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's MCMC algorithm

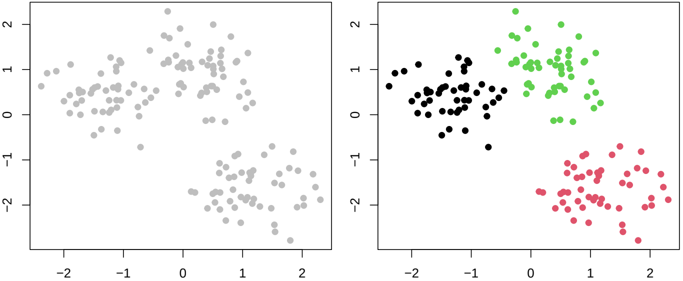
- produces up to spatially-informed clusters
  → I additionally introduced covariates information
- only accepts complete datasets
  → I made it work with missing values in the target variable
- has quite slow execution times (especially on large datasets)
  → I developed a brand-new and more efficient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

1. Description of the problem
   - What is the DRPM
   - How did we improve it
2. Implementation and optimizations
3. Analysis of our updates
4. Conclusion
5. Appendix (supplementary 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.

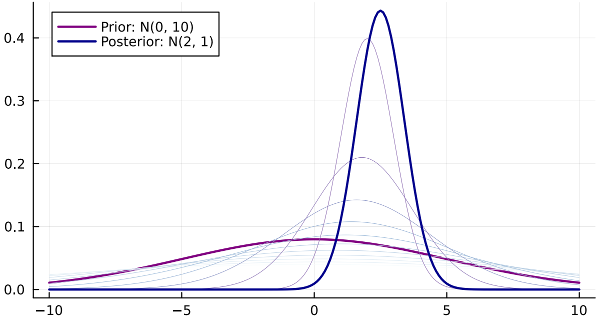
Clustering is a fundamental technique of unsupervised learning where a set of data points has to be divided into homogeneous groups of units which exhibit a similar behaviour.



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.

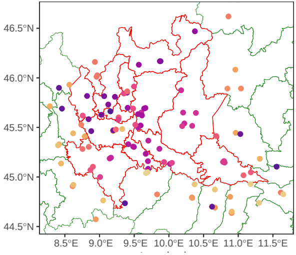
Bayesian models incorporate prior information on the model parameters and allow to assess uncertainty when performing inference on the results.

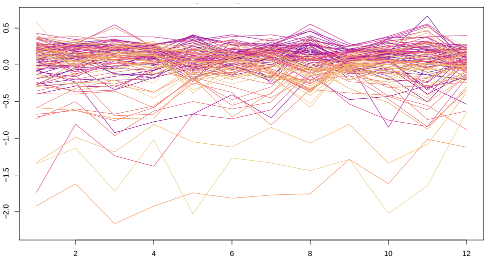


A bit of (spatio-temporal) context

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 $t = 1, \ldots, T$.





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.

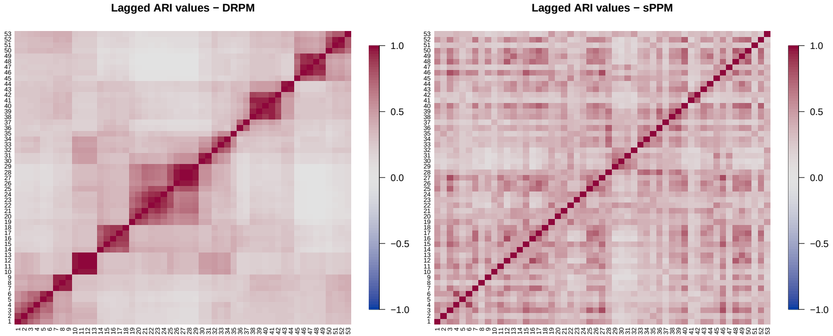
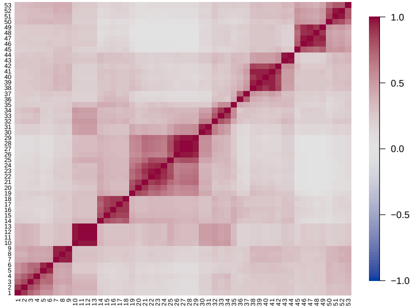


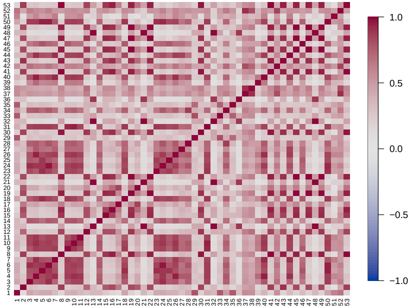
Figure: ARI($\hat{\rho}_t, \hat{\rho}_{t+k}$ computed for DRPM and a competitor model.

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 (Page et al., 2022)

To introduce temporal dependence in a collection of partitions, Page et al. (2022) assumed a first-order Markovian structure, leading to

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

where $P(\rho_1)$ is an exchangeable partition probability function (EPPF). Page et al. (2022) chose the EPPF induced by the Dirichlet Process

$$P(\rho_1) \propto \prod_{j=1}^{k_1} M \cdot (|S_{j1}| - 1)!$$

Modelling the temporal dependence (Page et al., 2022)

To characterize the other terms $P(\rho_t|\rho_{t-1})$, 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 ρ_{t-1} and ρ_t:

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

Modelling the temporal dependence (Page et al., 2022)

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, spanning from perfect temporal correlation ($\alpha_t = 0$) to full independence ($\alpha_t = 1$).

