POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Executive Summary of the Thesis

The DRPM Strikes Back: More Flexibility for a Bayesian Spatio-Temporal Clustering Model

Laurea Magistrale in Mathematical Engineering - Ingegneria Matematica

Author: Federico Angelo Mor

Advisor: Prof. Alessandra Guglielmi

Co-advisor: Alessandro Carminati

Academic year: 2023-2024

1. Introduction

Clustering is a fundamental technique of unsupervised learning where a set of data points has to be divided into homogenous groups of units which exhibit a similar behaviour relative to a target variable. It has long been a powerful tool for identifying structures and patterns within data, gaining increased popularity across a variety of scientific fields including social sciences, climate and environmental analysis, economics, and healthcare. Nowadays, as the accessibility of complex datasets has increased, the prevailing approach to tackling clustering challenges has transitioned from traditional algorithmic methods—such as hierarchical, partition-based, or density-based methods—to model-based methods, which assume a probabilistic modelling of the data. Thus, the development of these models finds its natural context within the Bayesian framework.

The effectiveness of Bayesian clustering models is particularly evident in the complex scenarios of spatio-temporal data, in which observations are collected over time and across various spatial locations. This scenario is inherently complex due to the interplay between spatial and temporal dimensions, a complexity that is further compounded when covariates are also available. Nevertheless, Bayesian models effectively integrate all these levels of information into the model, thereby improving the quality of the results.

Over the years, several models have been proposed in the Bayesian literature. Among these, the Dependent Random Partition Model (Page et al., 2022) stands out as it explicitly addresses the temporal dependence of partitions into the model formulation. However, the current implementation of its MCMC algorithm, written in C and accessible through an R interface, lacks some significant features, which comprise the inclusion of covariates, the handling of missing data, and a more efficient implementation.

In this work, we tackled these three issues by enhancing the flexibility of the original model, yielding improved performances from both the statistical and computational perspectives.

2. Description of the model

The main clustering tool employed in Bayesian clustering methods is a mixture model, which can naturally be extended to an infinite-mixture model within Bayesian nonparametrics. This extension specifies the conditional distribution of data points given a realization of the partition of the units, and a prior is assigned to this par-

tition by means of the discreteness of the assigned random probability measure (RPM). In the spatio-temporal context, where we analyse the sequence of partitions $\rho_1, \dots, \rho_T$, i.e. the clusters of the n experimental units at each time instant $t = 1, \dots, T$, this choice is not always effective. In fact, in infinite mixture models, partitions are only *induced* by the random partition model; yet, when the sequence of partitions is the actual inferential object of interest, then this sequence should be modelled *directly*, in order to capture more precisely the temporal dependencies exhibited by the partitions. While many Bayesian nonparametric methods that temporally correlate a sequence of random probability measures have been developed, they typically implement this temporal dependence through the atoms or weights associated with the representation of the discrete RPM. However, the correlation modelled in these underlying parameters does not necessarily reflects into correlations among the induced partitions. In contrast, the Dependent Random Partition Model (DRPM) (Page et al., 2022) explicitly models in its formulation the temporal dependencies among the partitions. This characteristic facilitates a more effective identification of temporal trends, resulting in clusters that evolve in a more gentle and interpretable way.

2.1. Temporal modelling for sequences of partitions

We now describe how DRPM models these temporal relationships within sequences of partitions. Before delving into the explanation, we first establish some general notation. We consider a spatio-temporal context where there are n units to be clustered at all time points $t = 1, \dots, T$. Each unit is represented as $i = 1, \dots, n$. We denote by $\rho_t = \{S_{1t}, \dots, S_{k_t t}\}$ the partition at time t of the n experimental units, composed by k_t clusters. An alternative representation of this partition is possible through cluster labels $\boldsymbol{c}_t = \{c_{1t}, \dots, c_{nt}\}$, with $c_{it} = j$ indicating that unit i belongs to cluster S_{jt}. Finally, we will denote with a $\star$ superscript all the variables or quantities which are cluster-specific.
Introducing temporal dependence in a collection of partitions requires the formulation of a joint probability model for $(\rho_1, \dots, \rho_T)$. Temporal dependence among the ρ_t's implies that the cluster configuration in ρ_t may be influenced by the cluster configurations found in $\rho_{t-1}, \rho_{t-2}, \dots, \rho_1$. However, Page et al. (2022) restricted this temporal connection to a first-order Markovian structure, where the conditional distribution of ρ_t given all the predecessors $\rho_{t-1}, \rho_{t-2}, \dots, \rho_1$ actually depends only on ρ_{t-1}. This led to the construction of the joint probability model for $(\rho_1, \dots, \rho_T)$ as

$$P(\rho_1, \dots, \rho_T) = P(\rho_T|\rho_{T-1}) \cdots P(\rho_2|\rho_1)P(\rho_1) \tag{1}$$

