiter and Schmidhuber, 1997) and gated recurrent units (GRUs, Chung et al., 2014), have demonstrated their power in capturing intricate processes without necessitating explicit domain understanding.
This is showcased by several papers focusing on learning differential equations from data using fully connected or recurrent neural networks (Raissi, 2018; Raissi et al., 2018; Guo et al., 2017). In particular, LSTM has been extensively expanded and applied in the context of dynamic systems, the interested reader can find a survey in Smagulova and James (2019).
We finally cite Kidger (2022) as a new but already standard reference on neural differential equations (NDE). In that PhD thesis, the author explores the links existing between (ordinary, controlled and stochastic) differential equations and several neural network architectures (e.g. residual neural network or recurrent neural network) which are nothing but discretizations of the former. This parallel makes NDEs, which basically are neural networks with continuous depth and an infinite number of layers, particularly suitable to tackle dynamic systems and time series.

Chapter 3

Co-clustering of evolving count matrices with the dynamic latent block model

This chapter introduces the dynamic latent block model (dLBM), a novel co-clustering approach for retrospective analysis in pharmacovigilance. The model detects temporal breaks in adverse drug reaction (ADR) data matrices, using a non-homogeneous Poisson process to capture interactions between rows and columns. By segmenting continuous time, the dLBM generates static matrices for each interval, we employ the SEM-Gibbs for the inference algorithm (Section 2.5.2) and the ICL criterion (Section 2.3.4) for model selection. The dLBM effectively summarizes extensive datasets with an high sparsity rate. Notably, the model identifies expected clusters and uncovers hidden patterns. Therefore, dLBM offers a powerful tool for retrospective analysis and insightful signal detection in pharmacovigilance.

This chapter presents the results of two research studies:

- G. Marchello, A. Fresse, M. Corneli, C. Bouveyron (2022). *Co-clustering of evolving count matrices with the dynamic latent block model: application to* *pharmacovigilance.* Statistics and Computing, 32(3):1–22;

- A. Destere, Giulia Marchello, D. Merino, N. Ben Othman, A. O. Gérard, T.Lavrut, D. Viard, F. Rocher, M. Corneli, C. Bouveyron, M. D. Drici (2023). *An artificial intelligence algorithm for co-clustering to help in Pharmacovigilance with a focus on COVID-19 vaccines.* Accepted for publication in the British Journal of Clinical Pharmacology.

3.1 Introduction

In Chapters 1 and 2 we discussed the lack of methods to perform dynamic co-clustering.
While some noteworthy approaches have been introduced, particularly in the context of metric-based models such as the dynamic spectral co-clustering method by Green et al. (2011) and co-clustering techniques designed for handling data streams based on non-negative matrix factorization as explored by Sang and Sun (2014), there remains a need for model-based dynamic co-clustering methods.
Also the increasing need to effectively handle count data, such as interaction data, has become more pronounced in various domains, including healthcare and pharmacovigilance. For instance, in pharmacovigilance, the collection of adverse drug reaction (ADR) reports is a prime example of count data. Healthcare systems and regulatory agencies continually receive reports of adverse events associated with medications. These reports represent counts of specific adverse reactions linked to particular drugs. Analyzing this count data is crucial to identify trends, unusual patterns, or potential safety concerns associated with medications. By quantifying and analyzing these occurrences, pharmacovigilance professionals can make informed decisions about drug safety, regulatory actions, and public health. In the context of applying co-clustering methods to pharmacovigilance, an important contribution was proposed by Robert et al. (2015). In this particular research, the authors introduced a significant advancement known as the multiple latent block model (MLBM). They extended the latent block model (Govaert and Nadif, 2008) by creating a row partition and two separate column partitions. These partitions were constructed based on two binary matrices: one representing the relationships between individuals and medical products, and the other capturing the relationships between individuals and ADRs.

3.1.1 Contributions of this work

This work introduces a generative co-clustering model, named the dynamic latent block model (dLBM), allowing in turn the detection of temporal breaks in the signals, as a retrospective tool for pharmacovigilance. We consider ADR count data matrices evolving over a time period $[0, T]$, whose number of rows and columns are fixed. We assume that the number of interactions occurring in time between rows and columns follows a non-homogeneous Poisson process

(NHPP) (Corneli et al., 2016; Matias et al., 2018). To handle the dynamic framework, we led a segmentation over the continuous time such that we obtain as many static matrices as the number of the identified time intervals. As inference procedure, we use the SEM-Gibbs algorithm (Section 2.5.2) while the ICL criterion (Section 2.3.4) is adopted for selecting the optimal number of clusters. Thus, dLBM provides a meaningful summary of massive datasets, possibly with a large number of missing data. To demonstrate the interest in pharmacovigilance, we run a large-scale retrospective experiment on an 10-year ADRs dataset from Regional Center of Pharmacovigilance (RCPV), located in the University Hospital of Nice (France). The interest of this application lies not only in summarizing the massive amount of drug adverse reactions (ADRs) data but also in identifying possible unexpected phenomena, such as atypical side effects of certain types of drugs. Indeed, dLBM was not only able to identify clusters that are coherent with retrospective knowledge, such as the Lévothyrox® and Mirena® crises, but also to detect an under-notification of bleeding ADRs during the Lévothyrox® crisis, the health professionals were unaware of.

3.1.2 Organization of the chapter

This chapter is organized as follows. Section 3.2 introduce the generative model of dLBM. In Section 3.3, the inference procedure is detailed and the model selection criterion is discussed. Section 3.4 presents various experiments on simulated data to test and evaluate the model performances. In Section 3.5, an application on a real ADRs dataset is presented to illustrate the potential of dLBM in pharmacovigilance. Section 3.6 provides some concluding remarks.

3.2 The dynamic latent block model

In this section, we introduce the dynamic latent block model (dLBM). For a review of the original LBM please refer to Section 2.3.1. The main goal of this time-dependent extension is the simultaneous clustering of rows and columns of high-dimensional sparse matrices in a dynamic time framework. The data we consider are organized such that the rows (drugs in pharmacovigilance application) are indexed by $i = 1, \dots N$ and the columns (adversarial effects) by $j = 1, \dots, M$. Moreover, we consider a fixed time period $[0, T]$ during which the total number of rows, N, and columns, M, is fixed. We indicate as $\mathcal{X}(t)$ the $N \times M$ matrix that contains the number of interactions occurring between the individual i and the item j at time $t \in [0, T]$.

3.2.1 Modeling the dynamic framework

Let us consider the time dimension such that $\mathcal{X}_{ij}$ is time dependent. Thus, $\mathcal{X}_{ij}(t)$, $t \in [0, T]$, represents the cumulative number of interactions at time t between i and j.
A possible approach for the dynamic modeling of such data relies on non-homogeneous Poisson processes (NHPPs), thus assuming that $\{\mathcal{X}_{ij}(.)\}_{i,j}$ are independent point processes, with instantaneous intensity functions $\Lambda_{ij}(t)$:

$$\mathcal{X}_{ij}(t) \sim \mathcal{P}\left(\int_0^t \Lambda_{ij}(u)du\right), \tag{3.1}$$

where $\mathcal{P}(\Lambda)$ denotes the Poisson probability mass function of parameter Λ. With the notation adopted so far, we thus assume the existence of $N \times M$ independent Poisson processes.
In order to cluster both the rows and the columns, we further assume that the intensity function $\Lambda_{ij}(t)$ only depends on the respective clusters of row i and column j:

$$\mathcal{X}_{ij}(t) \mid z_{iq}w_{j\ell} = 1 \sim \mathcal{P}\left(\int_0^t \Lambda_{q\ell}(u)du\right).$$

For further use, let us introduce the following time varying parameter $\boldsymbol{\Lambda} := (\Lambda_{q\ell}(t))_{q \leqslant Q; \ell \leqslant L}$. Given the above assumptions, the conditional distribution of the number of interactions between i and j, over the time interval $[s, t]$, where $0 \leqslant s \leqslant t \leqslant T$, is:

$$p(\mathcal{X}_{ij}(t) - \mathcal{X}_{ij}(s) \mid z_{iq}w_{j\ell} = 1, \Lambda) = \frac{(\int_s^t \Lambda_{q\ell}(v)dv)^{\mathcal{X}_{ij}(t) - \mathcal{X}_{ij}(s)}}{(\mathcal{X}_{ij}(t) - \mathcal{X}_{ij}(s))!} \exp\left(-\int_s^t \Lambda_{q\ell}(v)dv\right). \tag{3.2}$$

A discrete time version

In order to ease the understanding of the dynamic model and to make the inference tractable and computationally efficient, we also operate clustering over the time dimension. Let us first introduce a discretization of the considered time interval $[0, T]$. Thus, without loss of generality the following partition of $[0, T]$ is introduced:

$$0 = t_0 < t_1 < \cdots < t_U = T, \tag{3.3}$$

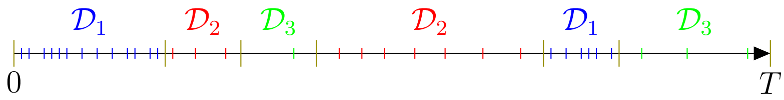


Figure 3.1: Time clusters.

where the T intervals, $I_u = [t_{u-1}, t_u[$, will also be clustered. The number of interactions between i and j on the time interval I_u can be therefore summarized by X_{iju}:

$$X_{iju} := \mathcal{X}_{ij}(t_u) - \mathcal{X}_{ij}(t_{u-1}), \qquad \forall(i,j,u),$$

where we recall that $\mathcal{X}_{ij}(t_u)$ represents the cumulative number of interactions at time t_u between i and j. Hence, we introduce the tensor $X := \{X_{iju}\}_{i\in 1,\dots,N, j\in 1,\dots,M, u\in 1,\dots T}$, with dimensions $N \times M \times T$, that contains the number of interactions between any observation and feature pair at any given time interval. We can also see X as a time series (along the third dimension) of incidence matrices.

Since our goal is to perform clustering over the time dimension as well, each time interval $I_1, \dots, I_U$ is also assumed to be assigned to a hidden time cluster $\mathcal{D}_1, \dots, \mathcal{D}_c$. To model the membership to time clusters, a new latent variable S is introduced, such that $s_u = c$ if the time interval I_u belongs to the time cluster $\mathcal{D}_c$. As it is shown in Figure 3.1, it is worth noticing that a specific time cluster can occur more than once in the temporal line when a similar interactivity pattern is repeated in time. Furthermore, as for Z and W, we assume that S follows a multinomial distribution:

$$p(S \mid \gamma) = \prod_{c=1}^{C} \gamma_c^{|\mathcal{D}_c|}, \tag{3.4}$$

where $\gamma_c = \mathbb{P}\{s_{uc} = 1\}$; $\sum_{c=1}^{C} \gamma_c = 1$ and $\mid \mathcal{D}_c \mid$ represents the number of time intervals in the cluster $\mathcal{D}_c$.

Once these additional assumptions have been made, we can rewrite Eq. (3.1) considering that the intensity functions are stepwise constant on each time cluster $\mathcal{D}_c$. Thus:

$$X_{iju} | z_{iq} w_{j\ell} s_{uc} = 1 \sim \mathcal{P}(\Lambda_{q\ell c} \Delta_u), \tag{3.5}$$

where Δ_u indicates the length of the interval I_u. Henceforth, in order to simplify the exposition, we assume that Δ_u is constant, $\Delta_u = \Delta$. We can finally set $\Delta = 1$ without loss of generality. A graphical representation of dLBM can be seen in Figure 3.2.

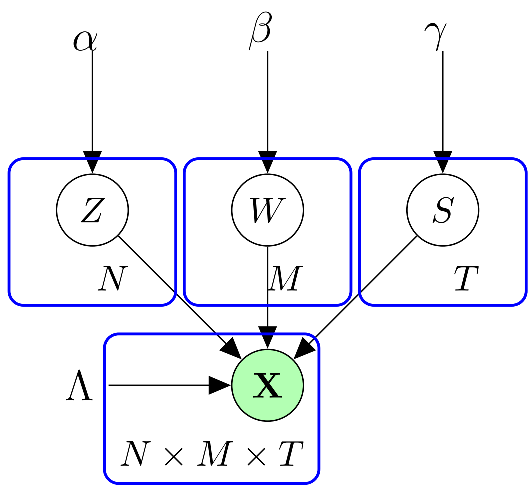


Figure 3.2: Graphical representation of dLBM.

From Eqs. (3.1)-(3.5), it holds that:

$$p(X_{iju} \mid z_{iq} w_{j\ell} s_{uc} = 1, \Lambda_{q\ell c}) = \left(\frac{(\Lambda_{q\ell c})^{X_{iju}}}{X_{iju}!} \exp\left(-\Lambda_{q\ell c}\right) \right). \tag{3.6}$$

Therefore, we can introduce the $Q \times L \times C$ tensor Λ, whose elements are denoted by $\Lambda_{q\ell c}$.

It is now possible to write the complete data likelihood of the model:

$$p(X, Z, W, S|\alpha, \beta, \gamma, \Lambda) = p(Z|\alpha)p(W|\beta)p(S|\gamma)p(X|Z, W, S, \Lambda), \tag{3.7}$$

where $p(Z|\alpha)$, $p(W|\beta)$ and $p(S|\gamma)$ were defined in the previous section. The conditional distribution of X, given Z, W, and S, can be easily obtained from Eq. (3.6) by independence:

$$p(X|Z, W, S, \Lambda) = \prod_{q,\ell,c} \left(\frac{(\Lambda_{q\ell c})^{R_{q\ell c}}}{P_{q\ell c}} \exp\left(- \mid \mathcal{A}_q \mid\mid \mathcal{B}_\ell \mid\mid \mathcal{D}_c \mid \Lambda_{q\ell c}\right) \right), \tag{3.8}$$

where $R_{q\ell c} = \sum_{i=1}^{N} \sum_{j=1}^{M} \sum_{u=1}^{T} z_{iq} w_{j\ell} s_{uc} X_{iju}$ and $P_{q\ell c} = \prod_{i=1}^{N} \prod_{j=1}^{M} \prod_{u=1}^{T} (z_{iq} w_{j\ell} s_{uc} X_{iju})!$. Denoting by θ the set of all model parameters, $\theta = (\alpha, \beta, \gamma, \Lambda)$, the log-likelihood can be finally written as:

$$\ell(\theta; X) = \sum_{Z} \sum_{W} \sum_{S} \log p(X, Z, W, S \mid \theta). \tag{3.9}$$

3.2.2 Link with related models

At this point, dLBM can be related with the following models:

- If we do not take into account the time dependency, assuming that the time period is restricted to a single time point t_0, dLBM coincides with the Poisson LBM.

- dLBM reduces to dSBM (Corneli et al., 2016) if the row individuals are the same as the column. In that case, in fact, Z would be equal to W and, consequently, $Q = L$. Therefore, $X_{\cdot,\cdot,u}$ passes from an incidence matrix to an adjacency matrix.

- If $C = T$, dLBM corresponds to TensorLBM (Boutalbi et al., 2020), where the third dimension is considered but the slices are not clustered. In fact, when the counting contingency table case is analyzed, the authors consider a Poisson TensorLBM, where for each slice a, the entries are distributed according to a $\mathcal{P}(\Lambda_{ij}^{a})$.

3.3 Inference algorithm and model selection

3.3.1 Inference

For the co-clustering model based approach outlined in the previous section, as well as for standard mixture models, a direct maximization of the log-likelihood with respect to the model parameters is not feasible (see discussion in Chapter 2.5).

Moreover, in the context of the present work the joint posterior distribution $p(Z, W, S \mid X, \theta)$ is not computationally tractable as well. To go through this limitation, we propose to approximate it through a Gibbs sampler within the E-step. Such an approach was proposed by Keribin et al. (2010) and exploited, for instance, by Bouveyron et al. (2018) for the functional latent block model (funLBM). The resulting SEM-Gibbs algorithm (see Chapter 2.5.2), starts with some initial values of the parameter $\theta^{(0)}$, the column clusters $W^{(0)}$ and the time clusters $S^{(0)}$. In this way, at the h^{th} iteration the algorithm alternates the following SE step and M step:

SE step: a partition for Z, W and S is drawn according to $q^*(Z, W, S) = p(Z, W, S \mid X, \theta)$, which is approximated by making use of a Gibbs sampler, using the current values of the parameter set θ. In this way, the unknown labels are simulated from their posterior distribution, given the observed data and the parameter set. It consists in executing a small number of iterations (usually 5 is enough) of the following three steps:

1) Generate the row partition $z_i^{(h+1)} = (z_{i1}^{(h+1)}, \ldots, z_{iQ}^{(h+1)})|X, W^{(h)}, S^{(h)}$ according to $z_i^{(h+1)} \sim \mathcal{M}(1,\widetilde{z}_{i1}, \ldots, \widetilde{z}_{iQ})$, for all $1 \leq i \leq N$ and $1 \leq q \leq Q$, where:

$$\widetilde{z}_{iq} = p(z_{iq} = 1 \mid X, W^{(h)}, S^{(h)}; \theta^{(h)}) = \frac{\alpha_q^{(h)} f_q(X_i \mid W^{(h)}, S^{(h)}; \theta^{(h)})}{\sum\limits_{q'} \alpha_{q'}^{(h)} f_{q'}(X_i|W^{(h)}, S^{(h)}; \theta^{(h)})},$$

where $X_i = (X_{iju})_{ju}$ and $f_q(X_i \mid W^{(h)}, S^{(h)}; \theta^{(h)}) = \prod_{j\ell} \prod_{uc} p(X_{iju}; \theta_{q\ell c}^{(h)})^{w_{j\ell}^{(h)} s_{uc}^{(h)}}$.

2) Generate the column partition $w_j^{(h+1)} = (w_{j1}^{(h+1)}, \ldots, w_{jL}^{(h+1)})|X, Z^{(h+1)}, S^{(h)}$ according to $w_j^{(h+1)} \sim \mathcal{M}(1,\widetilde{w}_{j1}, \ldots, \widetilde{w}_{jL})$, for all $1 \leq j \leq M$ and $1 \leq \ell \leq L$, where:

$$\widetilde{w}_{j\ell} = p(w_{j\ell} = 1|\ X, Z^{(h+1)}, S^{(h)}; \theta^{(h)}) = \frac{\beta_\ell^{(h)} f_\ell(X_j|Z^{(h+1)}, S^{(h)}; \theta^{(h)})}{\sum\limits_{\ell'} \beta_{\ell'} f_{\ell'}(X_j|Z^{(h+1)}, S^{(h)}; \theta^{(h)})},$$

where $X_j = (X_{iju})_{iu}$ and $f_\ell(X_j|Z^{(h+1)}, S^{(h)}; \theta^{(h)}) = \prod_{iq} \prod_{uc} p(X; \theta_{q\ell c}^{(h)})^{z_{iq}^{(h+1)} s_{uc}^{(h)}}$.

3) Generate the time cluster partition $s_u^{(h+1)} = (s_{u1}^{(h+1)}, \ldots, s_{uC}^{(h+1)})|X, Z^{(h+1)}, W^{(h+1)}$ according to $s_u^{(h+1)} \sim \mathcal{M}(1,\widetilde{s}_{u1}, \ldots, \widetilde{s}_{uC})$, for all $1 \leq u \leq T$ and $1 \leq c \leq C$, where:

$$\widetilde{s}_{uc} = p(s_{uc} = 1|X, Z^{(h+1)}, W^{(h+1)}; \theta^{(h)}) = \frac{\gamma_c^{(h)} f_c(X_u|Z^{(h+1)}, W^{(h+1)}; \theta^{(h)})}{\sum\limits_{c'} \gamma_{c'} f_{c'}(X_u|Z^{(h+1)}, W^{(h+1)}; \theta^{(h)})},$$

where $X_u = (X_{iju})_{ij}$ and $f_c(X_u|Z^{(h+1)}, W^{(h+1)}; \theta^{(h)}) = \prod_{iq} \prod_{j\ell} p(X_{iju}; \theta_{q\ell c}^{(h)})^{z_{iq}^{(h+1)} w_{j\ell}^{(h+1)}}$.

M step: in this step, $\mathcal{L}(q^*(Z, W, S); \theta^{(h)})$ is maximized with respect to θ, where:

$$\begin{aligned}\mathcal{L}(q^*(Z, W, S); \theta^{(h)}) &\simeq \sum_{Z,W,S} p(Z, W, S \mid X, \theta^{(h)}) \log \frac{p(X, Z, W, S \mid \theta)}{p(Z, W, S \mid X, \theta^{(h)})} \\ &\simeq E[\log(p(X, Z^{(h+1)}, W^{(h+1)}, S^{(h+1)} \mid \theta)) \mid X, \theta^{(h)}] + \xi,\end{aligned}$$

where ξ is a constant term related to θ^h. This conditional expectation of the complete data log-likelihood can be written in a developed form as follows:

$$\begin{aligned}E[\log(p(X, Z^{(h+1)}, W^{(h+1)}, S^{(h+1)} \mid \theta) \mid X, \theta^{(h)})] &= \sum_{i,q} z_{iq}^{(h+1)} \log \alpha_q + \sum_{j,\ell} w_{j\ell}^{(h+1)} \log \beta_\ell + \\ + \sum_{u,c} s_{u,c}^{(h+1)} \log \gamma_c &+ \sum_{i,q} \sum_{j,\ell} \sum_{u,c} z_{iq}^{(h+1)} w_{j\ell}^{(h+1)} s_{uc}^{(h+1)} \log(p(X_{iju} \mid \theta_{q\ell c})).\end{aligned} \tag{3.10}$$

The last term of the previous equation can be further developed as:

$$\begin{aligned}
&\sum_{i,q}\sum_{j,\ell}\sum_{u,c} z_{iq}^{(h+1)} w_{j\ell}^{(h+1)} s_{uc}^{(h+1)} \log(p(X_{iju} \mid \theta_{q\ell c})) = \\
&\sum_{i,q}\sum_{j,\ell}\sum_{u,c} z_{iq}^{(h+1)} w_{j\ell}^{(h+1)} s_{uc}^{(h+1)} \log\left[\frac{(\Lambda_{q\ell c}^{(h)})^{X_{iju}}}{X_{iju}!}\exp(-\Lambda_{q\ell c}^{(h)})\right] = \\
&\sum_{i,q}\sum_{j,\ell}\sum_{u,c} z_{iq}^{(h+1)} w_{j\ell}^{(h+1)} s_{uc}^{(h+1)} \left[X_{iju}\log(\Lambda_{q\ell c}^{(h)}) - \log(X_{iju}!) - \Lambda_{q\ell c}^{(h)}\right].
\end{aligned} \tag{3.11}$$

Thanks to the previous equation, the parameter set $\theta^{(h+1)}$ can be estimated. The mixture proportions are updated as follows (proof in Appendix A.1):

$$\alpha_q^{(h+1)} = \frac{1}{N}\sum_i z_{iq}^{(h+1)}, \quad \beta_\ell^{(h+1)} = \frac{1}{M}\sum_j w_{j\ell}^{(h+1)}, \quad \gamma_c^{(h+1)} = \frac{1}{T}\sum_u s_{uc}^{(h+1)}.$$

Moreover, the ML estimator of $\Lambda_{q\ell c}$ is as follows (proof in Appendix A.2):

$$\Lambda_{q\ell c}^{(h+1)} = \frac{R_{q\ell c}^{(h+1)}}{|\mathcal{A}_q||\mathcal{B}_\ell||\mathcal{D}_c|}, \qquad \forall(q,\ell,c).$$

Since assessing the convergence in stochastic inference algorithms is a challenging task, the algorithm runs for a certain number of iterations of the two steps and, to assess that convergence has been reached, we check both that the log-likelihood reached a plateau and that the values assumed by the model parameter values are stabilized during the iterations of the algorithm. We can obtain the final estimation of the parameter set $\widehat{\theta}$ by computing the mean from the sampled distribution, after the burn-in period.Finally, the optimal values for Z, W and S are estimated by the mode of their sampled distributions.

3.3.2 Model selection

Up to now, we have assumed that the number of row clusters (Q), column clusters (L) and time clusters (C) was known. However, for real datasets, this assumption is of course unrealistic. For this reason, our purpose in this section is to define a model selection criterion that can automatically identify the optimal number of clusters that are appropriate for the data at hand. The model selection approach is considered. We propose to rely on ICL (Integrated Completed Likelihood, (Biernacki et al., 2000)) to approximate the complete-data integrated log-likelihood. Hence, we derived the formulation of the ICL criterion for the model proposed above:

$$\begin{aligned} ICL(Q, L, C) = \log p(X, \hat{Z}, \hat{W}, \hat{S}; \hat{\theta}) - \frac{Q-1}{2} \log N + \\ -\frac{L-1}{2} \log M - \frac{C-1}{2} \log T - \frac{QLC}{2} \log(NPT) \end{aligned} \tag{3.12}$$

The triplet $(\hat{Q}, \hat{L}, \hat{C})$ that leads to the highest value for the ICL is considered as the most likely for those data. Alternative approaches may rely on greedy searches (Côme and Latouche, 2015; Keribin et al., 2017) which can be seen as less computationally expensive, but which operate sequentially and, therefore, cannot be parallelized.

3.4 Numerical experiments

The main purpose of this section is to highlight the most important features of dLBM over simulated datasets. We aim at demonstrating the validity of the inference algorithm and model selection criterion, presented in the previous sections. Regarding the initialization of the algorithm, in all the reported experiments the partitions of rows, columns and slices were initialized with a k-means applied to the rows, columns or slices of the proper unfolding of tensor X. The first experiment consists in applying dLBM to an easy scenario to explain its main outputs. Then, the second example shows a model selection application on 25 simulated datesets. In the third experiment, we compare the performances of dLBM with some state-of-the-art methods in three simulated scenarios. In the fourth experiment, we compare the performances of dLBM with the same state-of-the-art methods of the previous experiment, on a simulation setup that differs from our model assumptions.

3.4.1 Introductory example

As a first example, we simulate a dataset with $Q = 3$ groups of rows, $L = 2$ groups of columns and $C = 2$ groups of time clusters, with a level of sparsity $\tau = 0.97$. Table 3.1 shows the main features of this dataset.

We fit dLBM to the simulated dataset with the actual values for Q, L and C to show the ability of the model to fully recover the model parameters. Figure 3.3

N	M	T	τ	Q	L	C	α	β	γ
200	200	150	0.97	3	2	2	(0.15, 0.35, 0.55)	(0.2, 0.8)	(0.6, 0.4)

Table 3.1: Parameter values for the first simulated dataset.

shows the evolution of the complete data log-likelihood. As we can see the convergence is reached in less than 10 iterations. Figure 3.4 shows the evolution of the estimated mixture parameters $\hat{\alpha}$, $\hat{\beta}$ and $\hat{\gamma}$ along the iterations of the SEM-Gibbs algorithm. Comparing the values reached by each line with the actual values of the model parameters showed in Table 3.1, we can observe that dLBM fully recovers the original values in few iterations. Moreover, Figure 3.5 shows a bar plot of the number of interactions between rows and columns of the array X for each time period, where the two different time clusters are identified by different colors. We can easily deduce that dLBM selects the two time clusters in a meaningful way in terms of level of counted interactions in each time cluster.
Figure 3.6 shows the value of the estimated intensity parameter $\hat{\Lambda}$ for each cluster of rows and columns where different colors represent different time clusters. For instance, the algorithm detects that, in the first cluster of rows and the first cluster of columns (Block(1,1)), there is an high intensity of interactions in both of the time clusters. Figure 3.7 and Figure 3.8, display the data structure before running dLBM and the reorganized incidence matrices one for each time cluster. To this end, rows and columns of the incidence matrix are permuted, thanks to the estimates $\hat{Z}$ and $\hat{W}$, in such a way that nearby rows (columns) belong to the same cluster of rows (columns). The blocks are also delimited by black dashed lines.
Finally, to evaluate the performance of the model in identifying the correct rows, columns and times partitions, we use the adjusted Rand index (ARI) (Rand, 1971) for all of the three variables. The adjusted Rand index, from a mathematical point of view, is closely related to the accuracy measure, however it is a commonly used method for evaluating clustering problems since it can be applied for measuring the similarity between two partitions even with different number of clusters and it is invariant to label switching. The closer the index is to 1, the more two label vectors are similar to each other. We compared the original matrices Z, W and S, with the estimates $\hat{Z}$, $\hat{W}$ and $\hat{S}$ given by the output of the dLBM. The model obtained an ARI index of 1 for rows, columns and times partitions. Thus, we can conclude that our algorithm perfectly identifies the composition of the original clusters.

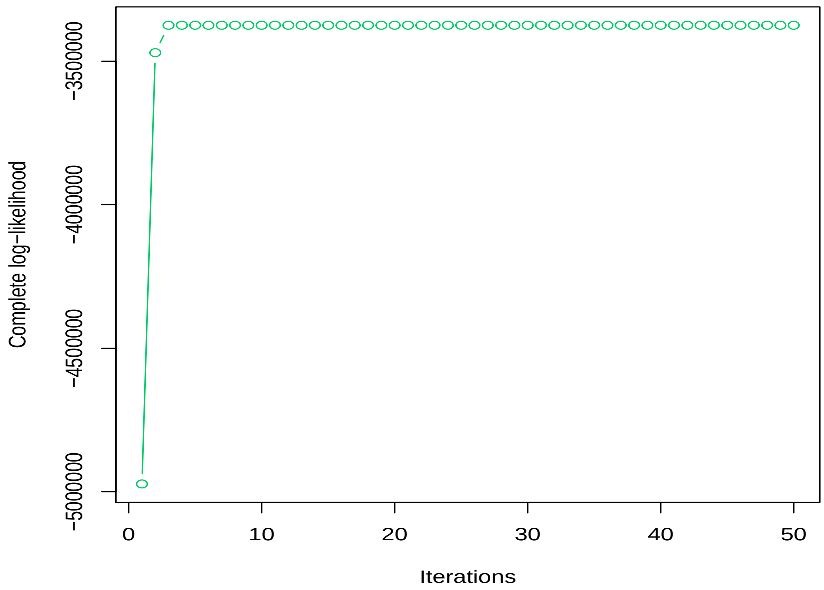


Figure 3.3: Complete data log-likelihood over the iterations of the SEM-Gibbs algorithm.

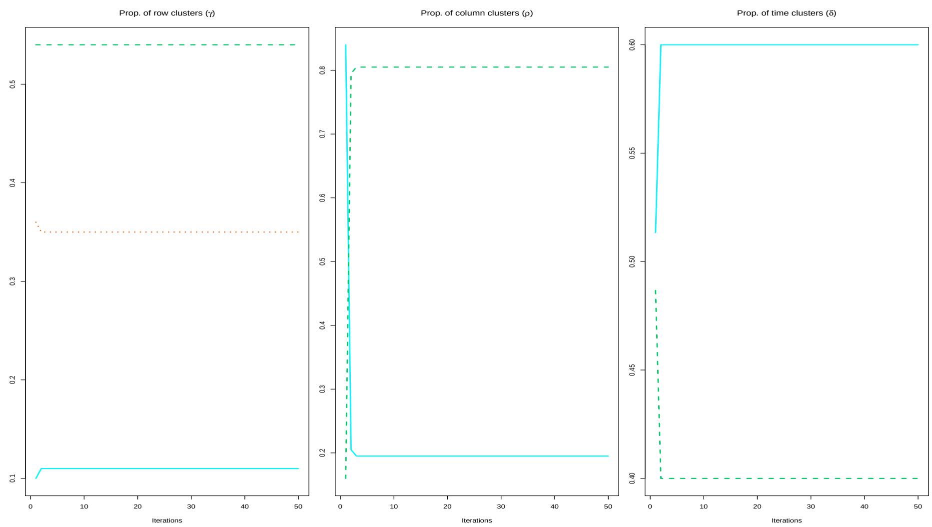


Figure 3.4: Estimates of the mixture parameters in the first simulated dataset.

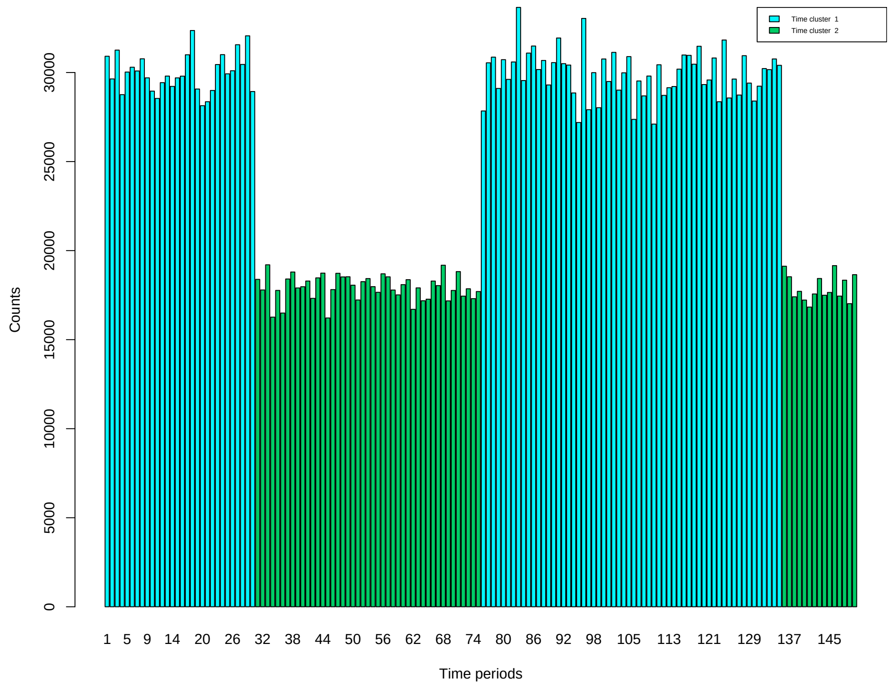


Figure 3.5: Time periods representation of the first simulated dataset, where different colors correspond to different time clusters.

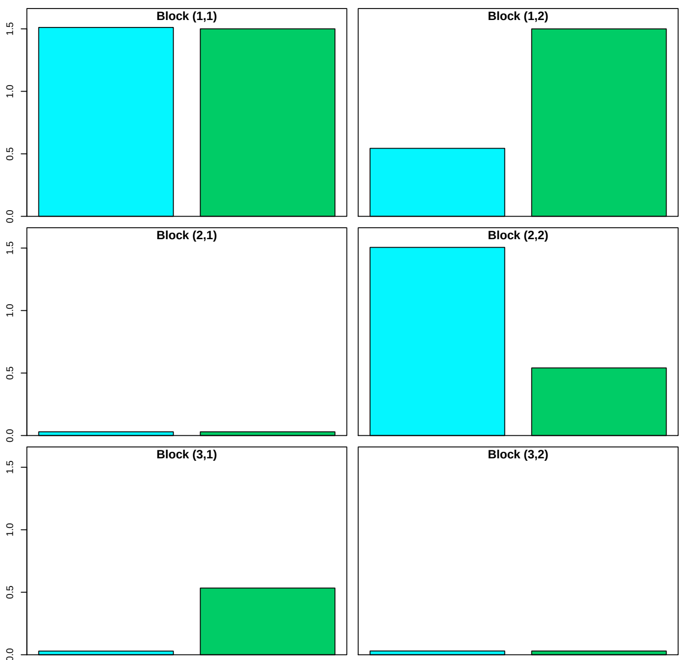


Figure 3.6: Estimated Λ values for each cluster of rows and columns.

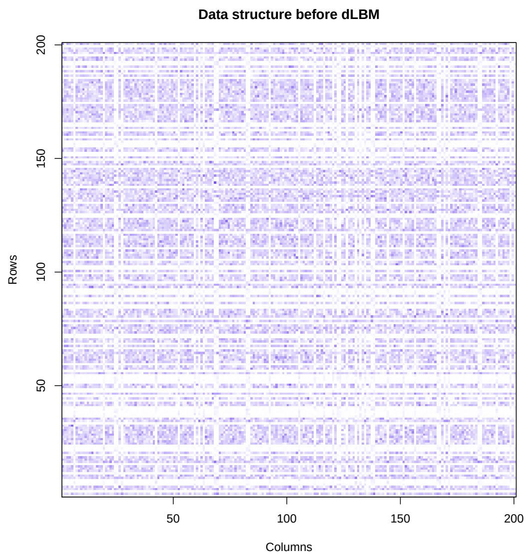


Figure 3.7: Unordered data structure before running dLBM.

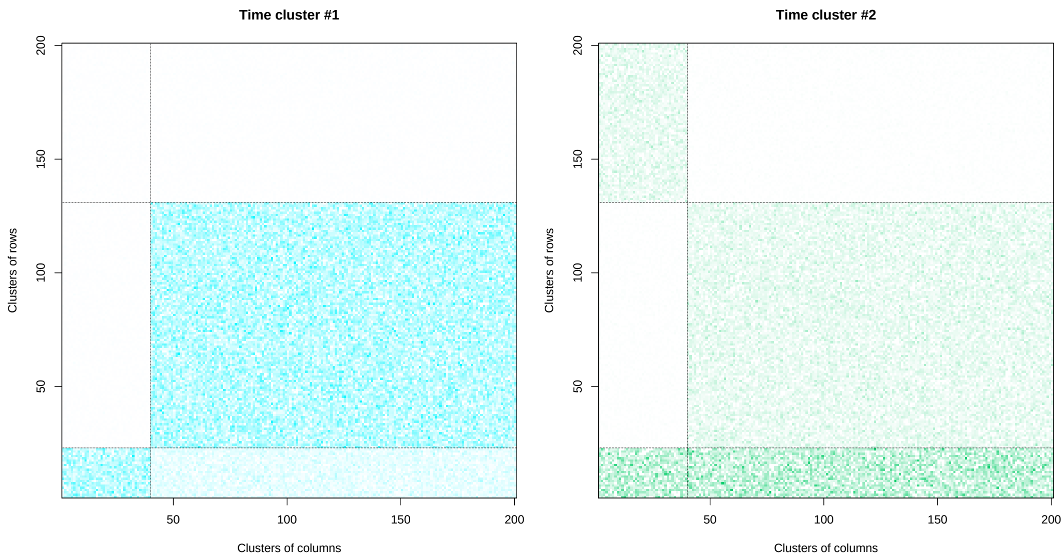


Figure 3.8: Reorganized incidence matrices, one for each time cluster, according to the estimates $\hat{Z}$ and $\hat{W}$. Nearby rows (columns) belong to the same cluster of rows (columns). The blocks are also delimited by black dashed lines.