In this way the formulation of the joint model becomes

$$\begin{aligned} P(\boldsymbol{\gamma}_1, \rho_1, \dots, \boldsymbol{\gamma}_T, \rho_T) = {} & P(\rho_T | \boldsymbol{\gamma}_T, \rho_{T-1}) P(\boldsymbol{\gamma}_T) \times \cdots \\ & \times P(\rho_2 | \boldsymbol{\gamma}_2, \rho_1) P(\boldsymbol{\gamma}_2) P(\rho_1) \end{aligned}$$

The DRPM (Page et al., 2022)

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

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

The DRPM (Page et al., 2022)

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.

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

The DRPM (Page et al., 2022)

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

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

Our generalized model

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

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

Our updated formulation

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

$$
\begin{aligned}
Y_{it}|Y_{it-1},\boldsymbol{\mu}^{\star}_{t},\boldsymbol{\sigma}^{2\star}_{t},\boldsymbol{\eta},\boldsymbol{c}_{t} &\overset{\text{ind}}{\sim} \mathcal{N}(\mu^{\star}_{c_{it}t}+\eta_{1i}Y_{it-1}+\boldsymbol{x}_{it}^{T}\beta_{t},\sigma^{2\star}_{c_{it}t}(1-\eta_{1i}^{2}))\\
Y_{i1} &\overset{\text{ind}}{\sim} \mathcal{N}(\mu^{\star}_{c_{i1}1}+\boldsymbol{x}_{i1}^{T}\beta_{1},\sigma^{2\star}_{c_{i1}1})\\
\beta_{t} &\overset{\text{ind}}{\sim} \mathcal{N}_{p}(\boldsymbol{b},s^{2}I)\\
\xi_{i}=\text{Logit}(\tfrac{1}{2}(\eta_{1i}+1)) &\overset{\text{ind}}{\sim} \text{Laplace}(a,b)\\
(\mu^{\star}_{jt},\sigma^{2\star}_{jt}) &\overset{\text{ind}}{\sim} \mathcal{N}(\vartheta_{t},\tau^{2}_{t})\times\text{invGamma}(a_{\sigma},b_{\sigma})\\
\vartheta_{t}|\vartheta_{t-1} &\overset{\text{ind}}{\sim} \mathcal{N}((1-\varphi_{1})\varphi_{0}+\varphi_{1}\vartheta_{t-1},\lambda^{2}(1-\varphi_{1}^{2}))\\
(\vartheta_{1},\tau^{2}_{t}) &\overset{\text{iid}}{\sim} \mathcal{N}(\varphi_{0},\lambda^{2})\times\text{invGamma}(a_{\tau},b_{\tau})\\
(\varphi_{0},\varphi_{1},\lambda^{2}) &\sim \mathcal{N}(m_{0},s_{0}^{2})\times\mathcal{U}(-1,1)\times\text{invGamma}(a_{\lambda},b_{\lambda})\\
\{\boldsymbol{c}_{t},\ldots,\boldsymbol{c}_{T}\} &\sim \text{tRPM}(\boldsymbol{\alpha},M)\text{ with }\alpha_{t}\overset{\text{iid}}{\sim}\text{Beta}(a_{\alpha},b_{\alpha})
\end{aligned}
$$

Our updated formulation

The regressor term $\boldsymbol{\beta}_t$ provides more flexibility to the model formulation through the insertion of covariates into the likelihood.

Prior: $\boldsymbol{\beta}_t \sim \mathcal{N}_p(\boldsymbol{b}, s^2 I)$
Update rule:

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

$$\boldsymbol{h}_{(\text{post})} = \left(\frac{\boldsymbol{b}}{s^2} + \sum_{i=1}^{n} \frac{(Y_{it} - \mu^{\star}_{c_{it}t})\boldsymbol{x}_{it}}{\sigma^{2\star}_{c_{it}t}} \right) \quad J_{(\text{post})} = \left(\frac{1}{s^2} I + \sum_{i=1}^{n} \frac{\boldsymbol{x}_{it}\boldsymbol{x}_{it}^T}{\sigma^{2\star}_{c_{it}t}} \right)$$

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

$$\boldsymbol{h}_{(\text{post})} = \left(\frac{\boldsymbol{b}}{s^2} + \sum_{i=1}^{n} \frac{(Y_{it} - \mu^{\star}_{c_{it}t} - \eta_{1i} Y_{it-1})\boldsymbol{x}_{it}}{\sigma^{2\star}_{c_{it}t}} \right) \quad J_{(\text{post})} = \left(\frac{1}{s^2} I + \sum_{i=1}^{n} \frac{\boldsymbol{x}_{it}\boldsymbol{x}_{it}^T}{\sigma^{2\star}_{c_{it}t}} \right)$$

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

Our updated formulation

The choice of the inverse gamma distribution for the variance parameters recovers conjugacy within the model.