Here, $P(\rho_1)$ is an exchangeable partition probability function (EPPF), implemented by (Page et al., 2022) through a Chinese restaurant process in the form

$$P(\rho_1|M) \propto \prod_{j=1}^{k_1} M \cdot (|S_j| - 1)! \tag{2}$$

which describes how the n experimental units at time period 1 are grouped into k_1 distinct clusters. In Section 2.2 we will detail how this EPPF can be adapted to incorporate spatial and covariates information.
To characterize the other terms $P(\rho_t|\rho_{t-1})$ in (1), the following auxiliary variables need to be introduced for all units $i = 1, \dots, n$.

$$\gamma_{it} = \begin{cases} 1 & \text{if unit } i \text{ is } \textit{not} \text{ reallocated when} \\ & \text{moving from time } t-1 \text{ to } t \\ 0 & \text{otherwise} \end{cases}$$

These parameters model the similarity between ρ_{t-1} and ρ_t. If the partitions ρ_{t-1} and ρ_t are highly dependent, their cluster configurations will change minimally, resulting in only few units changing their cluster assignments. Conversely, partitions exhibiting low dependence will likely manifest significantly different cluster configurations, leading to a majority of the units being reallocated. By construction, we set $\gamma_{i1} = 0$ for all i, meaning that at the first time instant all units will get reallocated as there is no partition at time $t = 0$. Page et al. (2022) assumed $\gamma_{it} \overset{\text{ind}}{\sim} \text{Ber}(\alpha_t)$ where $\alpha_t \in [0, 1]$ serves as a temporal dependence parameter. Specifically, $\alpha_t = 1$ denotes perfect temporal dependence, with $\rho_t = \rho_{t-1}$, while $\alpha_t = 0$ implies full independence of ρ_t from ρ_{t-1}. For clarity, the vector $\boldsymbol{\gamma}_t = (\gamma_{1t}, \dots, \gamma_{nt})$ is introduced, so that the T pairs of parameters $(\boldsymbol{\gamma}_j, \rho_j)$ are explicitly reported in the augmented formulation of the joint model (1), which becomes

$$P(\boldsymbol{\gamma}_1, \rho_1, \dots, \boldsymbol{\gamma}_T, \rho_T) = P(\rho_T|\boldsymbol{\gamma}_T, \rho_{T-1}P(\boldsymbol{\gamma}_T) \cdots P(\rho_2|\boldsymbol{\gamma}_2, \rho_1)P(\boldsymbol{\gamma}_2)P(\rho_1) \tag{3}$$

Once the partition model is specified, there is considerable flexibility in how to define the remainder of the Bayesian model. The complete formulation designed by Page et al. (2022) is provided in (4),

$$\begin{aligned}
Y_{it}|Y_{it-1}, \boldsymbol{\mu}_t^\star, \boldsymbol{\sigma}_t^{2\star}, \boldsymbol{\eta}, \boldsymbol{c}_t &\overset{\text{ind}}{\sim} \mathcal{N}(\mu^\star_{c_{it}t} + \eta_{1i} Y_{it-1}, \sigma^{2\star}_{c_{it}t}(1-\eta_{1i}^2)) \\
Y_{i1} &\overset{\text{ind}}{\sim} \mathcal{N}(\mu^\star_{c_{i1}1}, \sigma^{2\star}_{c_{i1}1}) \\
\xi_i = \text{Logit}(\tfrac{1}{2}(\eta_{1i}+1)) &\overset{\text{ind}}{\sim} \text{Laplace}(a, b) \\
(\mu^\star_{jt}, \sigma^\star_{jt}) &\overset{\text{ind}}{\sim} \mathcal{N}(\vartheta_t, \tau_t^2) \times \mathcal{U}(0, A_\sigma) \\
\vartheta_t|\vartheta_{t-1} &\overset{\text{ind}}{\sim} \mathcal{N}((1-\varphi_1)\varphi_0 + \varphi_1\vartheta_{t-1}, \lambda^2(1-\varphi_1^2)) \\
(\vartheta_1, \tau_t) &\overset{\text{iid}}{\sim} \mathcal{N}(\varphi_0, \lambda^2) \times \mathcal{U}(0, A_\tau) \\
(\varphi_0, \varphi_1, \lambda) &\sim \mathcal{N}(m_0, s_0^2) \times \mathcal{U}(-1, 1) \times \mathcal{U}(0, A_\lambda) \\
\{\boldsymbol{c}_t, \dots, \boldsymbol{c}_T\} &\sim \text{tRPM}(\boldsymbol{\alpha}, M) \text{ with } \alpha_t \overset{\text{iid}}{\sim} \text{Beta}(a_\alpha, b_\alpha)
\end{aligned} \tag{4}$$

where Y_{it} denotes the response variable measured by the i-th unit at time t and $\text{tRPM}(\boldsymbol{\alpha}, M)$ represents the temporal random partition model (3) parametrised by $\alpha_1, \dots, \alpha_T$ and the EPPF in (2). Our generalization of the original formulation is presented in (5), where changes and additions that differ from the original model (4) are highlighted in light blue.