3.4.2 Model selection experiment

In the previous experiment, we assumed to know the value of Q, L and C. In this section we aim at validating the ICL criterion for model selection. To do that, 25 independent datasets are generated with the setup indicated in Table 3.2. dLBM is applied on those simulated datasets for values of Q, L and C ranging from 1 to 6. The results are sorted according to the ICL values. Table 4.3 shows the percentage of selections by ICL criterion on 25 simulated datasets. The highlighted cell corresponds to the actual value of Q, L and C. ICL succeeds 64% of the time to identify the correct model. It is worth to notice that, when ICL does not select the right combination of Q, L and C, the wrongly selected models are close to the simulated one. In particular, 28% of the selections only differ from the actual model by one cluster, on one of the three cluster dimensions.

N	M	T	τ	Q	L	C	α	β	γ
200	200	200	0.97	4	3	3	(0.2,0.4,0.1,0.3)	(0.4,0.3,0.3)	(0.25,0.3,0.45)

Table 3.2: Parameter values for the second simulated dataset.

C = 3

Q/L	1	2	3	4	5	6
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	0	0	0	0	0	0
4	0	0	64	8	0	0
5	0	0	12	0	4	0
6	0	0	0	0	0	0

C = 4

Q/L	1	2	3	4	5	6
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	0	0	0	0	0	0
4	0	0	8	0	0	0
5	0	0	0	4	0	0
6	0	0	0	0	0	0

Table 3.3: Model selection. Percentage of selections by ICL criterion on 25 simulated datasets. The highlighted cell corresponds to the actual value of Q, L and C.

Scenario	N	M	T	τ	Q	L	C	α	β	γ	Λ
A - Easy	250	250	100	0.97	3	2	2	0.15,0.35,0.55	0.2,0.8	0.6, 0.4	Λ_A
B - Medium											Λ_B
C - Hard					4	3	3	0.2,0.4,0.1,0.3	0.4, 0.3, 0.3	0.25, 0.3, 0.45	Λ_C
D - Row_LBM	100	150	50	1	1	3	2	1	0.33, 0.33, 0.33	0.5, 0.5	Λ_D

Table 3.4: Parameter values for the four simulation senarios (see Appendix A.3 for details about $\Lambda_A, \Lambda_B, \Lambda_C$ and Λ_D).

3.4.3 Benchmark study

The goal of this third experiment is to compare dLBM with some state-of-the-art methods in terms of recovering the data structure. dLBM is compared with TensorLBM (Boutalbi et al., 2020) where, in absence of the original code, we set the number of time clusters of dLBM equal to the number of time intervals, $C = T$, and with the Poisson LBM by making use of the `bikm1` package (Robert et al., 2020). Since LBM supports only two dimensions, we shrink the third dimension summing up alternatively on rows, columns and slices, obtaining respectively the Row_LBM, Col_LBM and Slice_LBM methods.
We chose to evaluate the results with the ARI index by comparing the resulting cluster partitions with the simulated ones. To make this comparison more complete, we defined four simulation scenarios ("Easy", "Medium", "Hard" and "Row_LBM"), detailed in Table 3.4. In particular, in the "Easy", "Medium" and "Hard" scenarios, the data are simulated according to the dLBM model using different parameters to gradually increase the difficulty. In order to also add a fair comparison with competitors, the "Row_LBM" scenario generates data from the Poisson LBM model (Robert et al., 2020) using the R package bikm1 of the authors. As detailed in Table 3.4, the simulation parameters for this additional simulation scenario are the ones used in the default example of the R package `bikm1`.

Table 3.5 displays the results of this comparison, in terms of average ARI values, reported with standard deviations. The dash indicates that no value is reported because the calculation is not allowed by the model.
In the "Easy" situation, dLBM works perfectly. Also TensorLBM provides excellent results, even though calculated only on rows and columns. Row_LBM and Col_LBM only give good results on one dimension, while Slice_LBM produces extremely low results.
In "Medium" and "Hard" situations, dLBM continues to obtain excellent results, although not perfect, due to the increasing complexity of the proposed situations.

Scenario A - **Easy**			
	ARI_Rows	ARI_Cols	ARI_Slices
dLBM	1 ± 0	1 ± 0	1 ± 0
TensorLBM	0.8 ± 0.3	1 ± 0	-
Row_LBM	-	0.12 ± 0.21	0.96 ± 0.2
Col_LBM	0.1 ± 0.21	-	0.92 ± 0.2
Slice_LBM	0.09 ± 0.2	0.13 ± 0.22	-

Scenario B - **Medium**			
	ARI_Rows	ARI_Cols	ARI_Slices
dLBM	0.89 ± 0.17	1 ± 0	1 ± 0
TensorLBM	0.74 ± 0.18	1 ± 0	-
Row_LBM	-	0.13 ± 0.21	0.12 ± 0.23
Col_LBM	0.09 ± 0.21	-	0.15 ± 0.23
Slice_LBM	0.1 ± 0.2	0.14 ± 0.21	-

Scenario C - **Hard**			
	ARI_Rows	ARI_Cols	ARI_Slices
dLBM	0.79 ± 0.19	0.68 ± 0.22	0.63 ± 0.18
TensorLBM	0.64 ± 0.21	0.71 ± 0.19	-
Row_LBM	-	0.09 ± 0.14	0.09 ± 0.15
Col_LBM	0.12 ± 0.14	-	0.12 ± 0.15
Slice_LBM	0.2 ± 0.18	0.25 ± 0.22	-

Scenario D - **Row_LBM**			
	ARI_Rows	ARI_Cols	ARI_Slices
dLBM	-	0.96 ± 0.13	1 ± 0
TensLBM	-	0.94 ± 0.14	0.85 ± 0.19
Row_LBM	-	1 ± 0	1 ± 0
Col_LBM	-	-	0.94 ± 0.11
Slice_LBM	-	1 ± 0	-

Table 3.5: Co-clustering results for dLBM, TensorLBM, and LBM applied respectively by summing up the rows (Row_LBM), the columns (Col_LBM) and the slices (Slice_LBM) on 25 simulated data according to the four scenarios. Average ARI values are reported with standard deviations.

The three other LBM models perform poorly, while TensorLBM obtains rather high ARI values. Specifically, in the "Hard" situation the ARI value of TensorLBM on the columns is slightly higher than that of dLBM, even if the one calculated on the rows partitions is lower.
Finally, regarding the "Row_LBM" scenario, we can first observe that the results of Row_LBM are perfect, as expected, but dLBM demonstrates also to have very good performances in this less favorable simulation, conversely to the other competitors.

3.4.4 Robustness to model assumptions

The goal of this fourth experiment on simulated data is to test the performance of dLBM and compare it to its competitors when data are not simulated according to the model assumptions. Specifically, we decided to simulate the data from a negative binomial distribution. The negative binomial distribution is a discrete probability distribution that models the number of successes in a series of iid Bernoulli trials before a given number of failures, r. The probability mass function of the negative binomial is given by:

$$f(k,r,p) = \binom{k+r-1}{k} \cdot (1-p)^r \cdot p^k = \frac{\Gamma(k+r)}{k!\Gamma(r)}(1-p)^r p^k, \tag{3.13}$$

where k is the number of successes and p is the probability of success. When modeling counts data the negative binomial distribution is often a valid alternative to Poisson, because it allows the mean and the variance to be different: Mean: $\Lambda = \dfrac{pr}{1-p}$; Variance: $= \dfrac{pr}{(1-p)^2} = \Lambda + \frac{1}{r}\Lambda^2$.
A particular property of the negative binomial distribution is that it converges to the Poisson distribution, with expected value μ, when $r \to \infty$.
To accurately evaluate the performance of dLBM in comparison with its competitors we simulate from the negative binomial distribution 25 datasets for each value of r equal to 0.05, 0.1, 0.5, 0.8, 1, 5 and 10, while keeping the values of Λ unchanged to the ones of Scenario A in Section 3.4.3.
Figures 3.9, 3.10 and 3.11 report the results of the experiment, depicting line plots of the values taken by the ARI index in the partitions of rows, columns, and slices, respectively. The x-axis depicts the values of r based on which the data were simulated, and each color represents a different model. The models considered are the same as those described in Section 3.4.3: Row LBM, ColLBM, SliceLBM, and TensorLBM. The ARI values shown in these graphs refer to the

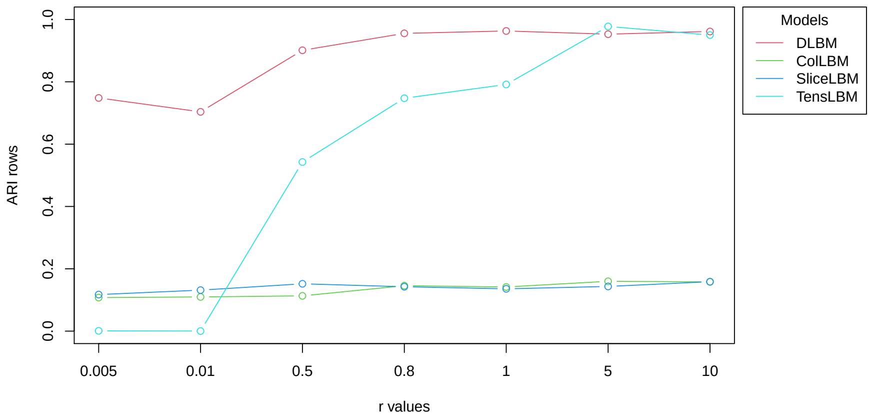


Figure 3.9: Evolution of the row ARI as a function of r for the different methods.

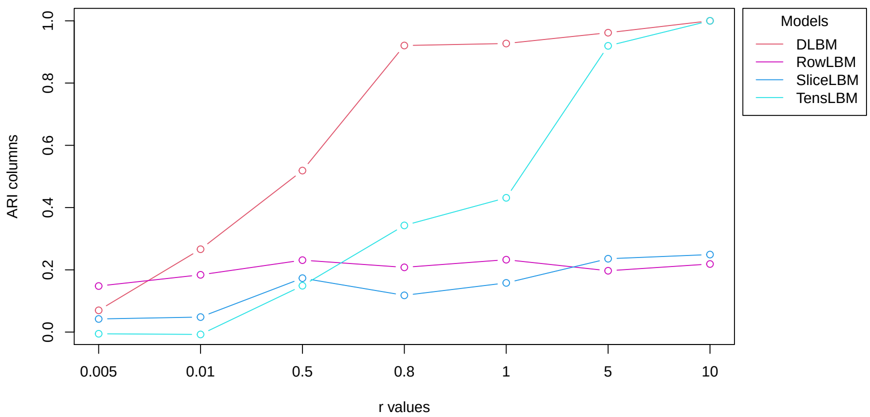


Figure 3.10: Evolution of the column ARI as a function of r for the different methods.

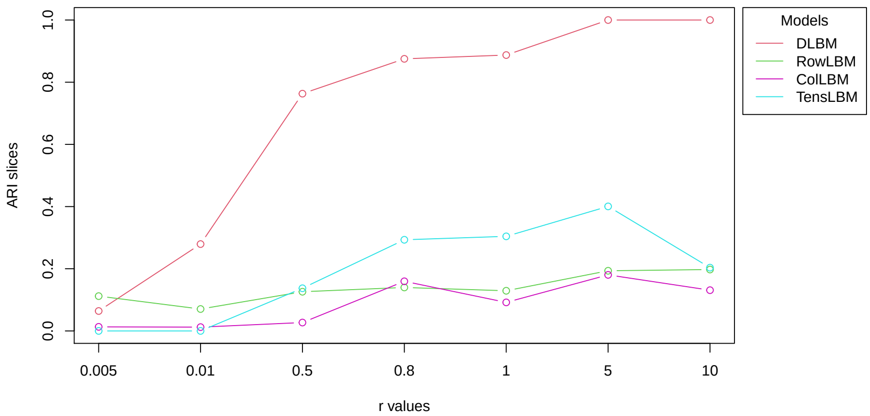


Figure 3.11: Evolution of the slice ARI as a function of r for the different methods.

average values reached by the index during the 25 simulations performed for each r value. Looking at these results we can observe that for values of r very close to zero, all models have difficulties in finding the correct cluster partition. This is due to the fact that when r is small (e.g. $r = 0.05, 0.1$), a negative binomial distribution is more spread than a Poisson distribution with the same mean and therefore the different clusters are extremely difficult to distinguish. For slightly higher values of r (e.g. $r = 0.5, 0.8$) dLBM outperforms its competitors, achieving excellent ARI values. Whereas, for sufficiently large values of r (e.g. $r = 5, 10$), the block distributions start to be enough different and dLBM managed to find the correct cluster partitions, with an ARI very close to 1 in all the dimensions. In this case, in fact, the results tend to be very similar to those of Scenario A in Section 3.4.3. It may be of interest to notice that also TensorLBM managed to have high values of ARI, for row and column partitions, although it needs a higher value of r to perform well. However, looking at the partition of slices we can see that the performance is poor, due to the fact that the assumption TensorLBM relies on is that the number of slice clusters is equal to the number of slices themselves, i.e. $C = T$.

3.5 Analysis of the adverse drug reaction dataset

This section focuses on the application of dLBM to a large-scale pharmacovigilance dataset, with the aim of illustrating the potential of the tool for such studies.

3.5.1 Protocol and data

This section considers a large dataset consisting of ADR data collected by the Regional Center of Pharmacovigilance (RCPV), located in the University Hospital of Nice (France). The center covers an area of over 2.3 million inhabitants and receives notifications about ADRs from different channels: a website[1] form, phone calls, emails, medical visits at the hospital units, etc. A time horizon of 10 years is considered, from January 1^{st}, 2010 to September 30^{th}, 2020, the unity measure for time intervals is a month ($\Delta_u = \Delta = 1$ month). The overall dataset is made of by 44,269 declarations, for which the market name of the drug, the notified ADR, the channel used for the declaration and its origin, as well as an identification number and the reception date are reported. To prevent the same

[1] https://signalement.social-sante.gouv.fr

medicine from being considered more than once if reported under slightly different names, we decided to use the international nonproprietary name (INN) of the drug (to simplify the comprehension, the INNs would be referred as drugs for the rest of the study). Moreover, we only considered molecules and ADRs that were notified more than 10 times over the 10 years. The resulting dataset contains 542 drugs, 586 ADRs and 129 months with 13,363 non-zero entries.
In fact, in that year an unexpected rise of reports for ADRs happened concerning two specific drugs: Mirena® and Lévothyrox®. Mirena® has been available in Europe since 1995. This birth control product contains a hormone called levonorgestrel. In 2017, concerns regarding ADRs associated with the use of levonorgestrel releasing intra-uterine device (IUD) started to grown with a media coverage peak occurred in May 2017, which resulted in a massive wave of ADRs reports from patients to French RCPVs (Langlade et al., 2019).
Also, Lévothyrox® has been marketed in France for about 40 years as a treatment for hypothyroidism and, in 2017, a new formula was introduced on the market. The Lévothyrox® case had an extremely high media coverage in France: the RCPVs received 18,241 reports of Lévothyrox® ADRs in 2017 only. Lévothyrox® spontaneous reports represent almost the 90% of all the spontaneous notifications that the Nice center received from patients in 2017. This phenomenon has been fully described in a recent article of Viard et al. (2019).
From Figure 3.12, one can understand the difficulty to work with such data which contain signals of very different amplitude. Indeed, behind those very visible effects, many ADR signals need to be detected for obvious public health reasons. In particular, those data also contain ADR reports regarding Médiator®, which is here far less visible than Lévothyrox® and Mirena®, but also led to many avoidable serious cardiovascular diseases. This is why, we expect dLBM to be a useful tool to reveal such hidden signals.

3.5.2 Summary of the results

Remembering that our aim is to find an underling latent structure in our dataset by applying co-clustering on the three dimensions of the array X with dimension $542 \times 586 \times 129$, we have run dLBM for different values of Q, L and C. Specifically, we tested rows (here drugs), columns (here ADRs) and times groups ranging from 1 to 12. The ICL criterion identified the optimal values for the triplet $(\hat{Q}, \hat{L}, \hat{C})$ as: $\hat{Q} = 7$, $\hat{L} = 10$, $\hat{C} = 6$, with a running time of about 6 hours.

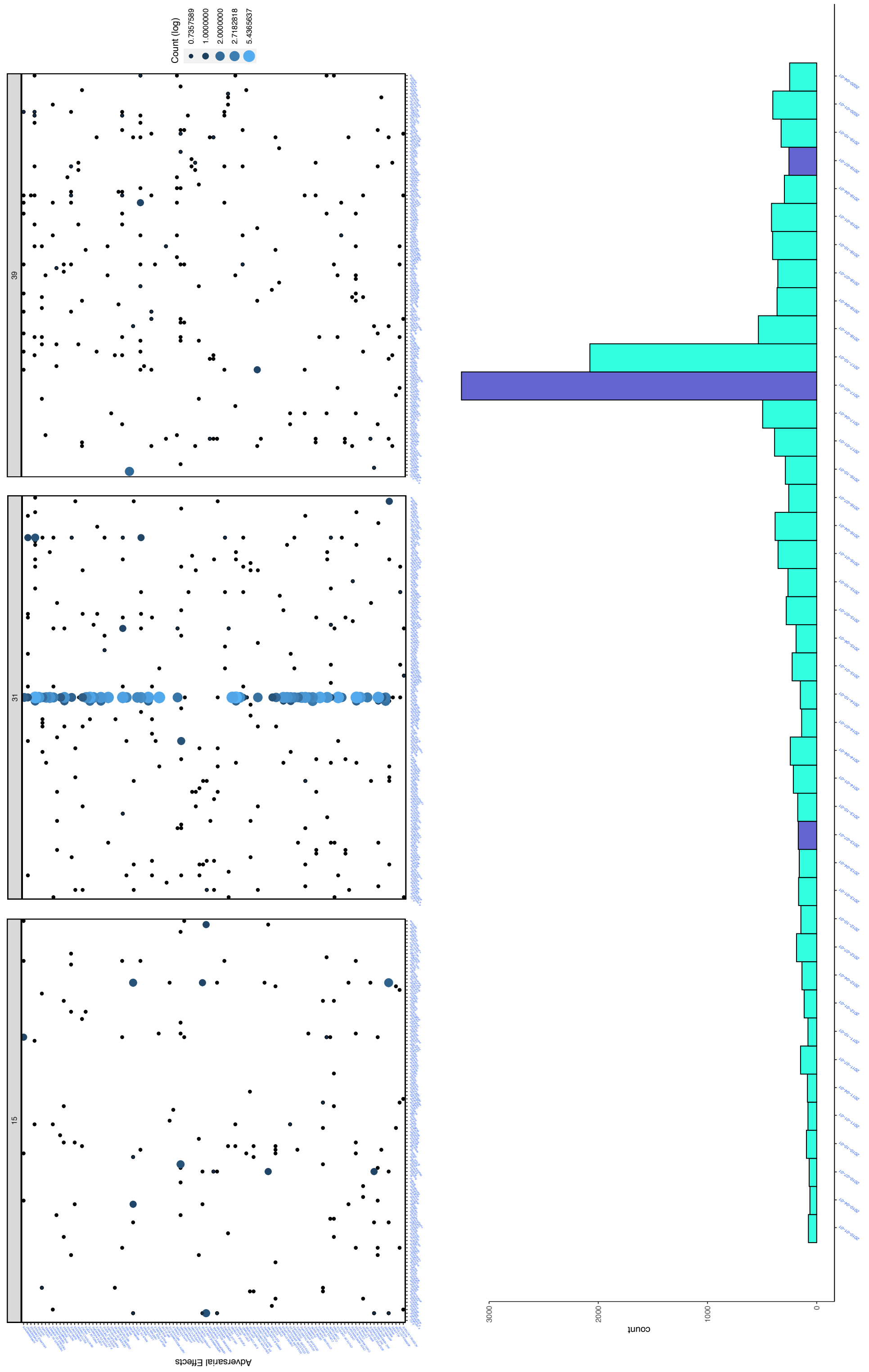


Figure 3.12: Notifications of adverse drug reactions (ADRs) reported to the Regional Center of Pharmacovigilance (RCPV) of Nice (France) in 3 different trimesters, highlighted in dark blue.

The process has been parallelized on 8 cores on a MacBook Pro, 2020, with a processor of 2,3 GHz Quad-Core Intel Core i7 and 16 GB of RAM.
Figure 3.13 shows the frequency of the declarations received by the RCPV from 2010 to 2020, sorted by month, where the colors represent the identified time clusters. Figure 3.14 displays the estimated intensity functions representation. In particular, this figure is very helpful for giving an overview of the relationships between drug clusters and ADR clusters and how they evolve over time. The colors refer to different time clusters and the brighter the color, the stronger the relation (i.e. the expected number of notifications in the time unit) between drug cluster and ADR cluster. Finally, Figure 3.15 shows more specifically the evolution of the relationship between drug clusters and ADR clusters over time. In fact, each panel represents a cluster of drugs and within them each line identifies a cluster of ADRs and its intensity changes over time.

Time clusters Starting from the analysis of the time clusters, one can easily notice in Figure 3.13 that the segmentation proposed by the algorithm confirms our knowledge about the previous mentioned health scandals while revealing a time structure more complex than expected. In fact, while cluster 1 and cluster 2 include various time intervals, cluster 3 clearly refers to the health crisis due to the Mirena® scandal while cluster 4 relates to the peak period in the Lévothyox® crisis. Time clusters 5 and 6 refer to the final stage of the Lévothyrox® crisis, when generics were introduced to the market. It is worth noticing that without the dLBM application it would have been impossible to detect the presence of other health scandal just before the one of Lévothyrox®. In fact, looking at Figure 3.13, one can see that the increase of declarations during the Mirena® health crisis are completely masked by the Lévothyrox® ones.

Drug clusters The clusters of drugs identified by the algorithm are also coherent with retrospective knowledge and adequately represent the variety of drugs present in the dataset. In particular, cluster 1, cluster 6 and cluster 7 are very specific, with one element only: they correspond respectively to lévothyroxine (Lévothyrox® and generics), benfluorex (Médiator®) and lévonorgestrel (Mirena®). It is worth noticing that Médiator® [2] was involved (like Lévothyrox® and Mirena®) in an important health scandal in 2009-2010. Moreover, cluster 2 contains the five

[2] https://www.ansm.sante.fr/Dossiers/Mediator-R/Mediator-R-et-accompagnement-des-personnes/(offset)/0

most frequently reported drugs and cluster 5 contains other common drugs, while cluster 4 is very large and heterogeneous, with drugs that are rarely reported and finally cluster 3 contains drugs that cause bleeding.

ADR clusters Concerning the clusters of ADRs, cluster 3 (e.g. coma, confusion, hepatic cytolysis, etc) and cluster 8 (e.g. agitation, agranulocytosis, arthralgia, etc.) contain the most frequently notified ADRs. Cluster 1 contains recurring ADRs (e.g. sweats, transient ischemic accident, lactic acidosis, etc.) but less than the other two previously mentioned. Cluster 2 (e.g. anemia, hemorrhagic stroke) and cluster 4 (e.g. hemorrhagic schock, deglobulisation, etc.) respectively include the most and the less frequent bleeding related ADRs. Cluster 7 is composed of ADRs clearly related to Lévothyrox® and Mirena® (e.g hair loss, cramps, insomnia, etc.). In cluster 10 there are general ADRs, although it contains some ADRs specifically related to Lévothyrox® and Médiator® (e.g. respectively abnomral TSH, valvular disease, etc.). Finally, cluster 5, 6 and 9 contain more general and nonspecific ADRs.

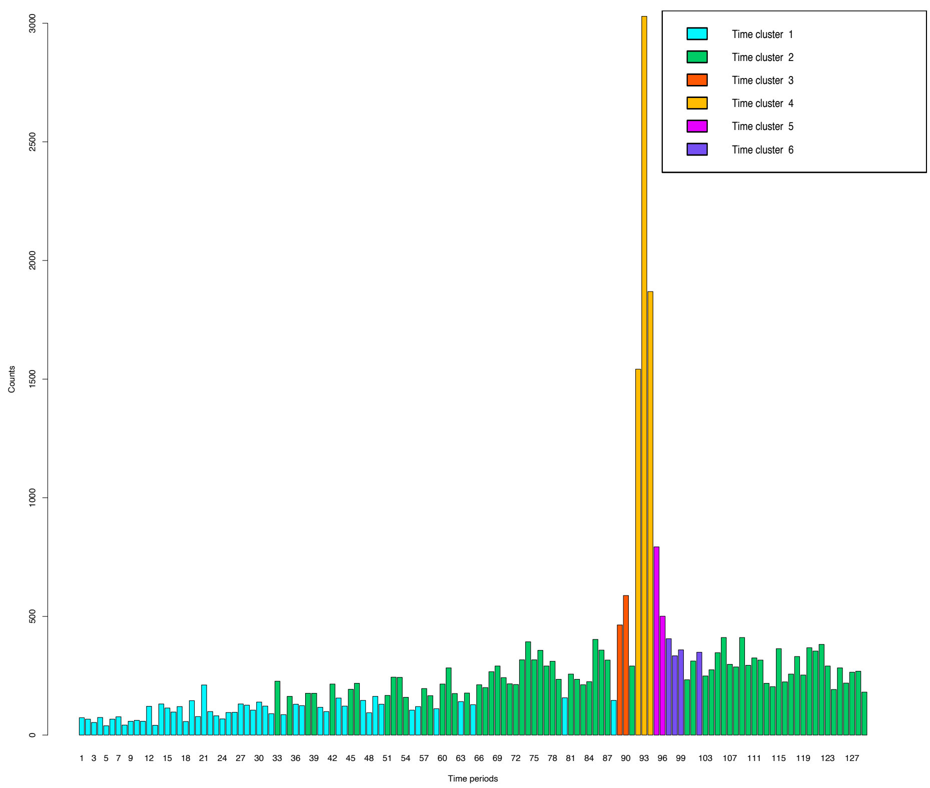


Figure 3.13: Number of declarations received by the pharmacovigilance center from 2010 to 2020, sorted by month, where the colors represent the time clusters.

Figure 3.14: Evolution of the relation between each drug cluster and the all ADR clusters over time. Each color corresponds to a different cluster of adverse drug reactions.

3.5.3 Detailed results

Time clusters Figure 3.14 is a graphical representation of the estimated intensity functions. It gives a clear idea about the relationships between clusters of molecules

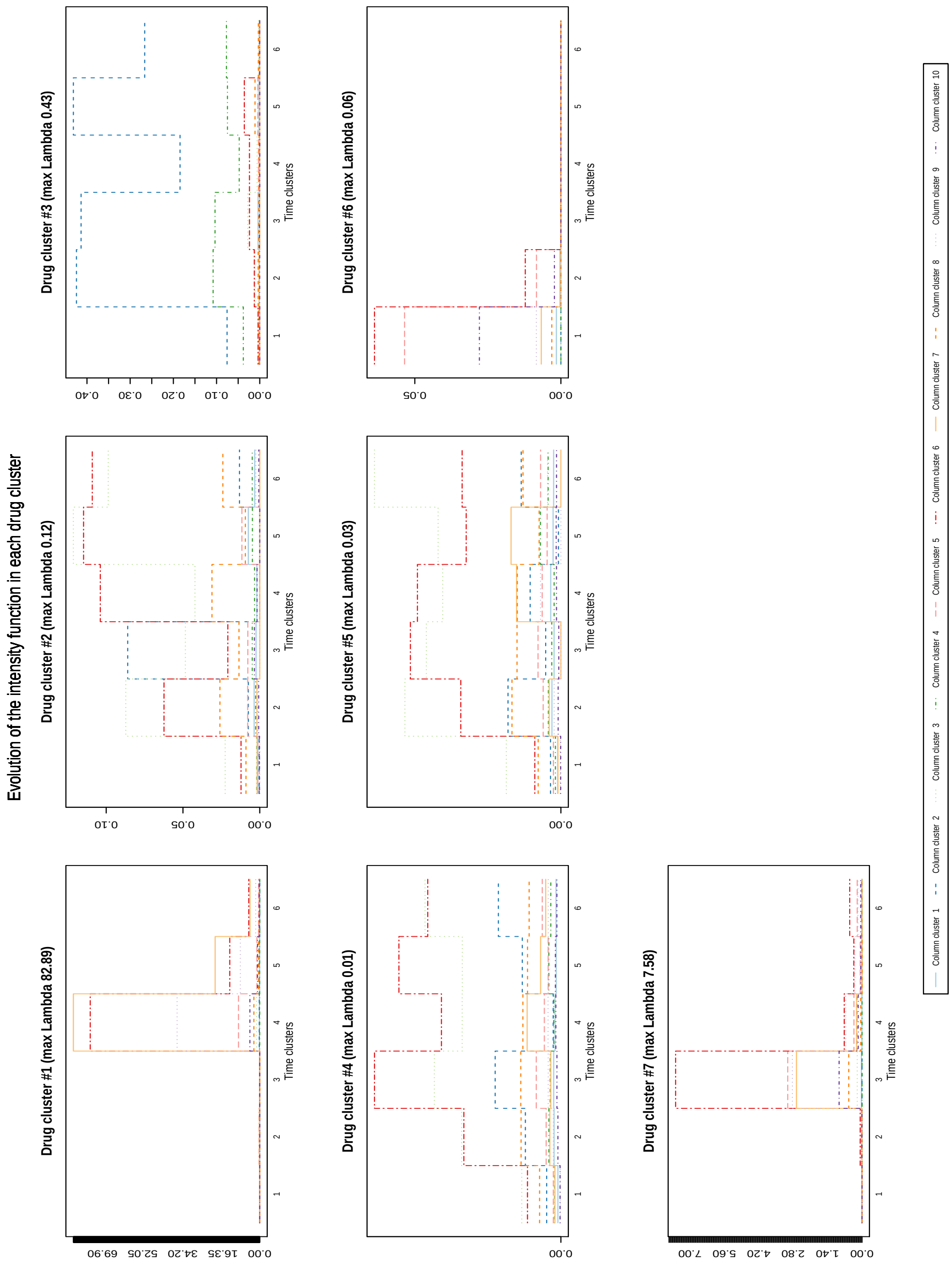


Figure 3.15: Evolution of the relation between each drug cluster and the all ADR clusters over time. Each color corresponds to a different cluster of adverse drug reaction.

and clusters of ADRs, with respect to time clusters. In particular:

- Time cluster 1: here one can notice the presence of all the drug clusters, with different levels of intensity. The peculiarity of this cluster lies in the strong presence of cluster 6 of drugs which gradually disappears in subsequent temporal clusters. In fact, it contains the drug benfluorex (Médiator®) which in 2010 was involved in a major health scandal and it has strong interactions with clusters 5, 6 and 10 of ADRs. It is worth noticing that dLBM managed to highlight this peculiarity that cannot be detected by simply looking at Figure 3.12. In the first time cluster it can also be noticed a strong relation between the cluster 3 of drugs (drugs that causes bleeding) and clusters 2 and 4 of ADRs which is coherent.

- Time cluster 2: the presence of Médiator® decreases while the interactions between the drugs that cause bleeding (cluster 3) and clusters 2 and 4 of ADRs is still strong. That cluster represent the actual profile of notifications received by RCPV. Similarly, the second cluster of drugs (the most frequently used) appears to have ADRs in almost all clusters, especially the third. We also notice the presence of Lévothyrox® (cluster 1) with ADRs especially in clusters 6 and 7.

- Time cluster 3: this cluster includes two months only: 05-06/2017. This period refers to the Mirena® scandal (cluster 7) with ADRs in clusters 5, 6, 7 and 9. They are not very specific but it may suggest a hormonal cause. In this matrix all the drug clusters are present (even though with a lower intensity with respect to Mirena®), with the exception of Médiator® which is not reappearing subsequently.

- Time cluster 4: it refers to the Lévothyrox® peak going from August to October 2017. Unlike previous clusters where most drugs clusters were present, this cluster only recognizes 2 drugs: Lévothyrox® (mainly) and Mirena® (weakly). Consequently, the interactions that stand out are those of Lévothyrox® with ADRs in clusters 6, 7 and 9. They are not very specific but coherent with the statements received by RCPV.

- Time cluster 5: it refers to 11-12/2017 and it is characterized by a reduction of Lévothyrox® declarations. Compared to the previous cluster, we note a reappearance of other drug clusters even if the intensities remain low compared to time cluster 1 and 2. The drugs/ADRs combinations remain those of the previous time cluster.

- Time cluster 6: it refers to the 1^{st} semester 2018, which corresponds to end of Lévothyrox® crisis. Globally it is similar to the two previous time clusters with some small variations on the intensities of the drug/ADRs pairs.

Drug clusters For a more in-depth analysis regarding the evolution of drug clusters over time and their interactions with the clusters of ADRs, we can refer to Figure 3.15. The following remarks derive:

- Drug cluster 1: this cluster refers to Lévothyrox® and its generics. There are almost no declared effects during the first three time clusters, from the fourth time cluster we observe a peak of declarations which corresponds to the start of the Lévothyrox® crisis, especially for ADRs in cluster 6, 7 and 9 and to a lesser extent on 5. These four clusters recognized by dLBM, include all of the ADRs described during Lévothyrox® crisis, namely hormonal ADRs (weight gain), general ADRs (fatigue, cramps) and neuro-psychic ADRs (anxiety, irritability, sleep disturbances). Time cluster 5 marks a decrease in Lévothyrox® reports in terms of numbers but the ADRs profile remains similar to the previous time cluster. It should be noticed that generics of Lévothyrox® began to be available from mid-October 2017, which could explain this decrease in the number of reports: patients started to have therapeutic alternatives. Finally, time cluster 6 represents the end of the Lévothyrox® crisis with a clear decrease in the number of reports.

- Drug cluster 2: this cluster includes drugs that are very frequently prescribed, the ADRs profile is globally constant over time with a predominance of clusters 3, 6 and 8, with variations in terms of proportions according to the time clusters. Cluster 3 of ADRs corresponds to frequent and generally serious ADRs. Cluster 8 also includes ADRs that are generally serious but a little less often reported than cluster 3. Cluster 6 corresponds to general ADRs that can be found with many other drugs (especially Lévothyrox®).

- Drug cluster 3: this cluster includes coagulation drugs whose main ADRs are bleeding, hence the predominance of cluster 2 and 4. The application of dLBM led us to identify, from the temporal point of view, 3 interesting events: the increase in ADRs between time clusters 1 and 2, a significant decrease in the number of declared ADRs in time cluster 4 (Lévothyrox® crisis) and a marked regression of cluster 2 in time cluster 6 (late Lévothyrox®) but without cluster 4 being affected.

- Drug cluster 4: this cluster includes a fairly large set of drugs declared relatively frequently and commonly prescribed, but disparate in terms of their therapeutic uses or their ADRs profile. However, we observe a predominance of ADRs of cluster 3 and 6 (general ADRs). At the temporal level, we observe an overall decrease in ADRs in time cluster 4 (Lévothyrox® crisis).

- Drug cluster 5: as for drug cluster 4, this cluster includes many heterogeneous drugs. The ADRs profile is similar to that of drug cluster 4, which is coherent with the fact that these two clusters are similar.

- Drug cluster 6: this is the Médiator® cluster, the ADRs are concentrated in time clusters 1 and 2 with a decrease in the number of ADRs in the second one. This is coherent with the history of the drug (Médiator® scandal happened in 2009 with withdrawal of the market). Regarding the profile of ADRs, we can notice a predominance of clusters 6, 5, and 10.

- Drug cluster 7: this is the Mirena® cluster, ADRs declarations predominate in time cluster 3 (Mirena® crisis) then drop drastically. At the level of ADRs profile, cluster 6 predominates (general ADRs), then come cluster 5, 7 and 9.

3.5.4 Discussion

In this application to pharmacovigilance, dLBM proved to be a very useful tool for identifying phenomena that would have been difficult to detect otherwise, even by an expert eye. In fact, dLBM revealed that in addition to Lévothyrox® health crisis, which was the one with the widest media coverage, two other major events have occurred. The first one concerning Médiator®, which took place in 2009-2010, and the second one concerning Mirena®, which took place in the first half of 2017. In addition, dLBM was also able to put in light some unexpected variations of notifications such as an under-notification of bleeding related ADRs during Lévothyrox® crisis. Bleeding related ADRs were expected to be constant over time because of the follow-up made the RCPV to monitor ADRs of direct oral anticoagulants (DOAs), a recent class of anticoagulant. However, the Lévothyrox®

crisis has caused such an overload of work that the DOAs follow-up have been temporarily interrupted. Another thing that dLBM has highlighted is the existence of 3 different phases during the Lévothyrox® crisis corresponding to the reporting peak, the marketing period of generics and the end of the crisis, respectively. Those phases were not noticed by the RCPV staff during the Lévothyrox® crisis. In general, we can conclude that dLBM could be extremly useful as a routine tool for signal detection, since it might help health professionals to identify structural changes or patterns of interest and, perhaps, prevent some of the consequences a health crisis can lead to.

