

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

Candidate: Federico Angelo Mor

Advisor: Prof. Alessandra Guglielmi  
Coadvisor: Prof. Alessandro Carminati



**POLITECNICO**  
MILANO 1863

December 11, 2024

# What is the thesis about?

The Dependent Random Partition Model from (Page et al., 2022) is a Bayesian spatio-temporal clustering model which directly models the dependencies in the sequence of clusters over time.

Currently, the model



- produces up to spatially-informed clusters  
→ I additionally introduced covariates information
- only accepts complete datasets  
→ I made it work with missing values in the target variable
- has quite slow execution times (especially on large datasets)  
→ I developed a brand-new and more efficient implementation

inserire o dire a voce il cambio da C a Julia

## ① Description of the problem

What is the DRPM

How did we improve it

## ② Implementation and optimizations

## ③ Analysis of the models

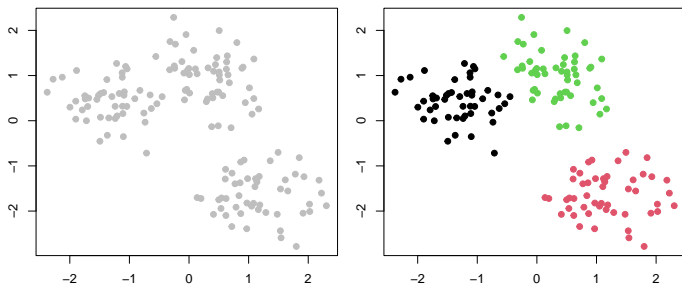
## ④ Conclusion

## ⑤ Appendix (additional material)

# Clustering

The Dependent Random Partition Model from (Page et al., 2022) is a Bayesian spatio-temporal **clustering** model which directly models the temporal dependencies in the sequence of clusters over time.

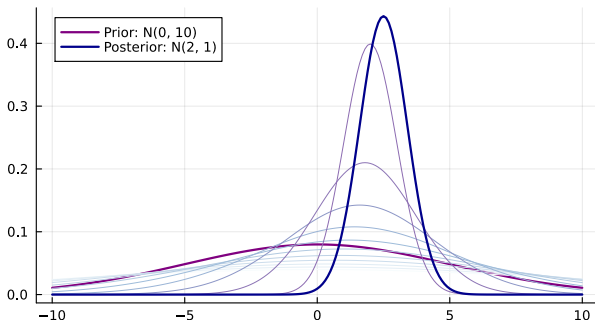
Clustering is a fundamental technique of unsupervised learning where a set of data points has to be divided into homogeneous groups of units which exhibit a similar behaviour.



# Why going Bayesian?

The Dependent Random Partition Model from (Page et al., 2022) is a **Bayesian** spatio-temporal clustering **model** which directly models the temporal dependencies in the sequence of clusters over time.

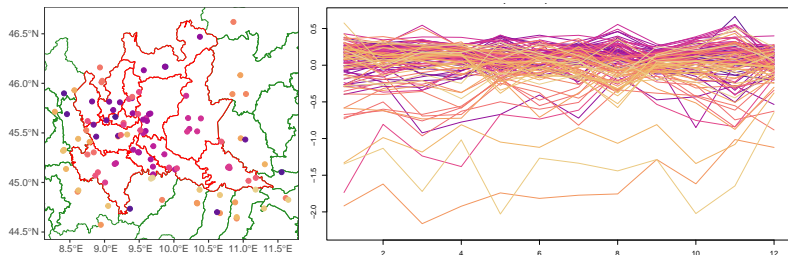
Bayesian models incorporate prior information on the model parameters and allow to assess uncertainty when performing inference on the results.



## A bit of (spatio-temporal) context

The Dependent Random Partition Model from (Page et al., 2022) is a Bayesian **spatio-temporal** clustering model which directly models the temporal dependencies in the sequence of clusters over time.

In spatio-temporal datasets, observations are collected over time and across various spatial locations. So we will have  $n$  units that have to be clustered at all time instants  $t = 1, \dots, T$ .

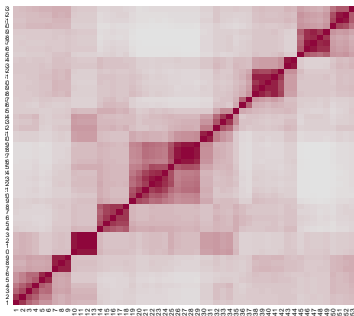


# Why should we care about temporal dependencies?

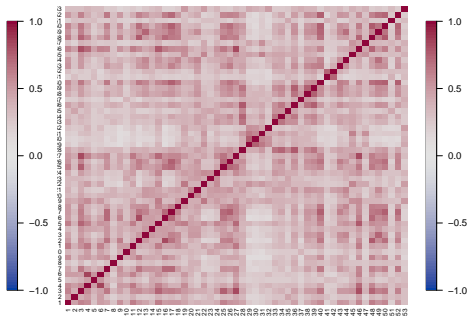
The **Dependent** Random Partition Model from (Page et al., 2022) is a Bayesian spatio-temporal clustering model which **directly models the temporal dependencies in the sequence of clusters over time**.

This allows to derive a more gentle and interpretable evolution of clusters.

Lagged ARI values – DRPM



Lagged ARI values – sPPM

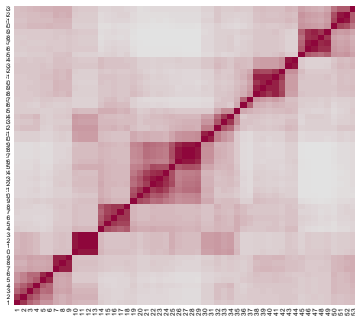


# Why should we care about temporal dependencies?

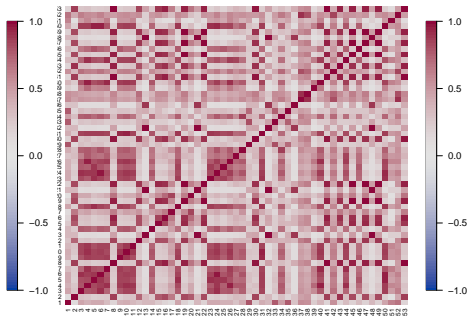
The **Dependent** Random Partition Model from (Page et al., 2022) is a Bayesian spatio-temporal clustering model which **directly models the temporal dependencies in the sequence of clusters over time**.

This allows to derive a more gentle and interpretable evolution of clusters.

Lagged ARI values – DRPM



Lagged ARI values – Curve PPMx





# Modelling the temporal dependence

Introducing temporal dependence in a collection of partitions requires the formulation of a joint probability model for  $(\rho_1, \dots, \rho_T)$ . Page et al. (2022) modelled this temporal connection as a first-order Markovian structure, where the conditional distribution of  $\rho_t$  given all the predecessors  $\rho_{t-1}, \rho_{t-2}, \dots, \rho_1$  actually depends only on  $\rho_{t-1}$ , leading to

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

Here,  $P(\rho_1)$  is an exchangeable partition probability function (EPPF), which describes how the  $n$  experimental units at time period 1 are grouped into  $k_1$  distinct clusters. Page et al. (2022) chose this EPPF to be  $P(\rho_1) \propto \prod_{j=1}^{k_1} M \cdot (|S_{j1}| - 1)!$ .

dire che è quella indotta dal

# Modelling the temporal dependence

To characterize the other terms  $P(\rho_t|\rho_{t-1})$  in (9), i.e. to explicitly model how  $\rho_{t-1}$  influences  $\rho_t$ , the following auxiliary variables need to be introduced to. For all units  $i = 1, \dots, n$  we define

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

These parameters model the similarity between  $\rho_{t-1}$  and  $\rho_t$ :

- if  $\rho_{t-1}$  and  $\rho_t$  are highly dependent, their cluster configurations will change minimally  $\implies$  the majority of  $\gamma_{it}$  will be 1
- if  $\rho_{t-1}$  and  $\rho_t$  exhibit low dependence, their cluster configurations will change significantly  $\implies$  the majority of  $\gamma_{it}$  will be 0

# Modelling the temporal dependence

Page et al. (2022) assumed  $\gamma_{it} \stackrel{\text{ind}}{\sim} \text{Ber}(\alpha_t)$ , where  $\alpha_t \in [0, 1]$  serves as a temporal dependence parameter, spanning from perfect temporal correlation ( $\alpha_t = 0$ ) to full independence ( $\alpha_t = 1$ ).

For clarity, the vector  $\gamma_t = (\gamma_{1t}, \dots, \gamma_{nt})$  is introduced, so that the  $T$  pairs of parameters  $(\gamma_j, \rho_j)$  are explicitly reported in the augmented formulation of the joint model (9), which becomes

$$P(\gamma_1, \rho_1, \dots, \gamma_T, \rho_T) = P(\rho_T | \gamma_T, \rho_{T-1}) P(\gamma_T) \cdots$$

$$\times P(\rho_2 | \gamma_2, \rho_1) P(\gamma_2) P(\rho_1)$$

$$\times P(\text{ecc})$$

Once the model for the partition is specified, there is considerable flexibility in how to define the remainder of the Bayesian model.