$$\begin{aligned}
Y_{it}|Y_{it-1}, \boldsymbol{\mu}_t^\star, \boldsymbol{\sigma}_t^{2\star}, \boldsymbol{\eta}, \boldsymbol{c}_t &\overset{\text{ind}}{\sim} \mathcal{N}(\mu^\star_{c_{it}t} + \eta_{1i} Y_{it-1} + \boldsymbol{x}_{it}^T\boldsymbol{\beta}_t, \sigma^{2\star}_{c_{it}t}(1-\eta_{1i}^2)) \\
Y_{i1} &\overset{\text{ind}}{\sim} \mathcal{N}(\mu^\star_{c_{i1}1} + \boldsymbol{x}_{i1}^T\boldsymbol{\beta}_1, \sigma^{2\star}_{c_{i1}1}) \\
\boldsymbol{\beta}_t &\overset{\text{ind}}{\sim} \mathcal{N}_p(\boldsymbol{b}, s^2 I) \\
\xi_i = \text{Logit}(\tfrac{1}{2}(\eta_{1i}+1)) &\overset{\text{ind}}{\sim} \text{Laplace}(a, b) \\
(\mu^\star_{jt}, \sigma^{2\star}_{jt}) &\overset{\text{ind}}{\sim} \mathcal{N}(\vartheta_t, \tau_t^2) \times \text{invGamma}(a_\sigma, b_\sigma) \\
\vartheta_t|\vartheta_{t-1} &\overset{\text{ind}}{\sim} \mathcal{N}((1-\varphi_1)\varphi_0 + \varphi_1\vartheta_{t-1}, \lambda^2(1-\varphi_1^2)) \\
(\vartheta_1, \tau_t^2) &\overset{\text{iid}}{\sim} \mathcal{N}(\varphi_0, \lambda^2) \times \text{invGamma}(a_\tau, b_\tau) \\
(\varphi_0, \varphi_1, \lambda^2) &\sim \mathcal{N}(m_0, s_0^2) \times \mathcal{U}(-1, 1) \times \text{invGamma}(a_\lambda, b_\lambda) \\
\{\boldsymbol{c}_t, \dots, \boldsymbol{c}_T\} &\sim \text{tRPM}(\boldsymbol{\alpha}, M) \text{ with } \alpha_t \overset{\text{iid}}{\sim} \text{Beta}(a_\alpha, b_\alpha)
\end{aligned} \tag{5}$$

In our updated formulation, we opted to model the variance parameters using an inverse gamma distribution rather than the the uniform distribution employed in (4). This more sophisticated choice should ensure better mixing in the Markov chain, as the inverse gamma distributions recover conjugacy within the model, allowing for variance updates to be performed using the analytically exact Gibbs sampler rather than relying on the acceptance-rejection method of Metropolis algorithm. Moreover, we introduced the regression parameter $\boldsymbol{\beta}_t$, which accounts for covariates in the likelihood, with the goal of improving the accuracy in the fitted estimates of the target variable.

2.2. MCMC algorithm

We now detail the MCMC algorithm developed by Page et al. (2022), which is necessary for sampling from the posterior distributions implied by model (4). The MCMC algorithm associated to our generalized model (5) required only minor adjustments to the original structure. For the sake of clarity, we will refer to CDRPM for the original model formulation by (Page et al., 2022), and to JDRPM for our updated version. The letters C and J denote the programming languages used to implement their corresponding MCMC algorithms: C for the former, Julia for the latter.

The iterative structure of (3) suggests the use of a Gibbs sampler, where $\boldsymbol{\gamma}_t$ and ρ_t are updated sequentially. The Markovian assumption reduces computational costs as we only need to consider ρ_{t-1} and ρ_{t+1} when updating ρ_t. To perform this update, the terms $P(\rho_1)$ and $P(\rho_t|\rho_{t-1})$ of (3) need to be clarified. While $P(\rho_1)$ is defined by (2), the derivation of $P(\rho_t|\rho_{t-1})$ requires the concept of compatibility.

Definition 2.1 (compatibility)**.** Two partitions ρ_t and ρ_{t-1} are said to be *compatible* with respect to $\boldsymbol{\gamma}_t$ if ρ_t can be obtained from ρ_{t-1} by reallocating items as indicated by γ_t; that is, by moving the units i for which $\gamma_{it} = 0$.