3.6 Conclusion

This work is born out of the need to analyze and summarize observations and features of a dynamic count matrix in a simultaneous way. We have proposed a dynamic co-clustering technique, with the purpose of simultaneously performing clustering of rows, columns and slices (time dimension). We consider a dynamic framework because it is of great interest to look for structural changes in the way clusters interact with each other along the time. To this end, we have introduced a generative model, named dynamic latent block model (dLBM). The dynamic time modeling relies on non-homogeneous Poisson processes, with a latent partition of time intervals. Inference is done using a SEM-Gibbs algorithm and the ICL criterion is used for model selection. The performance of dLBM was evaluated through applications to several simulated data scenarios and compared with that of competing methods. Then, dLBM was fit to a large-scale dataset supplied by the Regional Center of Pharmacovigilance of Nice (France). In this context, dLBM provided meaningful segmentations of drugs, adverse drug reactions and time. Its potential use by medical authorities for identifying meaningful pharmacovigilance patterns looks very promising.

Chapter 4

A Deep Dynamic Latent Block Model for Co-clustering of Zero-Inflated Data Matrices

In Chapter 3 we proposed a dynamic extension of LBM. A different extension is proposed in this chapter, allowing one to obtain results at a different granularity and better taking into account the number of zero entries in the data. To detect abrupt changes in the dynamics of both cluster memberships and data sparsity, the mixing and sparsity proportions are modeled through systems of ordinary differential equations. The inference relies on an original variational procedure whose maximization step trains fully connected neural networks in order to solve the dynamical systems. Numerical experiments on simulated and real world data sets demonstrate the effectiveness of the proposed methodology in the context of count data.

This chapter presents the results of two research studies:

- G. Marchello, M. Corneli, C. Bouveyron (2023). *A Deep Dynamic Latent Block Model for the Co-clustering of Zero-Inflated Data Matrices.* European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD);
- G. Marchello, M. Corneli, C. Bouveyron (2023). *A Deep Dynamic Latent Block Model for Co-clustering of Zero-Inflated Data Matrices.* Accepted for publication in the Journal of Computational Graphical Statistics.

4.1 Introduction

The challenge of high dimensionality often coincides with the issue of sparsity, potentially resulting in uninformative cluster outcomes and increased computational complexity (Raftery and Dean, 2006; Maugis et al., 2009).
In Chapter 3 we proposed an extension of LBM allowing one to perform the simultaneous clustering of rows, columns and time intervals of a three dimensional counting tensor. Although being a first attempt to extend LBM to the dynamic case, this model has at least two limitations: i) of not allowing cluster switches of rows/columns. Consequently, this model offers a limited level of granularity in its results, primarily supporting macroscopic and retrospective analyses; ii)it might not be suited to account for data sparsity due to the choice of the Poisson distribution.

4.1.1 Contribution of this work

In this chapter, we introduce a co-clustering method to deal with time evolving data matrices, potentially very sparse. In order to model the evolving generating process behind the data and simultaneously account for the data sparsity, we assume that the observations follow a time and block dependent mixture of zero-inflated distributions. Since we co-cluster the rows and columns of the data matrices, we introduce two evolving latent random variables that model the group memberships of observations and features, respectively. Moreover, the parameters of the random variables and the data sparsity proportion arise from three systems of ordinary differential equations that model the dynamics. Capturing the data dynamics is crucial in order to detect atypical phenomena that affected the generative process. For instance, if at a given time t_0 the value of some features suddenly increases for just one observation in a group, that observation will be very likely switched to another cluster, from t_0 on, when fitting our model to the data. This example suggests an interpretation of the results which is quite intuitive: a change in the affiliation of the observations/features to the clusters means that a change point has been detected, leaving space for further analysis to inspect the causes. Thus, we develop a highly interpretable co-clustering method allowing practitioners to obtain a faster visualization of the results in order to automate the data analysis. The proposed model can be used both as a tool for retrospective analysis and, above all, as a tool for near real-time and analysis of the data progression. The code is publicly available on GitHub at the link: https://github.com/giuliamar95/Zip-dLBM.

4.1.2 Organization of the chapter

This chapter is organized as follows. Section 4.2 introduces the proposed generative model. In Section 4.3, the inference procedure is detailed. Section 4.4 presents various experiments on simulated data to test and evaluate the model performances. In Section 4.5, an application to a real world dataset is presented. The London Bike sharing dataset is analyzed in order to illustrate the potential of the proposed model. Then, Section 4.6 details the results of applying the model to pharmacovigilance data. Finally, Section 4.7 provides some concluding remarks.

4.2 A Zero-Inflated dynamic LBM

In this section we introduce the Zero-Inflated Dynamic Latent Block model (Zero-Inflated dLBM).

Notation. The observed data are assumed to be collected into time evolving matrices, over the interval $[0, T]$. We work in discrete time[1] and assume that we have a time partition of equally spaced points:

$$0 = t_0 < t_1 < t_u \leq t_U = T.$$

Now up to rescaling, we can assume without loss of generality, that $t_{u+1} - t_u = 1$. Moreover, to simplify the exposition we omit the subscript u and, with a slight abuse of notation, we denote by t the generic time point t_u and by T the number of time points U. Thus, at (discretized) time t, we introduce the incidence matrix $X(t) \in \mathbb{N}^{N \times M}$ whose entry $X_{ij}(t)$ represents its generic element and it contains the observations and features that took place between t and $t-1$. The rows of $X(t)$ are indexed by $i = 1, ..., N$ and the columns by $j = 1, ..., M$.

[1] This assumption is needed to make the inference tractable. As such, we could describe the generative model in continuous time and move to discrete time in Section 4.3. However, in order to keep the exposition as simple as possible, we decided to introduce this assumption now.

4.2.1 A Zero-Inflated Dynamic Latent Block Model

For readers not familiar with the subject, it is advisable to begin by reading Section 2.3.1 of this manuscript, where the Latent Block Model is comprehensively explained. Our goal in this section is to simultaneously cluster the rows and columns of the collection of data matrices $\{X(t)\}_t$ evolving along the time.

Clusters modeling

The rows (i.e. observations) and columns (i.e. features) of $X(t)$ are clustered into Q and L groups, respectively. Although Q and L are assumed fixed over time, each row/column is nevertheless allowed to change cluster membership, in $[0, T]$. More formally, the cluster memberships of the rows and columns of X are identified by two evolving multinomial distributions, respectively parameterized by $\alpha(t)$ and $\beta(t)$:

$$Z_i(t) \overset{iid}{\sim} \mathcal{M}(1, \alpha(t) := (\alpha_1(t), \dots, \alpha_Q(t))),$$

where $Z_i(t)$ is the i-th row of $Z(t)$, $\mathcal{M}(1, \cdot)$ denotes the multinomial probability mass function and $\alpha_q(t) = \mathbb{P}\{z_{iq}(t) = 1\}$, with $\sum_{q=1}^{Q} \alpha_q(t) = 1$, at time $t \in \{0, \dots, T\}$.
Thus $Z(t) := \{z_{iq}(t)\}_{i\in 1,\dots,N; q\in 1,\dots,Q}$ represents the clustering of N rows into Q groups at a given time point t. In a similar fashion, for the column clusters, we assume:

$$W_j(t) \overset{iid}{\sim} \mathcal{M}(1, \beta(t) := (\beta_1(t), \dots, \beta_L(t))),$$

where $W_j(t)$ is the j-th row of $W(t)$, $\beta_\ell(t) = \mathbb{P}\{w_{j\ell}(t) = 1\}$ and $\sum_{\ell=1}^{L} \beta_\ell(t) = 1$.
The two random arrays Z and W are further assumed to be independent.

Sparsity modeling

In order to model a potential extreme sparsity, the observed data are assumed to be modeled by a mixture of block-conditional Zero-Inflated (ZI) distributions, where the entries $X_{ij}(t)$ are conditionally independent[2]:

$$X_{ij}(t)|Z_i(t), W_j(t) \sim ZI(\zeta_{Z_i(t),W_j(t)}; \pi(t)) \ , \tag{4.1}$$

[2]When no confusion arises we adopt in Eq. (4.1) a quite common convention in the clustering literature: $Z_i(t)$ denotes both the i-th row of $Z(t)$ and a random variable whose value is q if row i is in the q-th row cluster at time t, the same convention stands for for $W_j(t)$.

where ζ is a $Q \times L$ block-dependent parameter set of the distribution and $\pi(t)$ is a vector of length T that controls the proportion of data sparsity at any given time period. In more detail, being a mixture between a chosen distribution $\varphi(\cdot;\cdot)$ and a Dirac mass at zero, the Zero-Inflated distribution is used to account for a high sparsity in the data and can be formally written as:

$$\begin{cases} X_{ij}(t)|Z_i(t), W_j(t) \sim \delta_0(X_{ij}(t)) & \text{with probability } \pi(t) \\ X_{ij}(t)|Z_i(t), W_j(t) \sim \varphi(X_{ij}(t); \zeta_{Z_i(t);W_j(t)}) & \text{with probability } 1-\pi(t). \end{cases} \tag{4.2}$$

where $\delta_0(\cdot)$ is the Dirac mass function in 0.

Then, to model time-evolving sparsity, we rewrite Eq. (4.2) by introducing a hidden random matrix, $A \in \{0,1\}^{N\times M}$, where:

$$A_{ij}(t) \sim \mathcal{B}(\pi(t)),$$

with $\mathcal{B}(\cdot)$ denoting the Bernoulli probability mass function of parameter $\pi(t)$ and such that

$$\begin{aligned} A_{ij}(t) = 1 &\Rightarrow X_{ij}(t)|Z_i(t), W_j(t) \sim \delta_0(X_{ij}(t)) \\ A_{ij}(t) = 0 &\Rightarrow X_{ij}(t)|Z_i(t), W_j(t) \sim \varphi(X_{ij}(t); \zeta_{Z_i(t),W_j(t)}). \end{aligned} \tag{4.3}$$

Among the possible choices of $\varphi(\cdot)$, one of the most widely used is the Poisson distribution (Zero-Inflated Poisson, Lambert, 1992) for count data, or the log-normal and the Gamma distributions for continuous data.

Modeling the temporal evolution of the parameters

The last assumption concerns the modeling of clusters proportions and sparsity over time. In fact, the mixing parameters $\alpha(t)$ and $\beta(t)$ as well as the sparsity proportions $\pi(t)$ are assumed to be driven by systems of ordinary differential equations (ODEs). In this way, we are able to capture the temporal evolution of both the clusters composition and the (excess of) sparsity. In continuous time, the three dynamic systems read as:

$$\frac{d}{dt}a(t) = f_Z(a(t)), \qquad \frac{d}{dt}b(t) = f_W(b(t)), \qquad \frac{d}{dt}c(t) = f_A(c(t)), \tag{4.4}$$

where $t \in [0,T]$, $f_Z : \mathbb{R}^Q \to \mathbb{R}^Q$, $f_W : \mathbb{R}^L \to \mathbb{R}^L$ and $f_A : \mathbb{R} \to \mathbb{R}$ are three unknown continuous functions and $a : [0,T] \to \mathbb{R}^Q$, $b : [0,T] \to \mathbb{R}^L$ and $c : [0,T] \to \mathbb{R}$ are three continuously differentiable functions such that

$$\alpha_q(t) = \text{softmax}(a_q(t)) = \frac{e^{a_q(t)}}{\sum_{q_0=1}^{Q} e^{a}_{q_0}(t)},$$

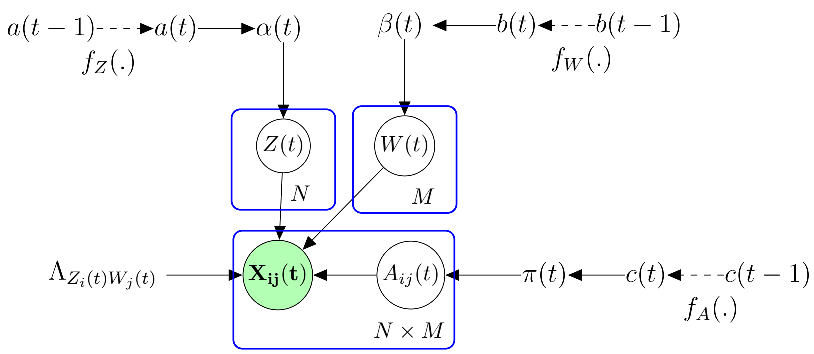


Figure 4.1: Graphical representation of the Zero-Inflated dLBM model.

$$\beta_\ell(t) = \text{softmax}(b_\ell(t)) = \frac{e^{b_\ell(t)}}{\sum_{\ell_0=1}^{L} e^{b}_{\ell_0}(t)},$$

$$\pi(t) = \frac{e^{c(t)}}{1 + e^{c(t)}}.$$

Then, since (as stated at beginning of Section 4.2) we work in discrete time, the above dynamic systems reduce to their Euler schemes. For instance, the first equation of Eq. (4.4) reduces to:

$$a(t+1) = a(t) + f_Z(a(t)).$$

Remark that, if we want to relax the condition of equally spaced points, we can introduce a parameter Δ_t such that:

$$a(t + \Delta_t) \cong a(t) + f_Z(a(t)) \cdot \Delta_t.$$

where Δ_t indicates the length of the considered time interval. Henceforth, in order to simplify the exposition, we assume that Δ_t is constant, denoted as Δ. Furthermore, for convenience, we set $\Delta = 1$ without loss of generality. A graphical representation of the model described so far, and named Zero-Inflated dLBM, can be seen in Figure 4.1.

4.2.2 The joint distribution

The model described so far can be adapted to any zero-inflated distribution. The first as well as the most well-known Zero-Inflated distribution is the Zero-Inflated Poisson, from an article by Lambert (1992). However, other distributions such as Zero-Inflated Negative Binomial (Ridout et al., 2001), Zero-Inflated Beta (Ospina and Ferrari, 2012), Zero-Inflated log-normal (Li et al., 2011) could be coupled

with the present modeling.
In the following, to ease the readability of the inference procedure we make use of the Zero-Inflated Poisson (ZIP) formulation to illustrate our approach. Hence, we can write $X_{ij}(t)|Z_i(t), W_j(t) \sim ZIP(\Lambda_{Z_i(t)W_j(t)}; \pi(t))$ and develop Eq. (4.2) as follows:

$$\begin{cases} X_{ij}(t)|Z_i(t), W_j(t) = 0 & \text{with probability } \pi(t) \\ X_{ij}(t)|Z_i(t), W_j(t) \sim \mathcal{P}(\Lambda_{Z_i(t)W_j(t)}) & \text{with probability } 1 - \pi(t) \end{cases}$$

where Λ is a $Q \times L$ matrix, denoting the block-dependent Poisson parameter and $\pi(t)$ represents the sparsity at any given time period, with $t = 0, \dots, T$.

The model described so far has a set of parameters denoted by $\theta = (\Lambda, \alpha, \beta, \pi)$ and a set of latent variables: $\{Z,\ W,\ A\}$, where $Z = \{Z(t)\}_{t=1,\dots,T}$, $W = \{W(t)\}_{t=1,\dots,T}$, $A = \{A(t)\}_{t=1,\dots,T}$ and $X = \{X(t)\}_{t=1,\dots,T}$.
As a first move, we can compute the likelihood of the complete data:

$$p(X, Z, W, A|\theta) = p(X|Z, W, A, \Lambda, \pi)p(A \mid \pi)p(Z|\alpha)p(W|\beta), \tag{4.5}$$

where:

$$p(X|A, Z, W, \Lambda, \pi) = \prod_{i=1}^{N}\prod_{j=1}^{M}\prod_{t=1}^{T} \mathbf{1}_{\{X_{ij}(t)=0\}}^{A_{ij}(t)} \left\{ \left(\frac{\Lambda_{Z_i(t)W_j(t)}^{X_{ij}(t)}}{X_{ij}(t)!} \exp(-\Lambda_{Z_i(t)W_j(t)}) \right)^{(1-A_{ij}(t))} \right\},$$

$$p(A|\pi) = \prod_{i=1}^{N}\prod_{j=1}^{M}\prod_{t=1}^{T} \pi(t)^{A_{ij}(t)} \Big(1 - \pi(t)\Big)^{(1-A_{ij}(t))},$$

$$p(Z|\alpha) = \prod_{i=1}^{N}\prod_{q=1}^{Q}\prod_{t=1}^{T} \alpha_q(t)^{Z_{iq}(t)}, \qquad p(W|\beta) = \prod_{j=1}^{M}\prod_{\ell=1}^{L}\prod_{t=1}^{T} \beta_\ell(t)^{W_{j\ell}(t)}.$$

4.3 The inference algorithm

Regarding the parameters estimation, the traditional procedure for such a model would be to maximize the log-likelihood $p(X|\theta)$. However, in our case, neither the direct maximization nor the classical EM algorithm (Dempster et al., 1977) are feasible because of the impossibility to compute the joint conditional distribution of the latent variables, $p(Z, W|X, \theta)$, due to their interdependent double missing structure. For details on this, please refer to Section 2.5.1. An additional difficulty is also the link that α, β and π have with their respective systems of differential equations, which do not allow the formulation of a closed updating formula. This is why, we rely on the Variational-EM algorithm combined with Stochastic Gradient Descent (SGD). We recall that variational inference is usually applied to complex models involving missing values or relying on latent structures (Jaakkola and Jordan, 1997; Jordan et al., 1998). The VEM algorithm has been shown to make relevant estimates of mixture models in different configurations (Govaert and Nadif, 2008).

4.3.1 Variational inference

Since we cannot compute the posterior distribution, $p(A, Z, W|X, \theta)$, we rely on a variational procedure which optimizes a lower bound of the likelihood. Let us thus introduce a variational distribution $q(.)$ in order to decompose the log-likelihood as follows:

$$\log p(X|\theta) = \mathcal{L}(q, \theta) + KL(q(.)||p(.|X, \theta)),$$

where $\mathcal{L}$ denotes a lower bound and is defined as:

$$\begin{aligned}
\mathcal{L}(q, \theta) &= \sum_Z \sum_W \sum_A q(Z, W, A) \log \frac{p(X, A, Z, W|\theta)}{q(Z, W, A)} \\
&= E_{q(A,Z,W)}\left[\log \frac{p(X, A, Z, W|\theta)}{q(A, Z, W)}\right] \\
&= E_{q(A,Z,W)}[\log(p(X, A, Z, W|\theta)] - E_{q(A,Z,W)}[\log(q(A, Z, W))],
\end{aligned} \tag{4.6}$$

and KL denotes the Kullaback-Liebler divergence between the true and the approximate posterior:

$$KL(q(.)||p(.|X, \theta)) = -\sum_A \sum_Z \sum_W q(A, Z, W) \log \frac{p(A, Z, W|X, \theta)}{q(A, Z, W)}.$$

Now, the objective is to find a tractable distribution $q(.)$ that maximizes the lower bound $\mathcal{L}(q, \theta)$. In order to allow the optimization of $\mathcal{L}(q, \theta)$, we further assume that $q(A, Z, W)$ factorizes as follows:

$$q(Z, W, A) = q(A)q(Z)q(W) = \prod_{i=1}^N \prod_{j=1}^M \prod_{t=1}^T q(A_{ij}(t)) \prod_{i=1}^N \prod_{t=1}^T q(Z_i(t)) \prod_{j=1}^M \prod_{t=1}^T q(W_j(t)).$$

4.3.2 Variational E-Step

The VE-step of the VEM algorithm aims at maximizing the lower bound in Eq. (4.6) with respect to the variational distribution $q(\cdot)$ while keeping θ fixed.

Following (Ch.10, Bishop, 2006), we derive the update equations for the factors $q(A)$, $q(Z)$, and $q(W)$, such that the log of the optimized factors are given by:

$$\log q^*(A) = E_{q(W,Z)}[\log p(X, A, Z, W \mid \theta)], \tag{4.7}$$

$$\log q^*(Z) = E_{q(A,W)}[\log p(X, A, Z, W \mid \theta)], \tag{4.8}$$

$$\log q^*(W) = E_{q(A,Z)}[\log p(X, A, Z, W \mid \theta)]. \tag{4.9}$$

Optimization of the factor q(A)

Let us consider the derivation of the update equation for the factor $q(A)$. The sequential update for the factor $q(A)$ can be computed through the log of the optimized factor, where all the terms that do not depend on A are absorbed in the constant term.

Proposition 1. *Denoting by $\delta_{ij}(t) := q(A_{ij}(t) = 1)$ the variational success probability for $A_{ij}(t)$, the optimal update of $q(A)$ is:*

$$\delta_{ij}(t) = \frac{\exp(R_{ij}(t))}{1 + \exp(R_{ij}(t))}, \tag{4.10}$$

with:

$$R_{ij}(t) = \log(\pi(t)\mathbf{1}_{\{X_{ij}(t)=0\}}) + \sum_{q=1}^{Q}\sum_{\ell=1}^{L}\Big[- E_{q(W,Z)}[Z_{iq}(t)]E_{q(W,Z)}[W_{j\ell}(t)]X_{ij}(t)\log\Lambda_{q\ell} + \\ + E_{q(W,Z)}[Z_{iq}(t)]E_{q(W,Z)}[W_{j\ell}(t)]\Lambda_{q\ell}\Big] + \log X_{ij}(t)! - \log(1-\pi(t)). \tag{4.11}$$

The proof is provided in Appendix B.1. Note that, formally, when $X_{ij}(t) \neq 0$, $R_{ij}(t) = -\infty$ and $\delta_{ij}(t) = 0$, which makes sense: non-null observations in X come from a Poisson distribution with probability one (see Eq. (4.3)).

Optimization of the factor q(Z)

Let us now consider the derivation of the update equation for the factor $q(Z)$. The sequential update for the factor $q(Z)$ can be computed through the log of the optimized factor, where all the terms that do not depend on Z are absorbed in the constant term.

Proposition 2. *Denoting by $\tau_{iq}(t) := q(Z_{iq}(t) = 1)$ the variational probability for $Z_{iq}(t)$, the optimal update of $q(Z)$ is:*

$$\tau_{iq}(t) = \frac{r_{iq}(t)}{\sum_{q_0=1}^{Q} r_{iq_0}(t)}, \tag{4.12}$$

with $r_{iq}(t)$ is denoted by:

$$r_{iq}(t) \propto \exp\left(\sum_{j=1}^{M}\sum_{\ell=1}^{L}\Big\{(1 - E_{q(A,W)}[A_{ij}(t)])\Big[E_{q(A,W)}[W_{j\ell}(t)]X_{ij}(t)\log(\Lambda_{q\ell}) + \\ - E_{q(A,W)}[W_{j\ell}(t)]\Lambda_{q\ell}\Big]\Big\} + \log(\alpha_q(t))\right). \tag{4.13}$$

The proof is provided in Appendix B.2.

Optimization of the factor q(W)

Let us now consider the derivation of the update equation for the factor $q(W)$. The sequential update for the factor $q(W)$ can be computed through the log of the optimized factor, where all the terms that do not depend on W are absorbed in the constant term.

Proposition 3. *Similarly for the latent variable W, denoting by $\eta_{j\ell}(t) := q(W_{j\ell}(t) = 1)$ the variational probability for $W_{j\ell}(t)$, the optimal update of $q(W)$ is:*

$$\eta_{j\ell}(t) = \frac{s_{j\ell}(t)}{\sum_{\ell_o=1}^{L} s_{j\ell_o}(t)}, \tag{4.14}$$

where :

$$s_{j\ell}(t) \propto \exp\left(\sum_{i=1}^{N}\sum_{q=1}^{Q}\Big\{(1 - E_{q(A,Z)}[A_{ij}(t)])\Big[E_{q(A,Z)}[Z_{iq}(t)]X_{ij}(t)\log(\Lambda_{q\ell}) + \\ - E_{q(A,Z)}[Z_{iq}(t)]\Lambda_{q\ell}\Big]\Big\} + \log(\beta_\ell(t))\right). \tag{4.15}$$

The proof is symmetrical to the one developed for $\tau_{iq}(t)$ in Proposition 2.

4.3.3 Variational M-Step

In order to obtain the updating of the parameter set θ, the objective of the M-Step is the maximization of the lower bound $\mathcal{L}(q,\theta)$ with respect to $\theta = (\Lambda, \alpha, \beta, \pi)$, while holding the variational distribution $q(\cdot)$ fixed.

Proposition 4. *By developing the Eq.* (4.6), *the variational lower bound* $\mathcal{L}(q,\theta)$ *can be written as:*

$$
\begin{aligned}
\mathcal{L}(q,\theta) = &\sum_{t=1}^{T}\sum_{i=1}^{N}\sum_{j=1}^{M}\Bigg\{\delta_{ij}(t)\log(\pi(t)\mathbf{1}_{\{X_{ij}(t)=0\}}) + (1-\delta_{ij}(t))\Bigg[\log(1-\pi(t)) +\\
&+\sum_{q=1}^{Q}\sum_{\ell=1}^{L}\Big\{\tau_{iq}(t)\eta_{j\ell}(t)X_{ij}(t)\log\Lambda_{q\ell} - \tau_{iq}(t)\eta_{j\ell}(t)\Lambda_{q\ell}\Big\}\Bigg]+\\
&-(1-\delta_{ij}(t))\log(X_{ij}(t)!)\Bigg\} + \sum_{t=1}^{T}\sum_{i=1}^{N}\sum_{q=1}^{Q}\tau_{iq}(t)\log(\alpha_q(t))+\\
&+\sum_{t=1}^{T}\sum_{j=1}^{M}\sum_{\ell=1}^{L}\eta_{j\ell}(t)\log(\beta_\ell(t)) - \sum_{t=1}^{T}\sum_{i=1}^{N}\sum_{q=1}^{Q}\tau_{iq}(t)\log\tau_{iq}(t)+\\
&-\sum_{t=1}^{T}\sum_{j=1}^{M}\sum_{\ell=1}^{L}\eta_{j\ell}(t)\log(\eta_{j\ell}(t)) - \sum_{t=1}^{T}\sum_{i=1}^{N}\sum_{j=1}^{M}\Big(\delta_{ij}(t)\log(\delta_{ij}(t)) + (1-\delta_{ij}(t))\log(1-\delta_{ij}(t))\Big).
\end{aligned}
\tag{4.16}
$$

The proof is provided in Appendix B.3.

Update of Λ

Here our goal is to derive the update of the Zero-inflated Poisson intensity parameter, Λ. The variational distribution $q(A, Z, W)$ is kept fixed, while the lower bound is maximized with respect to Λ, to obtain its update, $\hat{\Lambda}$. However, in case other zero-inflated distributions are chosen, this step must obviously be adapted to the new distribution, although the procedure of the derivation does not change.

Proposition 5. *The updating formula of* Λ *is obtained by maximizing* $\mathcal{L}(q,\theta)$ *with respect to the parameter and it can be written as follows:*

$$
\hat{\Lambda}_{q\ell} = \frac{\sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{t=1}^{T}\tau_{iq}(t)\eta_{j\ell}(t)\Big(X_{ij}(t) - \delta_{ij}(t)X_{ij}(t)\Big)}{\sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{t=1}^{T}\tau_{iq}(t)\eta_{j\ell}(t)\Big(1-\delta_{ij}(t)\Big)}. \tag{4.17}
$$

The proof is provided in Appendix B.4.

Update of α, β and π

The mixture proportions, $\alpha(t)$ and $\beta(t)$, and the sparsity parameter, $\pi(t)$, are driven by three systems of differential equations, in Eq. (4.4), respectively. As we assumed that the functions f_A, f_W and f_Z are continuous, we propose here to parametrize them using three fully connected neural networks (Gent and Sheppard, 1992). Thus, optimizing the lower bound in Eq. (4.16) with respect to $\alpha(t)$, $\beta(t)$ and $\pi(t)$, reduces to maximize it with respect to the parameters of the neural networks, ω_A, ω_Z and ω_W, as well as to the initial values $a(0), b(0)$ and $c(0)$. In particular, we denote with $\omega(h)$ the set of weights of the neural network settled for the updating of the related parameter at iteration h. The initial set of weights is randomly sampled, $\omega(0) = \{\omega(0)\}_{k=1}^{K}$, and along the iterations they are updated as follows:

$$\omega_k(h) = \omega_k(h-1) - \gamma \nabla \mathcal{L}_{\omega_k(h)},$$

where γ is the learning rate, in the experiments they are γ_A, γ_Z, $\gamma_W = 1e-4$. The maximization is implemented in PyTorch via automatic differentiation (Paszke et al., 2017) and relies on stochastic optimisation (ADAM, Kingma and Ba, 2014). For further details on other technical points see Appendix B.5. Thanks to back-propagation the updated networks provide us with estimates of α, β and π. The inference procedure is summarized in Algorithm 3.

4.3.4 Initialization and model selection

When dealing with clustering methods based on the EM algorithm, the initialization and the selection of the appropriate numbers of clusters (for rows and columns here) are two issues which deserve an appropriate treatment. The issues related to these two points are slightly complicated here by the use of deep neural networks for modeling the dynamics of cluster and sparsity proportions. Despite this apparent difficulty due to the intrinsic complexity of these networks, they will nevertheless offer some unexpected flexibility that we may use to lower the computational cost of the whole algorithm. Indeed, and as it will be illustrated in the following numerical experiments (Section 4.4), the use of deep neural networks for modeling the row and column cluster proportions will allow our algorithm to work with some empty clusters.

Therefore, with the objective to avoid the usual computationally demanding procedure of testing all pairs of row and column cluster numbers, we propose the

Algorithm 1 VEM-SGD algorithm (for the Zero-Inflated Poisson distribution)

Require: $X, Q, L, max.iter,\ \alpha, \beta, \pi, \Lambda$ from Initialization.

Initialization of $\tau(t)$ and $\eta(t)$: sampling from $\mathcal{M}(\alpha(t))$ and $\mathcal{M}(\beta(t))$, respectively;

Initialization of $\delta(t)$: matrix of 1, then setting $\delta(t) = 0$ when $X > 0$;

for it = 1 to *max.iter* **do**

VE-Step:

for p = 1 to p_max **do**

for t = 1 to T **do**

Update $\delta(t)$**,** $\tau(t)$**,** $\eta(t)$ **for all** $i = 1, \ldots, N; j = 1, \ldots, M$:

$$\delta_{ij}(t) = \frac{\exp(R_{ij}(t))}{(1 + \exp(R_{ij}(t)))},$$

where:

$$R_{ij}(t) = \log(\pi(t)\mathbf{1}_{\{X_{ij}(t)=0\}}) + \sum_{q=1}^{Q}\sum_{\ell=1}^{L}\Big[- \tau_{iq}(t)\eta_{j\ell}(t)X_{ij}(t)\log\Lambda_{q\ell} + \tau_{iq}(t)\eta_{j\ell}(t)\Lambda_{q\ell}\Big] + \log X_{ij}(t)! - \log(1 - \pi(t)).$$

$$\tau_{iq}(t) = \frac{1}{D_q}\exp\left(\sum_{j=1}^{M}\sum_{\ell=1}^{L}\Big\{(1 - \delta_{ij}(t))\Big[\eta_{j\ell}(t)X_{ij}(t)\log(\Lambda_{q\ell}) - \eta_{j\ell}(t)\Lambda_{q\ell}\Big]\Big\} + \log(\alpha_q(t))\right).$$

$$\eta_{j\ell}(t) = \frac{1}{D_\ell}\exp\left(\sum_{i=1}^{N}\sum_{q=1}^{Q}\Big\{(1 - \delta_{ij}(t))\Big[\tau_{iq}(t)X_{ij}(t)\log(\Lambda_{q\ell}) - \tau_{iq}(t)\Lambda_{q\ell}\Big]\Big\} + \log(\beta_\ell(t))\right).$$

with D_q and D_ℓ normalizing constants.

end for

end for

M-Step:

Update $\theta = (\Lambda, \pi, \alpha, \beta)$.

$$\hat{\Lambda}_{q\ell} = \frac{\sum_{i,j,t}\Big\{\tau_{iq}(t)\eta_{j\ell}(t)\Big(X_{ij}(t) - \delta_{ij}(t)X_{ij}(t)\Big)\Big\}}{\sum_{i,j,t}\Big\{\tau_{iq}(t)\eta_{j\ell}(t)\Big(1 - \delta_{ij}(t)\Big)\Big\}}.$$

for epoch = 1 to Epochs **do**

Update $\hat{\alpha}, \hat{\beta}, \hat{\pi}$:

Loss Evaluation;

Algorithm backpropagation;

Numerical optimization with SGD.

end for

end for

following strategy for both initialization and model selection.

- First, we select a single specific slice of the data $X_{t_{init}}$ and apply on it a static version of our Zip-dLBM algorithm for a list of pairs of cluster numbers, i.e. (q, ℓ) for $q = 2, \ldots, Q_{max}$ and $\ell = 2, \ldots, L_{max}$. We then use the ICL criterion (Integrated Completed Likelihood, (Biernacki et al., 2000)) to select the most appropriate row and column cluster numbers for this specific slice of data. Let us remind that ICL aims at approximating the complete-data integrated log-likelihood and can be derived for the Zip-dLBM model as follows:

$$ICL(Q, L) = \log p(X, \hat{Z}, \hat{W}; \hat{\theta}) - \frac{Q-1}{2} \log N +$$
$$-\frac{L-1}{2} \log M - \frac{QL}{2} \log(NM) - \frac{1}{2} \log(NM).$$

  The pair $(\hat{Q}, \hat{L})$ that leads to the highest value for the ICL is considered as the most likely cluster numbers for the considered slice of data $X_{t_{init}}$. Remark that, unless a further specific notice, the slice $X_{t_{init}}$ considered for this step in our experiments will be the first slice of the data, i.e. X_{t_0}.

- Second, in order to initialize our VEM-SGD algorithm (see Algorithm 3) with useful initial values for model parameters, we initiate a cascade process as follows in order to propagate the parameter estimates obtained on the slice $X_{t_{init}}$ to the following slices. Fixing for the moment the numbers of row and column clusters to $(\hat{Q}, \hat{L})$, we run the static version of our Zip-dLBM algorithm on the next slice $X_{t_{init}+1}$ with the parameters $\hat{\theta}_{t_{init}}$ as initial values. Then, the estimated parameters $\hat{\theta}_{t_{init}+1}$ are used as initialization of the static Zip-dLBM on the slice $X_{t_{init}+2}$, and so on for the following slices. This strategy allows to provide initial values for all model parameters $\hat{\theta}(t)$, for $t = 1, ..., T$.

- Finally, as we expect that the choice of $\hat{Q}$ row and $\hat{L}$ column cluster components could not be the best for all slices of the dataset, the VEM-SGD algorithm (see Algorithm 3) will be then run with more components than considered in the initialization. Indeed, we run the VEM-SGD algorithm with $Q_{max} \geq \hat{Q}$ and $L_{max} \geq \hat{L}$ cluster components. Then, part of the model parameters are initialized with $\hat{\theta}(t)$ obtained via the initialization procedure described above (see Algorithm 2) and the remaining parameters, corresponding to the additional row and column clusters are set to zero. Thus, we aim at exploiting the potential "blessing" of the use of deep neural networks allowing our VEM-SGD algorithm to start with some empty clusters. These empty clusters will have the possibility to be activated later in the inference process, if needed. Therefore, we avoid the usual computationally demanding procedure of running the whole algorithm with all pairs of row and column cluster numbers for the whole dataset. This strategy allows our approach to scale to massive datasets in a reasonable computation time and with satisfying results, as it will be illustrated in the next section. Similar solutions in a Bayesian framework have been proposed by Malsiner-Walli et al. (2016) in their work on model-based clustering based on sparse finite Gaussian mixtures and by Forbes et al. (2019) where two strategies are proposed for selecting the number of components in non-Gaussian mixture models.

Algorithm 2 Initialization algorithm

```
  Step 1: Static model selection
Require: X, Q_min, Q_max, L_min, L_max, max_iter, n.sim.
  for Q =Q_min, to Q=Q_max do
    for L =L_min, to L=L_max do
      Initialize randomly α, β, π, Λ;
      Run a static version of Zip-dLBM(Q, L) on t = 1, computing the ICL;
    end for
  end for
  Obtain Q* and L* that gives the highest ICL value.
  Step 2: Cascade Process
Require: X , Q*, L*, max_iter.
  for t = 1 to T do
    if t =1 then
      Initialize randomly α(t = 1), β(t = 1), π(t = 1), Λ;
      Run a static version of Zip-dLBM(Q*, L*) on t = 1;
      Store the results α(t = 1), β(t = 1), π(t = 1), Λ.
    else
      Initialize α(t − 1), β(t − 1), π(t − 1), Λ;
      Run a static version of Zip-dLBM(Q*, L*) on t;
      Store the results α(t), β(t), π(t), Λ.
    end if
  end for
```

4.4 Numerical experiments

The main purpose of this section is to highlight the most important features of our zero-inflated dLBM algorithm over simulated datasets in the Poisson scenario. We aim at demonstrating the validity of the inference algorithm and model selection criterion presented in the previous sections. The first experiment consists into applying Zip-dLBM to a specific dataset with evolving block pattern and sparsity to show that it recovers the data structure. The second experiment shows that Zip-dLBM is able to uncover clusters being initially empty, filling up over time, then emptying again. The third experiment shows the robustness of Zip-dLBM when the initial number of clusters is not the actual one, thus testing the performance of the model in case of poor initialization. The fourth experiment demonstrates the model selection procedure on 50 simulated date sets. All the experiments on simulated data were realized on datasets with $N = 600$ rows, $M = 400$ columns and $T = 50$ time instants. Then, in Appendix B.6, other experiments on simulated data are proposed, namely a benchmark study and an experiment to asses the robustness model assumption.