# The DRPM

DRPM formulation according to Page et al. (2022) (henceforth, CDRPM, with C as the C language used for the model's implementation).

$$Y_{it}|Y_{it-1}, \boldsymbol{\mu}_t^*, \boldsymbol{\sigma}_t^{2*}, \boldsymbol{\eta}, \mathbf{c}_t \stackrel{\text{ind}}{\sim} \mathcal{N}(\mu_{c_{it}t}^* + \eta_{1i} Y_{it-1}, \sigma_{c_{it}t}^{2*}(1 - \eta_{1i}^2))$$

$$Y_{i1} \stackrel{\text{ind}}{\sim} \mathcal{N}(\mu_{c_{i1}1}^*, \sigma_{c_{i1}1}^{2*})$$

$$\xi_i = \text{Logit}(\frac{1}{2}(\eta_{1i} + 1)) \stackrel{\text{ind}}{\sim} \text{Laplace}(a, b)$$

$$(\mu_{jt}^*, \sigma_{jt}^*) \stackrel{\text{ind}}{\sim} \mathcal{N}(\vartheta_t, \tau_t^2) \times \mathcal{U}(0, A_\sigma)$$

$$\vartheta_t|\vartheta_{t-1} \stackrel{\text{ind}}{\sim} \mathcal{N}((1 - \varphi_1)\varphi_0 + \varphi_1\vartheta_{t-1}, \lambda^2(1 - \varphi_1^2))$$

$$(\vartheta_1, \tau_t) \stackrel{\text{iid}}{\sim} \mathcal{N}(\varphi_0, \lambda^2) \times \mathcal{U}(0, A_\tau)$$

$$(\varphi_0, \varphi_1, \lambda) \sim \mathcal{N}(m_0, s_0^2) \times \mathcal{U}(-1, 1) \times \mathcal{U}(0, A_\lambda)$$

$$\{\mathbf{c}_t, \dots, \mathbf{c}_T\} \sim \text{tRPM}(\boldsymbol{\alpha}, M) \text{ with } \alpha_t \stackrel{\text{iid}}{\sim} \text{Beta}(a_\alpha, b_\alpha)$$

# The DRPM

To facilitate the propagation of temporal dependence throughout the model, an autoregressive AR(1) component is incorporated at both the data level and the cluster-specific parameter level.

$$Y_{it} | Y_{it-1}, \boldsymbol{\mu}_t^*, \boldsymbol{\sigma}_t^{2*}, \boldsymbol{\eta}, \mathbf{c}_t \stackrel{\text{ind}}{\sim} \mathcal{N}(\mu_{c_{it}}^* + \boxed{\eta_{1i} Y_{it-1}}, \sigma_{c_{it}}^{2*} (1 - \eta_{1i}^2))$$

$$Y_{i1} \stackrel{\text{ind}}{\sim} \mathcal{N}(\mu_{c_{i1}}^*, \sigma_{c_{i1}}^{2*})$$

$$\boxed{\xi_i = \text{Logit}(\frac{1}{2}(\eta_{1i} + 1))} \stackrel{\text{ind}}{\sim} \text{Laplace}(a, b)$$

$$(\mu_{jt}^*, \sigma_{jt}^{2*}) \stackrel{\text{ind}}{\sim} \mathcal{N}(\vartheta_t, \tau_t^2) \times \mathcal{U}(0, A_\sigma)$$

$$\vartheta_t | \vartheta_{t-1} \stackrel{\text{ind}}{\sim} \mathcal{N}((1 - \varphi_1)\varphi_0 + \boxed{\varphi_1 \vartheta_{t-1}}, \lambda^2 (1 - \varphi_1^2))$$

$$(\vartheta_1, \tau_t) \stackrel{\text{iid}}{\sim} \mathcal{N}(\varphi_0, \lambda^2) \times \mathcal{U}(0, A_\tau)$$

$$(\varphi_0, \boxed{\varphi_1}, \lambda) \sim \mathcal{N}(m_0, s_0^2) \times \mathcal{U}(-1, 1) \times \mathcal{U}(0, A_\lambda)$$

$$\{\mathbf{c}_t, \dots, \mathbf{c}_T\} \sim \text{tRPM}(\alpha, M) \text{ with } \alpha_t \stackrel{\text{iid}}{\sim} \text{Beta}(a_\alpha, b_\alpha)$$

# The DRPM

The  $\vartheta_t$  parameter serves as a temporal anchor for the cluster-specific means  $\mu_{jt}^*$ , ensuring that these means are not completely independent over time but instead exhibit a regular and interpretable progression.

$$Y_{it}|Y_{it-1}, \mu_t^*, \sigma_t^{2*}, \eta, \mathbf{c}_t \stackrel{\text{ind}}{\sim} \mathcal{N}(\mu_{c_{it}t}^* + \eta_{1i} Y_{it-1}, \sigma_{c_{it}t}^{2*}(1 - \eta_{1i}^2))$$

$$Y_{i1} \stackrel{\text{ind}}{\sim} \mathcal{N}(\mu_{c_{i1}1}^*, \sigma_{c_{i1}1}^{2*})$$

$$\xi_i = \text{Logit}(\frac{1}{2}(\eta_{1i} + 1)) \stackrel{\text{ind}}{\sim} \text{Laplace}(a, b)$$

$$(\mu_{jt}^*, \sigma_{jt}^{2*}) \stackrel{\text{ind}}{\sim} \mathcal{N}(\boxed{\vartheta_t}, \tau_t^2) \times \mathcal{U}(0, A_\sigma)$$

$$\boxed{\vartheta_t | \vartheta_{t-1}} \stackrel{\text{ind}}{\sim} \mathcal{N}((1 - \varphi_1)\varphi_0 + \varphi_1 \boxed{\vartheta_{t-1}}, \lambda^2(1 - \varphi_1^2))$$

$$(\boxed{\vartheta_1}, \tau_t) \stackrel{\text{iid}}{\sim} \mathcal{N}(\varphi_0, \lambda^2) \times \mathcal{U}(0, A_\tau)$$

$$(\varphi_0, \varphi_1, \lambda) \sim \mathcal{N}(m_0, s_0^2) \times \mathcal{U}(-1, 1) \times \mathcal{U}(0, A_\lambda)$$

$$\{\mathbf{c}_t, \dots, \mathbf{c}_T\} \sim \text{tRPM}(\alpha, M) \text{ with } \alpha_t \stackrel{\text{iid}}{\sim} \text{Beta}(a_\alpha, b_\alpha)$$

# Our generalized model

DRPM formulation according to our generalization (henceforth, JDRPM, with J as the Julia language used for the model's implementation).

$$Y_{it}|Y_{it-1}, \boldsymbol{\mu}_t^*, \boldsymbol{\sigma}_t^{2*}, \boldsymbol{\eta}, \mathbf{c}_t \stackrel{\text{ind}}{\sim} \mathcal{N}(\mu_{c_{it}t}^* + \eta_{1i}Y_{it-1} + \mathbf{x}_{it}^T \boldsymbol{\beta}_t, \sigma_{c_{it}t}^{2*}(1 - \eta_{1i}^2))$$

$$Y_{i1} \stackrel{\text{ind}}{\sim} \mathcal{N}(\mu_{c_{i1}1}^* + \mathbf{x}_{i1}^T \boldsymbol{\beta}_1, \sigma_{c_{i1}1}^{2*})$$

$$\boldsymbol{\beta}_t \stackrel{\text{ind}}{\sim} \mathcal{N}_p(\mathbf{b}, s^2 I)$$

$$\xi_i = \text{Logit}(\frac{1}{2}(\eta_{1i} + 1)) \stackrel{\text{ind}}{\sim} \text{Laplace}(a, b)$$

$$(\mu_{jt}^*, \sigma_{jt}^{2*}) \stackrel{\text{ind}}{\sim} \mathcal{N}(\vartheta_t, \tau_t^2) \times \text{invGamma}(a_\sigma, b_\sigma)$$

$$\vartheta_t | \vartheta_{t-1} \stackrel{\text{ind}}{\sim} \mathcal{N}((1 - \varphi_1)\varphi_0 + \varphi_1\vartheta_{t-1}, \lambda^2(1 - \varphi_1^2))$$

$$(\vartheta_1, \tau_1^2) \stackrel{\text{iid}}{\sim} \mathcal{N}(\varphi_0, \lambda^2) \times \text{invGamma}(a_\tau, b_\tau)$$

$$(\varphi_0, \varphi_1, \lambda^2) \sim \mathcal{N}(m_0, s_0^2) \times \mathcal{U}(-1, 1) \times \text{invGamma}(a_\lambda, b_\lambda)$$

$$\{\mathbf{c}_t, \dots, \mathbf{c}_T\} \sim \text{tRPM}(\boldsymbol{\alpha}, M) \text{ with } \alpha_t \stackrel{\text{iid}}{\sim} \text{Beta}(a_\alpha, b_\alpha)$$

# Updated formulation

- 1) We inserted a regression term in the likelihood and changed the prior distributions of the variance parameters

$$Y_{it}|Y_{it-1}, \mu_t^*, \sigma_t^{2*}, \eta, \mathbf{c}_t \stackrel{\text{ind}}{\sim} \mathcal{N}(\mu_{c_{it}t}^* + \eta_{1i} Y_{it-1} + \mathbf{x}_{it}^T \boldsymbol{\beta}_t, \sigma_{c_{it}t}^{2*} (1 - \eta_{1i}^2))$$

$$Y_{i1} \stackrel{\text{ind}}{\sim} \mathcal{N}(\mu_{c_{i1}1}^* + \mathbf{x}_{i1}^T \boldsymbol{\beta}_1, \sigma_{c_{i1}1}^{2*})$$

$$\boldsymbol{\beta}_t \stackrel{\text{ind}}{\sim} \mathcal{N}_p(\mathbf{b}, s^2 I)$$

$$\xi_i = \text{Logit}(\frac{1}{2}(\eta_{1i} + 1)) \stackrel{\text{ind}}{\sim} \text{Laplace}(a, b)$$

$$(\mu_{jt}^*, \sigma_{jt}^{2*}) \stackrel{\text{ind}}{\sim} \mathcal{N}(\vartheta_t, \tau_t^2) \times \text{invGamma}(a_\sigma, b_\sigma)$$

$$\vartheta_t | \vartheta_{t-1} \stackrel{\text{ind}}{\sim} \mathcal{N}((1 - \varphi_1)\varphi_0 + \varphi_1 \vartheta_{t-1}, \lambda^2 (1 - \varphi_1^2))$$

$$(\vartheta_1, \tau_t^2) \stackrel{\text{iid}}{\sim} \mathcal{N}(\varphi_0, \lambda^2) \times \text{invGamma}(a_\tau, b_\tau)$$

$$(\varphi_0, \varphi_1, \lambda^2) \sim \mathcal{N}(m_0, s_0^2) \times \mathcal{U}(-1, 1) \times \text{invGamma}(a_\lambda, b_\lambda)$$

$$\{\mathbf{c}_t, \dots, \mathbf{c}_T\} \sim \text{tRPM}(\boldsymbol{\alpha}, M) \text{ with } \alpha_t \stackrel{\text{iid}}{\sim} \text{Beta}(a_\alpha, b_\alpha)$$



# Updated formulation

parte nuova del modello creata da noi

The regressor term  $\beta_t$  improves the quality of the fitted estimates for the target variable and for the samples of the model parameters.