To perform this compatibility check, it suffices to ensure that the reduced partitions from ρ_t and ρ_{t-1} are identical, where "reduced" refers to the restriction of partitions ρ_t and ρ_{t-1} to the units that cannot move. Indeed, if these fixed units are clustered in the same way, then the free movers from ρ_t can be assigned labels to match the ones assigned in ρ_{t-1}. Denoting the set of fixed units at time t as $\mathfrak{R}_t = \{i : \gamma_{it} = 1\}$, this check translates into verifying that $\rho_t^{\mathfrak{R}_t} = \rho_{t-1}^{\mathfrak{R}_t}$.

This compatibility must be verified during the update steps of parameters γ_{it} and cluster labels c_{it} to ensure that the new sampled draws remain valid and coherent across all partitions and parameters involved. To derive $P(\rho_t|\rho_{t-1})$, Page et al. (2022) proved that, under the construction outlined thus far, marginally $\rho_1, \dots, \rho_T$ are identically distributed with law derived from the EPPF used to model ρ_1. This result, along with the previously outlined compatibility analysis, enables the derivation of the complete update rules rules for γ_{it} and c_{it}.

Once the sampling scheme for γ_{it} and ρ_t is established, updating the other parameters becomes straightforward. Each parameter will either require a Gibbs step or a Metropolis step, depending on the conjugacy of the prior associated with

that parameter. Thus, we will now outline the last theoretical step, which addresses how spatial and covariate information can be included in the prior for the partitions

The core of the clustering process, that is, updating γ_{it} and ρ_t, is inherently complex as it necessitates checking for compatibility issues. The approach entails simulating the assignment of each unit i, currently belonging to cluster j, to either one of the existing clusters or to a new singleton cluster. For each scenario, we compute the probability of this assignment to occur, from which we derive weights to inform the sampling decision for the next iteration. To influence the definition of these weights by spatial and, in our JDRPM updated formulation, covariates information, the EPPF (2) has to be updated through dedicated cohesion and similarity functions, respectively.

2.3. Spatial cohesions

The incorporation of spatial information can be effectively accommodated through the EPPF in our framework, resulting in spatially informed clusters that evolve over time. To introduce this extension, let $\boldsymbol{s}_i$ denote the spatial coordinates of the i-th (we note that these coordinates do not change over time), and let $\boldsymbol{s}^\star_{jt}$ denote the subset of spatial coordinates of the units belonging to cluster S_{jt}. Then, we can express the EPPF for the t-th partition in the following product form

$$P(\rho_t|M,\mathcal{S}) \propto \prod_{j=1}^{k_t} C(S_{jt}, \boldsymbol{s}^\star_{jt}|M,\mathcal{S}) \qquad (6)$$

Compared to the original formulation of (2), where $P(\rho_t|M) \propto \prod_{j=1}^{k_t} c(S_{jt}|M)$, (6) incorporates a spatial component into the partition weights through the spatial cohesion function $C(S_{jt}, \boldsymbol{s}^\star_{jt}|M,\mathcal{S})$. The original term $c(S_{jt}|M)$ describes how units inside cluster S_{jt} are likely to be clustered together a priori, while the cohesion function $C(S_{jt}, \boldsymbol{s}^\star_{jt}|M,\mathcal{S})$, parametrised by a set of parameters $\mathcal{S}$, measures the compactness of the spatial coordinates $\boldsymbol{s}^\star_{jt}$. In the JDRPM code, we implemented the same six cohesion functions which were also implemented in CDRPM, that is, the functions described in (Page et al., 2016).

2.4. Covariates similarities

The incorporation of covariates information, a characteristic feature of our generalized model, can be similarly integrated into the EPPF definition. To introduce this extension, let $X^\star_{jt}$ denote the $p \times |S_{jt}|$ matrix that contains the covariates of the units belonging to cluster S_{jt}, i.e. $X^\star_{jt} = \{\boldsymbol{x}^\star_{it} = (x_{it1}, \dots, x_{itp})^T : i \in S_{jt}\}$. In the current implementation of JDRPM we decided to treat each covariate individually. In this way, the new term in the definition of the EPPF for $P(\rho_t)$ will be a function of the vector $\boldsymbol{x}^\star_{jtr}$ that collects the values of the r-th covariate for the units inside cluster S_{jt}, i.e. row r of matrix $X^\star_{jt}$. Then, each contribution of the covariates will be considered independently, leading to an EPPF in the form

$$P(\rho_t|M,\mathcal{S},\mathcal{C}) \propto \prod_{j=1}^{k_t} C(S_{jt}, \boldsymbol{s}^\star_{jt}|M,\mathcal{S}) \left(\prod_{r=1}^{p} g(S_{jt}, \boldsymbol{x}^\star_{jtr}|\mathcal{C}) \right) \quad (7)$$

where $g(S_{jt}, \boldsymbol{x}^\star_{jtr}|\mathcal{C})$ is the covariate similarity, parametrised by a set of parameters $\mathcal{C}$, which measures the similarity of the covariates values $\boldsymbol{s}^\star_{jt}$. This approach of treating each covariate individually is convenient as it simplifies the definition of the similarity functions and allows to seamlessly accommodate numerical and categorical covariates. In the JDRPM code, we implemented four similarity functions, which are described in (Page et al., 2018).