4.4.1 Introductory example

As a first example, we simulate a dataset with dimension $600 \times 400 \times 50$ and with $Q = 3$ groups of rows, $L = 2$ groups of columns. The level of sparsity ranges from 80% to 90% in the time period. The values of the other simulated parameters in this experiment are shown in Table 4.1.

Cluster	α	β
1	0.2 to 0.8	0.1 to 0.99
2	0.18 to 0.14	0.99 to 0.1
3	0.6 to 0.06	-

$$\Lambda = \begin{bmatrix} 6 & 4 \\ 1 & 2 \\ 7 & 3 \end{bmatrix}$$

Table 4.1: Mixing proportion and Λ values.

We apply Zip-dLBM to the simulated dataset with the actual values of Q and L to show the ability of the model to fully recover the model parameters. The running time for this experiment is 23.5 minutes on CPU (see Appendix B.5 for details).

Figure 4.2 shows the evolution of the the lower bound, expressed in Eq. (4.16), that Zip-dLBM aims to maximize. We notice that the convergence is reached in less than 10 iterations. It is worth noticing that each iteration of the algorithm involves the optimization through gradient descent of the loss for 2000 epochs, for each parameter. Figure 4.3, displays the reorganized incidence matrices at time instants $t = 10$ and $t = 30$, respectively: the rows and columns of the incidence matrix are permuted according to the estimates of the latent variables $\hat{Z}$ and $\hat{W}$, in such a way that nearby rows (columns) belong to the same cluster of rows (columns). The blocks are also delimited by black dashed lines. The density of points within each block is determined by the intensity function of the Poisson distribution, represented by the parameter matrix Λ. The estimated values of Λ are as follows:

$$\Lambda = \begin{bmatrix} 3.001 & 7.001 \\ 2.000 & 1.004 \\ 3.990 & 5.998 \end{bmatrix}$$

In this example Figure 4.4 shows the evolution of the estimated mixture parameters $\hat{\alpha}$, $\hat{\beta}$ and $\hat{\pi}$ along the time period, represented on the x-axes. These parameters are estimated via stochastic gradient descent, linked with ODEs integration. By looking at these figures, we see the true parameters on the left column, the output of the initialization procedure in the middle and the results the Zip-dLBM estimates on the right. The comparison between the simulated and estimated parameter evolution shows that the model fully recovers the actual values over time, modulo the switched labels for the mixture proportions.

As expected, our algorithm succeeds to recover the simulated pattern. The similarity between the estimated and simulated values validates the accuracy of our modeling approach in capturing the underlying patterns and characteristics of the data, modulo the switched orders. Furthermore, to evaluate the quality of the clustering, we use a measure called CARI, recently introduced by Robert et al. (2020). This new criterion is based on the Adjusted Rand Index (Rand, 1971) and it was developed especially for being applied to co-clustering methods. The closer the index is to 1, the more both the row and column partitions are close to the actual ones, whereas the closer the value is to 0, the greater the difference between the true and estimated labels. In this experiment we obtained a CARI index of 1. From these results we can clearly see that our algorithm perfectly identifies the composition of the original clusters and it recovers the evolution of the mixing proportion over time.

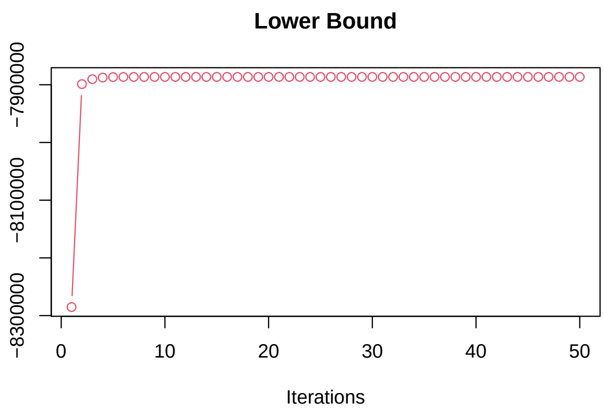


Figure 4.2: Lower bound maximization throughout the iterations of the Zip-dLBM algorithm.

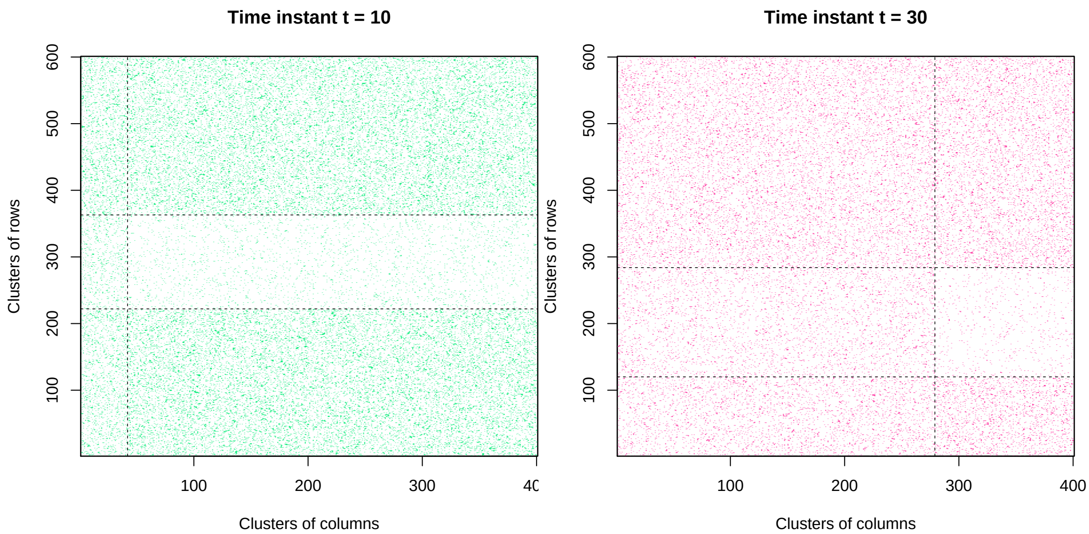
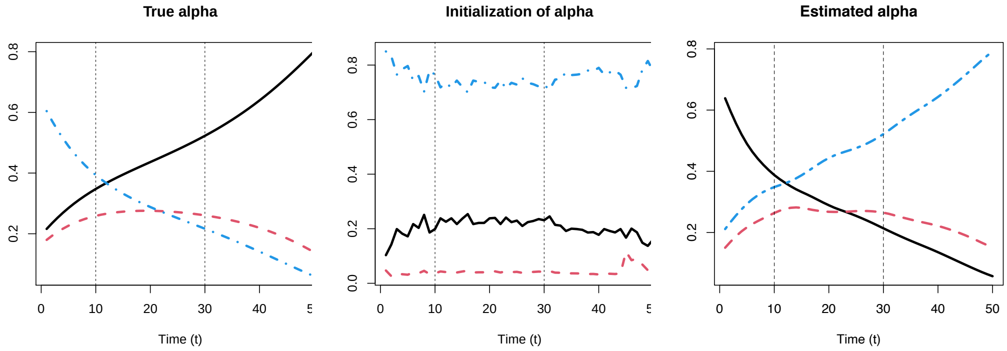


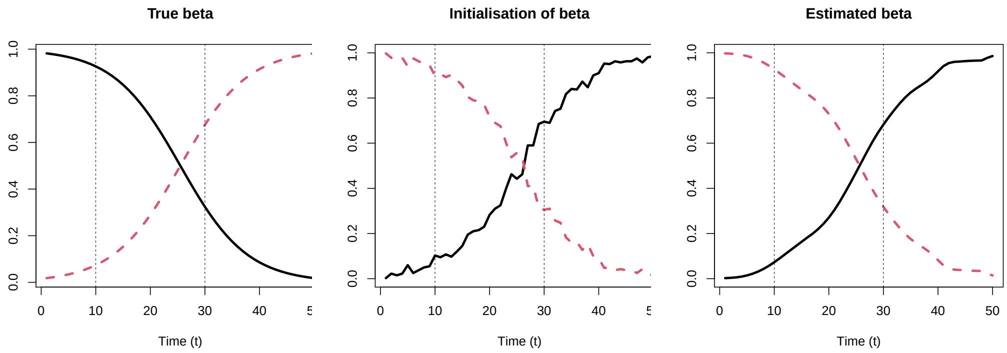
Figure 4.3: Reorganized incidence matrices at time instants $t = 10$ and $t = 30$ according to the estimates of the cluster memberships. Nearby rows (columns) belong to the same cluster of rows (columns). The blocks are also delimited by black dashed lines.

4.4.2 Robustness of the initialization procedure

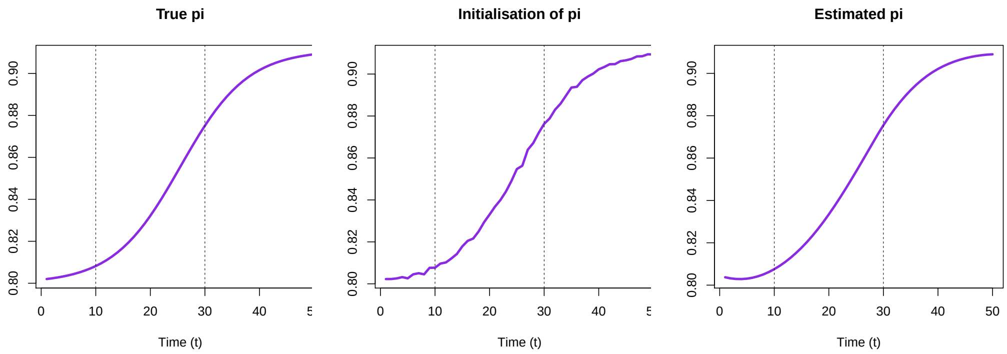
In this section we perform two experiments to test the robustness of the model to initialization. In the first experiment, we initialize parameters with a wrong number of clusters, while in the second experiment the data are simulated with



(a) True α. (b) Initialization of α. (c) Estimated α.



(d) True β. (e) Initialization of β. (f) Estimated β.



(g) True π. (h) Initialization of π. (i) Estimated π.

Figure 4.4: Evolution of the true (left), initialized (center) and estimated (right) proportions of the parameters α, β and π, respectively. Each curve represents the evolution of a column (row) cluster proportion.

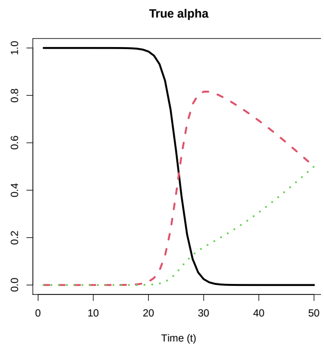
particularly complex dynamics. In fact, we test the model's ability to identify a cluster that is empty at the beginning of the period, which fills up and then empties again. As for the first experiment, in order to test the robustness of our initialization strategy, Zip-dLBM was intentionally initialized with a higher than the optimal number of clusters. In fact, although the data were simulated with $Q = 3$ and $L = 2$, both the initialization process and Zip-dLBM were run with $Q = 5$ and $L = 4$. Figure 4.5 shows on the left column the evolution of the simulated mixture proportions and the sparsity parameter, in the middle column their initialization, and on the right column the results of the estimates provided by Zip-dLBM. We can see that the initialization of α in Figure 4.5b is rather poor. Nevertheless, Zip-dLBM finds the right trend of the mixture proportions over time, effectively emptying the two superfluous clusters.

In this experiment a CARI index value was calculated for each time instant; the obtained CARI index is 0.98.

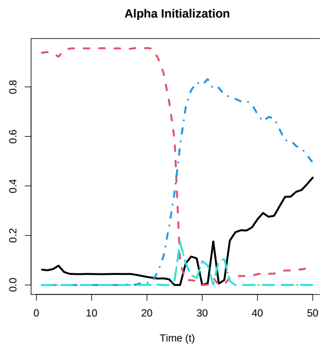
Now, as for the second experiment of testing the robustness of the model to initialization, we simulate the data in such a way that in the clusters in line there is one that is empty at the beginning of the period under consideration, then fills up towards the middle of the period, and then empties again at the end, Figure 4.6a shows this dynamic. Through this experiment we want to show how Zip-dLBM is able to find the right evolution of mixture proportions despite the complex dynamics. Figure 4.6 depicts the simulated parameters on the left, the initial estimates in the middle, and the final estimates on the right. Looking at the middle part, in Figure 4.6b, we can see that the initialization process is not particularly helpful to the model because of the switched labels and a poor parameters estimation. Despite the initialization, we see in Figure 4.6c that Zip-dLBM is perfectly able to recognize the initially empty cluster which then gradually fills up and empties again. In this experiment, the CARI index, obtained by averaging the indexes over time, is 0.95.

4.4.3 Model selection experiment

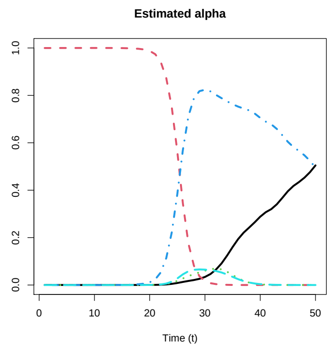
The previous experiments have allowed us to attest that the initialization strategy is globally robust and that the application of Zip-dLBM allows us to correct for poor initializations with respect to the number of clusters in rows or columns. Therefore, in this experiment we test the global capability in choosing the optimal number of clusters in rows and columns over a larger number of simulated datasets through



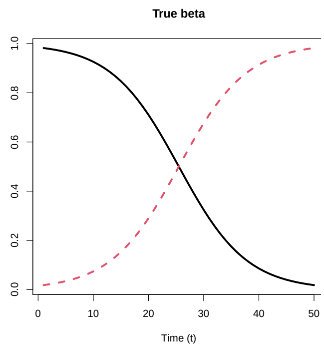
(a) True α.



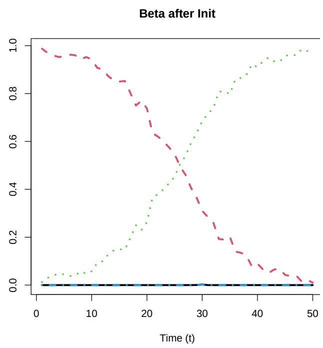
(b) Initialization of α.



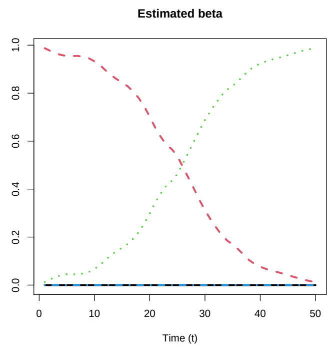
(c) Estimated α.



(d) True β.

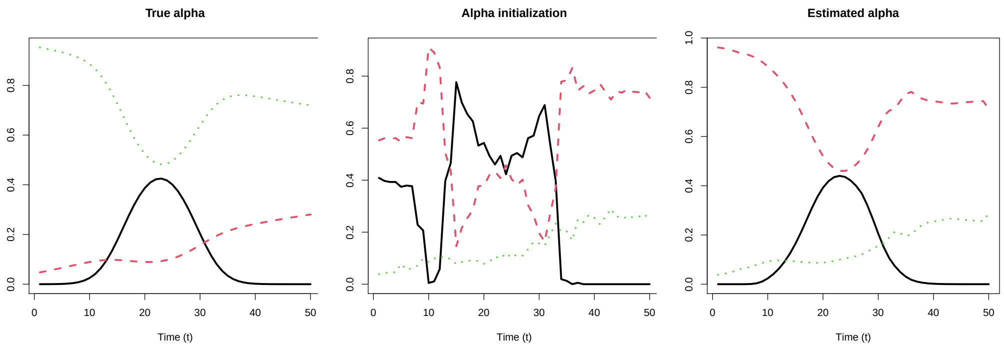


(e) Initialization of β.

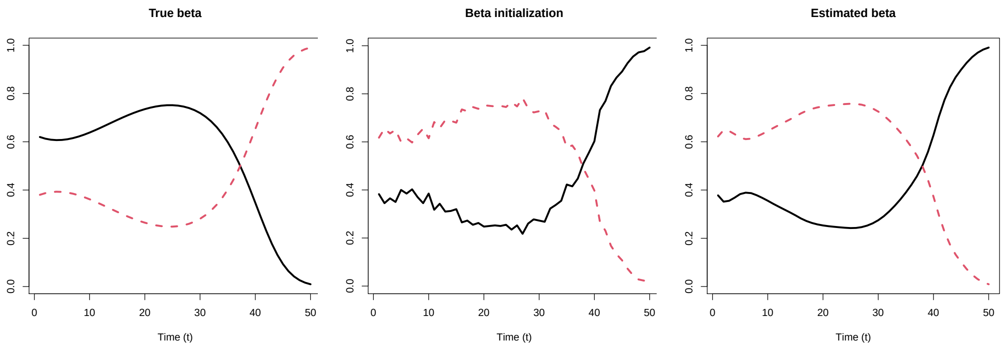


(f) Estimated β.

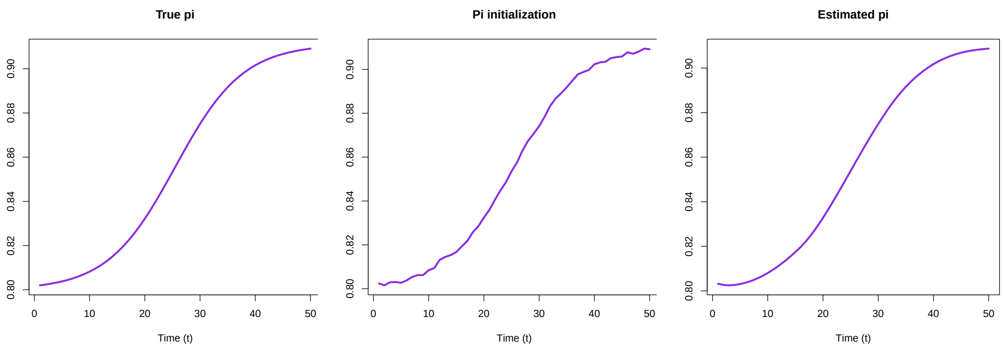
Figure 4.5: Evolution of the true (left), initialized (center) and estimated (right) group proportions of the parameters α and β, respectively. Each curve represents the evolution of a column (row) cluster proportion.



(a) True α. (b) Initialization of α. (c) Estimated α.



(d) True β. (e) Initialization of β. (f) Estimated β.



(g) True π. (h) Initialization of π. (i) Estimated π.

Figure 4.6: Evolution of the true (left), initialized (center) and estimated (right) proportions of the parameters α, β and π, respectively. In figures (a) to (f), each curve represents the evolution of a column (row) cluster proportion, while figures g to i represent the true (left), initialized (center) and estimated (right) parameter π.

the combination of the initialization procedure and the application of the Zip-dLBM algorithm. Let us recall that, as mentioned in Section 4.3, the ICL criterion identifies the optimal number of clusters only at one time instant in order to initialize the parameters optimally. Subsequently, Zip-dLBM is run with a higher number of clusters than those identified by ICL. Hence, to validate the performances on the component activation, 50 independent datasets are generated with the setup explained in Section 4.4.1, with $Q = 3$ row clusters and $L = 2$ column clusters, a level of sparsity varying between 80% and 90% and the other model parameters equal to:

Cluster	α	β
1	0.2 to 0.8	0.1 to 0.99
2	0.18 to 0.14	0.99 to 0.1
3	0.6 to 0.06	-

$$\Lambda = \begin{bmatrix} 6 & 4 \\ 1 & 2 \\ 7 & 3 \end{bmatrix}$$

Table 4.2: Mixing proportion and Λ values.

Then, Zip-dLBM is applied on those simulated datasets using values of Q and L equal to 10. Table 4.3 shows the percentage of selections. The highlighted cell corresponds to the actual value of Q and L. Zip-dLBM succeeds 86% of the time to identify the correct model. Specifically, to evaluate the results of this experiment, we averaged the membership probability of the two estimated mixing parameters, α and β; exceeding clusters having an average membership probability of less than 1e-3 were considered to be off. Among the results of the 50 simulated datasets, we report in Figure 4.7, as an illustrative example, one of the component activation results. We see that not only the unnecessary clusters remained empty, but also the estimates of the α and β are good, as Zip-dLBM manages to identify the evolution of the two mixing parameters over time, despite the number of clusters given as input is not the optimal one.

4.5 London bike sharing

This section focuses on the application of Zip-dLBM to a large-scale bike-sharing dataset in London, with the aim of illustrating the potential of our tool on a real dataset.

Q/L	1	2	3	4	5	6	7	8	9	10
1	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0
3	0	86	0	0	0	0	0	0	0	0
4	0	2	0	0	0	0	0	0	0	0
5	0	2	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0
8	0	4	0	0	0	0	0	0	0	0
9	0	2	0	0	0	0	0	0	0	0
10	0	4	0	0	0	0	0	0	0	0

Table 4.3: Model selection. Percentage of activated components on 50 simulated datasets. The highlighted cell corresponds to the actual value of Q and L.

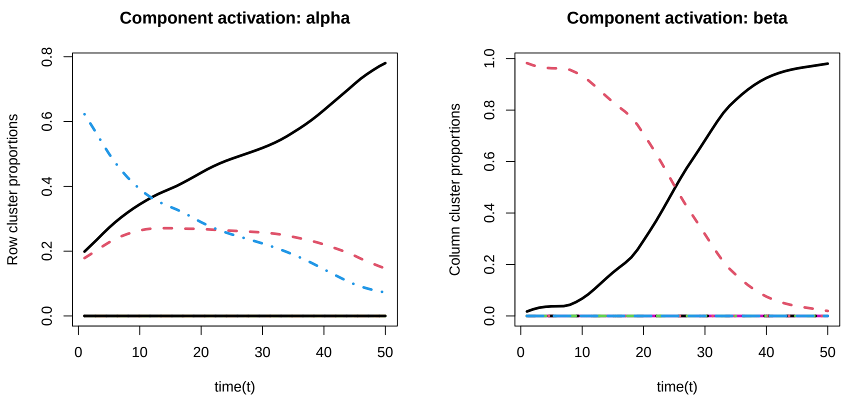


Figure 4.7: View of a model selection result: the useful clusters are activated while the useless ones lie empty on the basis of the figure.

4.5.1 Protocol and data

The data are collected and publicly distributed by Transport for London[3]. We focus on one-month, specifically June 2022, as it represents in our view a neutral choice for the use of shared bikes, both regarding the end of Covid pandemic and the weather conditions. The objective of this application is to analyze how inbound (Arrival) and outbound (Departure) bike rental stations take on different roles, and, consequently, different cluster memberships, depending on the hour

[3] https://cycling.data.tfl.gov.uk, also available on GitHub at: https://github.com/giuliamar95/Zip-dLBM

of the day. To analyze the data, we summed up the hourly interactions on the working days of the month, in order to order to obtain a "cumulative" day that we consider from 6 am to 10 pm. So, the time unity measure is one hour and the overall dataset is made of by 776,270 observations, for which we consider the departure station name, the arrival station name and the start time date. Moreover, we only consider stations (arrival and departure) that were rented more than 50 times over the month of June 2022. The resulting dataset contains 791 departure stations, 791 arrival stations and 16 hours corresponding to 475,586 non-zero entries in the incidence matrix. Figure 4.8 represents the number of bikes in use in the London sharing system at every hour over the whole month of June. It can be clearly noticed that there are two peaks at the rush hours when people go to or from work (7-8 am and 5-6 pm).

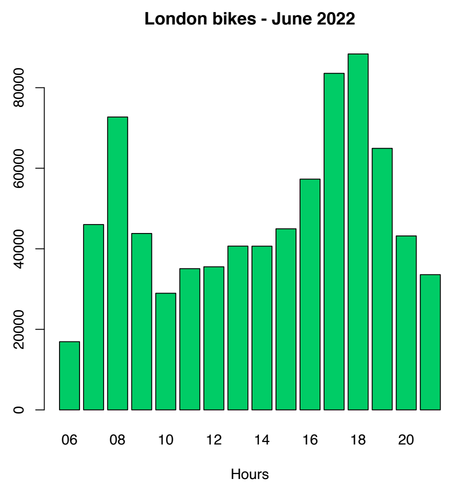


Figure 4.8: Barplot of the number of bikes taken from the London bike sharing system, from 6 am to 10 pm, in a cumulative day, corresponding to June 2022.

4.5.2 Summary of the results

We fit Zip-dLBM to the data with a running time of 22.3 minutes on CPU (see Appendix B.5 for details). For the initialization, as explained in Section 4.3.4, we computed the ICL criterion on one data slice, corresponding on the hour 9 am - 10 am, where the optimal number of clusters identified by the model selection criterion are $\hat{Q} = 6$ and $\hat{L} = 6$. Then, we initiated the model parameters through the cascade process described in Algorithm 2 and we ran Zip-dLBM with

$Q = 10$ and $L = 10$ to allow the model to fill or empty clusters as needed. Figure 4.9 represents the estimated Poisson intensities Λ for Zip-dLBM. This figure only focuses on the 6 groups of departure stations, denoted by the letter D, and the 6 groups of arrival stations, denoted by the letter A, that have been activated in the inference. Each color refers to a departure (rows) or arrival (columns) cluster and the higher is the value in each block, the strongest is the relationship (i.e the expected number of bikes exchanged in the time unit) between the related pair of clusters. Looking at this figure, it can be seen that

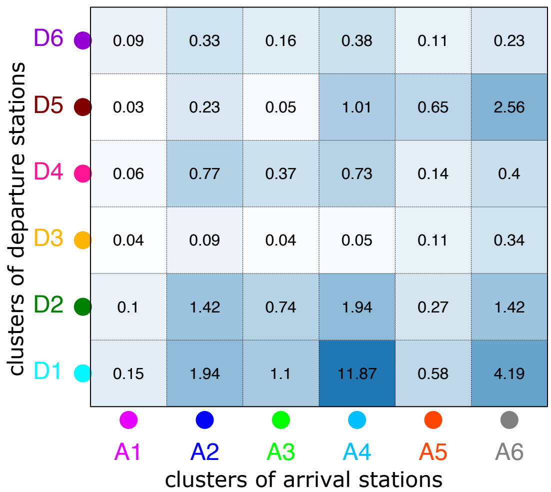


Figure 4.9: Estimated Poisson intensity parameter, each color represents a different departure (arrival) cluster.

some clusters are strongly related whereas others are not at all. For example, cluster D1 (light blue) of the departure stations is highly related with clusters A4 and A6 of arrival stations. Also, cluster D5 of departure stations shows the same behavior but with lower intensity levels. Hence, one may think that the arrival clusters A4 and A6 are located in the city center or in highly busy areas. In fact, looking at the coordinate belonging to cluster D1 of the departure stations, they are mostly concentrated at central locations and, more specifically, at the stations emitting the highest number of bikes, such as Hyde Park, King's Cross, and Queen Elizabeth Olympic Park. We also noted that during the month of June 2022 in Hyde Park and Queen Elizabeth Olympic Park several major events were held, such as the Rolling Stones concert (Hyde Park) and the Red Hot Chili Peppers concert (Queen Elizabeth Olympic Park). This suggests that this cluster includes stations that are subject to a higher-than-normal load. Also, we can see a particular behavior of cluster D2 (green) of outgoing bikes. This cluster is characterized by bikes leaving the suburbs during the morning peak hours (7 am-9 am) and then concentrating more in the city center during the day and especially in the evening rush hours (4 pm-6 pm). In addition, this cluster has a strong relationship with clusters A2, A4 and A6 of arrival stations. Among them, clusters A4 and A6 represent mostly central stations, while cluster A2 is concentrated in the city center in the morning rush hours and in the suburbs in the evening rush hours. We might infer that this dynamic is typical of workers who decide to bike to their workplaces. On the contrary, clusters D3 and D6 of departure stations have a very low intensity of interactions with the arrival clusters, however they are really spread all over the city. In particular, cluster D3 has the highest level of interactions with cluster A6 of arrival stations, while cluster D6 has the highest intensity level with cluster A2 and cluster A4 of arrival stations. The main difference between clusters D3 and D6 concerns their location. In fact, all over the day, there are really few points belonging to cluster D3 and they are mainly concentrated in the Greenwich peninsula and, only at the end of the day, in the city center. Cluster D6, on the contrary, concentrates in the city center and in some specific areas, such as Greenwich peninsula, Wandworth and Sheperd's Bush early in the morning, then it spreads all over the city. Therefore, from the large initial data matrix, Zip-dLBM was able to identify consistent and relevant clusters of the London sharing bike stations.

4.5.3 Interpretation of the estimated parameters

To better understand the results, we now focus on the estimates of the other model parameters. Figure 4.10 shows the estimated evolution of the sparsity parameter over time. We see that, at the beginning of the day, 6 am, the sparsity is at 95%, then as we approach the morning peak, the number of borrowed sharing bikes increases and consequently the sparsity decreases, reaching 86% at 8 am. Between 9 am and 2 pm, it again increases slightly (90%) and then decreases as we approach the peak at the end of the day, when workers leave work. In fact, at 6 pm the sparsity level reaches its daily minimum at a level of 80%.
Figures 4.11a and 4.11b show the estimation of the mixing parameters α and β. From Figure 4.11a, we see how cluster D2 has a precise evolution, corresponding to the daily work rhythm. Cluster D4 and Cluster D6 on the contrary, have exactly

the opposite behavior, filling up in the non-rush hours of the day, starting at 10 am, then emptying out in the rush hour of the afternoon and filling up again in the evening. We might infer that the outbound bikes in these clusters are mostly rented by tourists. Cluster D1, on the other hand, as mentioned earlier, is very peculiar because it contains a few stations that emit a lot of bikes, probably when there are special events that mobilize a large number of people. (e.g concerts, football matches, etc.)
Looking at Figure 4.11b instead, we see estimates of the mixing proportions of the arrival bike clusters. Cluster A2 again has the typical workday evolution, thus including those bike stations taken to get to the workplace and back home at the end of the day. Another particular group is cluster A3. Its proportions increase from 10 am reaching a peak between 2 pm and 3 pm, and while all other clusters tend to empty out in the evening, this one increases. Cluster A5 fills in mid-morning and then remains stable during the rest of the day. Clusters A4 and A6, on the other hand, are very small in terms of proportions. As we saw earlier, they in fact include few but very central stations with a very high density of interactions.

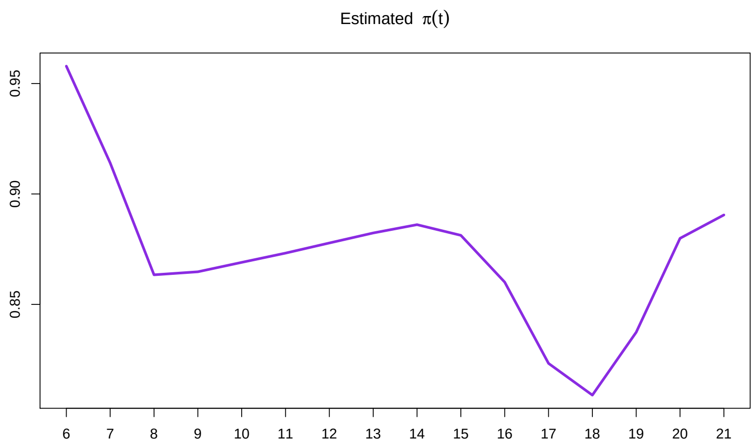
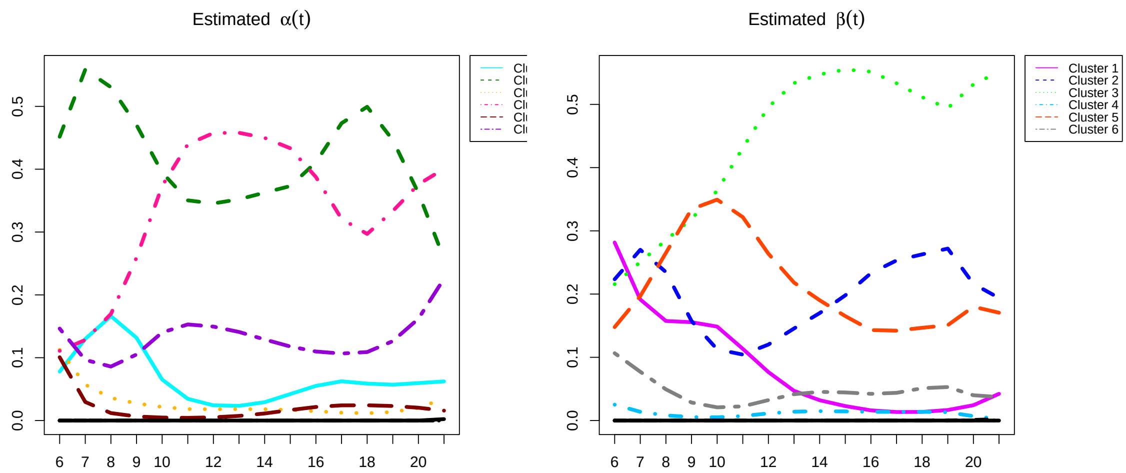


Figure 4.10: Evolution of $\hat{\pi}$.

4.5.4 Insights into the evolution of two departure and arrival clusters

For pedagogical purposes, in this section we choose two clusters, specifically the departure bike cluster D2 and the arrival bike cluster A2, by analyzing the results

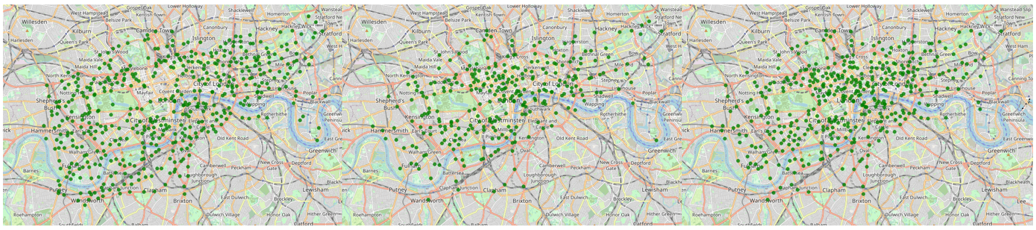


(a) Evolution of $\hat{\alpha}$.

(b) Evolution of $\hat{\beta}$.

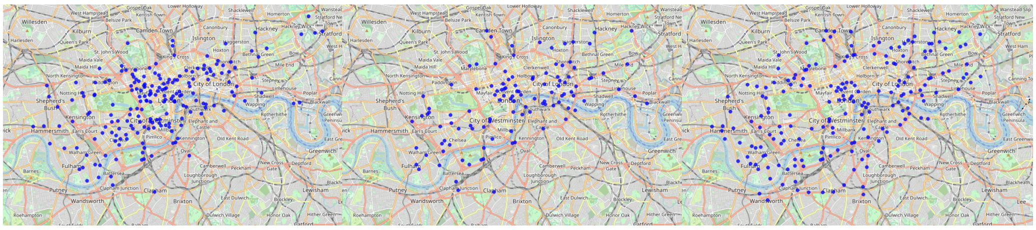
Figure 4.11: Evolution of the estimates $\hat{\alpha}$ and $\hat{\beta}$. Each color represents a different cluster.

in detail. Figure 4.12 depicts cluster D2 of the departing stations and its evolution in 3 time instants: 7 am, 1 pm and 6 pm. As mentioned, given its specific dynamics, this cluster is likely to be populated by bicycle stations used mostly by workers. In fact, in Figure 4.12a we see that at 7 am there are many points, mostly distributed outside the city center. Thereafter, around 1 pm (Figure 4.12b) the volume of points decreases. Then, toward the afternoon rush hour, in Figure 4.12c, the density of points increases again. Indeed, in Figure 4.12c we see that most of the points are now located in the center. This behavior is typically due to the workers from the suburbs going to work in the city center while at the end of the day taking the shared bikes back to leave the city center.
Similarly, Figure 4.13 shows the cluster A2 of the arrival stations and its evolution in 3 time instants: 7 am, 1 pm and 6 pm. Here we note how at 7 am cluster A2 contains arrival stations all located in the city center. In contrast, in Figure 4.13b, the number of bikes belonging to this cluster decreases and the locations are more scattered. Whereas, in Figure 4.13c, we note how at the end of the workday, most of the arrival stations in this cluster are no longer located in the center but in the suburbs. Thus, from the comparison of Figure 4.12 and Figure 4.13 we notice a complementary trend, dictated by the fact that the arrival and departure stations in the two selected clusters are both characteristic of the daily work schedules.



(a) Cluster D2 - 7 am. (b) Cluster D2 - 1 pm. (c) Cluster D2 - 6 pm.

Figure 4.12: Snapshots of the evolution of the departure cluster D2 at three different times in the day: 7 am, 1 pm, 6 pm.



(a) Cluster A2 - 7 am. (b) Cluster A2 - 1 pm. (c) Cluster A2 - 6 pm.

Figure 4.13: Snapshots of the evolution of the arrival cluster A2 at three different times in the day: 7 am, 1 pm, 6 pm.