Prior:  $\beta_t \sim \mathcal{N}_p(\mathbf{b}, s^2 I)$

Update rule:

for  $t = 1$ :  $f(\beta_t | -) \propto$  kernel of a  $\mathcal{N}\text{Canon}(\mathbf{h}_{(\text{post})}, J_{(\text{post})})$  with

$$\mathbf{h}_{(\text{post})} = \left( \frac{\mathbf{b}}{s^2} + \sum_{i=1}^n \frac{(Y_{it} - \mu_{c_{it}}^*) \mathbf{x}_{it}}{\sigma_{c_{it}}^{2*}} \right) \quad J_{(\text{post})} = \left( \frac{1}{s^2} I + \sum_{i=1}^n \frac{\mathbf{x}_{it} \mathbf{x}_{it}^T}{\sigma_{c_{it}}^{2*}} \right)$$

for  $t > 1$ :  $f(\beta_t | -) \propto$  kernel of a  $\mathcal{N}\text{Canon}(\mathbf{h}_{(\text{post})}, J_{(\text{post})})$  with

$$\mathbf{h}_{(\text{post})} = \left( \frac{\mathbf{b}}{s^2} + \sum_{i=1}^n \frac{(Y_{it} - \mu_{c_{it}}^* - \eta_{1i} Y_{it-1}) \mathbf{x}_{it}}{\sigma_{c_{it}}^{2*}} \right) \quad J_{(\text{post})} = \left( \frac{1}{s^2} I + \sum_{i=1}^n \frac{\mathbf{x}_{it} \mathbf{x}_{it}^T}{\sigma_{c_{it}}^{2*}} \right)$$

where  $\mathcal{N}\text{Canon}(\mathbf{h}, J)$  is the canonical formulation of the  $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$ , with  $\mathbf{h} = \Sigma^{-1} \boldsymbol{\mu}$  and  $J = \Sigma^{-1}$ .

# Updated formulation

The choice of the inverse gamma distribution recovers conjugacy within the model, leading to better mixing properties for the MCMC.

Prior:  $\sigma_{jt}^{2*} \sim \text{invGamma}(a_\sigma, b_\sigma)$

Update rule:

for  $t = 1$ :  $f(\sigma_{jt}^{2*} | -) \propto \text{kernel of a invGamma}(a_{\sigma(\text{post})}, b_{\sigma(\text{post})})$  with

$$a_{\tau(\text{post})} = a_\sigma + \frac{|S_{jt}|}{2} \quad b_{\tau(\text{post})} = b_\sigma + \frac{1}{2} \sum_{i \in S_{jt}} (Y_{it} - \mu_{jt}^* - \mathbf{x}_{it}^T \beta_t)^2$$

for  $t > 1$ :  $f(\sigma_{jt}^{2*} | -) \propto \text{kernel of a invGamma}(a_{\sigma(\text{post})}, b_{\sigma(\text{post})})$  with

$$a_{\tau(\text{post})} = a_\sigma + \frac{|S_{jt}|}{2} \quad b_{\tau(\text{post})} = b_\sigma + \frac{1}{2} \sum_{i \in S_{jt}} (Y_{it} - \mu_{jt}^* - \eta_{1i} Y_{it-1} - \mathbf{x}_{it}^T \beta_t)^2$$

Similar derivations apply to  $\tau_t^2$  and  $\lambda^2$ .

## Additional information level

(2) We introduced covariates information inside the prior for the partition.

To describe how we performed this inclusion, we recall how Page et al. (2022) included spatial information in the prior for the partition.

# Spatial information

In the original formulation we have  $P(\rho_t|M) \propto \prod_{j=1}^{k_t} c(S_{jt}|M)$ , where  $c(S_{jt}|M)$  described how units inside cluster  $S_{jt}$  are likely to be clustered together a priori.

Let  $\mathbf{s}_i$  denote the spatial coordinates of the  $i$ -th unit (noting that these coordinates do not change over time), and  $\mathbf{s}_{jt}^*$  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 form

$$P(\rho_t|M, \mathcal{S}) \propto \prod_{j=1}^{k_t} C(S_{jt}, \mathbf{s}_{jt}^*|M, \mathcal{S})$$

where the **cohesion function**  $C(S_{jt}, \mathbf{s}_{jt}^*|M, \mathcal{S})$ , parametrised by a set of parameters  $\mathcal{S}$ , measures the compactness of the spatial coordinates  $\mathbf{s}_{jt}^*$ . The implemented cohesion functions can be found in (Page et al., 2016).

# Covariates information

- our  
contribution

Let  $X_{jt}^*$  denote the  $p \times |S_{jt}|$  matrix that contains the covariates of the units belonging to cluster  $S_{jt}$ , i.e.  $X_{jt}^* = \{\mathbf{x}_{it}^* = (x_{it1}, \dots, x_{itp})^T : i \in S_{jt}\}$ . In the current implementation of JDRPM we chose to treat each covariate individually, leading to an EPPF in the form

$$P(\rho_t | M, S, \mathcal{C}) \propto \prod_{j=1}^{k_t} C(S_{jt}, \mathbf{x}_{jt}^* | M, S) \left( \prod_{r=1}^p g(S_{jt}, \mathbf{x}_{jtr}^* | \mathcal{C}) \right)$$

where the **similarity function**  $g(S_{jt}, \mathbf{x}_{jtr}^* | \mathcal{C})$ , parametrised by a set of parameters  $\mathcal{C}$ , measures the similarity of the  $r$ -th covariate values  $\mathbf{x}_{jtr}^*$ . The implemented similarity functions can be found in (Page et al., 2018).

spazio visivo

we implemented the similarity functions from the paper

see (page 16 table 1 and 2)

# Missing data

(3) We let the model accept missing data in the target variable through the derivation of an update rule for the missing  $Y_{it}$ 's.

for  $t = 1$ :  $f(Y_{it}|-) \propto \mathcal{N}(\mu_{Y_{it}(\text{post})}, \sigma_{Y_{it}(\text{post})}^2)$  with

$$\sigma_{Y_{it}(\text{post})}^2 = 1 / \left( \frac{1}{\sigma_{c_{it}t}^{2*}} + \frac{\eta_{1i}^2}{2\sigma_{c_{it+1}t+1}^{2*}(1 - \eta_{1i}^2)} \right)$$

$$\mu_{Y_{it}(\text{post})} = \sigma_{Y_{it}(\text{post})}^2 \left( \frac{\mu_{c_{it}t}^* + \mathbf{x}_{it}^T \beta_t}{\sigma_{c_{it}t}^{2*}} + \frac{\eta_{1i}(Y_{it+1} - \mu_{c_{it+1}t+1}^* - \mathbf{x}_{it+1}^T \beta_{t+1})}{\sigma_{c_{it+1}t+1}^{2*}(1 - \eta_{1i}^2)} \right)$$

for  $1 < t < T$ :  $f(Y_{it}|-) \propto \mathcal{N}(\mu_{Y_{it}(\text{post})}, \sigma_{Y_{it}(\text{post})}^2)$  with

$$\sigma_{Y_{it}(\text{post})}^2 = (1 - \eta_{1i}^2) / \left( \frac{1}{\sigma_{c_{it}t}^{2*}} + \frac{\eta_{1i}^2}{\sigma_{c_{it+1}t+1}^{2*}} \right)$$

$$\mu_{Y_{it}(\text{post})} = \sigma_{Y_{it}(\text{post})}^2 \left( \frac{\mu_{c_{it}t}^* + \eta_{1i} Y_{it-1} + \mathbf{x}_{it}^T \beta_t}{\sigma_{c_{it}t}^{2*}(1 - \eta_{1i}^2)} + \frac{\eta_{1i}(Y_{it+1} - \mu_{c_{it+1}t+1}^* - \mathbf{x}_{it+1}^T \beta_{t+1})}{\sigma_{c_{it+1}t+1}^{2*}(1 - \eta_{1i}^2)} \right)$$

for  $t = T$ :  $f(Y_{it}|-)$  is just the likelihood of  $Y_{it}$

# New implementation

(4) We developed a brand-new and more efficient implementation for the updated MCMC algorithm which we now describe in the following section.

- 1 Description of the problem
- 2 Implementation and optimizations
  - Language choice
  - Optimizations
- 3 Analysis of the models
- 4 Conclusion
- 5 Appendix (additional material)



# Implementation and optimizations

The MCMC algorithm to compute the posterior samples of our updated model has been implemented in Julia (Bezanson et al., 2017), which has several benefits compared to the C choice of Page et al. (2022).

MCMC: in julia  
(reference)  
why?



# Why Julia?

anche qui asciugare, togliere Julia

- Julia combines the ease and expressiveness of high-level languages (e.g. R, python, matlab) with the efficiency and performance characteristics of low-level languages (e.g. C, C++, Fortran)
- Julia code can be tested interactively, as with interpreted languages. . .
- . . . but performance is guaranteed through (just-in-time) compilation
- Linear algebra computations are optimized through BLAS and LAPACK libraries
- There is a vast collection of optimized and complete scientific packages (e.g. Statistics, Distributions (Besançon et al., 2021), MCMCChains (Ge et al., 2018), etc.)

togliere la parentesi prima di eg

# Spoiler alert

togliere la slide, tanto ne parleremo più avanti

Using Julia, we obtained **improved computational performance** compared to the original C implementation of (Page et al., 2022).

Our experiments proved that within the same amount of time in which CDRPM executes a spatially-informed fit, JDRPM can perform a spatially informed fit with up to five covariates in the prior and five covariates in the likelihood.

This performance gain was achieved through several optimizations steps which we now briefly highlight.