3. Implementation

The MCMC algorithm to compute the posterior samples of our updated model, described in Section 2.2, has been implemented in Julia (Bezanson et al., 2017).

Julia is a relatively new programming language that combines the ease and expressiveness of high-level languages with the efficiency and performance characteristics of low-level languages. This balance is primarily achieved through just-in-time compilation, using the LLVM framework, along with features as dynamic multiple dispatch and extensive code specialization against multiple runtime types.

This makes Julia a highly powerful tool for enhancing productivity and performance in implementations. As a reference, our development process comported an improvement by 74% in execution time and by 88% in the memory allocated when comparing an early-stage version to the final version of the Julia code. This performance gain was primarily achieved by a more effective memory management.

In fact, during the earlier development phases, we

observed that a significant portion of execution time was not consumed by actual calculations but, rather, by Julia's garbage collector, which had the burden of tracking all the allocated memory and reclaiming the unused one to make it available again for new computations. To solve this problem, we refactored the code to operate more in-place. This approach involved passing as arguments the variables that would be modified by a function, and applying those changes directly within the function, rather than returning values and subsequently using them to update the original variables. All this was achieved by leveraging Julia profiling tools such as `ProfileCanvas` and `BenchmarkTools`.

Another relevant performance improvement was achieved through the optimized implementation of spatial cohesions and covariates similarities. In fact, these functions are needed to derive the partitions weights in the update steps of γ_{it} and ρ_t, meaning that would be potentially executed millions of times during each fit. Consequently, refining these functions was crucial for providing fast execution times. In the cohesions functions, optimal performance was reached by using static vectors and matrices, provided by the `StaticArrays` package, rather than standard dynamic structures. For the similarity functions, we applied a SIMD paradigm (Single Instruction Multiple Data), through the `@simd` macro, which accelerated the processing of all the covariates values by working simultaneously on multiple datachunks rather than processing each element individually.

4. Analysis of the models

To evaluate the numerical performance of both algorithms, we analysed posterior samples and clusters estimates in two scenarios: first using a synthetic dataset that includes only the response variable, and secondly employing a real-world spatio-temporal dataset. For the latter, we used a dataset from the AgrImOnIA project (Fassò et al., 2023) which encompasses weekly measurements of air pollutants, together with many other environmental variables, in the Lombardy region of Italy in 2018. This dataset also provides covariates, whose effects will be examined in Sections 4.1.1 and 4.1.2.

4.1. Statistical improvements

In the first set of experiments we fitted both models, in both simulated and real-world scenarios, under equivalent conditions, i.e. identical datasets, hyperparameters, and MCMC configurations. With this setup, the models are expected to perform similarly to each other, as JDRPM is a generalization of CDRPM and should therefore manifest the same core behaviour of the original model. Indeed, the clusters estimates, as well as their temporal trend depicted in Figures 1 and 2, were remarkably similar, thus confirming the correctness of the JDRPM implementation. Moreover, the fitted values were more accurate in the JDRPM fits, as demonstrated by the mean squared errors (MSEs) improved by 20% and 8% in the simulated and real-world scenarios, respectively, compared to CDRPM fits. In absence of additional information from covariates, this improvement was likely attributable to the better mixing properties of the Markov chain in our model, thanks to the distribution choice for the variance parameters.

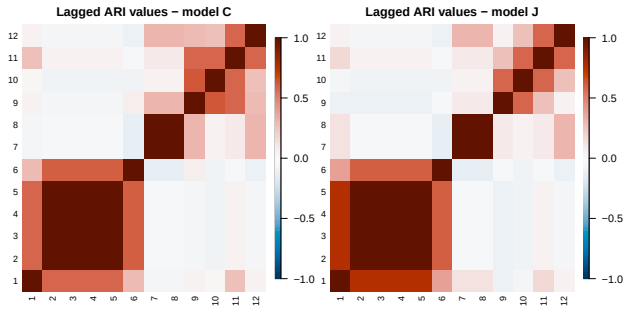

Figure 1: Lagged ARI values of CDRPM and JDRPM fits, in the simulated data scenario.