4.6 Analysis of the adverse drug reaction dataset

This section focuses on the application of Zip-dLBM to a large-scale pharmacovigilance dataset, similar to the one considered in 3, with the aim of illustrating the potential of the tool.

4.6.1 Protocol and data

We recall that the dataset consists of adverse drug reaction (ADR) declarations, collected by the Regional Center of Pharmacovigilance (RCPV), located in the University Hospital of Nice (France). A time horizon of 7 years is considered, from January 1^{st}, 2015 to March 3^{th}, 2022, the unity measure for the time interval is a trimester. The overall dataset is made of 27,754 declarations, for which the market name of the drug, the notified ADR and the reception date are considered. Moreover, we only considered drugs and ADRs that were notified more than 20 times over the 7 years. The resulting dataset contains 236 drugs, 324 ADRs and 29 time intervals with 12,336 non-zero entries. Looking at Figure 4.14, it can be clearly noticed that there are two peaks, one in 2017 and the other in 2021.

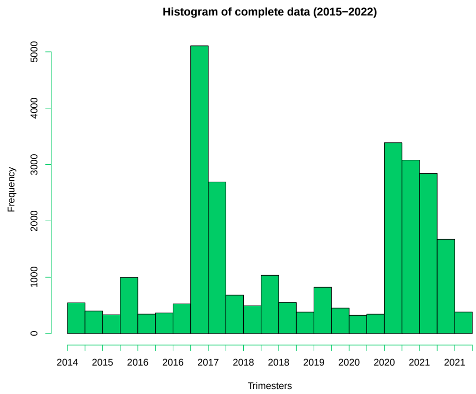


Figure 4.14: Number of declarations received by the pharmacovigilance center from 2015 to 2022, sorted by trimester.

In 2017, an unexpected rise of reports for ADRs happened concerning a specific drug called Lévothyrox®. This has been marketed in France for about 40 years as a treatment for hypothyroidism and, in 2017, a new formula was introduced on the market. The Lévothyrox® case had a huge media coverage in France: Lévothyrox® spontaneous reports represent the 90% of all the spontaneous notifications that the RCPV received in 2017 (Viard et al., 2019). In addition, since the end of the year 2020, vaccinations against Covid-19 have been introduced. At that time, three vaccines are licensed in Europe, Comirnaty® was the first Covid-19 vaccine available in France in December 2020, followed by Moderna® in January 2021 and Vaxzevria® in February 2021. From Figure 4.14, one can understand the difficulty to work with such data which contain signals of very different amplitude. Indeed, behind those very visible effects, many ADR signals need to be detected for obvious public health reasons. In particular, those data also contain ADR reports regarding another health scandal happened in 2017, involving Mirena®, which is here far less visible than Lévothyrox®, but also led to many avoidable serious health issues.

4.6.2 Summary of the results

To the initialize the algorithm, as explained in Section 4.3.4, we computed the ICL criterion on one data slice, corresponding to the first trimester, where the optimal numbers of clusters identified by the model selection criterion are $\hat{Q} = 4$ and $\hat{L} = 4$. Then, we initiated the model parameters through the cascade process described in Algorithm 2 and we ran Zip-dLBM with $Q = 7$ and $L = 7$ to allow the model

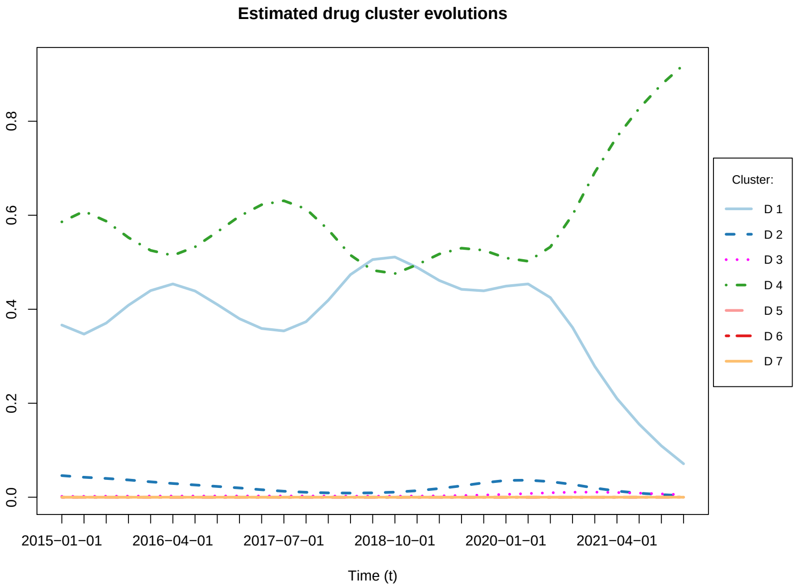


Figure 4.15: Evolution of the estimates $\hat{\alpha}$.

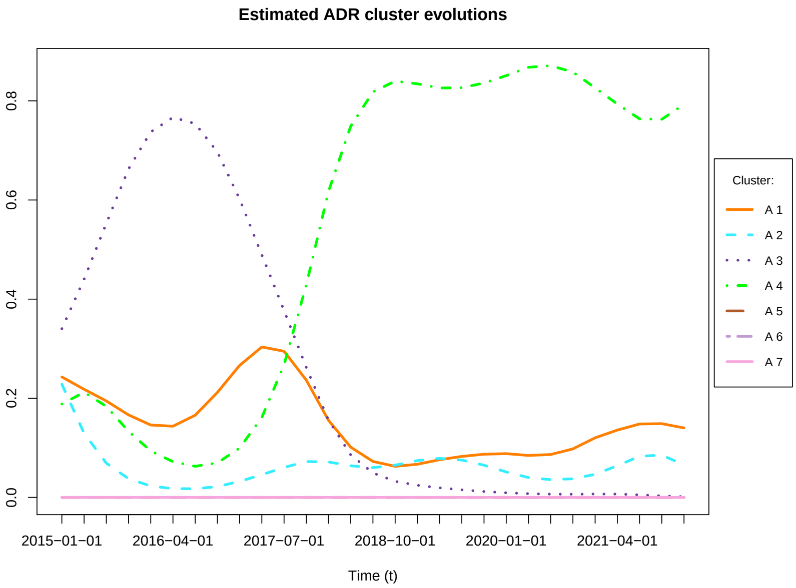


Figure 4.16: Evolution of the estimates $\hat{\beta}$.

to fill or empty clusters as needed. Figure 4.17 depicts the estimated Poisson intensities Λ for Zip-dLBM, focusing on 4 drug clusters (D) and 4 ADR clusters (A) that are activated during the inference. Each color represents a drug or ADR cluster, with higher values indicating stronger relationships (i.e., expected number of declarations received per time unit) between the respective clusters. The figure reveals varying degrees of association, for example, cluster D3 of the drug clusters is highly related with cluster A1 of ADR clusters. Figures 4.15, 4.16 and 4.18 show

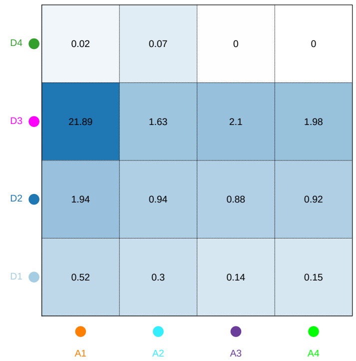


Figure 4.17: Estimated Poisson intensities.

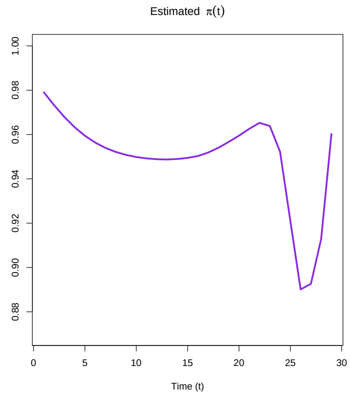


Figure 4.18: Evolution of the estimates $\hat{\pi}$.

the estimates of the model parameters $\hat{\alpha}$, $\hat{\beta}$ and $\hat{\pi}$, respectively. Figure 4.15 shows the estimation of the mixing parameter α. Cross-referencing the information from these results, we note that the clusters that have the highest intensity are also the less populated. For example, cluster D3 of drugs has a very high intensity of interactions with cluster A1 of adversarial effects, yet cluster D3 turns out to be very small in Figure 4.15. This is due to the fact that this cluster contains drugs that are declared with an an unusually high intensity. In fact, this cluster contains the drugs that are the causes of the major health crises that occurred during the reporting period: Mirena® in the first half of 2017, Lévothyrox® in the second part of 2017, and Covid-19 vaccines throughout 2021. Similarly, by analyzing the composition of cluster A1, it is possible to identify which ADRs were the most reported in each of the aforementioned crises. For instance, the most reported side effects during the Mirena® health crisis are mostly hormonal ones, such as anxiety, heat shock, and aggressive behavior. Then, looking at Figure 4.16, during the Lévothyrox® health crisis we notice a peak in the A1 cluster of adversarial effects, probably because the great media coverage that the scandal had in those years made people declare the most disparate side effects. Also, we see that in 2021 there is another peak, corresponding to the period of the Covid-19 vaccination. Here, the adversarial effects found in cluster A1 are mostly linked to problems related to the vaccination site (e.g. arm pain, arm inflammation, skin reaction) and flu syndrome as a result of the vaccine. Cluster D2, on the other hand, contains a few but very common and, consequently, much-reported drugs, for example, paracetamol and some of the most popular anticoagulants. From Figure 4.17 we note that this cluster has a stronger intensity of interactions with cluster A1 and

A2 of undesirable effects. Looking at Figure 4.16, we note that cluster A2 is thinly populated and seems to follow the trend of health crises discussed above less closely. In fact, this cluster contains less severe and more common adversarial effects, which can occur even with the more frequent medications (e.g., itching, headache, weight gain, etc.). Clusters D1 and D4, on the other hand, are characterized by very low interaction intensities and are densely populated by all other drugs. Then, looking at Figures 4.17 and 4.16, we see that the behavior of cluster A3 of adversarial effects is very peculiar. It is characterized by almost zero interaction intensity with drug clusters D1 and D4. After the Lévothyrox® crisis, the number of reported adversarial effects significantly decreased, indicating a turning point in pharmacovigilance as people became more aware of its importance and started reporting side effects more frequently. Moreover, analysing its composition, it was noticed that at the beginning of the period it also contained all the specific side effects of Covid-19 vaccines, which were not yet known. Later, in 2021, those side effects, changed clusters moving to cluster A1 as previously described. On the other hand, Figure 4.18 shows the estimated evolution of the sparsity parameter over time. We see that, at the beginning of the period, in 2015, the sparsity is at 98%, then as we approach the 2017 peak, the number of declarations increases and consequently the sparsity decreases. In 2019, it again increases slightly (97%) and then decreases as we approach the peak due to the Covid-19 vaccines. In fact, at the beginning of 2021 the sparsity level reaches its minimum at a level of 90%. Therefore, from the large initial data matrix, Zip-dLBM was able to identify meaningful clusters of such data.

4.6.3 Benchmark on real data

This section focuses on comparing Zip-dLBM with state of the art models on real-world data. We therefore carried out such an experiment by comparing Zip-dLBM with Zip-dLBM$_{\pi(\cdot)=0}$ and dLBM discussed in Appendix B.6. We also included in the comparison two models that do not consider the dynamic aspect: Poisson LBM, by making use of the bikm1 R package (Robert et al., 2020), baseline for model-based co-clustering methods, and k-means (MacQueen, 1967), applied on rows and columns separately. As we are in an unsupervised context, the model performances are evaluated by the silhouette score using cosine distance on rows and columns. Table 4.4 displays the results of this comparison, in terms of average silhouette scores, reported with standard deviations. From the reported

	Zip-dLBM	Zip-dLBM$_{\pi(\cdot)=0}$	dLBM	kmeans	LBM
Silhouette Score - Rows	**0.37 ± 0.12**	0.31 ± 0.12	-0.46 ± 0.25	0.21 ± 0.36	0.33 ± 0.12
Silhouette Score - Cols	**0.36 ± 0.23**	0.31 ± 0.25	-0.15 ± 0.06	0.31 ± 0.3	0.29 ± 0.23

Table 4.4: Results of Zip-dLBM, Zip-dLBM$_{\pi(\cdot)=0}$, dLBM, LBM and k-means on pharmacovigilance data. Average silhouette scores are reported with standard deviations.

results, one sees that Zip-dLBM outperforms its competitors. Also, it is worth noticing that unlike Zip-dLBM, LBM and k-means, being independently applied at each time instant, suffer from label switching, which is not penalized in the silhouette score. This should make the interpretation of these results even more in favor for Zip-dLBM.

4.7 Conclusion

In this chapter we have proposed a dynamic co-clustering technique, with the purpose of simultaneously performing clustering of rows and columns along the time dimensions. Since observations and features are allowed to change their cluster memberships in time, it is of great interest to look for structural changes in the way clusters interact with each other along the considered time period. We have introduced a generative zero-inflated dynamic latent block model, that can be further adapted to several zero-inflated probability distributions. For ease of reading the mathematical and experimental part, in this chapter we used the Zero-Inflated Poisson distribution, thus introducing the Zero-Inflated Poisson Dynamic Latent Block model (Zip-dLBM). The time modeling relies on three systems of ordinary differential equations. Inference is done using a Variational EM algorithm together with stochastic optimization for the parameters of the dynamic systems. The performance of our approach, called Zip-dLBM in the Poisson case, is evaluated through applications to several simulated data scenarios and compared with competing methods. Then, Zip-dLBM was fitted to two large-scale real datasets, the London sharing bikes and a pharmacovigilance dataset. In this context, Zip-dLBM provided meaningful and interpretable results.

Chapter 5

Deep dynamic co-clustering of count data streams: application to pharmacovigilance

In Chapters 3 and 4 of this thesis, our focus was primarily on dynamic co-clustering approaches that required the entire dataset to be readily available. Here, we introduce the logical evolution of the model. We modify the model inference to enable it to operate in an online mode, eliminating the need for the complete dataset upfront. To achieve this, we make use of LSTM neural networks, which enable us to model the evolution of the model parameters dynamically. Additionally, we add an online change point detection method, facilitating real-time alerts for evolving patterns in the data. As a demonstration of the model effectiveness and real-world applicability, we show its application to real pharmacovigilance data. This chapter presents the results of two research studies:

- G. Marchello, M. Corneli, C. Bouveyron (2023). *Deep dynamic co-clustering of streams of count data: a new online Zip-dLBM*. Proceedings of the 31th European Symposium on Artificial Neural Networks;
- G. Marchello, M. Corneli, C. Bouveyron (2023). *Deep dynamic co-clustering of count data streams: application to pharmacovigilance*. Submitted in a more detailed version to an international journal.

5.1 Introduction

In this chapter we mainly focus on the pharmacovigilance dataset described in Chapter 4. As said in the previous chapters, the detection of safety signals heavily relies on manual expert analysis, leading to potential incompleteness due to the workload involved and the need for substantial data before critical events can be detected. Such methods would enable pharmacovigilance experts to focus on searching for ADRs in all the digital documents and reports generated by healthcare establishments. Consequently, there is a pressing need to develop automated methods for safety signal detection in pharmacovigilance. Clustering techniques can help in this task to effectively summarize data to detect safety signals along the time and online change point detection algorithms can detect when the data generating process has changed and trigger further investigations. In this chapter, we address this challenge by exploring the development of an online model-based co-clustering tool for real-time safety signal detection. By treating adverse drug reaction (ADR) notifications as count data observed over time, our approach allows for the identification of temporal breaks in the safety signals. This facilitates the creation of alerts and provides room for further investigation by medical authorities. The primary objective of this research is to showcase the potential of our proposed method as a routine tool in pharmacovigilance.

To identify temporal breaks in the notifications progress, we rely on change point detection approaches. Change points refer to sudden changes in the underlying process generating the observed data points. Several change point detection algorithms have been proposed in the literature (e.g. Basseville et al. (1993); Cheng and Thaga (2005); Van den Burg and Williams (2020); Montgomery (2020)). A possible classification of these algorithms is into two types: online and offline. Online algorithms are designed to operate in real-time, with upcoming time series data. In contrast, offline algorithms are intended to run after the entire dataset has been collected. Regarding online change point detection algorithms, recent studies have shown that among the most effective methods there are the likelihood and probabilistic approaches (Kondratev et al., 2022; Kavitha and Punithavalli, 2010). On this subject, a seminal paper was proposed by Adams and MacKay (2007) introducing the Bayesian Online Change Point Detection (BOCD). The idea of BOCD is to identify change points using the so-called run lengths or segment. Whenever a new data point becomes available, the algorithm determines the likelihood of the corresponding run length increasing by one. If the probability of change is higher than that of growth, the run length resets to zero, and a change point is identified. Alternatively, another approach has been proposed by Kawahara and Sugiyama (2009), introducing subspace identification for change point detection. Here, the change points are detected by estimating a state-space model behind time series.

5.1.1 Our contribution

This chapter proposes an online extension of Zip-dLBM, introduced in Chapter4, a co-clustering technique designed to handle time-evolving data matrices that may contain many empty entries. We aim to introduce three novelties in this regard. The first is the ability of the estimation algorithm to work online, with streams of data. The second is the addition of an online change point detection method. By capturing the data dynamic behavior, the method can identify abrupt events that affect the generative process. To detect these changes we make use of the Bayesian Online Change Point Detection (BOCD, Adams and MacKay, 2007) that runs on the estimated model parameters in real time. The algorithm iteratively updates the posterior probabilities of the change points, based on the data observed so far. The third novelty relies on a different modeling choice for the time evolving parameter. In fact fully connected neural networks are substituted with LSTMs, as their structure is deemed more appropriate for the purpose. Therefore, this model introduces a new approach by incorporating an online inference method for Zip-dLBM and online changing point detection.

5.1.2 Organization of the chapter

The chapter is organized as follows. Section 5.2 introduces the proposed online inference for stream data. Section 5.3 presents various experiments on simulated data to test and evaluate the model performances. In Section 5.4, an application on the real ADRs dataset is presented to illustrate the potential of Stream Zip-dLBM in pharmacovigilance. Section 5.5 provides some concluding remarks.

5.2 Stream Zip-dLBM

Being the generative model Zip-dLBM, we kindly ask the reader to refer to Section 4.2.1 for a detailed description of the model assumptions. In that Section, the model has been proposed in a general way for any Zero-Inflated distribution. However, here we focus on the Zero-Inflated Poisson (ZIP) distribution. The set of the model parameters is denoted by $\theta = (\Lambda, \alpha, \beta, \pi)$ and the latent variables used so far are Z, W, and A, where we denote $\alpha \triangleq \{\alpha(t)\}_t$, $\beta \triangleq \{\beta(t)\}_t$, $\pi \triangleq \{\pi(t)\}_t$. Thus, the likelihood of the complete data reads:

$$p(X, Z, W, A|\theta) = p(X|Z, W, A, \Lambda, \pi)p(A \mid \pi)p(Z|\alpha)p(W|\beta). \tag{5.1}$$

The terms on the right hand side of the above equation can be further developed, for details see Section 4.2.2.

5.2.1 Online inference for stream data

In this section, we present the online extension of the Zip-dLBM method, called Stream Zip-dLBM. The objective is to perform co-clustering of rows and columns in real-time as new data become available. To prevent memory overload, we have revisited the original inference algorithm of Zip-dLBM, enabling the data to be processed without the need to store it in memory. To allow the algorithm to update the parameter estimates continuously as a new data is incorporated, we use a moving window, $G_d(t)$, of size d. In more detail, at time t, we keep in memory only the data in the interval $[t-d, t]$, namely $X(t-d), X(t-d+1), \ldots, X(t)$, that will be used for the estimation of the model parameters. The data outside the interval can be discarded to prevent memory overloads and maintain the algorithm's functionality. Once a time point t quits the time window (after passing through it) the parameter estimates at that point become fixed, and the "past" data can be discarded by the inference procedure.

5.2.2 Variational inference

As we saw in the previous chapter, we use a combination of Variational-EM inference and Stochastic Gradient Descent (SGD) to estimate the model parameters. We recall that a variational distribution $q(.)$, is introduced over (A, Z, W), in order to decompose the observed data log-likelihood as follows:

$$\log p(X|\theta) = \mathcal{L}(q, \theta) + KL(q(.)||p(.|X, \theta)),$$

where $\mathcal{L}$ denotes a lower bound of $p(X|\theta)$ and is defined as:

$$\begin{aligned}\mathcal{L}(q,\theta) &= \sum_{Z}\sum_{W}\sum_{A} q(Z,W,A)\log\frac{p(X,A,Z,W|\theta)}{q(Z,W,A)} \\ &= E_{q(A,Z,W)}\Big[\log\frac{p(X,A,Z,W|\theta)}{q(A,Z,W)}\Big] \\ &= E_{q(A,Z,W)}[\log(p(X,A,Z,W|\theta)] - E_{q(A,Z,W)}[\log(q(A,Z,W))],\end{aligned} \tag{5.2}$$

and KL indicates the Kullaback-Liebler divergence between the true and the approximate posterior $q(\cdot)$:

$$KL(q(.)||p(.|X,\theta)) = -\sum_{A}\sum_{Z}\sum_{W} q(A,Z,W)\log\frac{p(A,Z,W|X,\theta)}{q(A,Z,W)}.$$

The mean-field assumption is still employed:

$$q(A(t),Z(t),W(t)) = q(A(t))q(Z(t))q(W(t)) = \prod_{i=1}^{N}\prod_{j=1}^{M} q(A_{ij}(t))\prod_{i=1}^{N} q(Z_i(t))\prod_{j=1}^{M} q(W_j(t)). \tag{5.3}$$

5.2.3 Variational E-Step

As shown in Section 4.3.2, denoting by $\delta_{ij}(t) := q(A_{ij}(t) = 1)$ the variational probability of success for $A_{ij}(t)$, the optimal variational update is:

$$\delta_{ij}(t) = \frac{\exp(R_{ij}(t))}{1+\exp(R_{ij}(t))}, \tag{5.4}$$

with:

$$\begin{aligned}R_{ij}(t) = \log(\pi(t)\mathbf{1}_{\{X_{ij}(t)=0\}}) + \sum_{q=1}^{Q}\sum_{\ell=1}^{L}\Big[&- E_{q(W,Z)}[Z_{iq}(t)]E_{q(W,Z)}[W_{j\ell}(t)]X_{ij}(t)\log\Lambda_{q\ell} + \\ &+ E_{q(W,Z)}[Z_{iq}(t)]E_{q(W,Z)}[W_{j\ell}(t)]\Lambda_{q\ell}\Big] + \log X_{ij}(t)! - \log(1-\pi(t)).\end{aligned}$$

Regarding $q(Z)$, let us denote by $\tau_{iq}(t) := q(Z_{iq}(t) = 1)$ the variational probability of success of $Z_{iq}(t)$. The optimal update can be written as:

$$\tau_{iq}(t) = \frac{r_{iq}(t)}{\sum_{q_0=1}^{Q} r_{iq_0}(t)}, \tag{5.5}$$

with $r_{iq}(t)$ is denoted by:

$$\begin{aligned}r_{iq}(t) \propto \exp\Bigg(\sum_{j=1}^{M}\sum_{\ell=1}^{L}\Big\{(1-E_{q(A,W)}[A_{ij}(t)])\Big[&E_{q(A,W)}[W_{j\ell}(t)]X_{ij}(t)\log(\Lambda_{q\ell}) + \\ &- E_{q(A,W)}[W_{j\ell}(t)]\Lambda_{q\ell}\Big]\Big\} + \log(\alpha_q(t))\Bigg).\end{aligned}$$

Similarly for the latent variable W, denoting by $\eta_{j\ell}(t) := q(W_{j\ell}(t) = 1)$ the variational probability for $W_{j\ell}(t)$, the optimal update of $q(W)$ is:

$$\eta_{j\ell}(t) = \frac{s_{j\ell}(t)}{\sum_{\ell_o=1}^{L} s_{j\ell_o}(t)}, \tag{5.6}$$

where :

$$s_{j\ell}(t) \propto \exp\left(\sum_{i=1}^{N}\sum_{q=1}^{Q}\left\{(1 - E_{q(A,Z)}[A_{ij}(t)])\Big[E_{q(A,Z)}[Z_{iq}(t)]X_{ij}(t)\log(\Lambda_{q\ell}) + \right.\right.$$
$$\left.\left. - E_{q(A,Z)}[Z_{iq}(t)]\Lambda_{q\ell}\Big]\right\} + \log(\beta_\ell(t))\right).$$

We recall that the proofs of Eqs. (5.4), (5.5) and (5.6) are provided in Appendix B.1 and B.2. Here, it is worth noting that these update equations can be executed step by step as new data come, allowing for incremental updates of the variational parameters. Also, note that the update in these equations can be computed independently for any pair (i, j), at any time point t.

5.2.4 Online variational M-Step

While the updates in the E-step for $\tau(t)$, $\eta(t)$, and $\delta(t)$ depend solely on the current time instant t, the same cannot be said for the updates in the M-step. The M-step involves updating the model parameters, $\theta = (\Lambda,\ \alpha,\ \beta,\ \ \pi)$, based on the current estimates obtained in the E-step. In order to obtain the updates of the parameter set θ, the objective of the M-Step is the maximization of the lower bound $\mathcal{L}(q, \theta)$ with respect to $\theta = (\Lambda, \alpha, \beta, \pi)$, while holding the variational distribution $q(\cdot)$ fixed. Denoting t as the current time instant, we develop Eq.(5.2) such that the variational lower bound $\mathcal{L}(q, \theta)$ can be written as:

$$\begin{aligned}
\mathcal{L}(q, \theta) = & \sum_{u=1}^{t} \sum_{i=1}^{N} \sum_{j=1}^{M} \Bigg\{ \delta_{ij}(u) \log(\pi(u) \mathbf{1}_{\{X_{ij}(u)=0\}}) + (1 - \delta_{ij}(u)) \Bigg[\log(1 - \pi(u)) + \\
& + \sum_{q=1}^{Q} \sum_{\ell=1}^{L} \Big\{ \tau_{iq}(u) \eta_{j\ell}(u) X_{ij}(u) \log \Lambda_{q\ell} - \tau_{iq}(u) \eta_{j\ell}(u) \Lambda_{q\ell} \Big\} \Bigg] + \\
& - (1 - \delta_{ij}(u)) \log(X_{ij}(u)!) \Bigg\} + \sum_{u=1}^{t} \sum_{i=1}^{N} \sum_{q=1}^{Q} \tau_{iq}(u) \log(\alpha_q(u)) + \\
& + \sum_{u=1}^{t} \sum_{j=1}^{M} \sum_{\ell=1}^{L} \eta_{j\ell}(u) \log(\beta_\ell(u)) - \sum_{u=1}^{t} \sum_{i=1}^{N} \sum_{q=1}^{Q} \tau_{iq}(u) \log(\tau_{iq}(u)) + \\
& - \sum_{u=1}^{t} \sum_{j=1}^{M} \sum_{\ell=1}^{L} \eta_{j\ell}(u) \log(\eta_{j\ell}(u)) - \sum_{u=1}^{t} \sum_{i=1}^{N} \sum_{j=1}^{M} \Big(\delta_{ij}(u) \log(\delta_{ij}(u)) + \\
& + (1 - \delta_{ij}(u)) \log(1 - \delta_{ij}(u)) \Big).
\end{aligned} \tag{5.7}$$

Update of Λ

Here our goal is to derive the online update of the Zero-inflated Poisson intensity parameter, Λ. The variational distribution $q(A, Z, W)$ is kept fixed, while the lower bound is maximized with respect to Λ at every time instant t, to obtain its update, $\hat{\Lambda}$. In Section 4.3.3 we derived the optimal update as:

$$\hat{\Lambda}_{q\ell} = \frac{\sum_{i=1}^{N} \sum_{j=1}^{M} \sum_{u=1}^{t} \tau_{iq}(u) \eta_{j\ell}(u) \Big(X_{ij}(u) - \delta_{ij}(u) X_{ij}(u) \Big)}{\sum_{i=1}^{N} \sum_{j=1}^{M} \sum_{u=1}^{t} \tau_{iq}(u) \eta_{j\ell}(u) \Big(1 - \delta_{ij}(u) \Big)}. \tag{5.8}$$

Although the update of $\hat{\Lambda}_{q\ell}$ sums over all the past observations $(X_{ij}(1), \ldots, X_{ij}(t))$, we can develop Eq. (5.8) as follows:

$$\begin{aligned}
\hat{\Lambda}_{q\ell} = & \frac{\sum_{i=1}^{N} \sum_{j=1}^{M} \sum_{u=1}^{(t-1)} \tau_{iq}(u) \eta_{j\ell}(u) \Big(X_{ij}(u) - \delta_{ij}(u) X_{ij}(u) \Big) + \sum_{i=1}^{N} \sum_{j=1}^{M} \tau_{iq}(t) \eta_{j\ell}(t) \Big(X_{ij}(t) - \delta_{ij}(t) X_{ij}(t) \Big)}{\sum_{i=1}^{N} \sum_{j=1}^{M} \sum_{u=1}^{(t-1)} \tau_{iq}(u) \eta_{j\ell}(u) \Big(1 - \delta_{ij}(u) \Big) + \sum_{i=1}^{N} \sum_{j=1}^{M} \tau_{iq}(t) \eta_{j\ell}(t) \Big(1 - \delta_{ij}(t) \Big)} \\
= & \frac{N_{q\ell}^{old} + N_{q\ell}^{(t)}}{D_{q\ell}^{old} + D_{q\ell}^{(t)}} = \frac{N_{q\ell}^{old}}{D_{q\ell}^{old} + D_{q\ell}^{(t)}} + \frac{N_{q\ell}^{(t)}}{D_{q\ell}^{old} + D_{q\ell}^{(t)}}.
\end{aligned} \tag{5.9}$$

By splitting Λ in two different parts, we can distinguish between a part known at time $t-1$, namely $N_{q\ell}^{old}$ and $D_{q\ell}^{old}$, and the current updates at time t, $N_{q\ell}^{(t)}$ and $D_{q\ell}^{(t)}$. Then, we divide and multiply the first term for $D_{q\ell}^{old}$, such that, denoting

$\hat{\Lambda}_{q\ell}^{old} = \frac{N_{q\ell}^{old}}{D_{q\ell}^{old}}$, we obtain the final online update:

$$\hat{\Lambda}_{q\ell} = \hat{\Lambda}_{q\ell}^{old} \cdot \frac{D_{q\ell}^{old}}{D_{q\ell}^{old} + D_{q\ell}^{(t)}} + \frac{N_{q\ell}^{(t)}}{D_{q\ell}^{old} + D_{q\ell}^{(t)}}, \tag{5.10}$$

Hence, when a new observation comes Λ can be updated thanks to Eq.(5.10), along with the update of $D_{q\ell}^{old}$.

Update of α, β and π through deep neural networks

As mentioned in Section 4.2.1, the mixture proportions α and β, as well as the sparsity parameter π are driven by three systems of differential equations, respectively. Hence, the update process for these parameters in the online inference algorithm poses a challenge because they lack closed-form updating formulas. As a result, the decomposition strategy used for updating Λ cannot be applied. To address this issue, we introduce an approximation technique that leverages a moving window $G_d(t)$ of size d, allowing us to update the parameters based on the most recent d observations. In addition to its role in parameter updates, the moving window $G_d(t)$ serves another purpose as it is the input for a deep neural network. As we assumed that the functions f_A, f_W and f_Z are continuous, we propose to parametrize them with three LSTM networks (Hochreiter and Schmidhuber, 1997). LSTM is a type of recurrent neural network that operates on sequences of a specific length and produces a sequence of the same length, shifted one time step ahead. For instance, let's consider the current time t and the time window $G_d(t)$ with a length of d. The input for the LSTM networks consists of a series of values ranging from $t-1-d$ to $t-1$, representing the historical observations within the window. The LSTM networks then predict a sequence of values from $t-d$ to t, which correspond to the updated parameter values for α, β, and π. Also in this case, optimizing the lower bound in Eq.(5.7) with respect to α, β, and π reduces to maximize it with respect to the parameters of the respective neural networks. Here, instead of summing over the entire time period, we sum only over the moving window $G_d(t)$ of length d. For example, the loss function for α can be expressed as follows:

$$\mathcal{L} = \sum_{u \in G_d(t)} \sum_{i=1}^{N} \sum_{q=1}^{Q} \tau_{iq}(u) \log \alpha_q(u), \tag{5.11}$$

The loss functions of β and π can be similarly derived using their respective distribution-specific equations. In the experiments, this update is implemented in

PyTorch via automatic differentiation (Paszke et al., 2017) and relies on stochastic optimisation (ADAM, Kingma and Ba, 2014). Once the neural nets are trained via back-propagation (SGD) they provide us with the current ML estimates of $\alpha(t)$, $\beta(t)$ and $\pi(t)$. The whole inference procedure is summarized in Algorithm 3.

5.2.5 Initialization and model selection

In this section, modifications to the initialization method presented in 4.3.4 are proposed to make the algorithm adaptable to run online. First, we select the first slice of the data X_{t_0} and apply on it a static LBM algorithm for a list of pairs of cluster numbers, i.e. (q, ℓ) for $q = 2, \ldots, Q_{max}$ and $\ell = 2, \ldots, L_{max}$. Subsequently, we employ the ICL criterion (for more details see Section 2.3.4) to determine the optimal number of row and column clusters for this particular subset of data. The ICL criterion approximates the integrated log-likelihood of the complete dataset and can be derived as follows for our model:

$$ICL(Q, L) = \log p(X, \hat{Z}, \hat{W}; \hat{\theta}) - \frac{Q-1}{2}\log N - \frac{L-1}{2}\log M - \frac{QL}{2}\log(NM) + \\ - \frac{1}{2}\log(NM). \tag{5.12}$$

The pair $(\hat{Q}, \hat{L})$ that leads to the highest value for the ICL is considered as the most likely cluster numbers for the considered slice of data X_{t_0}. However, as we expect that the choice of $\hat{Q}$ row and $\hat{L}$ column cluster components could not be the best for all the future time instants, the VEM-SGD algorithm (see Algorithm 3) will be then run with more components than the ones found by the ICL. Indeed, we run the VEM-SGD algorithm with $Q_{max} \geq \hat{Q}$ and $L_{max} \geq \hat{L}$ cluster components. Then, every time there is a new data entry, part of the model parameters are initialized with $\hat{\theta}(t)$ obtained via a static run of LBM and the remaining parameters, corresponding to the additional row and column clusters are set to zero.

Therefore, our objective is to leverage the advantage of using deep neural networks, which enables our VEM-SGD algorithm to initialize with empty clusters. These empty clusters can potentially be activated in the future, if required. As a result, we can avoid the typically computationally intensive task of running the entire algorithm with all possible combinations of row and column cluster numbers for the complete dataset. This strategy enables our approach to handle large-scale

datasets within a reasonable computation time while achieving satisfactory results, as demonstrated in the forthcoming section.
Also, at the initial stage of the algorithm (i.e. the first d time points), the parameters $\alpha(t)$, $\beta(t)$ and $\pi(t)$ are modeled via a two-layer fully connected neural networks, following the approach of Zip-dLBM until $t = d$. Then, as explained in Section 5.2.4, at $t = d + 1$, the estimates from the previous time step, obtained via fully connected networks, serve as input to LSTM, which is used for online parameter estimation from this point on.

Algorithm 3 VEM-SGD Algorithm for Stream Zip-dLBM

Require: $X, \hat{Q}, \hat{L}, Q_{max}, L_{max}, max.iter, G_d(t)$.
  while New observations $X(t)$ come: **do**
    Initialization of $\alpha(t), \beta(t), \pi(t), \Lambda$ with LBM; % *with* $\hat{Q}$, *and* $\hat{L}$
    ▶ Add $Q_{max} - \hat{Q}$ columns of zeros to $\alpha(t)$;
    ▶ Add $L_{max} - \hat{L}$ columns of zeros to $\beta(t)$;
    ▶ Add $Q_{max} - \hat{Q}$ rows and $L_{max} - \hat{L}$ columns of zeros to Λ;
    for it $= 1$ to $max.iter$ **do**
      VE-Step:
      for p $= 1$ to Fixed.Point **do**
        alternatively update $\delta(t)$, $\tau(t)$, $\eta(t)$; % *fix point eqs*
      end for
      M-Step:
      Update $\theta = (\Lambda, \pi(t), \alpha(t), \beta(t))$.

$$\hat{\Lambda}_{q\ell} = \hat{\Lambda}_{q\ell}^{old} \cdot \frac{D_{q\ell}^{old}}{D_{q\ell}^{old} + D_{q\ell}^{(t)}} + \frac{N_{q\ell}^{(t)}}{D_{q\ell}^{old} + D_{q\ell}^{(t)}}.$$

      Update $\alpha(t), \beta(t), \pi(t)$ %*LSTM on the moving window* $t \in G_d(t)$
    end for
    Discard all the observation before $G_d(t)$
  end while

5.2.6 Bayesian online change point detection

As previously stated, one of the aims of Stream Zip-dLBM is to perform multivariate online change point detection. To accomplish this task, we combine the Bayesian Online Change Point Detection (BOCD) method, proposed in a seminal paper by Adams and MacKay (2007), with our strategy. BOCD detects change points based on the estimation of the posterior distribution over the current "run length", or time segment since the last change point, given the data observed so far, using a simple message-passing algorithm. Essentially, the run length is used to determine if a new data point belongs to the current partition based on previous observations. If the new data point belongs to the current partition, the run length will increase by 1 at the next time step, otherwise it will reset to 0. This process is continuously repeated at each time step. It is worth noticing that the BOCD algorithm is typically implemented in an online fashion, analyzing the data as it streams in. However, in our case, we directly apply the algorithm to detect change points on the estimates of $\alpha(t)$, $\beta(t)$, and $\pi(t)$ obtained via the integration of the dynamic systems once the LSTM are trained. To prevent detecting change points on parameters that will be recalculated in future time steps, we run the BOCD algorithm only on time points "behind" $G_d(t)$. Stated differently, at time t, BOCD operates on parameter values at time instances $t - d$.