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

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

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

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

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

Similar derivations apply to τ_t^2 and λ^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 (Page et al., 2022)

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

To include spatial information, we can move to a spatial cohesion function $C(S_{jt}, \boldsymbol{s}_{jt}^{\star}|M, \mathcal{S})$, which measures the compactness of the spatial coordinates $\boldsymbol{s}_{jt}^{\star}$.

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

We implemented the spatial cohesion functions from Page et al. (2016).

Covariates information - our update

In a similar way, we inserted covariates information by introducing an additional covariates similarity function $g(S_{jt}, \boldsymbol{x}^{\star}_{jtr} | \mathcal{C})$, which measures the similarity of the r-th covariate values $\boldsymbol{x}^{\star}_{jtr}$.

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

We implemented the covariates similarity functions Page et al. (2018).

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

$$\sigma^2_{Y_{it}(\text{post})} = 1 \Big/ \left(\frac{1}{\sigma^{2\star}_{c_{it}t}} + \frac{\eta^2_{1i}}{2\sigma^{2\star}_{c_{it+1}t+1}(1-\eta^2_{1i})} \right)$$

$$\mu_{Y_{it}(\text{post})} = \sigma^2_{Y_{it}(\text{post})} \left(\frac{\mu^\star_{c_{it}t} + \boldsymbol{x}^T_{it}\boldsymbol{\beta}_t}{\sigma^{2\star}_{c_{it}t}} + \frac{\eta_{1i}(Y_{it+1} - \mu^\star_{c_{it+1}t+1} - \boldsymbol{x}^T_{it+1}\boldsymbol{\beta}_{t+1})}{\sigma^{2\star}_{c_{it+1}t+1}(1-\eta^2_{1i})} \right)$$

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

$$\sigma^2_{Y_{it}(\text{post})} = \left(1-\eta^2_{1i}\right) \Big/ \left(\frac{1}{\sigma^{2\star}_{c_{it}t}} + \frac{\eta^2_{1i}}{\sigma^{2\star}_{c_{it+1}t+1}} \right)$$

$$\mu_{Y_{it}(\text{post})} = \sigma^2_{Y_{it}(\text{post})} \left(\frac{\mu^\star_{c_{it}t} + \eta_{1i}Y_{it-1} + \boldsymbol{x}^T_{it}\boldsymbol{\beta}_t}{\sigma^{2\star}_{c_{it}t}(1-\eta^2_{1i})} + \frac{\eta_{1i}(Y_{it+1} - \mu^\star_{c_{it+1}t+1} - \boldsymbol{x}^T_{it+1}\boldsymbol{\beta}_{t+1})}{\sigma^{2\star}_{c_{it+1}t+1}(1-\eta^2_{1i})} \right)$$

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.

Implementation and optimizations

How to implement the MCMC algorithm of our updatede model? 🤔
In Julia! 🤩

Why Julia?

- combines the ease and expressiveness of high-level languages (e.g. R, python, matlab) with the efficiency and performance of low-level languages (e.g. C, C++, Fortran)
- 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
- vast collection of optimized and complete scientific packages, e.g. `Statistics`, `Distributions` (Besançon et al., 2021), and `MCMCChains` (Ge et al., 2018)

Spoiler alert

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

⌛ CDRPM fit with space $\Longleftrightarrow$ JDRPM fit with space *and* five covariates in the prior *and* five covariates in the likelihood ⌛

This performance gain was achieved through several optimizations steps.

Optimizing covariates similarities

Problem: optimizing covariates similarity g_4.

```julia
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.
    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
```

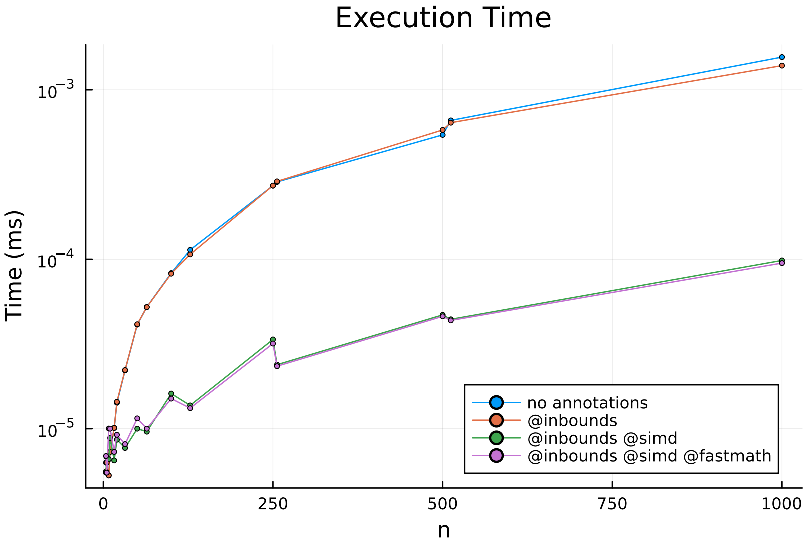
Optimizing covariates similarities

Problem: optimizing covariates similarity g_4.