# General optimizations

spostare tutta questa parte in fondo in appendice come materiale aggiuntivo

- preallocation of all modelling and working variables
- ensuring type stability of the code, using the package Cthulhu

```
indexes::Vector{Int64} = findall((gamma_iter::Matrix{Bool}[::Core.Const(...,t
+1] .== 1)
Si_comp1 = @view rho_tmp::Vector{Int64}[indexes::Vector{Int64}]
Si_comp2 = @view Si_iter::Matrix{Int64}[indexes::Vector{Int64},(t::Int64+1)
::In...]
rho_comp::Int64 = compatibility(Si_comp1::SubArray{Int64, 1, Vector{Int64}},
..., Si_comp2)

if (rho_comp::Int64 != 1)::Bool
    ph::Vector{Float64}[k::Int64] = log(0)::Core.Const(-Inf) # assignment to
    a new cluster is not compatible
else
    # sample new params for this new cluster
    muh_draw::Float64 = rand(Normal(theta_iter::Vector{Float64}[t::Int64]
::Float64, sqrt(tau2_iter::...[t]))
    sig2h_draw::Float64 = rand(InverseGamma(sig2h_priors::Vector{Float64}[1]
::Float64, sig2h_priors::Vector{...[2]))
```

# General optimizations

- avoiding unnecessary allocations through the view instruction

```
ph[k] = loglikelihood(Normal(
    muh_draw + (lk_xPPM ? dot(view(Xlk_covariates,j,:,t), beta_iter[t]) : 0),
    sqrt(sig2h_draw)),
    Y[j,t]) + lPP[1]
```

and through in-place operations

```
copy!(s1n, @view sp1[aux_idx])
copy!(s2n, @view sp2[aux_idx])

spatial_cohesion!(spatial_cohesion_idx, s1n, s2n, sp_params_struct,
true, M_dp, S,1,true,lPP)
# LPP += spatial_cohesion!(spatial_cohesion_idx, s1n, s2n,
sp_params_struct, true, M_dp, S,1,true)
```

# General optimizations

- benchmarking different possible solutions through the package BenchmarkTools (Chen et al., 2016).

```
using BenchmarkTools
nh_tmp = rand(100)
@btime nclus_temp = sum($nh_tmp .> 0)
# 168.956 ns (2 allocations: 112 bytes)
@btime nclus_temp = count(x->(x>0), $nh_tmp)
# 11.612 ns (0 allocations: 0 bytes)
```

- refining the definition of cohesion and similarity functions, which are performance-critical since they are called thousands or even millions of times at every fit

# Optimizing covariates similarities

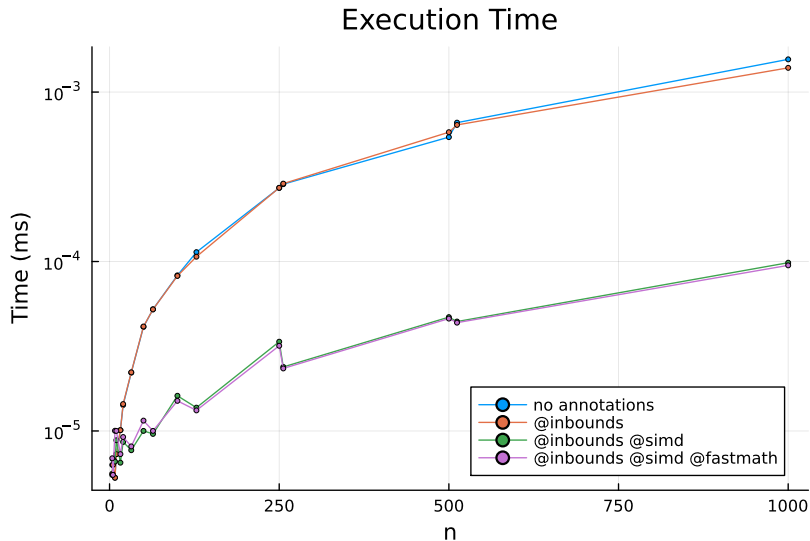
lasciamo questa qui come unico esempio

Problem: optimizing similarity  $g_4$ .

Solution: employing some optimizing macros on the inner loop.

```
function similarity4(X_jt::AbstractVector{<:Real}, mu_c::Real,
↳ lambda_c::Real, a_c::Real, b_c::Real, lg::Bool)
  n = length(X_jt); nm = n/2
  xbar = mean(X_jt)
  aux2 = 0.
  @inbounds @fastmath @simd for i in eachindex(X_jt)
    aux2 += X_jt[i]^2
  end
  aux1 = b_c + 0.5 * (aux2 - (n*xbar + lambda_c*mu_c)^2/(n+lambda_c) +
↳ lambda_c*mu_c^2 )
  out = -nm*log2pi + 0.5*log(lambda_c/(lambda_c+n)) + lgamma(a_c+nm) -
↳ lgamma(a_c) + a_c*log(b_c) + (-a_c-nm)*log(aux1)
  return lg ? out : exp(out)
end
```

# Optimizing covariates similarities





- 1 Description of the problem
- 2 Implementation and optimizations

### 3 Analysis of the models

- Comparing the two algorithms

- Performance with missing values

- Effects of the covariates

- Scaling performances

spostare questo "elenco" nell'altra slide

- 4 Conclusion
- 5 Appendix (additional material)

# Comparing the two algorithms

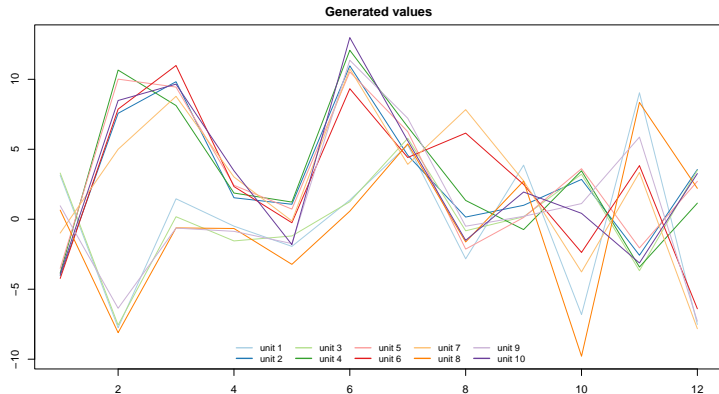
Our model represents a generalization of the original DRPM and its associated MCMC algorithm. Therefore, under identical datasets, hyperparameters, and MCMC configurations, both models are expected to perform similarly and produce comparable clusters estimates.

To evaluate the numerical performance of both algorithms, we will analyse posterior samples and clusters estimates in two scenarios:

- ① using a synthetic dataset that includes only the response variable
- ② employing a real-world spatio-temporal dataset, derived from the AgrImOnIA project (Fassò et al., 2023)

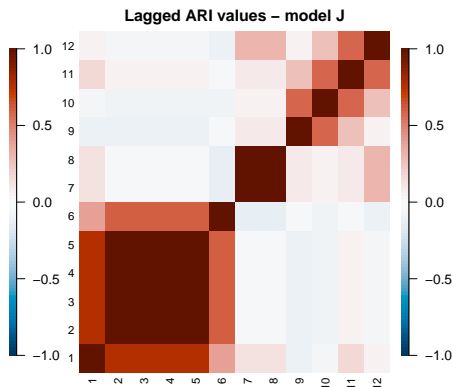
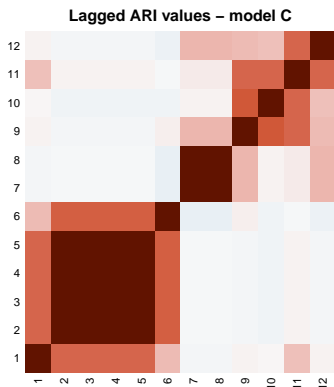
# Simulated data scenario

For the first comparison we generated a dataset consisting of  $n = 10$  units and  $T = 12$  time instants. Both algorithms were executed collecting 2000 iterates from a total of 50000 iterations, by discarding the first 40000 as burnin and then thinning by 5.



# Simulated data scenario

	MSE mean	MSE median	execution time
CDRPM	1.6221	1.5823	19s
JDRPM	<b>1.2634</b>	<b>1.2034</b>	<b>13s</b>



# Simulated data scenario

Computing the adjusted Rand index  $ARI(\rho_{JDRPM}(t), \rho_{CDRPM}(t))$  for all time instants  $t = 1, \dots, 12$ , we obtained a **mean of 0.95** and a **median of 1**.

model C

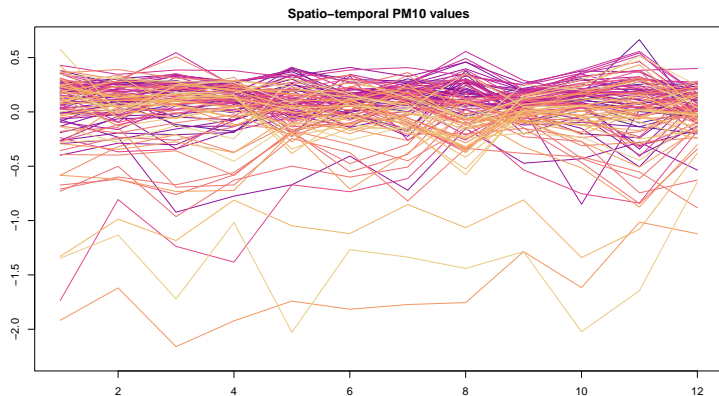
10	10	10	10	10	10	10	10	10	10	10	10	10
9	9	9	9	9	9	9	9	9	9	9	9	9
8	8	8	8	8	8	8	8	8	8	8	8	8
7	7	7	7	7	7	7	7	7	7	7	7	7
6	6	6	6	6	6	6	6	6	6	6	6	6
5	5	5	5	5	5	5	5	5	5	5	5	5
4	4	4	4	4	4	4	4	4	4	4	4	4
3	3	3	3	3	3	3	3	3	3	3	3	3
2	2	2	2	2	2	2	2	2	2	2	2	2
1	1	1	1	1	1	1	1	1	1	1	1	1
1	2	3	4	5	6	7	8	9	10	11	12	

model J

10	10	10	10	10	10	10	10	10	10	10	10	10
9	9	9	9	9	9	9	9	9	9	9	9	9
8	8	8	8	8	8	8	8	8	8	8	8	8
7	7	7	7	7	7	7	7	7	7	7	7	7
6	6	6	6	6	6	6	6	6	6	6	6	6
5	5	5	5	5	5	5	5	5	5	5	5	5
4	4	4	4	4	4	4	4	4	4	4	4	4
3	3	3	3	3	3	3	3	3	3	3	3	3
2	2	2	2	2	2	2	2	2	2	2	2	2
1	1	1	1	1	1	1	1	1	1	1	1	1
1	2	3	4	5	6	7	8	9	10	11	12	

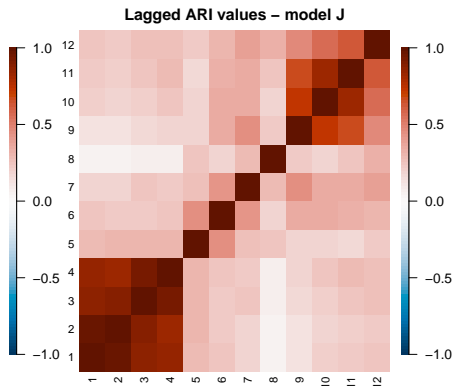
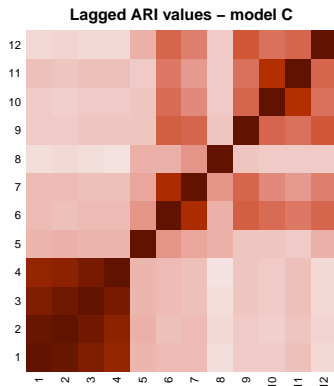
## Real-world scenario

For the second comparison, we employed a dataset comprising weekly averages of  $\text{PM}_{10}$  values from the year 2018, measured by  $n = 105$  units for  $T = 12$  time instants. We collected 4000 iterates from 110000 total iterations, by discarding the first 90000 as burnin and then thinning by 5.