5.3 Numerical experiments

The main purpose of this section is to highlight the most important features of Stream Zip-dLBM over simulated datasets and to demonstrate the validity of the inference algorithm presented in the previous sections. The first experiment consists in applying Stream Zip-dLBM to a specific dataset with evolving block pattern and sparsity to show that it recovers the data structure in real-time. While the second experiment demonstrates the model selection procedure on a simulated dataset.

5.3.1 Introductory example

A simulated dataset with dimension $350 \times 300 \times 150$ has been generated according to our model to perform this experiment. The simulated dynamics of α, β and π can be seen on the left-hand side of Figure 5.2. Concretely, α, β and π are three time series independently fluctuating around constant trends. Fluctuations are obtained at each time by means of independent Gaussian distributions with constant variance. The mean levels change when a change point occurs. The levels of the simulated change points and the values of the simulated parameter Λ are indicated in Table 5.1.

Based on the mixture proportions α, β, the values of the latent variables were then simulated through their distributions. Next, we used the sparsity proportions,

α			
cp	16	66	103
β			
cp	25	62	112
π			
cp	16	66	103

$$\Lambda = \begin{bmatrix} 6 & 4 \\ 1 & 2 \\ 7 & 3 \end{bmatrix}$$

Figure 5.1: Simulated time instants for change points of α, β and π and simulated values of Λ.

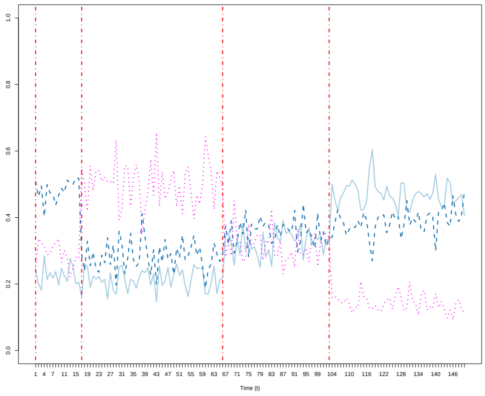
π, and the intensity parameter, Λ, to simulate the three-dimensional tensor X as Zero-Inflated Poisson variables. We then applied the Stream Zip-dLBM inference algorithm to the simulated dataset, using the actual values of $Q = 3$ and $L = 2$ to demonstrate the model's ability to recover the parameters. Figure 5.2 displays the true mixture proportions on the left side and the online estimates on the right side. The red dashed lines depict the simulated and estimated change points, respectively. From these results we see that Stream Zip-dLBM perfectly recovers the evolution of the mixing proportion and the sparsity parameter over time, including the change points.

5.3.2 Model selection experiment

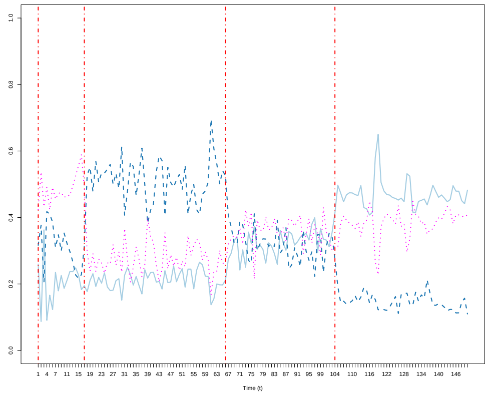
In this experiment, we assess the ability of Stream Zip-dLBM to determine the optimal number of clusters for rows and columns. Initially, we utilize the Integrated Completed Likelihood (ICL) criterion to compute the optimal number of clusters on the first slice. This ensures that the algorithm is initialized reasonably. Once initialized, the algorithm maintains consistent results without making alterations, thus retaining the selected number of clusters. To evaluate the effectiveness of the algorithm and verify if it activates new clusters, we generate a dataset based on the configuration described in Section 5.3.1. The dataset consists of 3 row clusters (Q=3) and 2 column clusters (L=2). We then apply Stream Zip-dLBM to this dataset, with maximum values of $Q_{max} = 7$ and $L_{max} = 7$. Figure 5.3 provides an illustrative demonstration of the algorithm's behavior, specifically regarding the activation of clusters. It is clear from the figure that the unnecessary clusters remain empty and that the estimates of the α, β, and π parameters are also accurate. Finally, it is worth noticing that Stream Zip-dLBM successfully identifies the changing points in α, β and π over time, despite not using the optimal number of input clusters. Finally, to evaluate the performance of the model in identifying the correct rows and columns partitions, we use the adjusted Rand index (ARI) (Rand, 1971). The adjusted Rand index, from a mathematical point of view, is closely related to the accuracy measure, however it is a commonly used method for evaluating clustering problems since it can be applied for measuring the similarity between two partitions even with different number of clusters and it is invariant to label switching. We also use a measure called CARI, recently introduced by Robert et al. (2020). This new criterion is based on the Adjusted Rand Index (Rand, 1971) and it was developed especially for being applied to co-clustering methods. The closer these indexes are to 1, the more two label vectors are similar to each other. We compared the original matrices Z and W, with the estimates τ and η given by the output of Stream Zip-dLBM. We evaluate the performance indexes at each time step, obtaining the following results:

ARI rows	ARI columns	CARI
0.99 ± 0.03	1 ± 0	0.99 ± 0.02

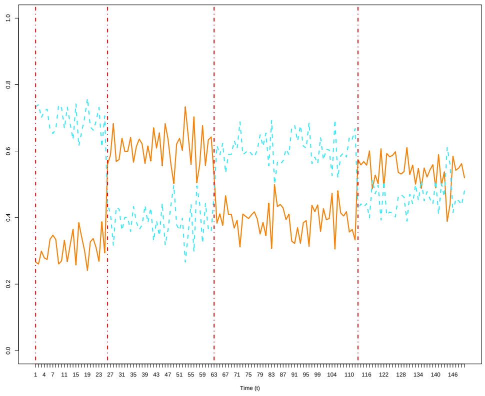
Thus, we can conclude that our algorithm satisfyingly identifies the composition of the original clusters in time.



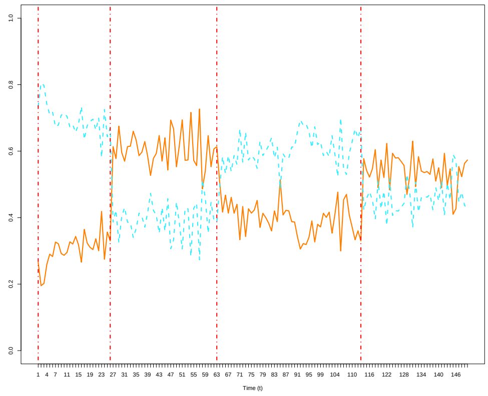
(a) True α.



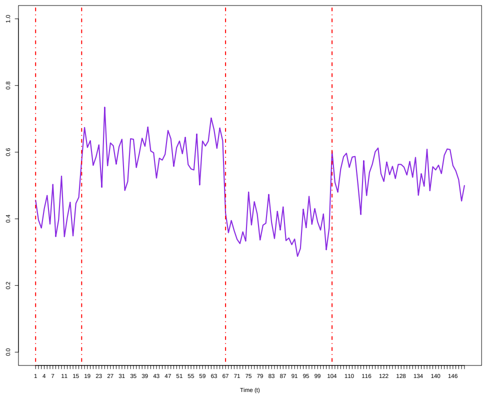
(b) Estimated α.



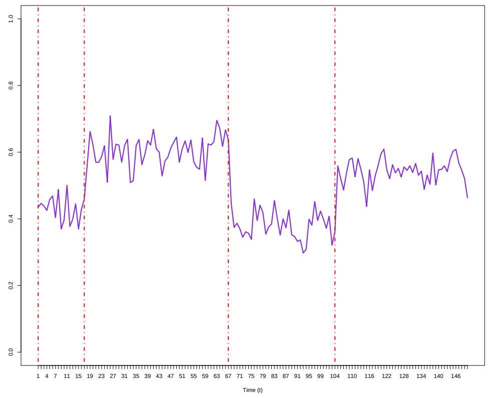
(c) True β.



(d) Estimated β.

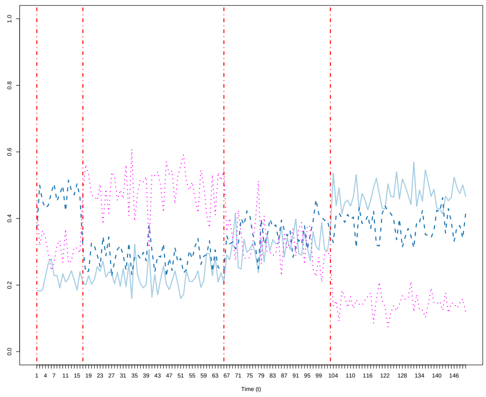


(e) True π.

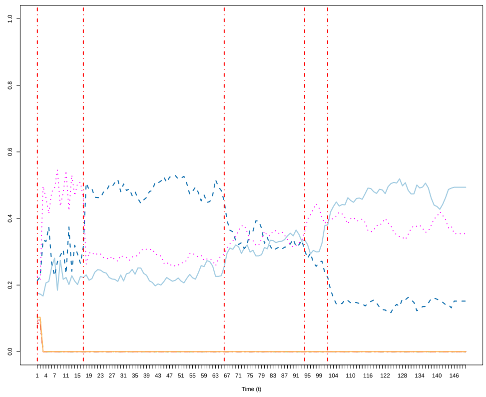


(f) Estimated π.

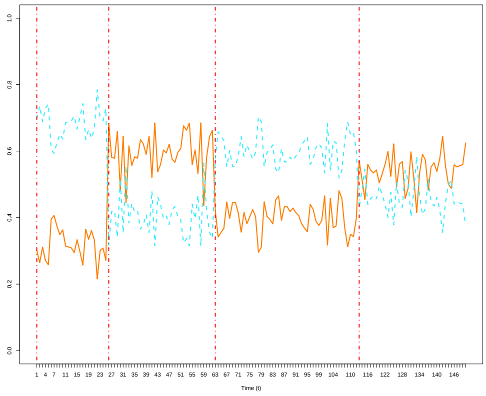
Figure 5.2: Evolution of the true (left) and estimated (right) proportions of the parameters α, β and π, respectively.



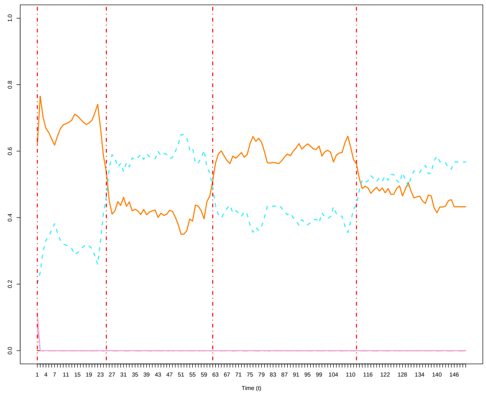
(a) True α.



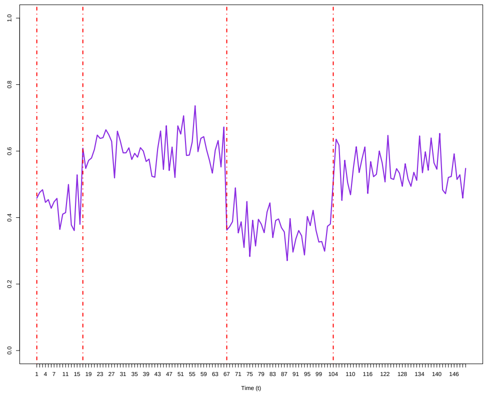
(b) Estimated α.



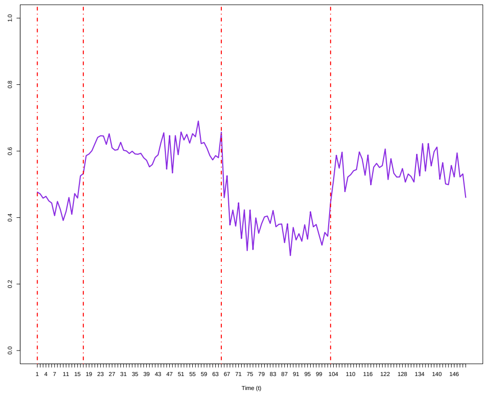
(c) True β.



(d) Estimated β.



(e) True π.



(f) Estimated π.

Figure 5.3: Evolution of the true (left) and estimated (right) proportions of the parameters α, β and π, respectively. In Figures (b) and (d), we observe how the excess clusters remain empty, depicted by the lines of the mixture proportions that stay at zero throughout the entire time period.

5.4 Analysis of the adverse drug reaction dataset

This section focuses on the online application of Stream Zip-dLBM to a large-scale pharmacovigilance dataset, with the aim of illustrating the potential of the tool for such studies.

5.4.1 Protocol and data

This section considers a large dataset consisting of an adverse drug reaction (ADR) dataset, collected by the Regional Center of Pharmacovigilance (RCPV), located in the University Hospital of Nice (France). The RCPV survey an area of three departments totalling 2.3 millions inhabitants.A time horizon of 7 years is considered, from January 1^{st}, 2015 to March 3^{rd}, 2022. Since the data are very sparse, we aggregate them summing up along the time dimension, such that one time instant corresponds to one month. The overall dataset is made of 39,267 declarations, for which the market name of the drug, the notified ADR and the reception date are considered. Moreover, we only considered drugs and ADRs that were notified more than 10 times over the 7 years. The resulting dataset contains 419 drugs, 614 ADRs and 87 time intervals with 23,264 non-zero entries. Figure 5.4 shows the frequency of declarations arrived at the RCPV in the considered period, sorted by month. For more details on this dataset please refer to the data description in Section 4.6.1.

It is important to highlight that the data being used exhibit extreme sparsity, ranging from a minimum of 99.25% to a maximum of 99.98% per month. To avoid encountering numerical issues, the LSTM network was not employed for inferring the parameter π. Instead, point estimates of $\hat{\pi}$were used in the inference process.

5.4.2 Summary of the results

To initialize the algorithm, as explained in Section 5.2.5, we computed the ICL criterion on one data slice, corresponding to the first month, where the optimal numbers of clusters identified by the model selection criterion are $\hat{Q} = 3$ and $\hat{L} = 3$. Then, we run Stream Zip-dLBM and every time a new entry is added in the tensor X. We initiate the model parameters through the process described in Section 5.2.5. Also, we ran Stream Zip-dLBM with $Q_{max} = 7$ and $L_{max} = 7$ to allow the model to fill or empty clusters as needed. The process takes a running

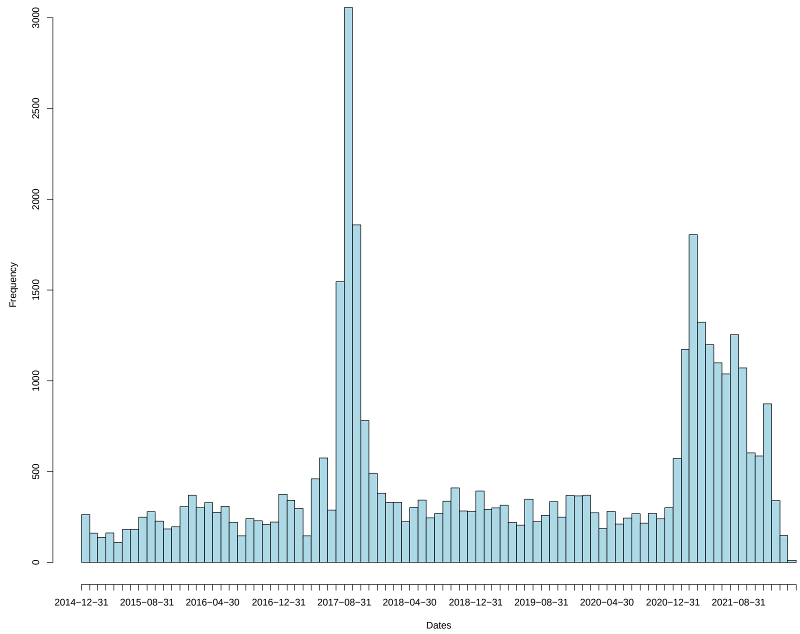
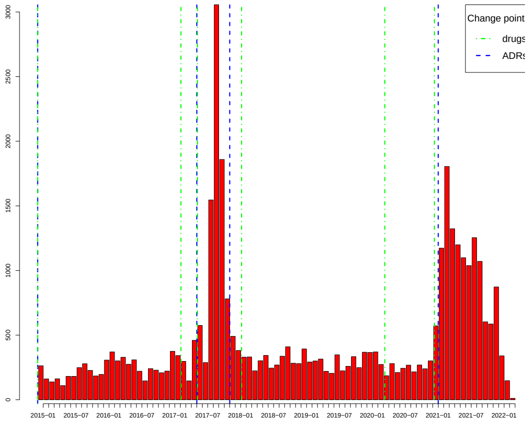


Figure 5.4: Number of declarations received by the pharmacovigilance center from 2015 to 2022, sorted by month.

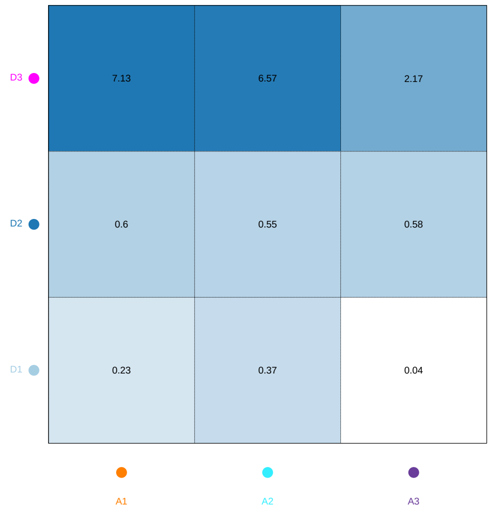
time of about one hour on a MacBook Pro, 2020, with a processor of 2,3 GHz Quad-Core Intel Core i7 and 16 GB of RAM. It's important to emphasize that once new data arrives at a new time instant, the time required to fit the model and obtain updated results is around 12 seconds. When we compare this result to the time it takes to obtain retrospective results using the model presented in Chapter 4, we can see that this new version is more time-efficient.

In Figure 5.5a, the frequency of declarations received by the RCPV from 2015 to 2022 is depicted, organized by month. Here, the identified change points are represented by dashed lines. Specifically, the green dashed lines indicate the change points detected in the evolution of drug clusters, while the blue dashed lines indicate the change points detected in the evolution of ADR clusters.

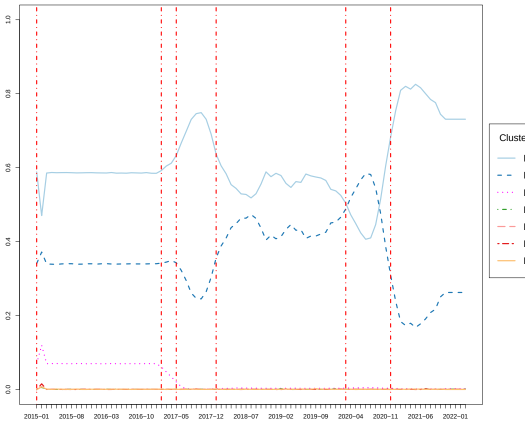
Figure 5.5b displays the estimated Poisson intensity parameter, Λ. This figure only focuses on the 3 groups of drug clusters, denoted by the letter D, and the 3 groups of ADR clusters, denoted by the letter A, that have been activated in the inference. This representation provides valuable insights for model interpretation as it gives an overview of the relationships between drug clusters and ADR clusters and how they evolve over time. Each color refers to a drug (rows) or ADR (columns) cluster and the higher is the value in each block, the strongest



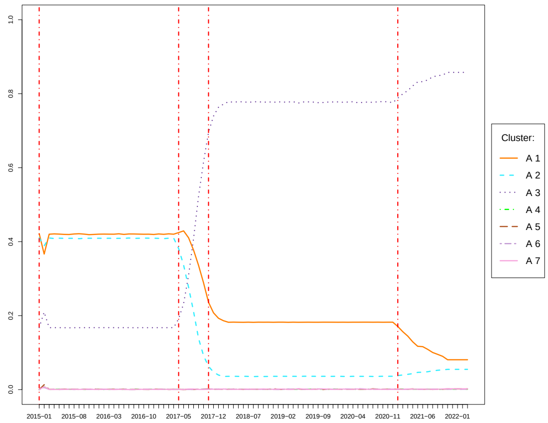
(a) Histogram of declarations over time, with change points.



(b) Estimated Poisson intensity parameter.



(c) Evolution of drug clusters proportions.



(d) Evolution of ADR clusters proportions.

Figure 5.5: Histogram of declarations over time, with the change points detected for the drugs and ADRs (top left) and estimated Poisson intensities, each color represents a different drug (ADR) cluster (top right); evolution of the estimates of $\hat{\alpha}$ (bottom left); evolution of the estimates of $\hat{\beta}$ (bottom right).

is the relationship (i.e the expected number of declarations received in the time unit) between the related pair of clusters. Looking at this figure, it can be seen that some clusters are strongly related whereas others are not at all.
Figures 5.5c and 5.5d display the estimated mixture proportions of drug clusters ($\hat{\alpha}$) and ADR clusters ($\hat{\beta}$) respectively. The dashed lines in the figures represent the change points identified by the BOCD algorithm.

Figure 5.6 illustrates the estimated sparsity parameter, with the y-axis scale ranging from 0.99 to 1. It is worth noting that no change points have been detected in $\hat{\pi}$ because the values did not exhibit sufficient variability.
By examining the information provided in Figures 5.5b, 5.5c, and 5.5d, we can observe an interesting pattern. The clusters with the highest intensity are also the least populated. For instance, cluster D3 (drug cluster) demonstrates a remarkably high intensity of interactions with clusters A1 and A2 (ADR clusters). Despite its high intensity, cluster D3 appears to be relatively small in Figure 5.5c. This phenomenon is attributed to the presence of drugs associated with significant health crises that happened during the reporting period. Notably, Mirena® in the first half of 2017, Lévothyrox® in the latter part of 2017, and Covid-19 vaccines throughout 2021 were the primary drivers of these crises. Interestingly, each crisis period is marked by a detected change point in both the drug cluster proportions (Figure 5.5c) and ADR cluster proportions (Figure 5.5d).
Similarly, by analyzing the composition of clusters A1 and A2, it is possible to identify which ADRs were the most reported in each of the aforementioned crises. For instance, from the composition of cluster A2 we notice that the most reported side effects during the Mirena® health crisis are mostly hormonal ones, such as anxiety, heat shock, and aggressive behavior. Then, looking at Figure 5.5d, during the Lévothyrox® health crisis we notice a peak in the A3 cluster of adversarial effects, probably because the great media coverage that the scandal had in those years made people declare the most disparate side effects. Also, we see that in 2021 there is another peak, corresponding to the period of the Covid-19 vaccination. Here, the adversarial effects found in cluster A1 are mostly linked to problems related to the vaccination site (e.g. arm pain, arm inflammation, skin reaction) and flu syndrome as a result of the vaccine. Cluster D2 presents an interesting contrast as it remains empty until August 2017. Subsequently, it contains a few but widely used drugs that are frequently reported, such as paracetamol, amoxicillin, and some popular antidepressants. The evolution of cluster D2's proportion, as shown in Figure 5.5c, aligns with the change points identified by the algorithm. This cluster begins to populate after the Lévothyrox® health crisis, with a peak in the early stages of the Covid-19 crisis, before the introduction of the vaccines. From Figure 5.5b, it can be observed that the intensity of interactions with clusters of undesirable effects does not differ significantly for cluster D2. Cluster D1, on the other hand, exhibits remarkably low interaction intensities and is densely populated by all other drugs. Initially, from the beginning of the analysis until 2017, it contains all drugs. However, after 2017, it includes only

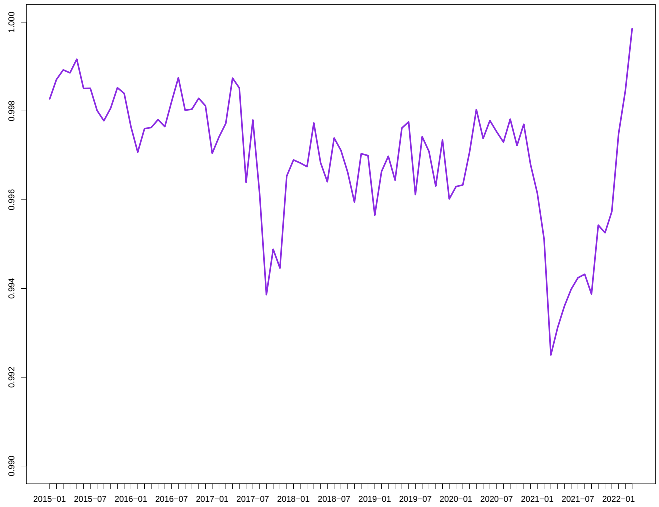


Figure 5.6: Evolution of the estimates $\hat{\pi}$.

drugs with a low frequency of reports. From Figure 5.5c, we see that the evolution of Cluster D1 over time aligns with the change points identified by the algorithm. Upon examining Figures 5.5b and 5.5d, a distinct behavior is observed in the clusters of adverse effects. Initially, until June 2017, cluster A1 contains all adverse effects. However, it gradually empties over time. Specifically, during the Mirena® crisis, only the adverse effects not associated with that particular health scandal remain in cluster A1. Subsequently, a significant change in the cluster membership occurs following the change point identified in October 2017. From this point onward, the number of ADRs in cluster A1 decreases significantly, and they are specifically related to Lévothyrox® (e.g. hair loss, cramps, insomnia, etc.). After the Lévothyrox® crisis, cluster A1 becomes empty until the subsequent change point detected in January 2021. From this moment until the peak of Covid-19 vaccine reports in February 2022, cluster A1 includes the main adverse effects reported for Covid-19 vaccines (e.g. pain at the vaccination site, skin rash, pericarditis, etc.).

Also, from Figure 5.5d we clearly notice as the Lévothyrox® crisis marked a turning point in the history of pharmacovigilance, probably because from this moment on people realized its importance and began to declare side effects of drugs and vaccines much more frequently.

Lastly, Figure 5.6 provides the estimated evolution of the sparsity parameter over time. The y-axis scale is set from 0.99 to 1 in order to visualize the changes

over time. Notably, no change points were detected in the estimated parameter $\hat{\pi}$ due to insufficient variability in the values. Initially, in 2015, the sparsity is recorded at 99.83%. As we approach the peak in 2017, the number of declarations increases, leading to a decrease in sparsity, reaching a local minimum of 99.38%. Subsequently, as we approach the peak related to Covid-19 vaccines, the sparsity level reaches its global minimum at 99.25% in March 2021.
As a summary, Stream Zip-dLBM method successfully identified meaningful clusters within the extensive initial data matrix of ADR reports. This provides a proof of concept of the possible use of this algorithm to detect public safety events from streams of ADR data.

5.5 Conclusions

This work is born out of the need to analyze and summarize observations and features of a dynamic matrix in an online setting for an application to pharmacovigilance. We have proposed an online dynamic co-clustering technique that enables simultaneous clustering of rows and columns along the time dimensions. As observations and features can change their cluster memberships over time, detecting structural changes in cluster interactions throughout the time period becomes crucial. We have introduced a generative zero-inflated dynamic latent block model as online extension of Zip-dLBM. The time modeling approach relies on three systems of ordinary differential equations. Inference is conducted using a Variational EM algorithm, combined with stochastic optimization of LSTM network parameters for the dynamic systems. Also, we added an online change points detection method to the process such that Stream Zip-dLBM is able to detect abrupt changes and create alerts in real time. The performance of our approach is evaluated through applications to some simulated data scenarios. Then, Stream Zip-dLBM was fit to a large-scale dataset supplied by the Regional Center of Pharmacovigilance of Nice (France). In this context, the model provided meaningful online segmentation of drugs and adverse drug reaction. Its potential use by medical authorities for identifying meaningful pharmacovigilance patterns looks very promising. The online inference algorithm, combined with change point detection, allows Stream Zip-dLBM to operate in real-time, continuously analyzing the flow of ADR declarations and triggering alerts as soon as a change point is detected. This provides an opportunity for further investigation and intervention by medical authorities.

Chapter 6

On going work, conclusion and perspectives

6.1 On going work: SBM for point clouds registration

6.1.1 Introduction

Registration of point clouds is crucial in various computer vision applications, playing a pivotal role in tasks such as 3D image retrieval, segmentation, and shape recognition. The main goal of point set registration consists in establishing correspondences between two sets of points and determining transformations that maps one point set onto another. Point set registration is a challenging task due to several factors, such as the presence of an unknown nonrigid spatial transformation, the high dimensionality of point sets, potential noise, and the existence of outliers. The transformation considered in point set registration typically falls in two categories: rigid or nonrigid. A rigid transformation is constrained to translation, rotation, and scaling. On the other hand, nonrigid transformations encompass a broader range of alterations, such as stretching and skewing. Figure 6.1 shows an example of point set registration with a 2D toy example. In this picture, the second image is visibly rotated by 90 degrees and it and contains an additional point denoted by a question mark compared to the first image. This discrepancy adds complexity to the matching task. Extending this concept to more intricate scenarios involving high-dimensional data, noise, outliers, and 3D images, it becomes evident that point sets registration can quickly become a challenging task.

Regarding the pairwise point set registration, depending on the modeling assumptions, several approaches have been proposed in the literature. They can be classified into distance-based methods (Zhang, 1994; Zhou and De la Torre, 2015), filter-based methods (Zhu et al., 2018; Li et al., 2016) and probability-based methods (Myronenko and Song, 2010; Zhou et al., 2014). It has been observed that probability-based methods tend to outperform other approaches. However, it is worth noting that the probability-based methods come at a higher computational cost in contrast to distance-based and filter-based methods (Zhu et al., 2019).

This section aims to formulate a probability-based point set registration method, designed to effectively address rigid transformations in the context of 3D point clouds.

In our application, we have multiple matrices representing a sub-sampling of points from distinct 3D images of the astragalus (a small bone in the ankle) of

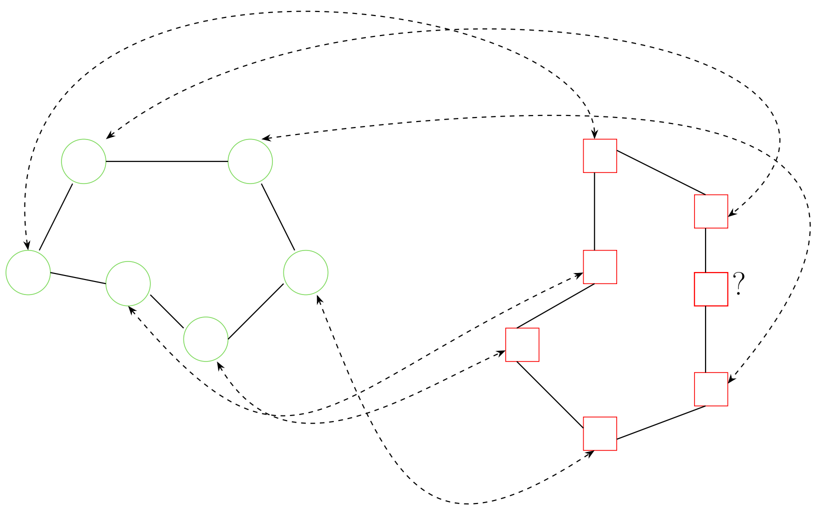


Figure 6.1: A toy example of the point set registration problem.

some specific animal species: sheep, goats, muflon and chamois. The ultimate objective is to perform species identification based on these images. This research is conducted within the framework of the *chiar in AI dor History and Archaeology* of my advisor, Marco Corneli, in collaboration with another PhD student, Davide Adamo.

The point clouds are obtained via a 3D scan of bones in a modern collection, being part of the PhD thesis of Vuillien (2020) in archaeo-zoology. The scanned images must undergo rigid transformations (rotation and symmetries) in order to be aligned. Since point clouds are invariant with respect to points permutation, the registration task is not trivial. It remains nonetheless necessary since, due to the small number of 3D images in our collection (around 40), recent state of the art methods (i.e. PointNet (Qi et al., 2017)) that would allow us to avoid registration are not exploitable.

Each point cloud is represented by a matrix, featuring rows corresponding to the points within the cloud and three columns designating each dimension to identify the spatial location of the point. These matrices exhibit disparate dimensions, implying that the number of points varies across images. For instance, considering just two matrices, one might have dimensions $P \times 3$, and the other $N \times 3$, where $P \neq N$.

Moreover, the rows of these matrices lack of alignment. For instance, the point identified by the first row of one matrix is in no way associated with the point identified by the first row of another matrix. Figure 6.2 shows an example of two point clouds representing two astragalus, where the second one is slightly rotated compared to the first one. Also, the red dots represent the first elements of the representation matrices, respectively. Our objective is to establish a meaningful correspondence between the rows of the two matrices, even in presence of rigid transformations, as it shown, for instance in Figure 6.3. Here, the second point cloud (in orange) has been rotated such that the two images overlap: Coherent Point Drift (CPD) method, as proposed by Myronenko and Song (2010) was used to register the two point clouds. In this probabilistic approach, the alignment of two point sets is treated as a probability density estimation. The authors employ Gaussian Mixture Model (GMM) centroids, representing the first point set, fitted to the data representing the second point set by maximizing the likelihood. However, despite its effectiveness in cases of slight misalignments, the algorithm exhibits limitations when dealing with mirrored images or more substantial rotations, as we see in Figure 6.4. This is attributed to the algorithm sensitivity to converging to local minima, particularly when the initialization is random.

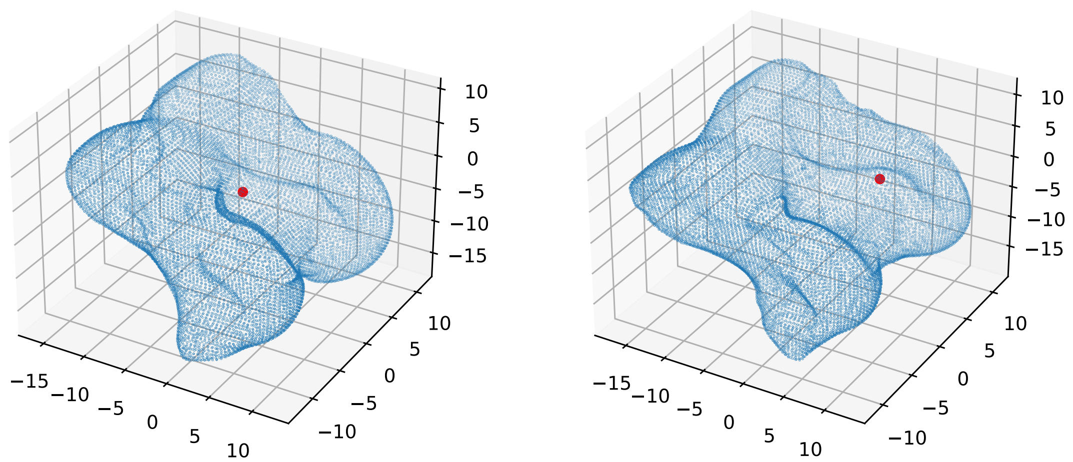


Figure 6.2: 3D point cloud representation of two bones, with different orientation. The red dot identifies the first row of the respective representation matrices.

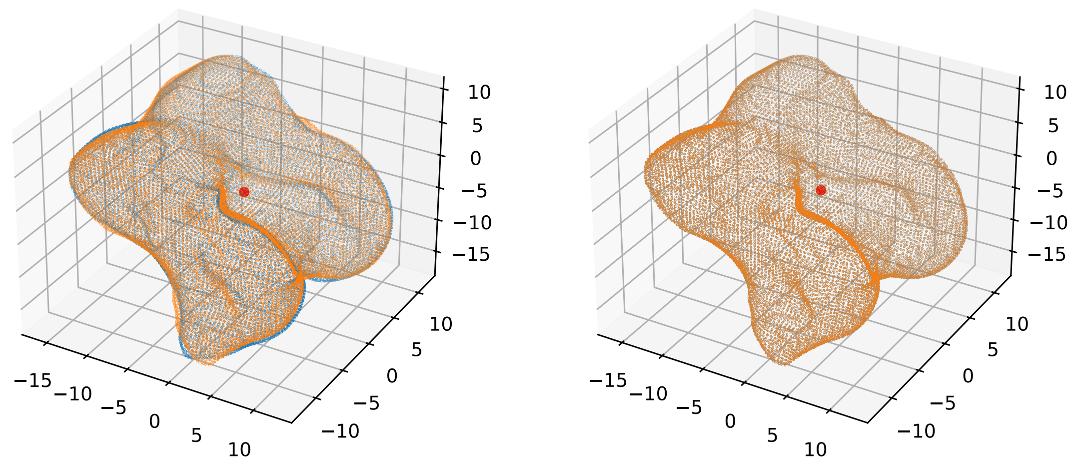


Figure 6.3: 3D point cloud representation of two astragalus after the point set registration with CPD. The orange point cloud represents the second cloud overlapping the first one in blue.