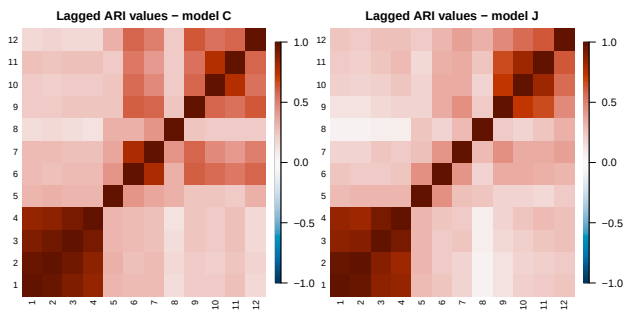

Figure 2: Lagged ARI values of CDRPM and JDRPM fits, in the real-world scenario.

4.1.1 Covariates in the likelihood

Next, we examined the effects of covariates-informed fits. Given their goal of improving the

accuracy of the fitted values, we examined two natural use-cases of covariates in the likelihood: first, fitting on spatio-temporal dataset with missing data, secondly, performing inference on a new location as in a kriging scenario.
In the first experiment, an incomplete dataset was created by randomly removing 10% of the data points. Indeed, in the real-world scenario, the inclusion of covariates in the likelihood improved all metrics, with reduced MSEs and improved fitting metrics LPML and WAIC. The results are reported in Table 1.

Table 1: Comparison of JDRPM fits, in the real-world scenario, with and without the inclusion of covariates in the likelihood, on a complete dataset and on a dataset with missing values.

	Full data	Full data + Xlk	NA data	NA data + Xlk
MSE (mean)	0.0131	**0.0112**	0.0160	**0.0127**
MSE (median)	0.0138	**0.0113**	0.0170	**0.0130**
LPML	624.91	**778.96**	502.86	**625.81**
WAIC	-1898.05	**-2029.84**	-1793.64	**-1902.74**
Execution time	**48m**	56m	**43m**	58m

We note that the standard fits without covariates in the likelihood did not exhibit a significant drop in performance with respect to their full dataset counterparts; meaning that even if no additional information is available, the update rule for the missing data Y_{it} can still be effective. For instance, in the simulated scenario (thus without covariates information) with missing data, all the true values of the missing points lied within the 95% credible intervals of the corresponding fitted estimates.
Regarding the kriging analysis, we simulated the new units added at new locations by removing all data entries from three randomly selected units within the spatio-temporal dataset. In this experiment, we aimed to evaluate the JDRPM's accuracy in predicting the behaviour of units for which sensors may be absent or inactive. We expected that the estimation accuracy would improve as model complexity increases, and truly the best performances, in terms of MSE, LPML, and WAIC, occurred for JDRPM fits including covariates in the likelihood; proving again the effectiveness of this addition to the model formulation. However, some inaccuracies arose with a unit which was often clustered as a singleton, exhibited extreme values in some of the included covariates, and was quite separate from other units; factors which possibly hindered a more precise estimation of her response variable values.

4.1.2 Covariates in the prior

Regarding covariates in the prior, which have the goal of producing more informative and accurate results, we performed several JDRPM fits, on the PM_{10} target variable, including in the prior three relevant covariates: wind speed, precipitation levels, and boundary layer height. Again, this spatially and covariates-informed fit displayed a significant improvement in all metrics, as reported in Table 2.

Table 2: Comparison between CDRPM and JDRPM fits and their associated algorithms, in the real-world scenario, with and without covariates in the prior.

	CDRPM	JDRPM	JDRPM + Xcl
MSE (mean)	0.0142	0.0131	**0.0126**
MSE (median)	0.0149	0.0138	**0.0135**
LPML	694.81	624.91	**677.71**
WAIC	-1768.42	-1898.05	**-1969.76**
Execution time	1h38m	**48m**	1h20m

Although incorporating multiple covariates introduces the possibility of divergent "clustering suggestions," their contributions can still be effectively assessed through cluster-specific boxplots of the target variable PM_{10} along with the three covariates included in the analysis. Figure 3 demonstrates how the spatially and covariates-informed JDRPM fit in the prior captured a more refined and coherent clustering structure, providing a clear separation in the target variable PM_{10}, as well as among the covariates, particularly regarding wind speed and total precipitation. For instance, the dark red cluster is well characterized by the highest levels of PM_{10} and the second-last levels of wind speed and total precipitations; factors which probably hindered the pollutant dispersion.
The contribution of covariates is particularly evident comparing the CDRPM spatially-informed fit to the JDRPM fit including space and covariates in the prior. Such comparison, exemplified by Figures 4 and 5, shows how the JDRPM fit managed to better identify clusters which were instead sparser, less specialized in the CDRPM fit.

4.2. Computational improvements

Finally, to rigorously assess whether the objective of faster execution times was achieved, we designed a series of experiments to compare the two models and their corresponding implementations.