Solution: applying 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



Analysis of the models

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)

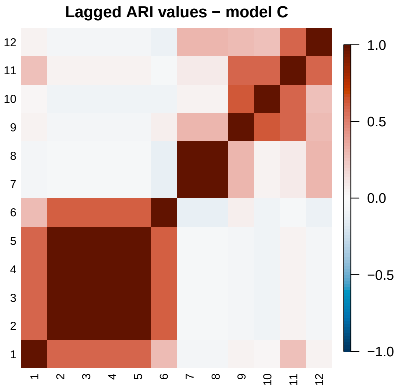
Fair comparison

Under identical conditions on the datasets, MCMC setup (burnin, thinning, number of iterations), and hyperparameters?

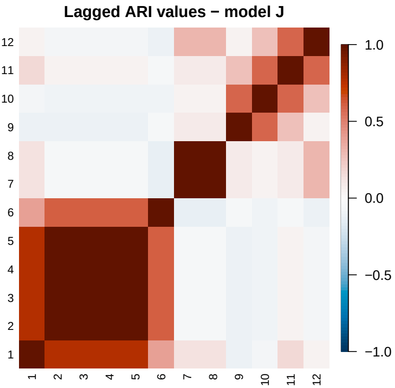
A fair comparison

Under identical conditions on the datasets, MCMC setup (burnin, thinning, number of iterations), and hyperparameters?

Everything works nicely, they produce very similar results.



Lagged ARI values − model C

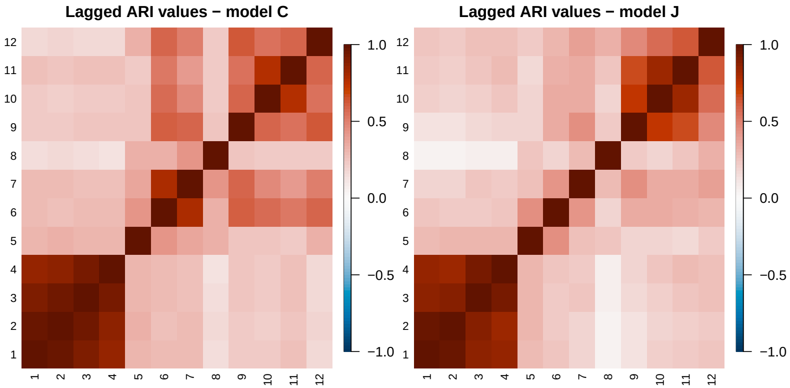


Lagged ARI values − model J

A fair comparison

Under identical conditions on the datasets, MCMC setup (burnin, thinning, number of iterations), and hyperparameters?

Everything works nicely, they produce very similar results.



A fair comparison

Under identical conditions on the datasets, MCMC setup (burnin, thinning, number of iterations), and hyperparameters?

Everything works nicely, they produce very similar results.
. . . *and our model is faster* 😏

simulated scenario	MSE mean	MSE median	execution time
CDRPM	1.6221	1.5823	19s (3.8 ms/it)
JDRPM	**1.2634**	**1.2034**	**13s (2.6 ms/it)**

real-world scenario	MSE mean	MSE median	execution time
CDRPM	0.0142	0.0149	1h38m (54 ms/it)
JDRPM	**0.0131**	**0.0138**	**48m (26 ms/it)**

Performance with missing values

In presence of missing values, how will our generalized model perform?

We randomly removed 10% of the target values Y_{it} from the dataset, so that they would become our “missing values”.

Performance with missing values

All the true values lie within the credible intervals.

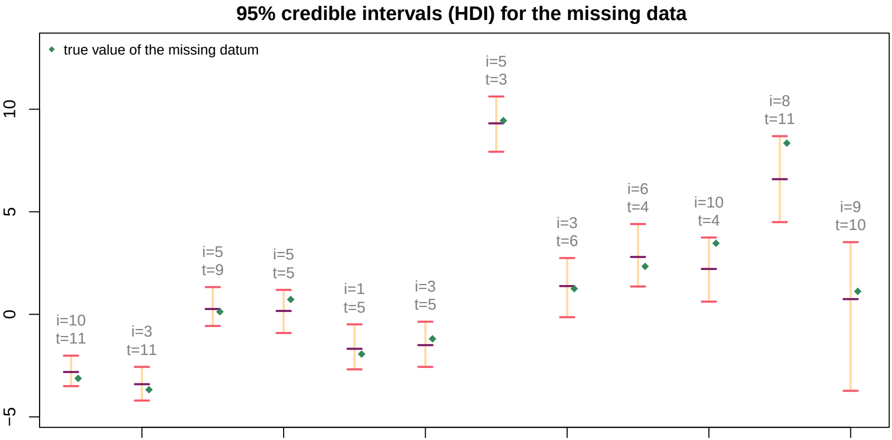


Figure: CIs of the fitted estimates for the missing Y_{it}'s, in the simulated data scenario.