6.1.2 Our contribution

Our approach would be to use a probabilistic method as well. Specifically, we plan on making use of the Stochastic Block Model instead of Gaussian Mixture Models (GMMs) in order to model the points of a source cloud with those of a target cloud. In more detail, we represent a point cloud with N points via a graph with N nodes, thus leading to a (weighted) adjacency $N \times N$ matrix, denoted by $\mathcal{D}$. The way the nodes are connected to each other and hence the nature of the graph is of course crucial and several approaches are possible: a k-nearest neighbors graph or an ϵ-neighborhood graph where $\mathcal{D}_{ij}$ being either the Euclidean distance between to neighbor nodes or zero is an option. Otherwise a fully connected graph, with $\mathcal{D}_{ij}$ being the geodesic distance between points i and j might be considered. This list of options is of course not exhaustive.

For simplicity we focus on two point clouds $\mathcal{X}$ and $\mathcal{Y}$ represented by two matrices of dimension $N \times 3$ and $P \times 3$, respectively, with $N > P$. We compute their "distance" adjacency matrices denoted as D^x and D^y, respectively.

Now, let Z be a latent matrix such that: $Z := \{z_{ip}\}_{i \in 1,\dots,N, p \in 1,\dots,P}$. This matrix represents the clustering of point of $\mathcal{X}$ into P groups, where point i belongs to cluster p if $z_{ip} = 1$, 0 otherwise. Moreover, the rows of Z are assumed to be

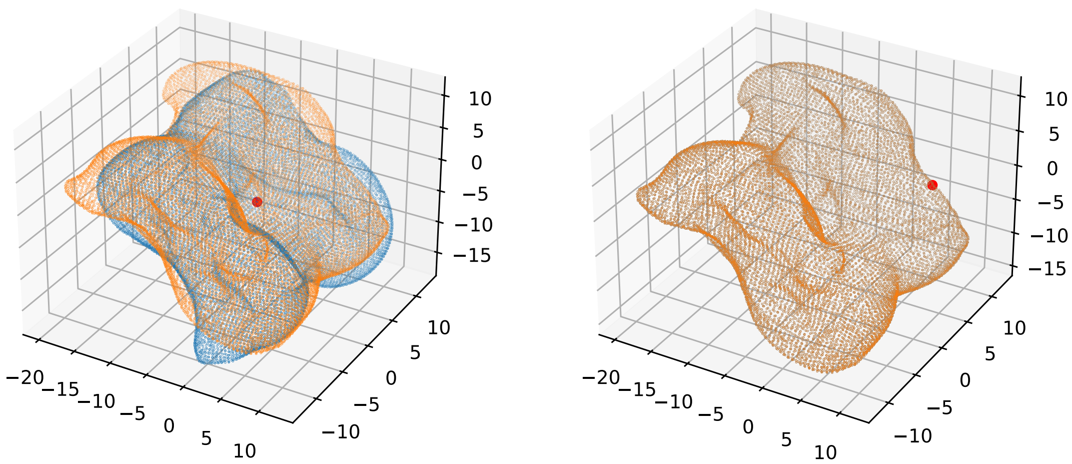


Figure 6.4: 3D point cloud representation of two astragalus after the point set registration with CPD in a more difficult setting. The orange point cloud represents the second cloud overlapping the first one in blue.

independently distributed according to a multinomial distributions:

$$p(Z|\alpha) = \prod_{i=1}^{N}\prod_{p=1}^{P} \alpha_p^{z_{ip}},$$

where $\alpha_p = \mathbb{P}\{z_{ip} = 1\}$ and $\sum_{p=1}^{P} \alpha_p = 1$. We further assume that, conditionally to Z, $\mathcal{D}^x$ follows a log-normal distribution, such that:

$$\log \mathcal{D}^x_{ij}|Z_i, Z_j \sim \mathcal{N}(\mathcal{D}^y_{Z_iZ_j}; \sigma^2_{Z_iZ_j}). \tag{6.1}$$

Hence we can write:

$$p(\mathcal{D}^x|Z,\cdot) = \prod_{\substack{i,j\\ j>i}}^{N} p(\mathcal{D}^x_{ij}|Z,\cdot). \tag{6.2}$$

and taking the logarithm:

$$\begin{aligned} \log p(\mathcal{D}^x|Z,\cdot) &= \sum_{j>i}^{N} \log p(\mathcal{D}^x_{ij}|Z,\cdot) \\ &= \sum_{j>i}^{N} \frac{1}{\sqrt{2\pi\sigma^2_{Z_iZ_j}}} \exp\Big(-\frac{1}{2}\cdot\frac{(\mathcal{D}^x_{ij} - \mathcal{D}^y_{Z_iZ_j})^2}{\sigma^2_{Z_iZ_j}}\Big). \end{aligned} \tag{6.3}$$

The model described so far has a set of parameters denoted by $\theta = (\sigma, \alpha)$, and one latent variable: Z. Then, we can write the log-likelihood of the complete data as follows:

$$
\begin{aligned}
\log p(\mathcal{D}^x, Z|\theta) &= \log p(D^x|Z, \mathcal{D}^y, \sigma) + \log p(Z|\alpha) \\
&= \sum_{j>i}^{N} \frac{1}{\sqrt{2\pi\sigma^2_{Z_iZ_j}}} \cdot \exp\Big(-\frac{1}{2} \cdot \frac{(\mathcal{D}^x_{ij} - \mathcal{D}^y_{Z_iZ_j})^2}{\sigma^2_{Z_iZ_j}} \Big) + \sum_{i=1}^{N} \log \alpha_{Z_i} \\
&= \sum_{i=1}^{N}\sum_{j>i}^{N}\sum_{p=1}^{P}\sum_{\ell=1}^{P} \Big(Z_{ip}Z_{j\ell}\Big[\frac{1}{\sqrt{2\pi\sigma_{p\ell}}} \cdot \exp\Big(-\frac{1}{2} \cdot \frac{(\mathcal{D}^x_{ij} - \mathcal{D}^y_{p\ell})^2}{\sigma^2_{p\ell}} \Big) \Big] \Big) \\
&+ \sum_{i=1}^{N}\sum_{p=1}^{P} z_{ip} \log \alpha_p.
\end{aligned}
\tag{6.4}
$$

Therefore, our goal is to leverage the posterior estimates of Z along with the model parameters, σ and α, to rearrange the rows of $\mathcal{D}^x$. and so to find correspondences between the points in $\mathcal{X}$ and those of $\mathcal{Y}$. The objective is to ensure that, after permutation, in $\mathcal{D}^x$ nearby rows, assigned to the same clusters, correspond to the same point of $\mathcal{Y}$. To achieve this, we plan to employ a Variational EM (VEM) algorithm for the inference process, possibly coupled with numerical optimization in the M-step, in order to estimate σ^2.

6.2 Overview of the contributions

This thesis is born out of the need to tackle the multifaceted challenges posed by high-dimensional, time-dependent and discrete data. It was driven by the aim to develop autonomous methodologies capable of detecting meaningful patterns, summarizing vast information, and aiding in effective data visualization.

To this end, in Chapter 3 we proposed the Dynamic Latent Block Model (dLBM) as a first versatile approach for co-clustering of evolving count matrices. We demonstrated its ability to uncover temporal patterns within a sequence of data matrices first with several experiments on simulated dataset, and then with a real dataset. The Dynamic Latent Block Model (dLBM) was in fact applied to an extensive dataset provided by the Regional Center of Pharmacovigilance (RCPV) in Nice, France. In this application, dLBM exhibited its ability to generate valuable partitions of drugs, adverse drug reactions and temporal patterns. The potential adoption of dLBM by medical authorities, with the aim of discerning significant pharmacovigilance patterns, appears to hold considerable promise. The potential of dLBM as a regular tool for detecting safety signals is further substantiated by additional studies conducted by the RCPV. These studies have led to the redaction of a another paper that provides an in-depth analysis of the medical results and implications obtained through this application.

Then, Chapter 4 introduced the Zero-Inflated Dynamic Latent Block Model. Here, both observations and features have the flexibility to change their cluster membership over time, allowing one to explore structural changes in the interactions between clusters throughout the observed time period. Also, this model can be eventually adapted to various zero-inflated probability distributions. The temporal aspect of the model is captured through the utilization of three distinct systems of ordinary differential equations. The inference process relies on an innovative variational method. In fact, the maximization step involves training fully connected neural networks to solve the underlying dynamical systems.

Finally, in Chapter 5 we proposed the Stream Zip-dLBM, tailored for online applications. Notably, its application to pharmacovigilance data underscored the potential of these methodologies in addressing real-world healthcare challenges.

The innovative models presented in this thesis provide one with multiple solutions for uncovering hidden patterns in high-dimensional dynamic discrete datasets. These methodologies hold the potential to facilitate early detection and timely responses to safety signal detection of medical products.

6.3 Perspectives

During this research, several promising avenues for future exploration and development have emerged:

- **Development of a web platform**: We are actively engaged in the creation of a dedicated software application based on the Stream Zip-dLBM model. This application is intended for use by the Regional Center of Pharmacovigilance in Nice, France. Once implemented, the application will resularly analyze daily or weekly time series, on a machine at the pharmacovigilance center. Its primary function is to automatically fit the model to incoming data. Leveraging the online change point detection method within the model, the application will promptly identify structural changes in the data and send automatic email notifications containing concise reports of the results. The software's effectiveness will be evaluated over a 6-month period, with the aim of eventually offering it on a national scale.

- **Block-dependent sparsity parameter**: In Chapter 4, we introduced a time-dependent sparsity parameter, denoted as π. However, this parameter is not currently block-dependent, unlike the Poisson parameter Λ. A future research direction involves investigating the feasibility of making the sparsity parameter block-dependent. Hence, we would write: $X_{ij}(t)|Z_i(t), W_j(t) \sim ZIP(\Lambda_{Z_i(t)W_j(t)}; \pi_{Z_i(t)W_j(t)})$, such that:

$$\begin{cases} X_{ij}(t)|Z_i(t), W_j(t) \sim \delta_0(X_{ij}(t)) & \text{with probability } \pi_{Z_i(t)W_j(t)} \\ X_{ij}(t)|Z_i(t), W_j(t) \sim \mathcal{P}(\Lambda_{Z_i(t)W_j(t)}) & \text{with probability } 1 - \pi_{Z_i(t)W_j(t)} \end{cases}$$

  where $\delta_0(\cdot)$ is the Dirac mass function in 0, Λ is a $Q \times L$ matrix, denoting the block-dependent Poisson parameter and $\pi_{Z_i(t)W_j(t)}$ represents the block dependent sparsity parameter. However, this extension would require careful identifiability studies to assess its practicality and effectiveness.

- **Model selection**: In Chapters 4 and 5, we employed the Integrated Classification Likelihood (ICL) criterion for model selection. While this criterion serves its purpose, it does not necessarily account for the presence of a large number of zero entries in interaction datasets. Future work may involve developing a model selection criterion that specifically deals with only non-zero entries, thus improving computational efficiency. Additionally, adapting a model selection criterion to account for the temporal evolution of the data would be a major contribution to this research area. Such criteria would enable more accurate and context-aware model selection, enhancing the overall performance of the proposed models.

- **Choice of the deep architecture**: The models presented in this thesis offer an innovative approach to co-clustering of discrete data in temporal contexts. However, the adoption of deep learning highlights a crucial challenge: the choice of deep architecture. Unlike supervised tasks where performance metrics often guide this choice, the selection of the neural network architecture poses a significant open problem in an unsupervised framework. In fact, managing the complexities of finding the right balance between model complexity, generalization capacity, and understanding latent structures can be quite challenging. Progress in this direction has the potential to not only improve the capabilities of dynamic latent block models but also provide valuable insights into the interaction between unsupervised learning and deep architectures.

These perspectives highlight the ongoing and evolving nature of research in the field of dynamic co-clustering and dynamic latent block models.

Chapter 7

Appendix

A Appendix Chapter 3

A.1 Estimation of the mixture proportions

The proof about how to obtain the updated mixture proportions is only shown for the estimation of parameter $\alpha_q^{(h+1)}$ because for the estimation of the other parameters, β and γ, the procedure is similar:

$$p(Z|\alpha) = \mathcal{L}(\alpha; Z) = N! \prod_{i=1}^{N} \prod_{q=1}^{Q} \frac{\alpha_q}{z_{iq}!};$$

$$\ell(\alpha_q; z_{iq}^{(h+1)}) = \log \mathcal{L}(\alpha_q, z_{iq}^{(h+1)}) = \log \left(N! \prod_{i=1}^{N} \prod_{q=1}^{Q} \frac{\alpha_q}{z_{iq}^{(h+1)}!} \right) =$$

$$= \log N! + \sum_{i=1}^{N} \sum_{q=1}^{Q} z_{iq}^{(h+1)} \log \alpha_q - \sum_{i=1}^{N} \sum_{q=1}^{Q} \log z_{iq}^{(h+1)}!$$

For a constrained maximization of this quantity we employ the Lagrange Multipliers, taking into account the constraint $\sum_{q=1}^{Q} \alpha_q = 1$.

$$\mathcal{L}(\alpha_q; \Lambda) = \ell(\alpha_q; z_{iq}^{(h+1)}) + \Lambda(1 - \sum_{q=1}^{Q} \alpha_q)$$

$$\frac{\partial \mathcal{L}(\alpha_q; \Lambda)}{\partial \alpha_q} = \frac{\partial \ell(\alpha_q; z_{iq}^{(h+1)})}{\partial \alpha_q} + \frac{\partial \Lambda(1 - \sum_q \alpha_q)}{\partial \alpha_q} = 0$$

$$\frac{\partial \sum_{i=1}^{N} \sum_{q=1}^{Q} z_{iq}^{(h+1)} \log \alpha_q}{\partial \alpha_q} - \Lambda \frac{\partial \sum_{q=1}^{Q} \alpha_q}{\partial \alpha_q} = 0$$

$$\frac{\sum_{i=1}^{N} z_{iq}^{(h+1)}}{\alpha_q} - \Lambda = 0$$

$$\sum_{i=1}^{N} z_{iq}^{(h+1)} = \Lambda \alpha_q \Rightarrow \frac{\sum_{i=1}^{N} z_{iq}^{(h+1)}}{\Lambda} = \alpha_q$$

Since Λ is equal to N:

$\sum_{q=1}^{Q} \sum_{i=1}^{N} \frac{z_{iq}^{(h+1)}}{\Lambda} = \sum_{q=1}^{Q} \alpha_q \Rightarrow \frac{1}{\Lambda} \sum_{q=1}^{Q} \sum_{i=1}^{N} z_{iq}^{(h+1)} = 1$; we can conclude that the estimation of $\alpha_q^{(h+1)}$is the following:

$$\alpha_q^{(h+1)} = \frac{1}{N}\sum_{i=1}^{N} z_{iq}^{(h+1)}$$

A.2 Maximum likelihood estimator of $\Lambda_{q\ell c}$

The maximum likelihood estimator of $\Lambda_{q\ell c}$ is obtained through the following process:

$$\log L(\Lambda|X,Z,W,S) = \sum_{q=1}^{Q}\sum_{\ell=1}^{L}\sum_{c=1}^{C}(R_{q\ell c}\log\Lambda_{q\ell c} - |\mathcal{A}_q||\mathcal{B}_\ell||\mathcal{D}_c|\Lambda_{q\ell c} + c)$$

where c is a constant that includes all the terms that does not depend on Λ.

$$\frac{\partial \log \mathcal{L}(\Lambda|X,Z,W,S)}{\partial\Lambda} = \frac{R_{q\ell c}}{\Lambda_{q\ell c}} - |\mathcal{A}_q||\mathcal{B}_\ell||\mathcal{D}_c| = 0 \Rightarrow \widehat{\Lambda}_{q\ell c} = \frac{R_{q\ell c}}{|\mathcal{A}_q||\mathcal{B}_\ell||\mathcal{D}_c|}$$

A.3 Intensity functions in the three scenarios

From Table 3.4, the scenarios "Easy" and "Medium" may look the same. However, the main difference between the two scenarios is the value assumed by the intensity function Λ. The values of this parameter in the three different scenarios are:

- Scenario A - Easy: $\Lambda = \Lambda_A$
  $\Lambda_A[,,1] = \begin{bmatrix} 50 & 18 \\ 1 & 1 \\ 1 & 50 \end{bmatrix}$; $\Lambda_A[,,2] = \begin{bmatrix} 50 & 50 \\ 18 & 1 \\ 1 & 18 \end{bmatrix}$
- Scenario B - Medium: $\Lambda = \Lambda_B$
  $\Lambda_B[,,1] = \begin{bmatrix} 1 & 1 \\ 1 & 7 \\ 7 & 20 \end{bmatrix}$; $\Lambda_B[,,2] = \begin{bmatrix} 20 & 20 \\ 7 & 1 \\ 1 & 7 \end{bmatrix}$
- Scenario C - Hard: $\Lambda = \Lambda_C$
  $\Lambda_C[,,1] = \begin{bmatrix} 70 & 12 & 1 \\ 35 & 1 & 35 \\ 1 & 70 & 12 \\ 12 & 35 & 70 \end{bmatrix}$; $\Lambda_C[,,2] = \begin{bmatrix} 35 & 70 & 12 \\ 70 & 70 & 70 \\ 12 & 1 & 35 \\ 1 & 70 & 1 \end{bmatrix}$; $\Lambda_C[,,3] = \begin{bmatrix} 12 & 70 & 35 \\ 35 & 12 & 70 \\ 70 & 35 & 12 \\ 12 & 1 & 35 \end{bmatrix}$
- Scenario D - Row_LBM: $\Lambda = \Lambda_D$
  $\Lambda_D[,,1] = \begin{bmatrix} 1 & 6 & 4 \end{bmatrix}$; $\Lambda_D[,,2] = \begin{bmatrix} 1 & 7 & 1 \end{bmatrix}$

A.4 Data structure representation

Fig. 1 shows a representation of the interactivity patterns between all the drugs and adversarial effects at any given time interval. Each panel represents a time interval and the size and the color of the points depend on the number of declarations received.

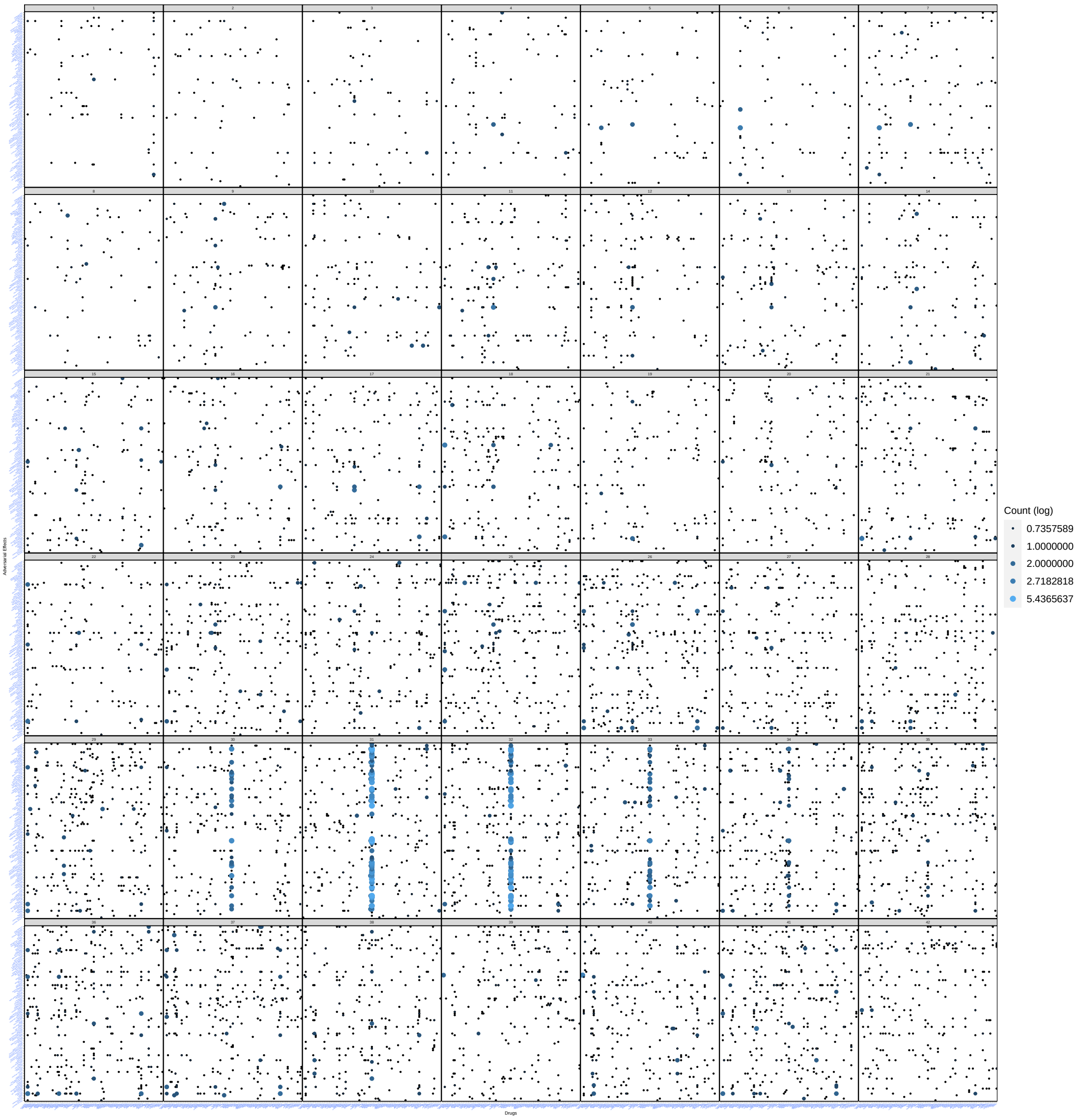


Figure 1: Representation of the interactivity patterns between drugs and adversarial effects at any given time interval. A small sample of the whole data set is considered.

B Appendix Chapter 4

B.1 Proof: Optimization of the factor q(A)

Starting from Eq. 4.7, we use the decomposition in Eq. 5.1, and then we substitute the conditional distributions on the right-hand side. We denote by *const* those terms in the lower bound not depending on $A_{ij}(t)$.

$$\begin{aligned}
\log(q^*(A)) &= E_{q(W,Z)}[\log(p(X,Z,W,A|\theta)] + const;\\
&= E_{q(W,Z)}[\log(p(X|Z,W,A,\Lambda,\pi)] + E_{q(W,Z)}[\log(A|\pi)] + const\\
&= \sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{t=1}^{T}\Bigg\{A_{ij}(t)[\log \mathbf{1}_{\{X_{ij}(t)=0\}}] + (1-A_{ij}(t))\Bigg[\sum_{q=1}^{Q}\sum_{\ell=1}^{L}\Big[E_{q(W,Z)}[Z_{iq}(t)]E_{q(W,Z)}[W_{j\ell}(t)]X_{ij}(t)\log\Lambda_{q\ell}\\
&- E_{q(W,Z)}[Z_{iq}(t)]E_{q(W,Z)}[W_{j\ell}(t)]\Lambda_{q\ell}\Big] - \log X_{ij}(t)!\Bigg] + A_{ij}(t)\log\pi(t) + (1-A_{ij}(t))\log(1-\pi(t))\Bigg\} + const\\
&= \sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{t=1}^{T}\Bigg\{A_{ij}(t)\log\pi(t)\mathbf{1}_{\{X_{ij}(t)=0\}} + (1-A_{ij}(t))\Bigg[\sum_{q=1}^{Q}\sum_{\ell=1}^{L}\Big[E_{q(W,Z)}[Z_{iq}(t)]E_{q(W,Z)}[W_{j\ell}(t)]X_{ij}(t)\log\Lambda_{q\ell}\\
&- E_{q(W,Z)}[Z_{iq}(t)]E_{q(W,Z)}[W_{j\ell}(t)]\Lambda_{q\ell}\Big] - \log X_{ij}(t)! + \log(1-\pi(t))\Bigg]\Bigg\} + const\\
&= \sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{t=1}^{T}\Bigg\{A_{ij}(t)\log\pi(t)\mathbf{1}_{\{X_{ij}(t)=0\}} + A_{ij}(t)\Bigg[\sum_{q=1}^{Q}\sum_{\ell=1}^{L}\Big[-E_{q(W,Z)}[Z_{iq}(t)]E_{q(W,Z)}[W_{j\ell}(t)]X_{ij}(t)\log\Lambda_{q\ell}\\
&+ E_{q(W,Z)}[Z_{iq}(t)]E_{q(W,Z)}[W_{j\ell}(t)]\Lambda_{q\ell}\Big] + \log X_{ij}(t)! - \log(1-\pi(t))\Bigg]\Bigg\} + const\\
&= \sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{t=1}^{T}\Bigg\{A_{ij}(t)\Bigg[\log\pi(t)\mathbf{1}_{\{X_{ij}(t)=0\}} + \sum_{q=1}^{Q}\sum_{\ell=1}^{L}\Big[-E_{q(W,Z)}[Z_{iq}(t)]E_{q(W,Z)}[W_{j\ell}(t)]X_{ij}(t)\log\Lambda_{q\ell}\\
&+ E_{q(W,Z)}[Z_{iq}(t)]E_{q(W,Z)}[W_{j\ell}(t)]\Lambda_{q\ell}\Big] + \log X_{ij}(t)! - \log(1-\pi(t))\Bigg]\Bigg\} + const.
\end{aligned}$$

We can then recognize the functional form of the Bernoulli distribution by indicating:

$$\begin{aligned}
\log q^*(A) &\propto \sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{t=1}^{T} A_{ij}(t)\log\delta_{ij}(t) + (1-A_{ij}(t))\log(1-\delta_{ij}(t)),\\
&\propto \sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{t=1}^{T} A_{ij}(t)\frac{\log\delta_{ij}(t)}{1-\log\delta_{ij}(t)}
\end{aligned}$$

where $\delta_{ij}(t)$ is defined as:

$$\delta_{ij}(t) = \frac{\exp(R_{ij}(t))}{1+\exp(R_{ij}(t))},$$

with $R_{ij}(t)$ defined as:

$$\begin{aligned}
R_{ij}(t) = \log(\pi(t)\mathbf{1}_{\{X_{ij}(t)=0\}}) + \sum_{q=1}^{Q}\sum_{\ell=1}^{L}\Big[&-E_{q(W,Z)}[Z_{iq}(t)]E_{q(W,Z)}[W_{j\ell}(t)]X_{ij}(t)\log\Lambda_{q\ell}+\\
&+ E_{q(W,Z)}[Z_{iq}(t)]E_{q(W,Z)}[W_{j\ell}(t)]\Lambda_{q\ell}\Big] + \log X_{ij}(t)! - \log(1-\pi(t)).
\end{aligned}$$

B.2 Proof: Optimization of the factor q(Z)

Starting from Eq. 4.8, we use the decomposition in Eq. 5.1, and then we substitute the conditional distributions on the right-hand side. We denote by *const* those terms in the lower bound not depending on $Z_{iq}(t)$.

$$
\begin{aligned}
\log q^*(Z|\theta) &= E_{q(A,W)}[\log p(X, A, Z, W \mid \theta)] \\
&= E_{q(A,W)}[\log(p(X \mid A, Z, W, \Lambda, \pi) + \log p(Z \mid \alpha)] \\
&= E_{q(A,W)}\Bigg[\sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{t=1}^{T}\Bigg\{(1 - A_{ij}(t))\sum_{q=1}^{Q}\sum_{\ell=1}^{L}\Big\{Z_{iq}(t)W_{j\ell}(t)X_{ij}(t)\log(\Lambda_{q\ell}) - Z_{iq}(t)W_{j\ell}(t)\Lambda_{q\ell}\Big\} \\
&- (1 - A_{ij}(t))\log(X_{ij}(t)!)\Bigg\}\Bigg] + \sum_{i=1}^{N}\sum_{q=1}^{Q} Z_{iq}(t)\log(\alpha_q(t)) + \text{const}, \\
&= \sum_{i=1}^{N}\sum_{j=1}^{M}(1 - E_{q(A,W)}[A_{ij}(t)])\Bigg[\sum_{q=1}^{Q}\sum_{\ell=1}^{L}\Big\{Z_{iq}(t)E_{q(A,W)}[W_{j\ell}(t)]X_{ij}(t)\log(\Lambda_{q\ell}) + \\
&- Z_{iq}(t)E_{q(A,W)}[W_{j\ell}(t)]\Lambda_{q\ell}\Big\}\Bigg] + \sum_{i=1}^{N}\sum_{q=1}^{Q} Z_{iq}(t)\log(\alpha_q(t)) + \text{const}, \\
&= \sum_{i=1}^{N}\sum_{q=1}^{Q} Z_{iq}(t)\Bigg[\sum_{j=1}^{M}\sum_{\ell=1}^{L}\Big\{(1 - E_{q(A,W)}[A_{ij}(t)])\Big[E_{q(A,W)}[W_{j\ell}(t)]X_{ij}(t)\log(\Lambda_{q\ell}) + \\
&- E_{q(A,W)}[W_{j\ell}(t)]\Lambda_{q\ell}\Big]\Big\} + \log(\alpha_q(t))\Bigg] + \text{const}.
\end{aligned}
$$

We can then recognize the functional form of the multinomial distribution. Thus, we can write:

$$
\log q^*(Z|\theta) = \sum_i \sum_t \sum_q Z_{iq}(t)\log r_{iq(t)} + const. \tag{1}
$$

Taking the exponential on the two sides, we obtain:

$$
q(Z_i) = \prod_{t=1}^{T}\prod_{q=1}^{Q} r_{iq}(t)^{Z_{iq}(t)},
$$

where $r_{iq}(t)$ is denoted by:

$$
\begin{aligned}
r_{iq}(t) \propto \exp\Bigg(&\sum_{j=1}^{M}\sum_{\ell=1}^{L}\Big\{(1 - E_{q(A,W)}[A_{ij}(t)])\Big[E_{q(A,W)}[W_{j\ell}(t)]X_{ij}(t)\log(\Lambda_{q\ell}) + \\
&- E_{q(A,W)}[W_{j\ell}(t)]\Lambda_{q\ell}\Big]\Big\} + \log(\alpha_q(t))\Bigg).
\end{aligned}
$$

However, this distribution needs to be normalized because the matrix $Z(t)$ is a binary matrix and the elements sum to 1 over the values of Q. We can then obtain:

$$q(Z_i) = \prod_{t=1}^{T} \prod_{q=1}^{Q} \tau_{iq}(t)^{Z_{iq}(t)},$$

where

$$\tau_{iq}(t) = \frac{r_{iq}(t)}{\sum_{q_0=1}^{Q} r_{iq_0}(t)}$$

B.3 Derivation of the lower bound

Starting from Eq. 4.6 we obtain the final expression of the variational lower bound $\mathcal{L}(q, \theta)$ by developing the expression:

$$\begin{aligned}
\mathcal{L}(q,\theta) &= \sum_{A,Z,W} q(A,Z,W) \log \frac{p(X|A,Z,W,\Lambda)p(A \mid \pi)p(Z|\alpha)p(W|\beta)}{q(A,Z,W)} \\
&= E_{q(A,Z,W)}\left[\log \frac{p(X|A,Z,W,\Lambda)p(A \mid \pi)p(Z|\alpha)p(W|\beta)}{\prod_{i=1}^{N}\prod_{j=1}^{M}\prod_{t=1}^{T} q(A_{ij}(t)) \prod_{i=1}^{N}\prod_{t=1}^{T} q(Z_i(t)) \prod_{j=1}^{M}\prod_{t=1}^{T} q(W_j(t))}\right] \\
&= E_{q(A,Z,W)}[\log p(X|A,Z,W,\Lambda)] + E_{q(A)}[\log p(A \mid \pi)] + E_{q(Z)}[\log p(Z|\alpha)] + \\
&+ E_{q(W)}[\log p(W|\beta))] - E_{q(Z)}[\log \prod_i q(Z_i)] + \\
&- E_{q(W)}[\log \prod_j q(W_j)] - E_{q(A)}[\log \prod_i \prod_j q(A_{ij})].
\end{aligned}$$

Then we substitute the results obtained in the VE-Step, denoting $E_{q(A,Z,W)}[A_{ij}(t)] = \delta_{ij}(t)$, $E_{q(A,Z,W)}[Z_{iq}(t)] = \tau_{iq}(t)$ and $E_{q(A,Z,W)}[W_{j\ell}(t)] = \eta_{j\ell}(t)$, in order to obtain the final expression of the lower bound that can be written as follows:

$$\begin{aligned}
\mathcal{L}(q,\theta) &= \sum_{t=1}^{T}\sum_{i=1}^{N}\sum_{j=1}^{M} \Bigg\{ \delta_{ij}(t) \log(\pi(t) \mathbf{1}_{\{X_{ij}(t)=0\}}) + (1-\delta_{ij}(t)) \Bigg[\log(1-\pi(t)) + \\
&+ \sum_{q=1}^{Q}\sum_{\ell=1}^{L} \Big\{ \tau_{iq}(t)\eta_{j\ell}(t) X_{ij}(t) \log \Lambda_{q\ell} - \tau_{iq}(t)\eta_{j\ell}(t)\Lambda_{q\ell} \Big\} \Bigg] + \\
&- (1-\delta_{ij}(t)) \log(X_{ij}(t)!) \Bigg\} + \sum_{t=1}^{T}\sum_{i=1}^{N}\sum_{q=1}^{Q} \tau_{iq}(t) \log(\alpha_q(t)) + \\
&+ \sum_{t=1}^{T}\sum_{j=1}^{M}\sum_{\ell=1}^{L} \eta_{j\ell}(t) \log(\beta_\ell(t)) - \sum_{t=1}^{T}\sum_{i=1}^{N}\sum_{q=1}^{Q} \tau_{iq}(t) \log \tau_{iq}(t) + \\
&- \sum_{t=1}^{T}\sum_{j=1}^{M}\sum_{\ell=1}^{L} \eta_{j\ell}(t) \log(\eta_{j\ell}(t)) - \sum_{t=1}^{T}\sum_{i=1}^{N}\sum_{j=1}^{M} \Big(\delta_{ij}(t) \log(\delta_{ij}(t)) + (1-\delta_{ij}(t)) \log(1-\delta_{ij}(t)) \Big).
\end{aligned}$$

B.4 Proof: Update of Λ

To find the optimal update expression of Λ we compute the derivative of the lower bound $\mathcal{L}(q,\theta)$ in Eq. 5.7 with respect to Λ and set it equal to zero, as follows:

$$\frac{\partial \log \mathcal{L}(q,\theta)}{\partial \Lambda_{q\ell}} = \sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{t=1}^{T}(1-\delta_{ij}(t))\Big[\frac{\tau_{iq}(t)\eta_{j\ell}(t)X_{ij}(t)}{\Lambda_{q\ell}} - \tau_{iq}(t)\eta_{j\ell}(t)\Big]$$

$$\sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{t=1}^{T}(1-\delta_{ij}(t))\Big[\tau_{iq}(t)\eta_{j\ell}(t)X_{ij}(t) - \tau_{iq}(t)\eta_{j\ell}(t)\Lambda_{q\ell}\Big] = 0$$

$$\sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{t=1}^{T}(1-\delta_{ij}(t))\tau_{iq}(t)\eta_{j\ell}(t)\Lambda_{q\ell} = \sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{t=1}^{T}\tau_{iq}(t)\eta_{j\ell}(t)\Big[X_{ij}(t) - X_{ij}(t)\delta_{ij}(t)\Big]$$

$$\hat{\Lambda}_{q\ell} = \frac{\sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{t=1}^{T}\tau_{iq}(t)\eta_{j\ell}(t)\Big(X_{ij}(t) - \delta_{ij}(t)X_{ij}(t)\Big)}{\sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{t=1}^{T}\tau_{iq}(t)\eta_{j\ell}(t)\Big(1-\delta_{ij}(t)\Big)}$$

B.5 Algorithmic Consideration

In this section, we provide detailed technical specifications of the neural networks used in the M-Step of the inference algorithm. We employed fully connected neural networks with two hidden layers, each consisting of 200 neurons. The choice of two hidden layers allows for capturing complex patterns and relationships within the data. We did not used mini-batch training, opting for a full-batch approach where the entire training dataset was used in each iteration. In co-clustering, the clusters formed by rows and columns are interconnected. Any updates made to a subset of rows may impact the clustering of the corresponding columns and vice versa. This interdependency makes it challenging to update mini-batches independently, as changes in one batch may affect the quality and coherence of the clustering solution. Within the VEM algorithm, each iteration involved the optimization of the lower bound. For this purpose, we performed 2000 epochs, where an epoch represents a complete pass through the entire dataset. It is worth noting that our results have demonstrated robustness to the choices we made regarding the size of the neural network. Our experiments and evaluations have shown that the approach remains effective and yields reliable results across a range of network sizes. All the experiments in Section 4.4 run on CPU on a MacBook Pro, 2020, with a processor of 2,3 GHz Quad-Core Intel Core i7 and 16 GB of RAM

B.6 Other experiment on simulated data

In this section, we present two experiments conducted on simulated data. The first experiment constitutes a benchmark study in which we compare the performance of our model with other state-of-the-art methods. In the second experiment, we simulate datasets that deviate from the assumptions of the original model. This enables us to assess the robustness of our model's assumptions and evaluate its performance under varying scenarios.