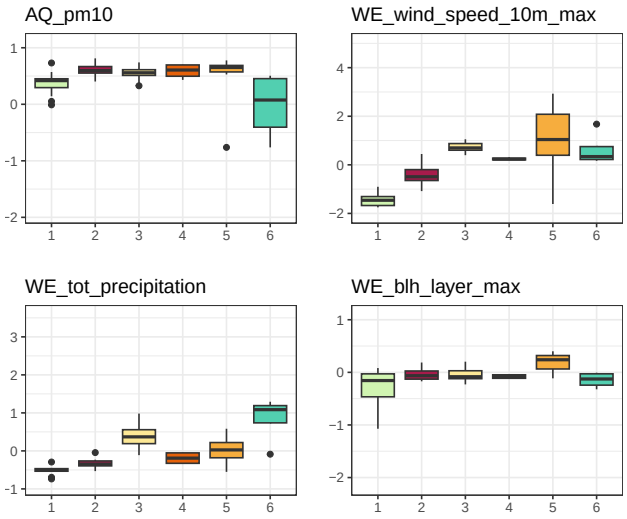

Figure 3: Example of the clusters estimates produced by JDRPM fit, in the real-world scenario, with covariates in the prior. The boxplots refer to the target variable `AQ_pm10` along with the three covariates included in the fit.

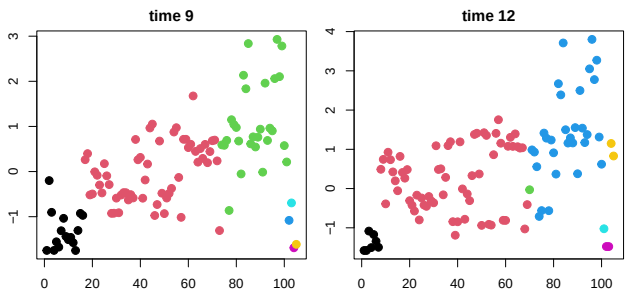

Figure 4: Example of clusters estimates with respect to the wind speed covariate, in the CDRPM spatially-informed fit.

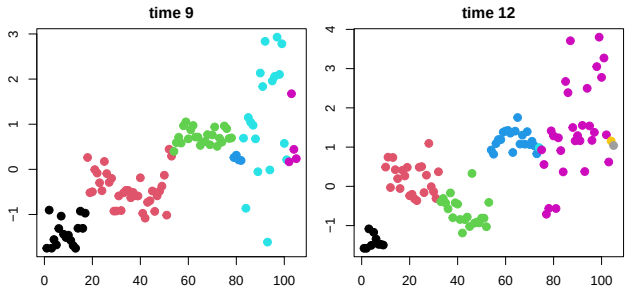

Figure 5: Example of clusters estimates with respect to the wind speed covariate, in the JDRPM spatially-informed fit with covariates in the prior.

The different information levels affect the model complexity and therefore reflect into the computational load, however also the size of the testing dataset plays a significant role. To systematically explore these factors, we conducted the experiments across incremental levels of information—from including only the target variable, then space, covariates in the likelihood, and finally covariates in the prior—as well as across a "mesh" of dataset sizes, with both the number of units n and the time horizons T ranging through the set $\{10, 50, 100, 250\}$. The results on extreme-sized dataset, e.g. with n or T being 250, may reflect hardware limitations of the machine where fits were conducted, i.e. my simple laptop, rather than actual model performance, therefore more attention should be devoted to the more accurate intermediate-sized experiments, which reliably demonstrate the JDRPM's improved performance. For example, Figure 6 shows how, in the case of spatially-informed fits, JDRPM can perform up to 2x faster compared to CDRPM.

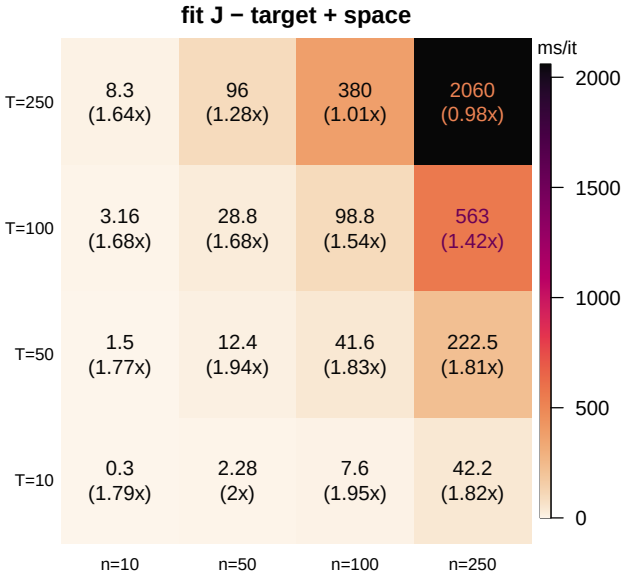

Figure 6: Execution times, in milliseconds per iteration, when fitting JDRPM on a spatio-temporal dataset. In brackets are reported the speedups relative to the CDRPM timings, where higher values indicate better performance.