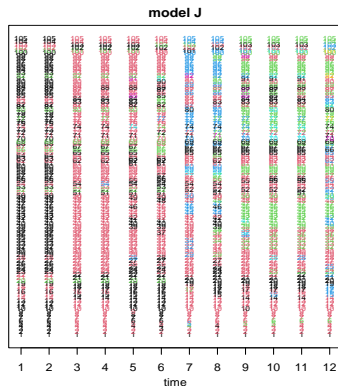
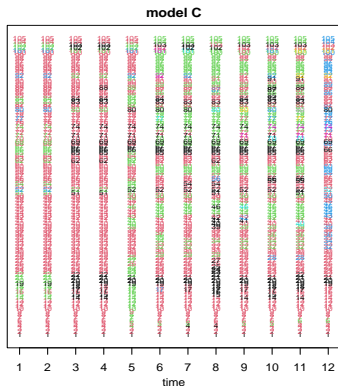
# Real-world scenario

	MSE mean	MSE median	execution time
CDRPM	0.0142	0.0149	1h38m
JDRPM	<b>0.0131</b>	<b>0.0138</b>	<b>48m</b>



# Real-world scenario

Computing the adjusted Rand index  $ARI(\rho_{\text{JDRPM}}(t), \rho_{\text{CDRPM}}(t))$  for all time instants  $t = 1, \dots, 12$ , we obtained a **mean of 0.80** and a **median of 0.86**, denoting a strong agreement between the clusters estimates.





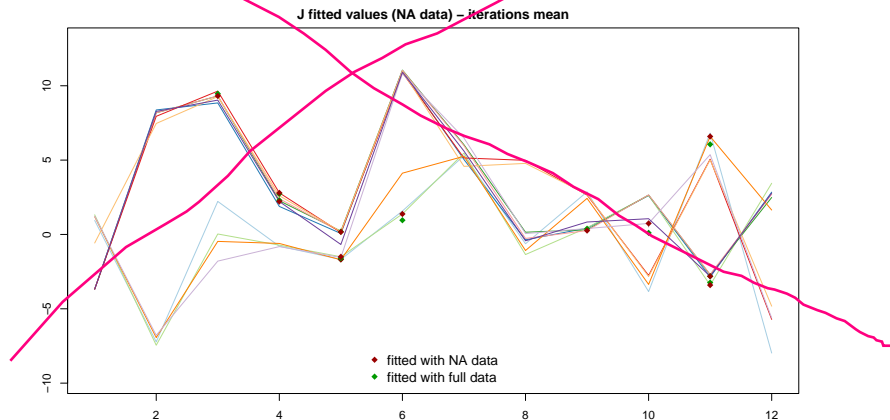
# Performance with missing values

We then replicated the analyses focusing on scenarios involving missing values, with the objective of investigating how the JDRPM performs in the absence of complete datasets and to determine whether it maintains effective performances under such conditions.

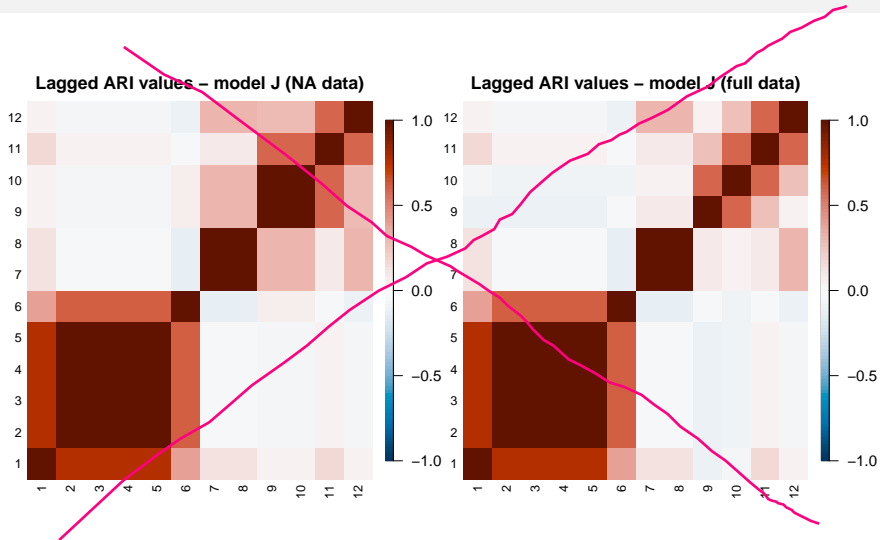
Given the extent of missing values in the AgrImOnIA dataset (Fassò et al., 2023), which was used for the spatio-temporal analysis, we opted to set 10% of the values as missing (NAs). To implement this, we randomly selected  $nT/10$  indexes from the sets  $[1, \dots, n]$  and  $[1, \dots, T]$  to identify all the pairs  $(i, t)$  that would be designated as missing entries in the target variable  $Y_{it}$ .

# Simulated data

	MSE mean	MSE median	LPML	WAIC	exec. time
full data	1.2634	1.2034	-223.36	393.97	13s
NA data	1.4721	1.2101	-236.93	401.44	13s

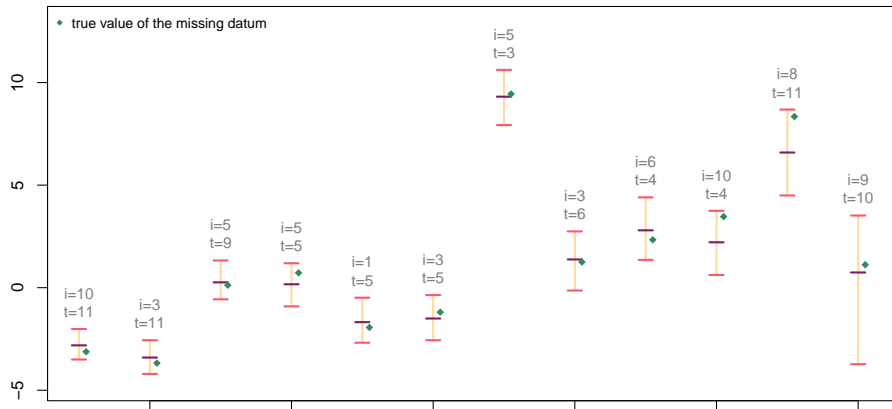


# Simulated data



# Simulated data

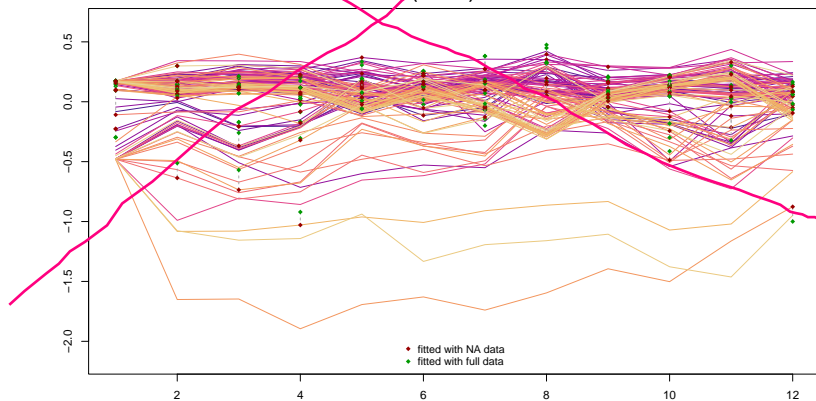
95% credible intervals (HDI) for the missing data



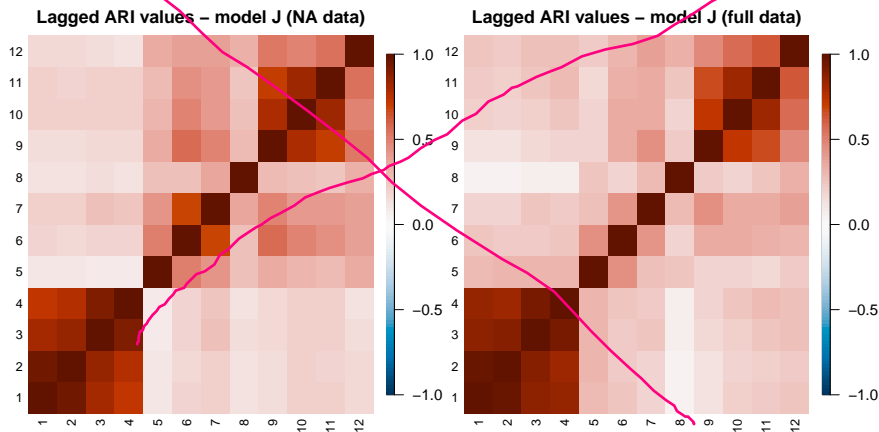
# Real-world scenario

	MSE mean	MSE median	LPML	WAIC	exec. time
full data	0.0131	0.0138	624.91	-1898.05	48m
NA data	0.0160	0.0170	502.86	-1793.64	43m

J fitted values (NA data) – iterations mean



# Real-world scenario



# Real-world scenario

95% credible intervals (HDI) for the missing data

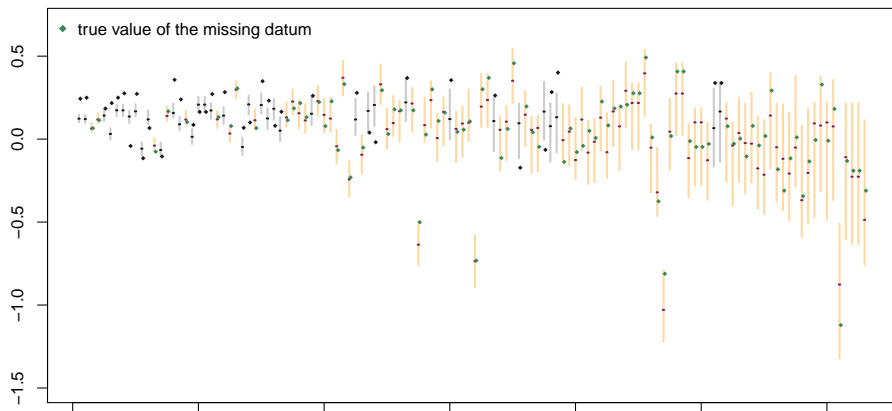


Figure: 74% of the true values lie within the credible intervals.