Performance with missing values

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

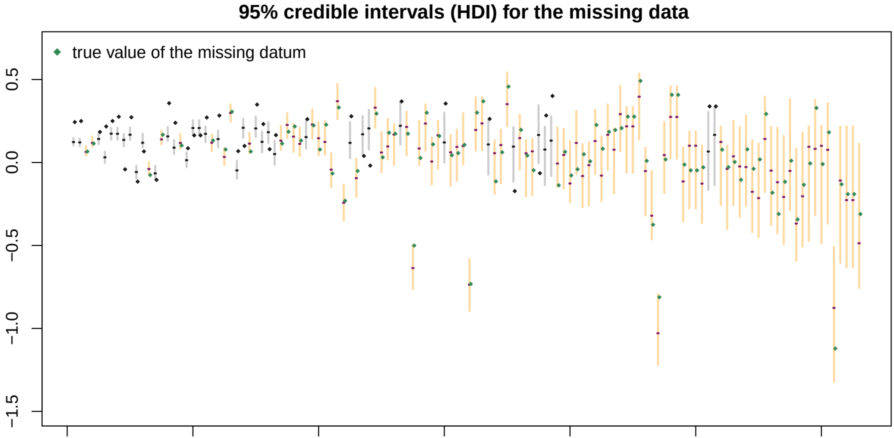


Figure: CIs of the fitted estimates for the missing Y_{it}'s, in the real-world scenario.

Effects of the covariates

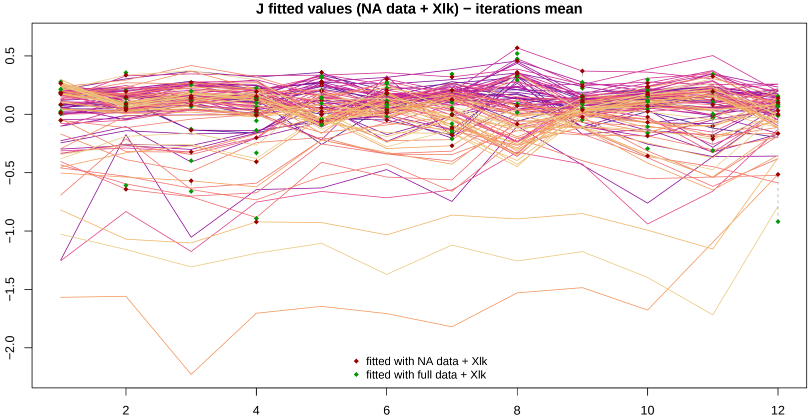
We then conducted several experiments to explore the key advancement introduced by the JDRPM: the inclusion of covariates.

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

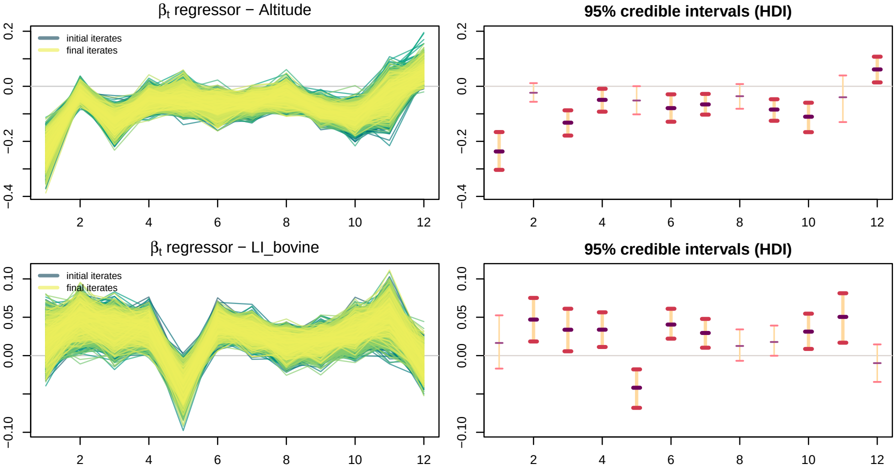
- for covariates in the likelihood, we repeated the spatio-temporal experiments, with missing data, to determine whether this approach would yield more refined fitted estimates
- for covariates in the prior, we repeated the spatio-temporal experiments to assess whether this approach would result in more 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	**43m**
NA data + Xlk	**0.0127**	**0.0130**	**625.81**	**-1902.74**	58m



Covariates in the likelihood



β_t regressor – Altitude

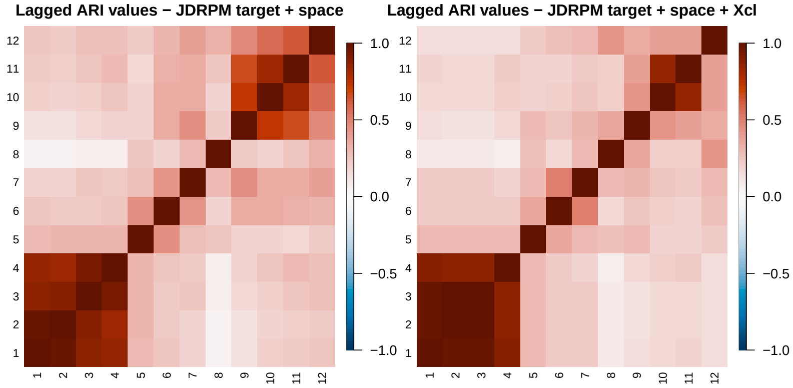
95% credible intervals (HDI)

β_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	**48m**
JDRPM + Xcl	**0.0126**	**0.0135**	**677.71**	**-1969.76**	1h20m