Moreover, even JDRPM fits including covariates information demonstrate superior performance compared to simple CDRPM fits, despite the additional complexity and subsequent increase in computational load. Figure 7 shows how on a medium sized dataset, composed by $n = 50$ units, fully-informed JDRPM fits can perform faster than spatially-informed CDRPM fits. In addition, JDRPM fits with spatial information can peform faster than CDRPM fits which only include the target variable.
In Figure 7, the number of covariates in the likelihood and prior levels was fixed to $p = 5$. In fact, $p = 5$ covariates seems to be the equilibrium point between JDRPM and CDRPM performance. More precisely, other experiments were conducted, with a varying number of covariates in

likelihood and prior levels, and reported how in the same time that CDRPM performs a spatially-informed fit, JDRPM can perform a spatially-informed fit with up to five covariates in the prior and with an almost indifferent number of covariates in the likelihood, as they do not add particular computational load. These results strongly validate the performance speedup reached with our new implementation.

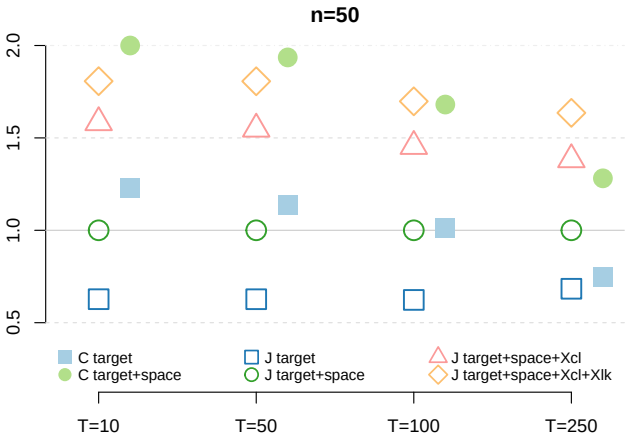

Figure 7: Visual representation of the performance of all the fits, for the $n = 50$ case. Specifically, the execution time per iteration of JDRPM spatially-informed fit has been taken as a reference, to which then all other fits have been compared to derive their speedup (or slowdown) factor. Points above the reference line indicate slower fits, while points below denote faster fits.

5. Conclusions

In conclusion, we can state that JDRPM represents a significant improvement over the original CDRPM. From a theoretical perspective, JDRPM retains the foundational structure of its predecessor, while also introducing covariates in the prior and likelihood levels, as well as providing more flexibility through the handling of missing data in the target variable. From a computational perspective, we successfully reduced execution times by up to 50% compared to the original implementation. Moreover, the innovative choice of the Julia language should accelerate the implementation of future modifications or and updates.

Possible drawbacks derive from the increased complexity of our generalized model. The more sophisticated distribution choice of inverse gamma rather than uniform for the variance parameters, as well as the selection of parameters for spatial cohesions and covariates similarities, could require a more careful reasoning and tuning process when performing the fits. In this way, future development paths could try to reduce this complexity, e.g. through heuristics on how to assign the various parameters and hyperparameters given the dataset at hand. Moreover, the Julia powerful ecosystem could be leveraged to provide real-time feedbacks about the sampling process, e.g. through trace plots, computation logs of the various update steps, etc.

In short, while the JDRPM's complexity may present certain challenges, it also lays the groundwork for significant methodological advancements and practical enhancements. Our developments are expected to improve the model's applicability and accuracy, for more effective research outcomes.

References

Bezanson, Jeff et al. (2017). "Julia: A fresh approach to numerical computing". In: *SIAM Review* 59.1, pp. 65–98. DOI: 10.1137/141000671 (cit. on p. 4).

Fassò, A. et al. (2023). *AgrImOnIA: Open Access dataset correlating livestock and air quality in the Lombardy region, Italy (3.0.0)*. DOI: https://doi.org/10.5281/zenodo.7956006 (cit. on p. 5).

Page, Garritt L. and Fernando A. Quintana (Sept. 2018). "Calibrating covariate informed product partition models". In: *Statistics and Computing* 28, pp. 1–23. DOI: 10.1007/s11222-017-9777-z (cit. on p. 4).

– (2016). "Spatial Product Partition Models". In: *Bayesian Analysis* 11.1, pp. 265–298. DOI: 10.1214/15-BA971 (cit. on p. 4).

Page, Garritt L., Fernando A. Quintana, and David B. Dahl (2022). "Dependent Modeling of Temporal Sequences of Random Partitions". In: *Journal of Computational and Graphical Statistics* 31.2, pp. 614–627. DOI: 10.1080/10618600.2021.1987255 (cit. on pp. 1–3).