# Effects of the covariates

We then conducted several experiments to explore the key advancement introduced by the JDRPM: the inclusion of covariates.

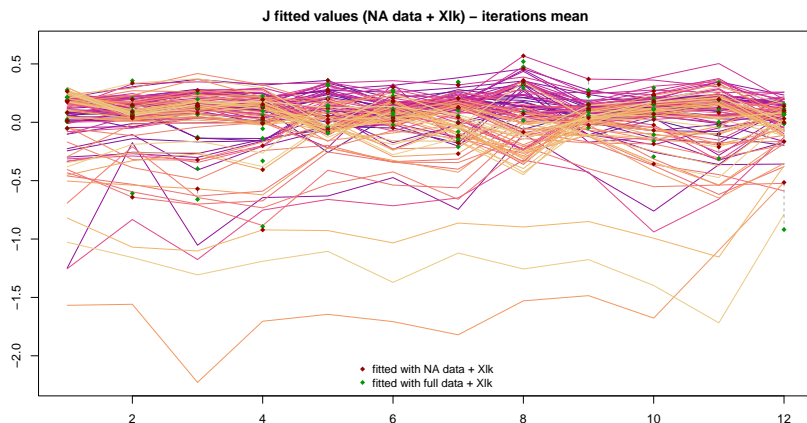
Given their distinctly different purposes, we studied separately their effects:

- for covariates in the likelihood, we repeated the spatio-temporal experiments, with missing data, to determine whether this approach would yield more refined fitted estimates
- for covariates in the prior, we repeated the spatio-temporal experiments to assess whether this approach would result in more accurate and interpretable cluster estimates



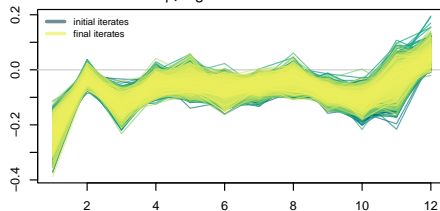
# Covariates in the likelihood

	MSE mean	MSE median	LPML	WAIC	exec. time
full data	0.0131	0.0138	624.91	-1898.05	48m
NA data	0.0160	0.0170	502.86	-1793.64	<b>43m</b>
NA data + Xlk	<b>0.0127</b>	<b>0.0130</b>	<b>625.81</b>	<b>-1902.74</b>	58m

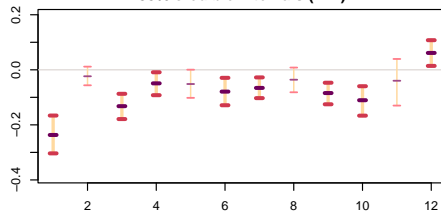


# Covariates in the likelihood

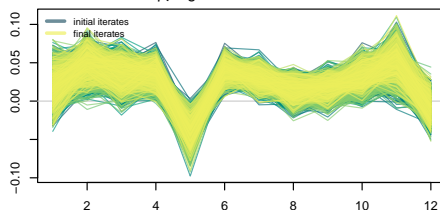
$\beta_t$  regressor – Altitude



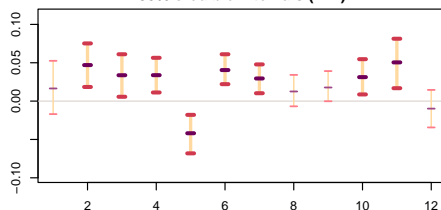
95% credible intervals (HDI)



$\beta_t$  regressor – LI\_bovine



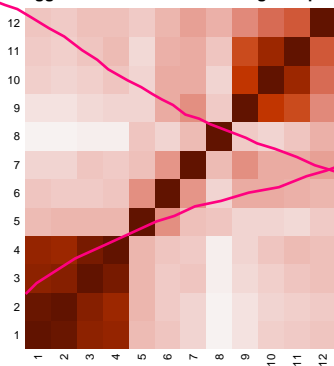
95% credible intervals (HDI)



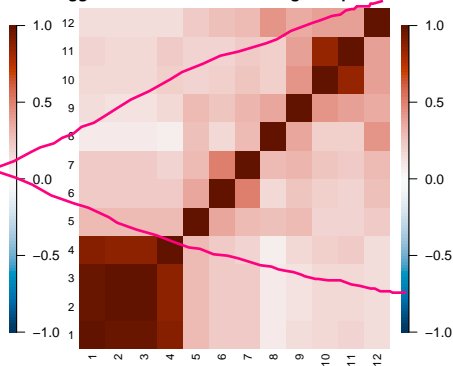
# Covariates in the prior

	MSE mean	MSE median	LPML	WAIC	exec. time
CDRPM	0.0142	0.0149	694.81	-1768.42	1h38m
JDRPM	0.0131	0.0138	624.91	-1898.05	<b>48m</b>
JDRPM + Xcl	<b>0.0126</b>	<b>0.0135</b>	<b>677.71</b>	<b>-1969.76</b>	1h20m

Lagged ARI values – JDRPM target + space



Lagged ARI values – JDRPM target + space + Xcl



# Covariates in the prior

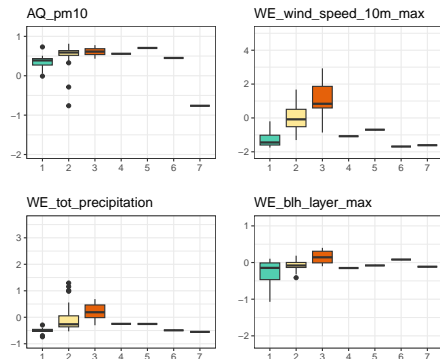
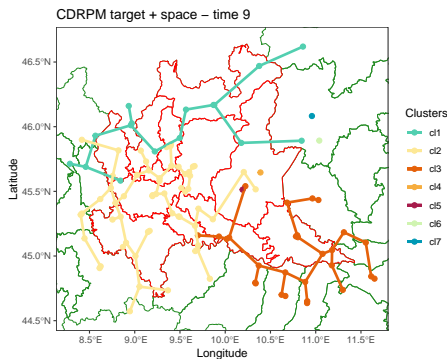


Figure: CDRPM spatially-informed fit.

# Covariates in the prior

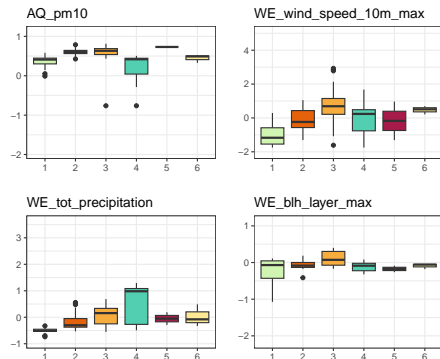
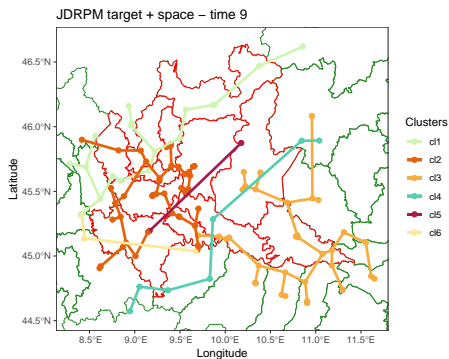


Figure: JDRPM spatially-informed fit.

# Covariates in the prior

questa parte veloce perché non è molto il focus

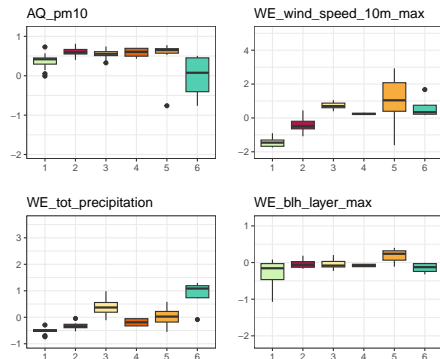
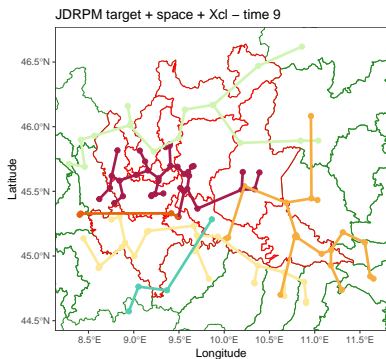
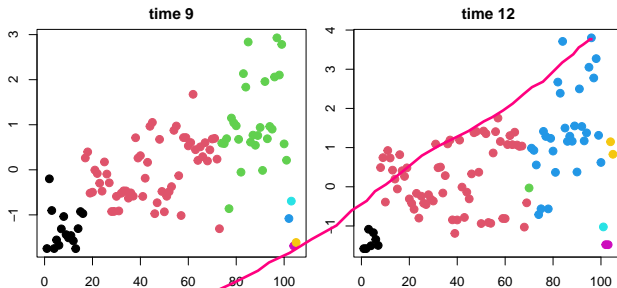
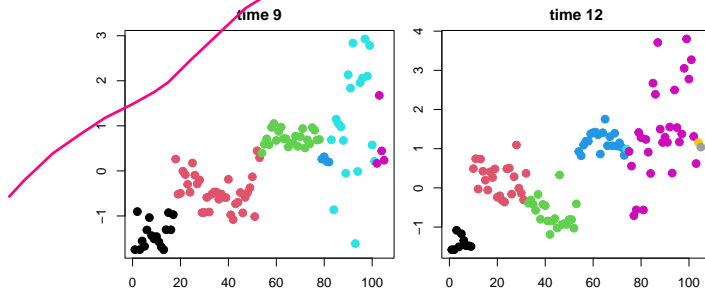


Figure: JDRPM spatially-informed fit with covariates in the prior.