Covariates in the prior

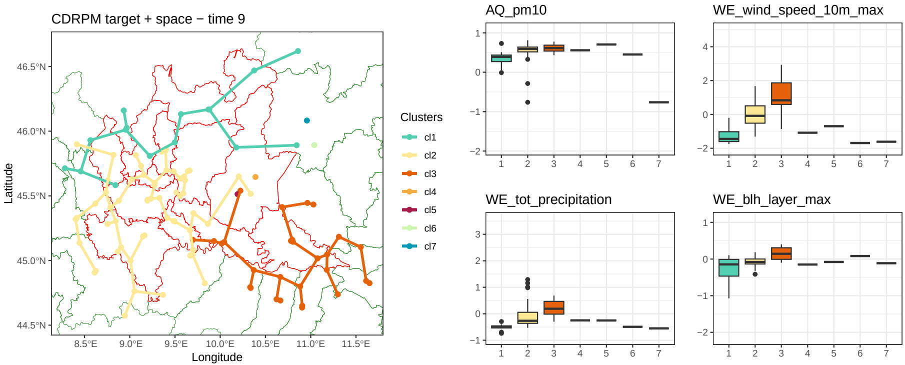


Figure: CDRPM spatially-informed fit.

Covariates in the prior

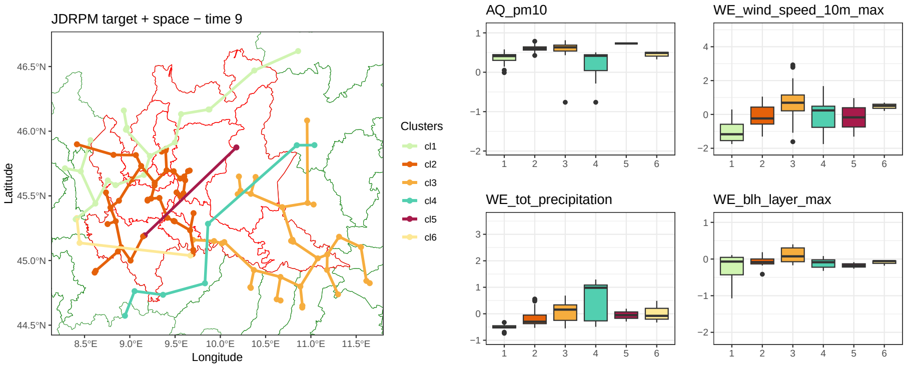


Figure: JDRPM spatially-informed fit.

Covariates in the prior

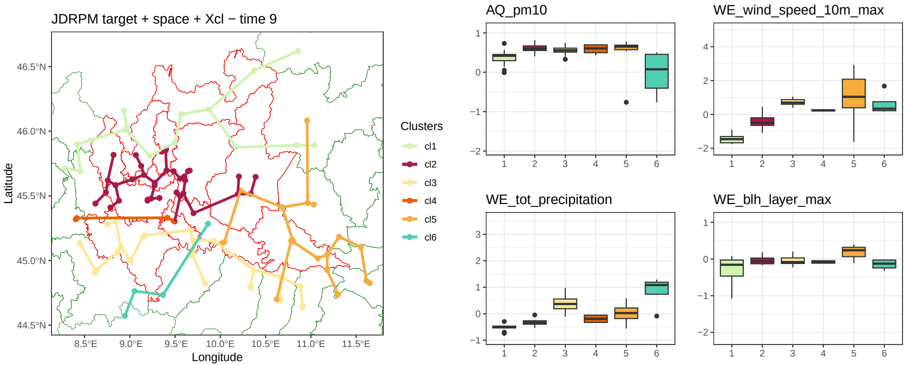
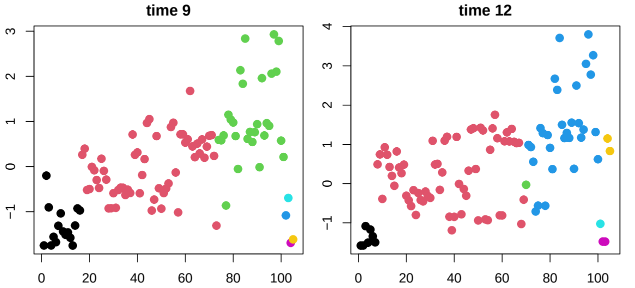
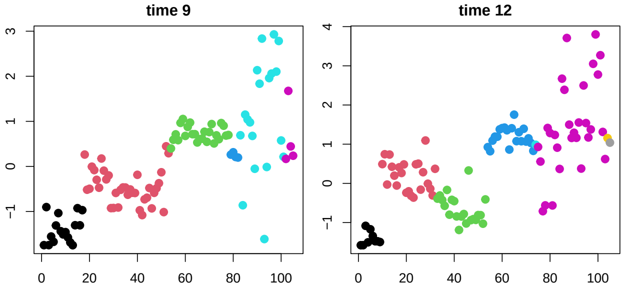


Figure: JDRPM spatially-informed fit with covariates in the prior.