Benchmark study

The goal of this experiment is to compare Zip-dLBM with two state-of-the-art methods to recover the data structure. First, Zip-dLBM is compared with a model based on the same assumptions but which does not take into account the sparsity modeling over time. Denoted by Zip-dLBM$_{\pi(\cdot)=0}$, the model does not take into account the excess of zeros in the data and it is obtained by setting the sparsity parameter $\pi(t)$, with t in $[0, T]$, equal to zero. The other model Zip-dLBM is compared with is dLBM, proposed in Chapter 3, where not only the sparsity is not taken into account but the cluster memberships Z and W are not time-dependent, i.e. cluster switches are not allowed. However, the expected number of interactions between co-clusters (the parameter Λ) changes in time in dLBM.
We also included in the comparison two models that do not consider the dynamic aspect: LBM (Govaert and Nadif, 2008), baseline for model-based co-clustering methods, and k-means (MacQueen, 1967), applied on rows and columns separately. Since the two models do not consider the time aspect, they have been applied at each time instant separately. We chose to evaluate the results with the CARI index. In order to consider the dynamic aspects of the dataset and account for possible switched cluster labels across consecutive time instants, we store the row and column cluster membership results in a unified vector that includes all clustering outcomes over time. This results in vectors with sizes of $N \cdot T$ for row memberships and $M \cdot T$ for column memberships, where N and M represent the number of rows and columns, respectively, and T denotes the total number of time instants. In particular, in order to compare the affectations to the clusters over time, the cluster labels in dLBM were repeated as many times as the number of time instants, and then compared to the affectations of the simulated data using the CARI index. To make this comparison more complete, we defined two simulation scenarios. In Scenario A, the data are simulated as described in Section

4.4.1 but with a constant sparsity level of 80%, fixed in time. In Scenario B the sparsity evolves in time from 80% to 90% . Table 1 displays the results of this comparison, in terms of average CARI values, reported with standard deviations. In Scenario A, Zip-dLBM performs well reaching a CARI value of 0.93, on the other hand Zip-dLBM$_{\pi(\cdot)=0}$ suffers from the excessive number of zeros, whose treatment is not considered, probably affecting the clustering performance. Even worse for dLBM, LBM and kmeans whose CARI index is around 0. For dLBM this is certainly due to the fact that the two latent clustering variables, Z and W, do not evolve over time. LBM and k-means are penalized by the switching clustering labels across different time instants since they are applied independently at each time instant.

In scenario B, Zip-dLBM performs comparably with the previous scenario, with an average CARI index of 0.94 and a smaller standard deviation. Thus, we see that an increasing level of sparsity does not degrade the performance of the model since it is able to distinguish structural zeros from those coming from the Poisson process. This could even help in improving clustering performance. On the contrary, Zip-dLBM$_{\pi(\cdot)=0}$ performs worse than the results obtained in the scenario A probably due to the increased sparsity in the data.

	Zip-dLBM	Zip-dLBM$_{\pi(\cdot)=0}$	dLBM	kmeans	LBM
Scenario A	0.93 ± 0.13	0.27 ± 0.1	0 ± 0	0.01 ± 0	0.01 ± 0.1
Scenario B	0.94 ± 0.03	0.16 ± 0.11	0 ± 0.01	0 ± 0	0.01 ± 0.1

Table 1: Co-clustering results for Zip-dLBM, Zip-dLBM$_{\pi(\cdot)=0}$, dLBM, LBM and kmeans on 50 simulated data according to the two scenarios. Average CARI values are reported with standard deviations.

Robustness to model assumptions

The goal of this experiment on simulated data is to test the performance of Zip-dLBM when data are not simulated according to the model assumptions. Specifically, we decided to simulate the data from a Zero-Inflated negative binomial distribution. The negative binomial distribution is a discrete probability distribution that models the number of successes in a series of iid Bernoulli trials before a given number of failures, r. Following the notation of Section 4.2, being a mixture between the negative binomial distribution and a Dirac mass at zero, the Zero-Inflated negative binomial distribution can be formally written as:

$$\begin{cases} X_{ij}(t)|Z_i(t), W_j(t) = 0 & \text{with probability } \pi(t) \\ X_{ij}(t)|Z_i(t), W_j(t) \sim \mathcal{NB}(X_{ij}(t); r, p) & \text{with probability } 1 - \pi(t). \end{cases}$$

The probability mass function of the negative binomial is given by:

$$f(k, r, p) = \binom{k + r - 1}{k} \cdot (1 - p)^r \cdot p^k = \frac{\Gamma(k + r)}{k!\Gamma(r)}(1 - p)^r p^k,$$

where k is the number of successes and p is the probability of success. When modeling counts data the negative binomial distribution is often a valid alternative to the Poisson one, because it allows the mean and the variance to be different: Mean: $\Lambda = \dfrac{pr}{1-p}$; Variance: $= \dfrac{pr}{(1-p)^2} = \Lambda + \frac{1}{r}\Lambda^2$. A particular property of the negative binomial distribution is that it converges to the Poisson distribution, with expected value Λ, when $r \to \infty$.

To accurately evaluate the performance of Zip-dLBM we simulate with the

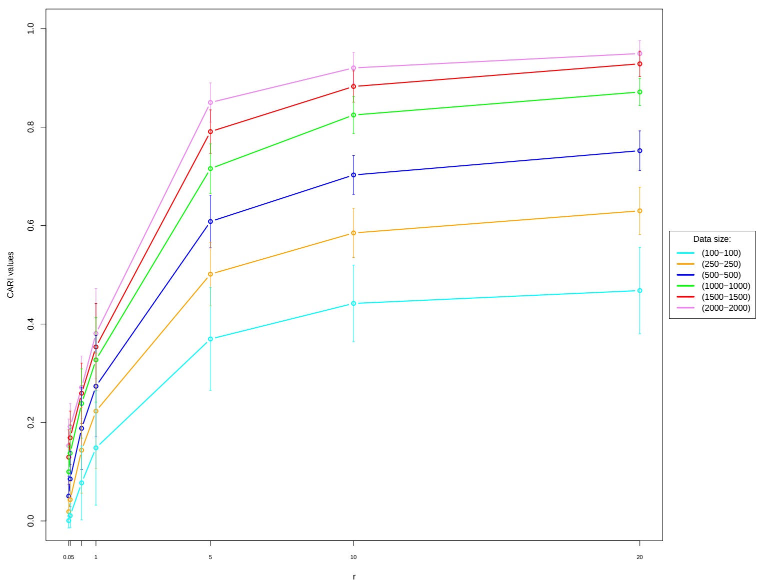


Figure 2: Evolution of the CARI (on the y-axis) according to r (on the x-axis), where each line represent a different data size.

negative binomial distribution data sets for each value of r equal to 0.05, 0.1, 0.5, 1, 5, 10, 20, while keeping the values of Λ unchanged to the ones of the introductory example in Section 4.4.1. Also, to evaluate the robustness of the model to data size we simulate data sets with different size for each value of r. The datasets have been simulated with (row-column) size 100-100, 250-250,

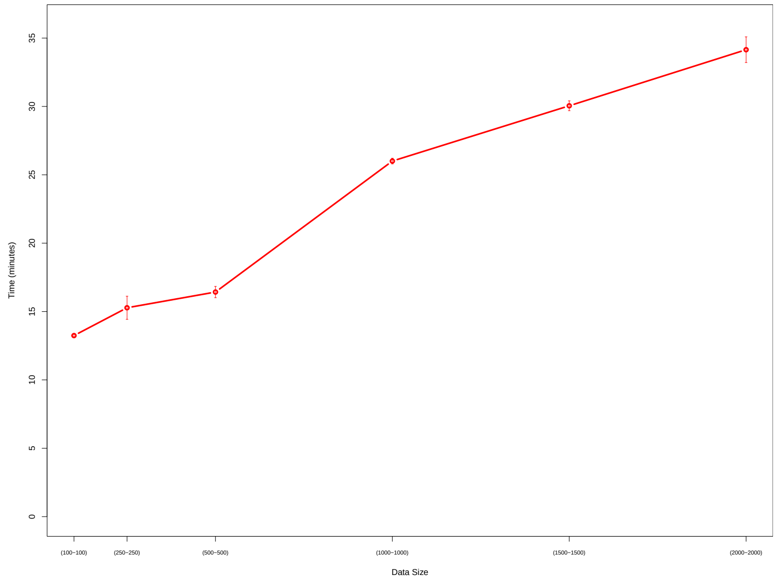


Figure 3: Evolution of the average running time (on the y-axis), displayed with the standard deviation, according to different data sizes (on the x-axis).

500-500, 1000-1000, 1500-1500, 2000-2000, respectively. Figure 2, reports the results of the experiment, depicting the evolution of the values taken by the CARI index (on the y-axis) across different values of the parameter r (on the x-axis) and for different data sizes. Looking at these results we can observe that for values of r very close to zero, all models have difficulties in identifying the correct cluster partition. However, as r increases, models with larger data sizes achieve an average CARI value close to 1, indicating improved clustering performance. Also, we see that the method becomes increasingly robust as the data size increases. In the same experiment, we also evaluate the average running time for each simulated dataset across different r values. Figure 3 presents these results, with the x-axis representing the data size and the y-axis indicating the running time in minutes. Each point on the graph corresponds to the average running time, accompanied by its standard deviation. Notably, we observe that the running time demonstrates a relatively linear relationship with the data size, suggesting scalability of the algorithm.

Bibliography

Adams, R. P. and MacKay, D. J. (2007). Bayesian online changepoint detection. *arXiv preprint arXiv:0710.3742*. Cited on pages 121, 122, and 129.

Ailem, M., Role, F., and Nadif, M. (2015). Co-clustering document-term matrices by direct maximization of graph modularity. In *Proceedings of the 24th ACM international on conference on information and knowledge management*, pages 1807–1810. Cited on page 23.

Ailem, M., Role, F., and Nadif, M. (2017a). Model-based co-clustering for the effective handling of sparse data. *Pattern Recognition*, 72:108–122. Cited on page 21.

Ailem, M., Role, F., and Nadif, M. (2017b). Sparse poisson latent block model for document clustering. *IEEE Transactions on Knowledge and Data Engineering*, 29(7):1563–1576. Cited on page 37.

Banerjee, A., Dhillon, I., Ghosh, J., Merugu, S., and Modha, D. S. (2004). A generalized maximum entropy approach to bregman co-clustering and matrix approximation. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 509–514. Cited on page 24.

Basseville, M., Nikiforov, I. V., et al. (1993). *Detection of abrupt changes: theory and application*, volume 104. prentice Hall Englewood Cliffs. Cited on page 121.

Baudry, J.-P. (2015). Estimation and model selection for model-based clustering with the conditional classification likelihood. Cited on page 31.

Bellman, R. (1957). Dynamic programming princeton university press princeton. *New Jersey Google Scholar*, pages 24–73. Cited on page 5.

Bergé, L. R., Bouveyron, C., Corneli, M., and Latouche, P. (2019). The latent topic block model for the co-clustering of textual interaction data. *Computational Statistics & Data Analysis*, 137:247–270. Cited on pages 19 and 28.

Biernacki, C., Celeux, G., and Govaert, G. (2000). Assessing a mixture model for clustering with the integrated completed likelihood. *IEEE transactions on* *pattern analysis and machine intelligence*, 22(7):719–725. Cited on pages 30, 59, and 96.

Biernacki, C. and Jacques, J. (2016). Model-based clustering of multivariate ordinal data relying on a stochastic binary search algorithm. *Statistics and Computing*, 26:929–943. Cited on page 28.

Biernacki, C., Jacques, J., and Keribin, C. (2023). A survey on model-based co-clustering: High dimension and estimation challenges. *Journal of Classification*, pages 1–50. Cited on pages 18, 27, and 43.

Bishop, C. M. (2006). Approximate inference. pages 461–517. Springer-Verlag, Berlin, Heidelberg. Cited on page 91.

Bock, H. (1979). Simultaneous clustering of objects and variables. *Analyse des données et Informatique*, pages 187–203. Cited on page 19.

Boutalbi, R., Labiod, L., and Nadif, M. (2020). Tensor latent block model for co-clustering. *International Journal of Data Science and Analytics*, pages 1–15. Cited on pages 37, 57, and 66.

Boutalbi, R., Labiod, L., and Nadif, M. (2021). Implicit consensus clustering from multiple graphs. *Data Mining and Knowledge Discovery*, 35(6):2313–2340. Cited on page 37.

Bouveyron, C., Bozzi, L., Jacques, J., and Jollois, F.-X. (2018). The functional latent block model for the co-clustering of electricity consumption curves. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 67(4):897–915. Cited on pages 36 and 57.

Bouveyron, C. and Brunet-Saumard, C. (2014). Model-based clustering of high-dimensional data: A review. *Computational Statistics & Data Analysis*, 71:52–78. Cited on page 18.

Bouveyron, C., Jacques, J., Schmutz, A., Simoes, F., and Bottini, S. (2022). Co-clustering of multivariate functional data for the analysis of air pollution in the south of france. *The Annals of Applied Statistics*, 16(3):1400–1422. Cited on page 36.

Brault, V. (2014). *Estimation et sélection de modèle pour le modèle des blocs latents*. PhD thesis, Université Paris Sud-Paris XI. Cited on pages 43 and 44.

Brault, V. and Mariadassou, M. (2015). Co-clustering through latent bloc model: A review. Cited on page 28.

Casa, A., Bouveyron, C., Erosheva, E., and Menardi, G. (2021). Co-clustering of time-dependent data via the shape invariant model. *Journal of Classification*, 38(3):626–649. Cited on page 37.

Celeux, G. (1985). The sem algorithm: a probabilistic teacher algorithm derived from the em algorithm for the mixture problem. *Computational statistics quarterly*, 2:73–82. Cited on page 41.

Celeux, G. and Diebolt, J. (1992). A stochastic approximation type em algorithm for the mixture problem. *Stochastics: An International Journal of Probability and Stochastic Processes*, 41(1-2):119–134. Cited on page 41.

Celeux, G. and Govaert, G. (1992). A classification em algorithm for clustering and two stochastic versions. *Computational statistics & Data analysis*, 14(3):315–332. Cited on page 44.

Celisse, A., Daudin, J.-J., and Pierre, L. (2012). Consistency of maximum-likelihood and variational estimators in the stochastic block model. Cited on pages 30 and 43.

Cheng, K.-O., Law, N.-F., Siu, W.-C., and Liew, A. W.-C. (2008). Identification of coherent patterns in gene expression data using an efficient biclustering algorithm and parallel coordinate visualization. *BMC bioinformatics*, 9(1):210. Cited on page 19.

Cheng, S. W. and Thaga, K. (2005). Multivariate max-cusum chart. *Quality Technology & Quantitative Management*, 2(2):221–235. Cited on page 121.

Chiquet, J., Mariadassou, M., and Robin, S. (2018). Variational inference for probabilistic poisson pca. *The Annals of Applied Statistics*, 12(4):2674–2698. Cited on page 19.

Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*. Cited on page 47.

Côme, E. and Latouche, P. (2015). Model selection and clustering in stochastic block models based on the exact integrated complete data likelihood. *Statistical Modelling*, 15(6):564–589. Cited on page 60.

Corneli, M. (2017). *Dynamic stochastic block models, clustering and segmentation in dynamic graphs*. PhD thesis. 2017PA01E012. Cited on page 35.

Corneli, M., Bouveyron, C., and Latouche, P. (2020). Co-clustering of ordinal data via latent continuous random variables and not missing at random entries. *Journal of Computational and Graphical Statistics*, pages 1–15. Cited on page 28.

Corneli, M., Latouche, P., and Rossi, F. (2016). Block modelling in dynamic networks with non-homogeneous poisson processes and exact icl. *Social Network Analysis and Mining*, 6(1):55. Cited on pages 35, 53, and 57.

Daudin, J.-J., Picard, F., and Robin, S. (2008). A mixture model for random graphs. *Statistics and computing*, 18(2):173–183. Cited on page 18.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22. Cited on pages 38, 39, and 89.

Deodhar, M. and Ghosh, J. (2010). Scoal: A framework for simultaneous co-clustering and learning from complex data. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 4(3):1–31. Cited on page 19.

Dhillon, I. S. (2001). Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 269–274. Cited on page 22.

Dhillon, I. S., Mallela, S., and Kumar, R. (2003a). A divisive information-theoretic feature clustering algorithm for text classification. *Journal of machine learning research*, 3(Mar):1265–1287. Cited on page 19.

Dhillon, I. S., Mallela, S., and Modha, D. S. (2003b). Information-theoretic co-clustering. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 89–98. Cited on pages 19, 22, and 23.

Ding, C., Li, T., Peng, W., and Park, H. (2006). 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. Cited on page 22.

DuBois, C., Butts, C., and Smyth, P. (2013). Stochastic blockmodeling of relational event dynamics. In *Artificial intelligence and statistics*, pages 238–246. PMLR. Cited on page 35.

Forbes, F., Arnaud, A., Lemasson, B., and Barbier, E. (2019). Component elimination strategies to fit mixtures of multiple scale distributions. In *Statistics and Data Science: Research School on Statistics and Data Science, RSSDS 2019, Melbourne, VIC, Australia, July 24–26, 2019, Proceedings 1*, pages 81–95. Springer. Cited on page 97.

Fruhwirth-Schnatter, S., Celeux, G., and Robert, C. P. (2019). *Handbook of mixture analysis*. CRC press. Cited on page 39.

Gent, C. and Sheppard, C. (1992). Special feature. predicting time series by a fully connected neural network trained by back propagation. *Computing & Control Engineering Journal*, 3(3):109–112. Cited on page 94.

George, T. and Merugu, S. (2005). A scalable collaborative filtering framework based on co-clustering. In *Fifth IEEE International Conference on Data Mining (ICDM'05)*, pages 4 pp.–. Cited on page 19.

Giraud, C. (2021). *Introduction to high-dimensional statistics*. CRC Press. Cited on page 5.

Goffinet, E., Lebbah, M., Azzag, H., Loïc, G., and Coutant, A. (2022). Functional non-parametric latent block model: A multivariate time series clustering approach for autonomous driving validation. *Computational Statistics & Data Analysis*, 176:107565. Cited on page 36.

Good, I. (1965). Categorization of classification. mathematics and computer science in biology and medicine. *Her Majesty's Stationary O ce, London*. Cited on page 19.

Govaert, G. (1983). *Classification croisée*. Éditeur inconnu. Cited on page 19.

Govaert, G. and Nadif, M. (2003). Clustering with block mixture models. *Pattern Recognition*, 36(2):463–473. Cited on pages 18, 24, and 25.

Govaert, G. and Nadif, M. (2008). Block clustering with bernoulli mixture models: Comparison of different approaches. *Computational Statistics & Data Analysis*, 52(6):3233–3245. Cited on pages 29, 38, 52, 90, and 159.

Govaert, G. and Nadif, M. (2010). Latent block model for contingency table. *Communications in Statistics - Theory and Methods*, 39(3):416–425. Cited on page 27.

Green, N., Rege, M., Liu, X., and Bailey, R. (2011). Evolutionary spectral co-clustering. In *The 2011 international joint conference on neural networks*, pages 1074–1081. IEEE. Cited on pages 36 and 52.

Guo, D., Xu, F., Li, Z., Nie, Z., and Shao, H. (2017). Design, verification, and application of new discrete-time recurrent neural network for dynamic nonlinear equations solving. *IEEE Transactions on Industrial Informatics*, 14(9):3936–3945. Cited on page 47.

Guyon, I. and Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182. Cited on page 19.

Hanisch, D., Zien, A., Zimmer, R., and Lengauer, T. (2002). Co-clustering of biological networks and gene expression data. *Bioinformatics*, 18(suppl_1):S145–S154. Cited on page 19.

Hartigan, J. (2000). Bloc voting in the united states senate. *Journal of Classification*, 17:29–49. Cited on page 24.

Hartigan, J. A. (1975). Printer graphics for clustering. *Journal of Statistical Computation and Simulation*, 4(3):187–213. Cited on page 19.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780. Cited on pages 47 and 127.

Holland, P. W., Laskey, K. B., and Leinhardt, S. (1983). Stochastic blockmodels: First steps. *Social networks*, 5(2):109–137. Cited on page 31.

Jaakkola, T. S. and Jordan, M. I. (1997). A variational approach to bayesian logistic regression models and their extensions. In *Sixth International Workshop on Artificial Intelligence and Statistics*, pages 283–294. PMLR. Cited on page 90.

Jacques, J. and Biernacki, C. (2018). Model-based co-clustering for ordinal data. *Computational Statistics & Data Analysis*, 123:101–115. Cited on page 28.

Jagalur, M., Pal, C., Learned-Miller, E., Zoeller, R. T., and Kulp, D. (2007). Analyzing in situ gene expression in the mouse brain with image registration, feature extraction and block clustering. In *BMC bioinformatics*, volume 8, pages 1–21. BioMed Central. Cited on page 19.

Jernite, Y., Latouche, P., Bouveyron, C., Rivera, P., Jegou, L., and Lamassé, S. (2014). The random subgraph model for the analysis of an ecclesiastical network in merovingian gaul. Cited on page 46.

Jolliffe, I. T. (2002). *Principal component analysis for special types of data*. Springer. Cited on page 19.

Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., and Saul, L. K. (1998). An introduction to variational methods for graphical models. In *Learning in graphical models*, pages 105–161. Springer. Cited on page 90.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. Cited on page 46.

Kavitha, V. and Punithavalli, M. (2010). Clustering time series data stream-a literature survey. *arXiv preprint arXiv:1005.4270*. Cited on page 121.

Kawahara, Y. and Sugiyama, M. (2009). Change-point detection in time-series data by direct density-ratio estimation. In *Proceedings of the 2009 SIAM international conference on data mining*, pages 389–400. SIAM. Cited on page 122.

Kemp, C., Tenenbaum, J. B., Griffiths, T. L., Yamada, T., and Ueda, N. (2006). Learning systems of concepts with an infinite relational model. In *AAAI*, volume 3, page 5. Cited on page 24.

Keribin, C., Brault, V., Celeux, G., and Govaert, G. (2015). Estimation and selection for the latent block model on categorical data. *Statistics and Computing*, 25(6):1201–1216. Cited on pages 27, 28, 29, 30, and 44.

Keribin, C., Brault, V., Celeux, G., Govaert, G., et al. (2012). Model selection for the binary latent block model. In *Proceedings of COMPSTAT*, volume 2012. Cited on pages 38, 42, and 44.

Keribin, C., Celeux, G., and Robert, V. (2017). The Latent Block Model: a useful model for high dimensional data. In *ISI 2017 - 61st world statistics congress*, pages 1–6, Marrakech, Morocco. Cited on page 60.

Keribin, C., Govaert, G., and Celeux, G. (2010). Estimation d'un modèle à blocs latents par l'algorithme SEM. Cited on pages 42 and 57.

Kidger, P. (2022). On neural differential equations. *arXiv preprint arXiv:2202.02435*. Cited on page 47.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*. Cited on pages 94 and 128.

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*. Cited on page 44.

Kock, L., Klein, N., and Nott, D. J. (2022). Variational inference and sparsity in high-dimensional deep gaussian mixture models. *Statistics and Computing*, 32(5):70. Cited on page 45.

Kondratev, A., Salminen, K., Rantala, J., Salpavaara, T., Verho, J., Surakka, V., Lekkala, J., Vehkaoja, A., and Müller, P. (2022). A comparison of online methods for change point detection in ion-mobility spectrometry data. *Array*, page 100151. Cited on page 121.

Krishnan, R., Shalit, U., and Sontag, D. (2017). Structured inference networks for nonlinear state space models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31. Cited on page 46.

Kumar, S., Gao, X., and Welch, I. (2016). Co-clustering for dual topic models. In *AI 2016: Advances in Artificial Intelligence: 29th Australasian Joint Conference, Hobart, TAS, Australia, December 5-8, 2016, Proceedings 29*, pages 390–402. Springer. Cited on page 28.

Labiod, L. and Nadif, M. (2011a). Co-clustering for binary and categorical data with maximum modularity. In *2011 IEEE 11th international conference on data mining*, pages 1140–1145. IEEE. Cited on page 23.

Labiod, L. and Nadif, M. (2011b). Co-clustering under nonnegative matrix tri-factorization. In *International Conference on Neural Information Processing*, pages 709–717. Springer. Cited on page 22.

Lambert, D. (1992). Zero-inflated poisson regression, with an application to defects in manufacturing. *Technometrics*, 34(1):1–14. Cited on pages 87 and 88.

Langlade, C., Gouverneur, A., Bosco-Lévy, P., Gouraud, A., Pérault-Pochat, M.-C., Béné, J., Miremont-Salamé, G., Pariente, A., and of Pharmacovigilance Centres, F. N. (2019). Adverse events reported for mirena levonorgestrel-releasing intrauterine device in france and impact of media coverage. *British Journal of Clinical Pharmacology*, 85(9):2126–2133. Cited on page 71.

Latouche, P., Birmelé, E., and Ambroise, C. (2011). Overlapping stochastic block models with application to the french political blogosphere. Cited on page 45.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553). Cited on pages 38 and 46.

Lee, D. and Seung, H. S. (2000). Algorithms for non-negative matrix factorization. *Advances in neural information processing systems*, 13. Cited on page 22.

Li, L., Yang, M., Wang, C., and Wang, B. (2016). Cubature kalman filter based point set registration for slam. In *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pages 847–852. IEEE. Cited on page 142.

Li, N., Elashoff, D. A., Robbins, W. A., and Xun, L. (2011). A hierarchical zero-inflated log-normal model for skewed responses. *Statistical Methods in Medical Research*, 20(3):175–189. Cited on page 88.

Lindstrom, M. J. (1995). Self-modelling with random shift and scale parameters and a free-knot spline shape function. *Statistics in medicine*, 14(18):2009–2021. Cited on page 37.

Lomet, A. (2012). *Sélection de modèle pour la classification croisée de données continues.* PhD thesis, Compiègne. Cited on page 27.

MacQueen, J. (1967). Classification and analysis of multivariate observations. In *5th Berkeley Symp. Math. Statist. Probability*, pages 281–297. University of California Los Angeles LA USA. Cited on pages 116 and 159.

MacQueen, J. et al. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA. Cited on page 17.

Madeira, S. C. and Oliveira, A. L. (2004). Biclustering algorithms for biological data analysis: a survey. *IEEE/ACM transactions on computational biology and bioinformatics*, 1(1):24–45. Cited on page 21.

Malsiner-Walli, G., Frühwirth-Schnatter, S., and Grün, B. (2016). Model-based clustering based on sparse finite gaussian mixtures. *Statistics and computing*, 26(1-2):303–324. Cited on page 97.

Mariadassou, M. and Matias, C. (2015). Convergence of the groups posterior distribution in latent or stochastic block models. Cited on page 43.

Matias, C. and Miele, V. (2017). Statistical clustering of temporal networks through a dynamic stochastic block model. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 79(4):1119–1141. Cited on page 35.

Matias, C., Rebafka, T., and Villers, F. (2018). A semiparametric extension of the stochastic block model for longitudinal networks. *Biometrika*, 105(3):665–680. Cited on pages 35, 36, and 53.

Maugis, C., Celeux, G., and Martin-Magniette, M.-L. (2009). Variable selection for clustering with gaussian mixture models. *Biometrics*, 65(3):701–709. Cited on pages 19 and 84.

McLachlan, G. J., Lee, S. X., and Rathnayake, S. I. (2019). Finite mixture models. *Annual review of statistics and its application*, 6:355–378. Cited on page 18.

Mehta, N., Duke, L. C., and Rai, P. (2019). Stochastic blockmodels meet graph neural networks. In *International Conference on Machine Learning*, pages 4466–4474. PMLR. Cited on page 45.

Montgomery, D. C. (2020). *Introduction to statistical quality control*. John Wiley & Sons. Cited on page 121.

Myronenko, A. and Song, X. (2010). Point set registration: Coherent point drift. *IEEE transactions on pattern analysis and machine intelligence*, 32(12):2262–2275. Cited on pages 142 and 144.

Nadif, M. and Govaert, G. (2010). Model-based co-clustering for continuous data. In *2010 Ninth international conference on machine learning and applications*, pages 175–180. IEEE. Cited on page 27.

Norris, J. R. (1998). *Markov chains*. Number 2. Cambridge university press. Cited on page 35.

Nowicki, K. and Snijders, T. A. B. (2001). Estimation and prediction for stochastic blockstructures. *Journal of the American statistical association*, 96(455):1077–1087. Cited on pages 31 and 34.

Ospina, R. and Ferrari, S. L. (2012). A general class of zero-or-one inflated beta regression models. *Computational Statistics & Data Analysis*, 56(6):1609–1623. Cited on page 88.

Parisi, G. and Shankar, R. (1988). Statistical field theory. Cited on page 43.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. (2017). Automatic differentiation in pytorch. Cited on pages 94 and 128.

Qi, C. R., Su, H., Mo, K., and Guibas, L. J. (2017). Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660. Cited on page 143.

Raftery, A. E. and Dean, N. (2006). Variable selection for model-based clustering. *Journal of the American Statistical Association*, 101(473):168–178. Cited on pages 19 and 84.

Raiffa, H., Schlaifer, R., et al. (1961). Applied statistical decision theory. Cited on page 28.

Raissi, M. (2018). Deep hidden physics models: Deep learning of nonlinear partial differential equations. *The Journal of Machine Learning Research*, 19(1):932–955. Cited on page 47.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. (2018). Multistep neural networks for data-driven discovery of nonlinear dynamical systems. *arXiv preprint arXiv:1801.01236*. Cited on page 47.

Ramsay, J. and Silverman, B. (2005). Principal components analysis for functional data. *Functional data analysis*, pages 147–172. Cited on page 36.

Rand, W. M. (1971). Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850. Cited on pages 61, 99, and 132.

Rangapuram, S. S., Seeger, M. W., Gasthaus, J., Stella, L., Wang, Y., and Januschowski, T. (2018). Deep state space models for time series forecasting. *Advances in neural information processing systems*, 31. Cited on page 46.

Ridout, M., Hinde, J., and Demétrio, C. G. (2001). A score test for testing a zero-inflated poisson regression model against zero-inflated negative binomial alternatives. *Biometrics*, 57(1):219–223. Cited on page 88.

Robert, V. (2017). *Classification croisée pour l'analyse de bases de données de grandes dimensions de pharmacovigilance*. PhD thesis, Université Paris-Saclay. Cited on page 31.

Robert, V., Celeux, G., and Keribin, C. (2015). Un modèle statistique pour la pharmacovigilance. In *47èmes Journées de Statistique de la SFdS*, Lille, France. Cited on pages 29 and 52.

Robert, V., Vasseur, Y., and Brault, V. (2020). Comparing high-dimensional partitions with the co-clustering adjusted rand index. *Journal of Classification*, pages 1–29. Cited on pages 66, 99, 116, and 132.

Rocci, R. and Vichi, M. (2008). Two-mode multi-partitioning. *Computational Statistics & Data Analysis*, 52(4):1984–2003. Cited on page 22.

Rooth, M. (1995). Two-dimensional clusters in grammatical relations. In *AAAI Symposium on representation and acquisition of lexical knowledge*. Cited on page 24.

Sang, C.-Y. and Sun, D.-H. (2014). Co-clustering over multiple dynamic data streams based on non-negative matrix factorization. *Applied intelligence*, 41:487–502. Cited on page 52.

Schwarz, G. (1978). Estimating the dimension of a model. *The annals of statistics*, pages 461–464. Cited on page 30.

Selosse, M., Gormley, C., Jacques, J., and Biernacki, C. (2020). A bumpy journey: exploring deep gaussian mixture models. In *"I Can't Believe It's Not Better!"NeurIPS 2020 workshop*. Cited on page 45.

Shan, H. and Banerjee, A. (2008). Bayesian co-clustering. In *2008 Eighth IEEE International Conference on Data Mining*, pages 530–539. IEEE. Cited on pages 19, 24, and 28.

Smagulova, K. and James, A. P. (2019). A survey on lstm memristive neural network architectures and applications. *The European Physical Journal Special Topics*, 228(10):2313–2324. Cited on page 47.

Steinbach, M., Ertöz, L., and Kumar, V. (2004). The challenges of clustering high dimensional data. In *New directions in statistical physics: econophysics, bioinformatics, and pattern recognition*, pages 273–309. Springer. Cited on page 5.

Stephens, M. (2000). Dealing with label switching in mixture models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 62(4):795–809. Cited on page 42.

Tang, Y., Salakhutdinov, R., and Hinton, G. (2012). Deep mixtures of factor analysers. *arXiv preprint arXiv:1206.4635*. Cited on page 44.

Tipping, M. E. and Bishop, C. M. (1999). Probabilistic principal component analysis. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 61(3):611–622. Cited on pages 19 and 45.

Van den Burg, G. J. and Williams, C. K. (2020). An evaluation of change point detection algorithms. *arXiv preprint arXiv:2003.06222*. Cited on page 121.

Van den Oord, A. and Schrauwen, B. (2014). Factoring variations in natural images with deep gaussian mixture models. *Advances in neural information processing systems*, 27. Cited on page 45.

van Dijk, B., van Rosmalen, J., and Paap, R. (2009). A bayesian approach to two-mode clustering. Technical report. Cited on pages 28 and 29.

Viard, D., Parassol-Girard, N., Romani, S., Van Obberghen, E., Rocher, F., Berriri, S., and Drici, M.-D. (2019). Spontaneous adverse event notifications by patients subsequent to the marketing of a new formulation of levothyrox® amidst a drug media crisis: atypical profile as compared with other drugs. *Fundamental & clinical pharmacology*, 33(4):463–470. Cited on pages 71 and 113.

Viroli, C. and McLachlan, G. J. (2019). Deep gaussian mixture models. *Statistics and Computing*, 29:43–51. Cited on page 45.

Von Luxburg, U. (2007). A tutorial on spectral clustering. *Statistics and computing*, 17:395–416. Cited on page 17.

Vuillien, M. (2020). *Systèmes d'élevage et pastoralisme en Provence et dans les Alpes méridionales durant la Protohistoire: Nouvelles perspectives en archéozoologie*. PhD thesis, Université Côte d'Azur. Cited on page 143.

Wang, P., Domeniconi, C., and Laskey, K. B. (2009). Latent dirichlet bayesian co-clustering. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 522–537. Springer. Cited on pages 19 and 28.

Wang, Y. J. and Wong, G. Y. (1987). Stochastic blockmodels for directed graphs. *Journal of the American Statistical Association*, 82(397):8–19. Cited on page 31.

Witten, D. M. and Tibshirani, R. (2010). A framework for feature selection in clustering. *Journal of the American Statistical Association*, 105(490):713–726. Cited on page 19.

Wu, C. J. (1983). On the convergence properties of the em algorithm. *The Annals of statistics*, pages 95–103. Cited on page 40.

Wyse, J. and Friel, N. (2012a). Block clustering with collapsed latent block models. *Statistics and Computing*, 22:415–428. Cited on pages 19, 28, 29, and 38.

Wyse, J. and Friel, N. (2012b). Block clustering with collapsed latent block models. *Statistics and Computing*, 22(2):415–428. Cited on page 31.

Wyse, J., Friel, N., and Latouche, P. (2017). Inferring structure in bipartite networks using the latent blockmodel and exact ICL. *Network Science*, 5(1):45–69. Cited on pages 31 and 38.

Xu, B., Bu, J., Chen, C., and Cai, D. (2012). An exploration of improving collaborative recommender systems via user-item subgroups. In *Proceedings of the 21st international conference on World Wide Web*, pages 21–30. Cited on page 19.

Xu, D., Cheng, W., Zong, B., Ni, J., Song, D., Yu, W., Chen, Y., Chen, H., and Zhang, X. (2019). Deep co-clustering. In *Proceedings of the 2019 SIAM International Conference on Data Mining*, pages 414–422. SIAM. Cited on pages 45 and 46.

Xu, K. S. and Hero, A. O. (2014). Dynamic stochastic blockmodels for time-evolving social networks. *IEEE Journal of Selected Topics in Signal Processing*, 8(4):552–562. Cited on page 46.

Yang, T., Chi, Y., Zhu, S., Gong, Y., and Jin, R. (2011). Detecting communities and their evolutions in dynamic social networks—a bayesian approach. *Machine learning*, 82(2):157–189. Cited on page 34.

Zhang, Z. (1994). Iterative point matching for registration of free-form curves and surfaces. *International journal of computer vision*, 13(2):119–152. Cited on page 142.

Zhou, F. and De la Torre, F. (2015). Factorized graph matching. *IEEE transactions on pattern analysis and machine intelligence*, 38(9):1774–1789. Cited on page 142.

Zhou, Z., Zheng, J., Dai, Y., Zhou, Z., and Chen, S. (2014). Robust non-rigid point set registration using student's-t mixture model. *PloS one*, 9(3):e91381. Cited on page 142.

Zhu, H., Guo, B., Zou, K., Li, Y., Yuen, K.-V., Mihaylova, L., and Leung, H. (2019). A review of point set registration: From pairwise registration to groupwise registration. *Sensors*, 19(5):1191. Cited on page 142.

Zhu, X., Ding, M., Huang, T., Jin, X., and Zhang, X. (2018). Pcanet-based structural representation for nonrigid multimodal medical image registration. *Sensors*, 18(5):1477. Cited on page 142.

Zreik, R., Latouche, P., and Bouveyron, C. (2017). The dynamic random subgraph model for the clustering of evolving networks. *Computational Statistics*, 32(2):501–533. Cited on page 46.