CDRPM target + space – WE\_wind\_speed\_10m\_max (sorted by cluster)



JDRPM target + space + Xcl – WE\_wind\_speed\_10m\_max (sorted by cluster)



# Scaling performances

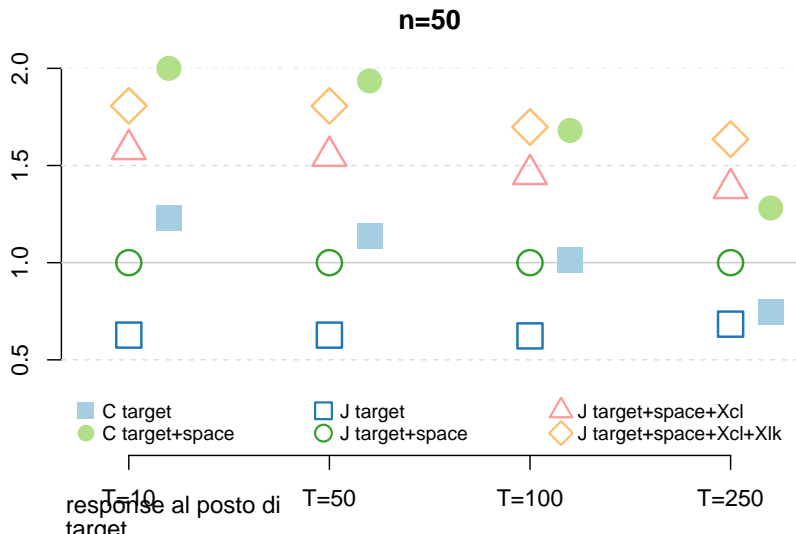
~~Finally~~, we designed a set of experiments to evaluate the computational performances of the CDRPM's and JDRPM's implementations.

We fitted both models across a “mesh” of dataset sizes, with both the number of units  $n$  and the time horizons  $T$  ranging through the set  $\{10, 50, 100, 250\}$ , and with information layers inserted incrementally on top of each other.

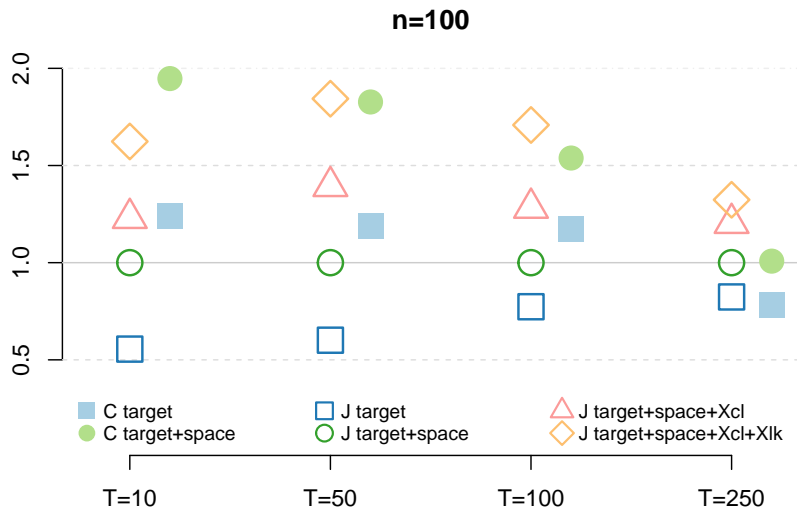
To measure the average execution time per iteration of each fit we defined the number of iterations to be inversely proportional to the size of the dataset, and repeated each fit was repeated multiple times to record the minimum execution time observed.



# Summary performances



# Summary performances



- ① Description of the problem
- ② Implementation and optimizations
- ③ Analysis of the models
- ④ **Conclusion**
  - Strengths
  - Drawbacks
- ⑤ Appendix (additional material)

# Strengths

al posto di JDRPM mettere "our model"

- JDRPM retains the foundational structure of its predecessor CDRPM, with the temporal modelling of the sequence of partitions
- the introduction of covariates information, as well as the accommodation of missing values, should allow more flexibility in the real-world researches
- despite the increased complexity, we provided more efficiency in the implementation, significantly reducing execution times
- the choice of the Julia language should facilitate easier code developments and future variations

in generale: raccontare bene il lavoro che ho fatto io, e in caso glissare su altre parti

# Drawbacks

the estimates of the parameters derived with out model

inserire la parte di github e interfaccia R

- the robustness of the fits may decrease due to the intricacies of parameters selection both in the prior distributions as well as in the cohesion and similarity functions
- reaching an appropriate balance between spatial and covariates information may require empirical testing (however, to address this problem, the Julia code already provides an optional argument, `cv_weight`, defaulted to 1, that allows to adjust the influence of covariates similarities)
- the choice of the inverse gamma distribution as the prior of the variance parameters allows better mixing properties but is more delicate to tune, compared to a simpler uniform distribution

il nostro modello e l'altro sono complessi e quindi in generale sono sensibili alla scelta degli iperparametri



*That's all Folks!*

# Bibliography I



Besançon, Mathieu et al. (2021). “Distributions.jl: Definition and Modeling of Probability Distributions in the JuliaStats Ecosystem”. In: *Journal of Statistical Software* 98.16, pp. 1–30. ISSN: 1548-7660. DOI: 10.18637/jss.v098.i16 (cit. on p. 26).



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. 25).



Chen, Jiahao and Jarrett Revels (Aug. 2016). “Robust benchmarking in noisy environments”. In: *arXiv e-prints*, arXiv:1608.04295. arXiv: 1608.04295 [cs.PF] (cit. on p. 30).



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 pp. 34, 41).

# Bibliography II



Ge, Hong, Kai Xu, and Zoubin Ghahramani (2018). “Turing: a language for flexible probabilistic inference”. In: *International Conference on Artificial Intelligence and Statistics, AISTATS 2018, 9-11 April 2018, Playa Blanca, Lanzarote, Canary Islands, Spain*, pp. 1682–1690. URL:

<http://proceedings.mlr.press/v84/ge18b.html> (cit. on p. 26).



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. 21).



— (2016). “Spatial Product Partition Models”. In: *Bayesian Analysis* 11.1, pp. 265–298. DOI: 10.1214/15-BA971 (cit. on p. 20).



# Bibliography III



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. 2,  
4–9, 11, 12, 19, 25, 27).

- ① Description of the problem
- ② Implementation and optimizations
- ③ Analysis of the models
- ④ Conclusion
- ⑤ Appendix (additional material)
  - Optimizing spatial cohesions
  - Inference on a new location
  - More performance analysis

# Optimizing spatial cohesions

Problem: optimizing cohesions  $C_3$  and  $C_4$ .

Solution v1: naive vector implementation.

```
sbar = [mean(s1), mean(s2)]  
vtmp = sbar - mu_0  
Mtmp = vtmp * vtmp'  
Psi_n = Psi + S + (k0*sdim) / (k0+sdim) * Mtmp  
:  
:
```

# Optimizing spatial cohesions

Problem: optimizing cohesions  $C_3$  and  $C_4$ .

Solution v2: implementation using only scalar variables.

```
sbar1 = mean(s1)
sbar2 = mean(s2)
vtmp_1 = sbar1 - mu_0[1]
vtmp_2 = sbar2 - mu_0[2]
Mtmp_1 = vtmp_1^2
Mtmp_2 = vtmp_1 * vtmp_2
Mtmp_3 = copy(Mtmp_2)
Mtmp_4 = vtmp_2^2
aux1 = k0 * sdim; aux2 = k0 + sdim
Psi_n_1 = Psi[1] + S1 + aux1 / (aux2) * Mtmp_1
Psi_n_2 = Psi[2] + S2 + aux1 / (aux2) * Mtmp_2
Psi_n_3 = Psi[3] + S3 + aux1 / (aux2) * Mtmp_3
Psi_n_4 = Psi[4] + S4 + aux1 / (aux2) * Mtmp_4
:
:
```

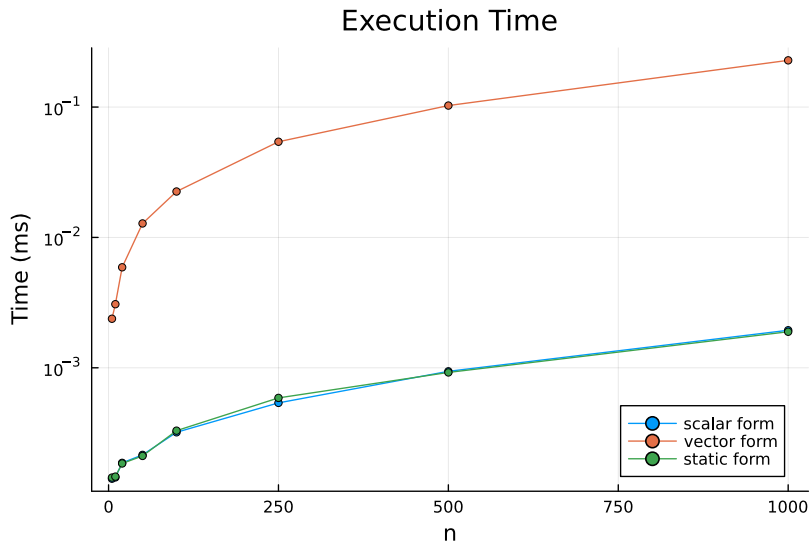
# Optimizing spatial cohesions

Problem: optimizing cohesions  $C_3$  and  $C_4$ .