CDRPM target + space − WE_wind_speed_10m_max (sorted by cluster)



JDRPM target + space + Xcl − WE_wind_speed_10m_max (sorted by cluster)



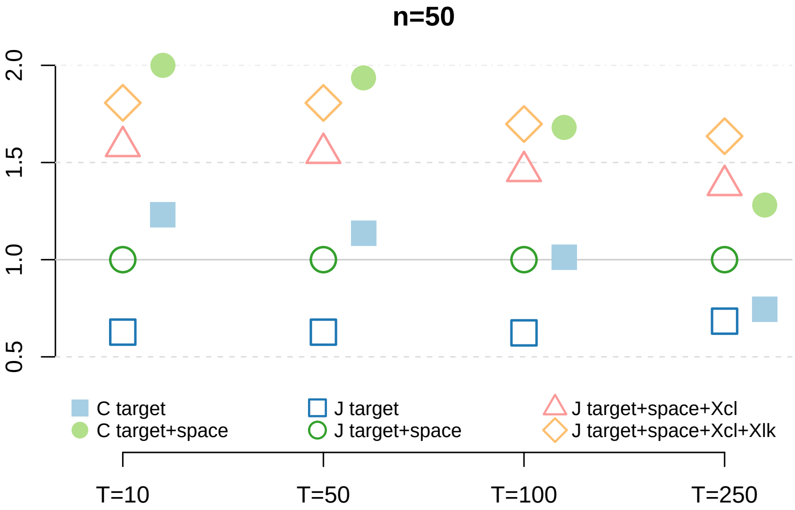
Scaling performances

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

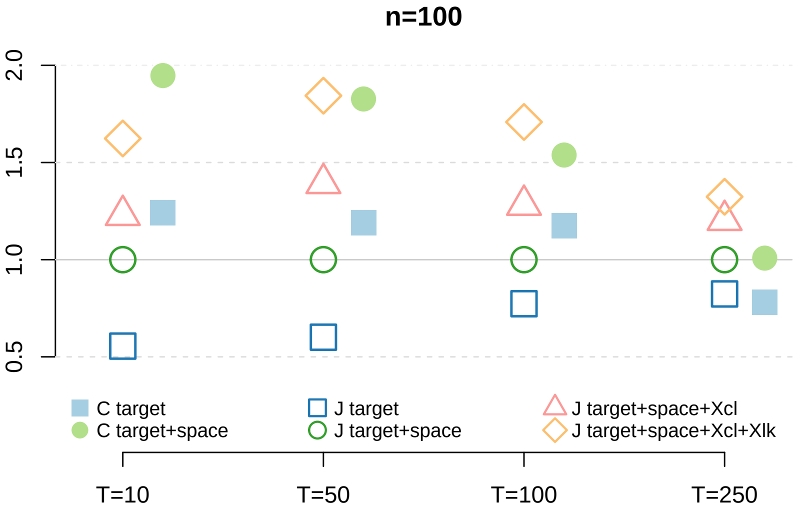
We fitted both models across a "mesh" of dataset sizes, with both the number of units n and the time horizons T ranging through the set $\{10, 50, 100, 250\}$, and with information layers inserted incrementally on top of each other.

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



Strengths

- 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

Drawbacks

- 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

That's all Folks!

Bibliography I

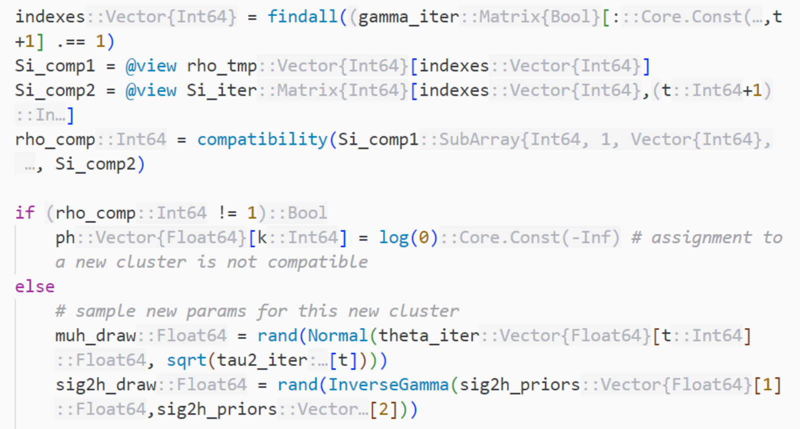
- Besançon, Mathieu et al. (2021). “Distributions.jl: Definition and Modeling of Probability Distributions in the JuliaStats Ecosystem”. In: *Journal of Statistical Software* 98.16, pp. 1–30. ISSN: 1548-7660. DOI: `10.18637/jss.v098.i16`.
- Chen, Jiahao and Jarrett Revels (Aug. 2016). “Robust benchmarking in noisy environments”. In: *arXiv e-prints*, arXiv:1608.04295. arXiv: `1608.04295 [cs.PF]`.
- 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`.