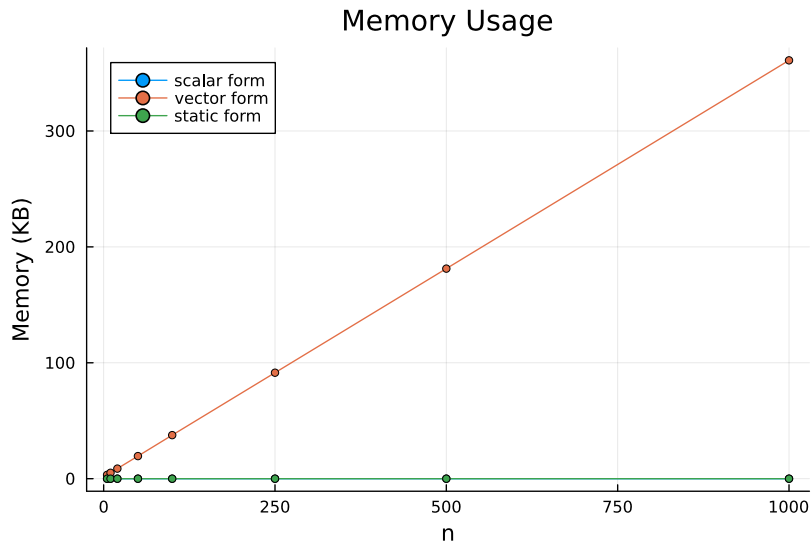
Solution: implementation using static vectors and matrices.

```
using StaticArrays
sbar1 = mean(s1); sbar2 = mean(s2)
sbar = SVector((sbar1, sbar2))
vtmp = sbar .- mu_0
Mtmp = vtmp * vtmp'
aux1 = k0 * sdim; aux2 = k0 + sdim
Psi_n = Psi .+ S .+ aux1 / (aux2) .* Mtmp
⋮
```

# Optimizing spatial cohesions



# Optimizing spatial cohesions



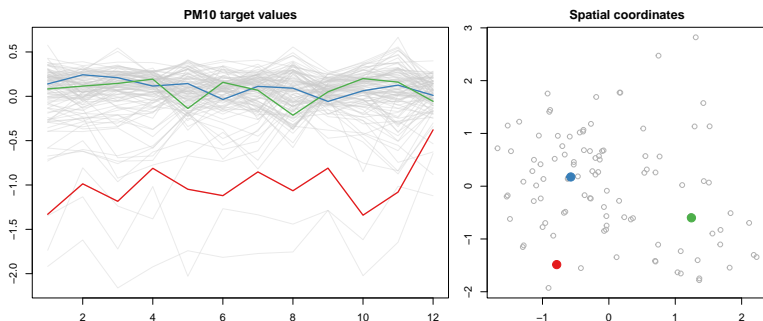
# Inference on a new location

As a final experiment on the effects of covariates, we considered a spatio-temporal scenario in which new units were added at new locations, with the objective of inferring the values of their target variable time series. We reproduced this scenario by removing all data entries from three randomly-selected units within the spatio-temporal dataset.

This context resembles the real use-case of predicting the behaviour of a unit for which sensors may be absent or inactive, with the expectation that the estimation accuracy will improve as model complexity increases.



		space	space+Xlk	space+Xlk+Xcl
unit 92 (red)	MSE mean	0.112452	<b>0.042037</b>	0.044957
	MSE median	0.111573	<b>0.041676</b>	0.045216
unit 61 (blue)	MSE mean	0.004117	<b>0.002449</b>	0.002527
	MSE median	0.004711	0.002547	<b>0.002534</b>
unit 44 (green)	MSE mean	<b>0.003919</b>	0.006368	0.005945
	MSE median	<b>0.003997</b>	0.006419	0.005950



# Inference on a new location

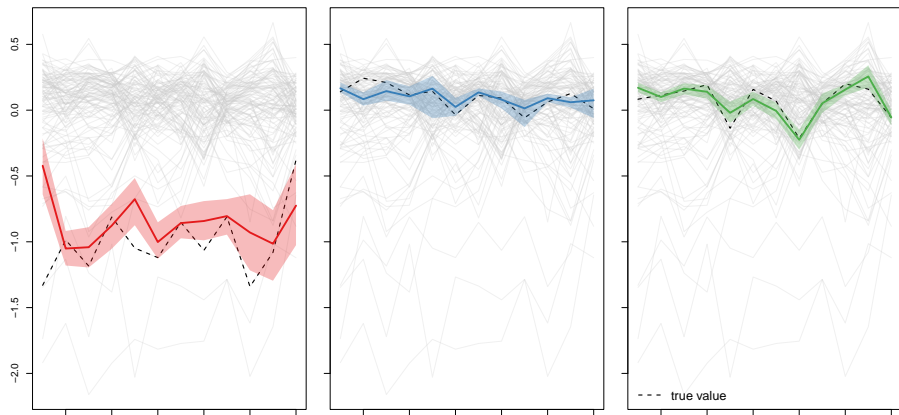
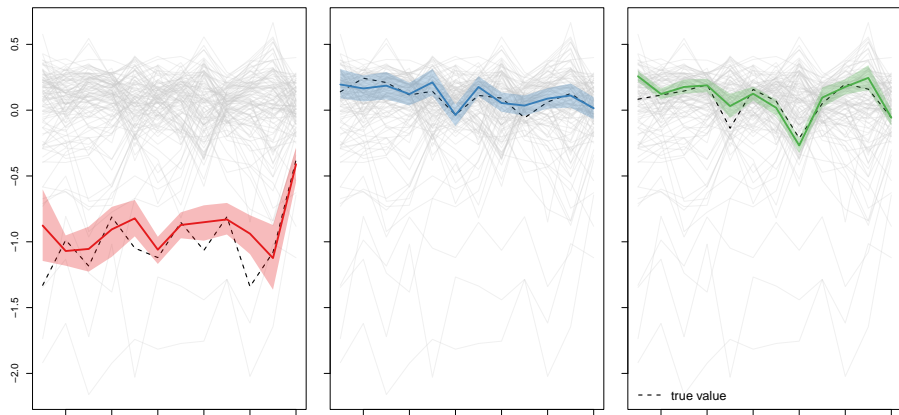


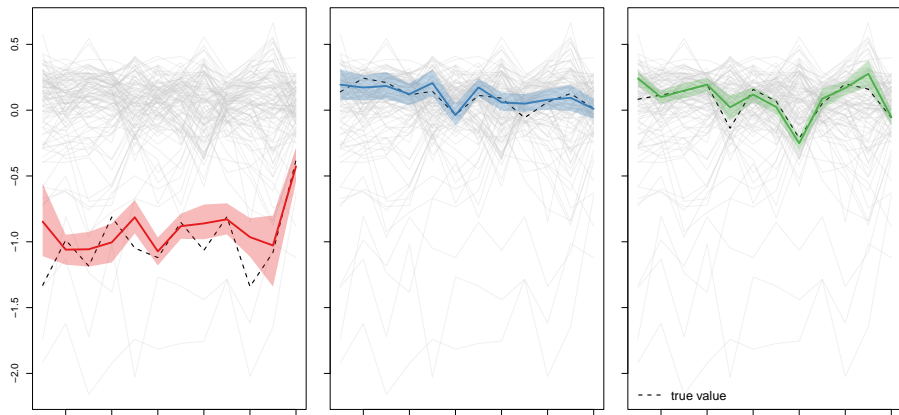
Figure: JDRPM spatially-informed fit.

# Inference on a new location



**Figure:** JDRPM spatially-informed fit with covariates in the likelihood.

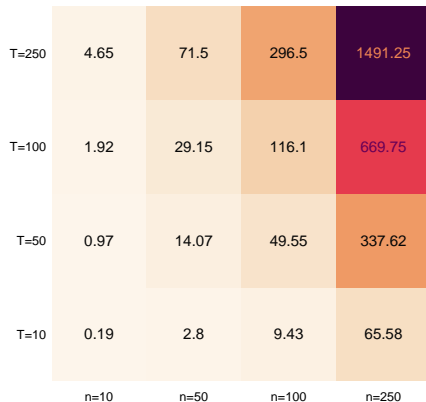
# Inference on a new location



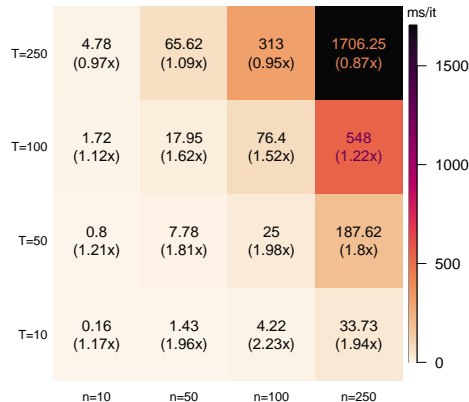
**Figure:** JDRPM spatially-informed fit with covariates in the likelihood and in the prior.

# Performances in the simulated data scenario

fit C – target only

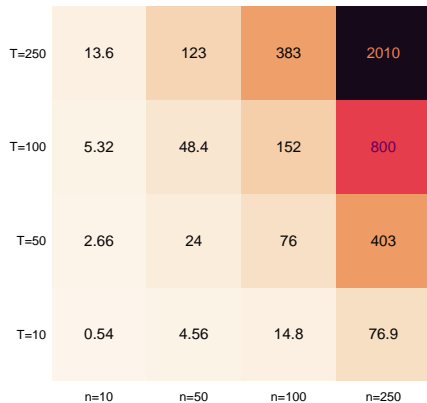


fit J – target only

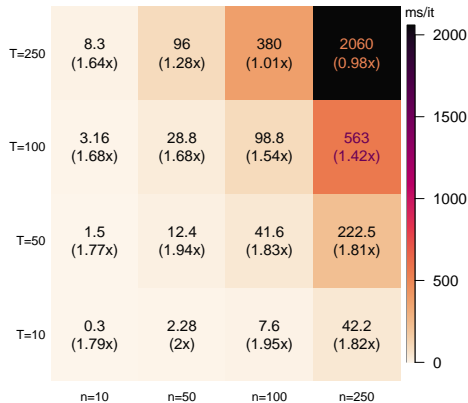


# Performances in the real-world scenario

fit C – target + space



fit J – target + space



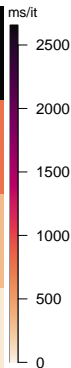
# Performances - varying $n$ and $T$ , fixed $p_{lk}$ and $p_{cl}$

fit J – target + space + Xcl (p=5)

T=250	14.2 (0.58x)	132.75 (0.72x)	457.25 (0.83x)	2551.25 (0.81x)
T=100	5.36 (0.59x)	41.9 (0.69x)	126.6 (0.78x)	702 (0.8x)
T=50	2.59 (0.58x)	19.2 (0.65x)	57.95 (0.72x)	298 (0.75x)
T=10	0.49 (0.61x)	3.61 (0.63x)	9.34 (0.81x)	45.73 (0.92x)
	n=10	n=50	n=100	n=250

fit J – target + space + Xcl + Xlk (p=5)

T=250	17.25 (0.48x)	157 (0.61x)	503 (0.76x)	2657.5 (0.78x)
T=100	6.52 (0.48x)	48.9 (0.59x)	168.8 (0.59x)	770.5 (0.73x)
T=50	3.22 (0.47x)	22.4 (0.55x)	76.7 (0.54x)	317.75 (0.7x)
T=10	0.63 (0.48x)	4.12 (0.55x)	12.34 (0.62x)	46.75 (0.9x)
	n=10	n=50	n=100	n=250



# Performances - fixed $n$ and $T$ , varying $p_{lk}$ and $p_{cl}$