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

General optimizations

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

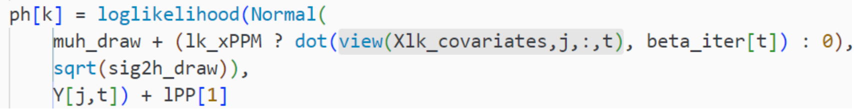


```
indexes::Vector{Int64} = findall((gamma_iter::Matrix{Bool}[::Core.Const(…,t
+1] .== 1)
Si_comp1 = @view rho_tmp::Vector{Int64}[indexes::Vector{Int64}]
Si_comp2 = @view Si_iter::Matrix{Int64}[indexes::Vector{Int64},(t::Int64+1)
::In…]
rho_comp::Int64 = compatibility(Si_comp1::SubArray{Int64, 1, Vector{Int64},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

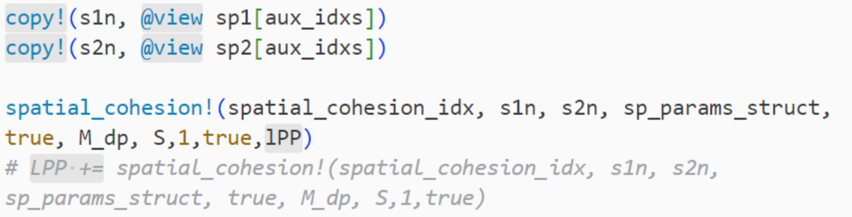
General optimizations

- 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_idxs])
copy!(s2n, @view sp2[aux_idxs])

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).

```julia
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 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
```

$\vdots$

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
```

$\vdots$

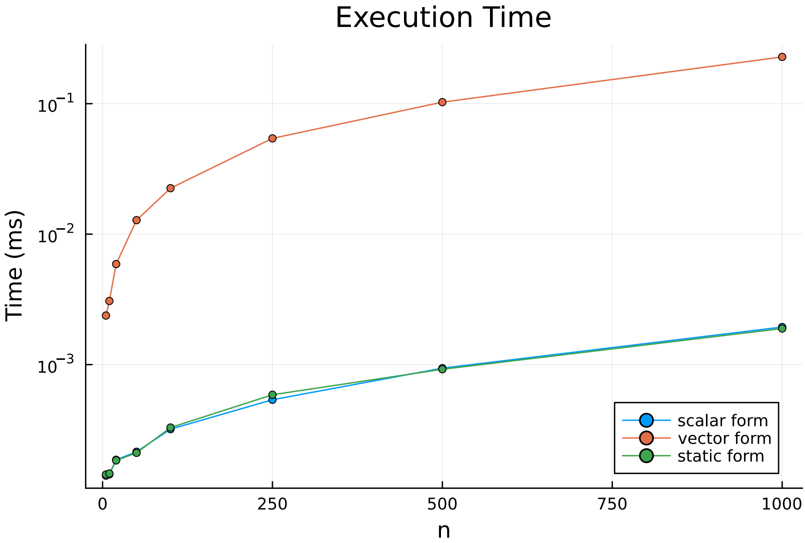
Optimizing spatial cohesions

Problem: optimizing cohesions C_3 and C_4.

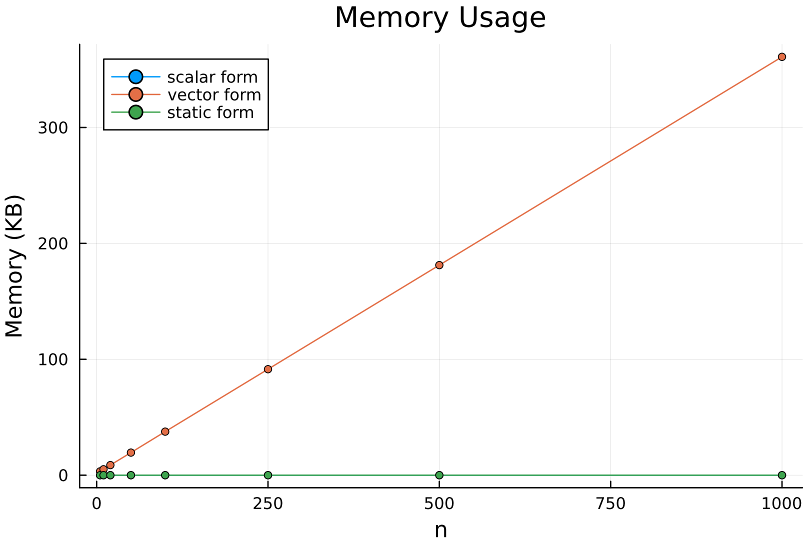
Solution: implementation using static vectors and matrices.

```
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

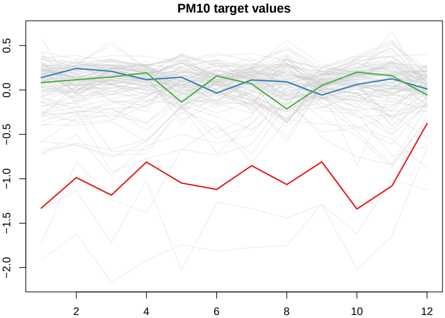


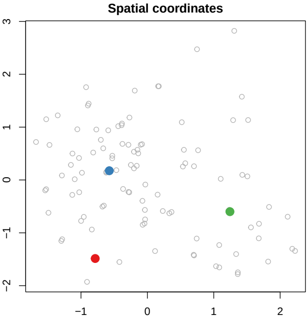
Inference on a new location

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	**0.042037**	0.044957
	MSE median	0.111573	**0.041676**	0.045216
unit 61 (blue)	MSE mean	0.004117	**0.002449**	0.002527
	MSE median	0.004711	0.002547	**0.002534**
unit 44 (green)	MSE mean	**0.003919**	0.006368	0.005945
	MSE median	**0.003997**	0.006419	0.005950





Inference on a new location

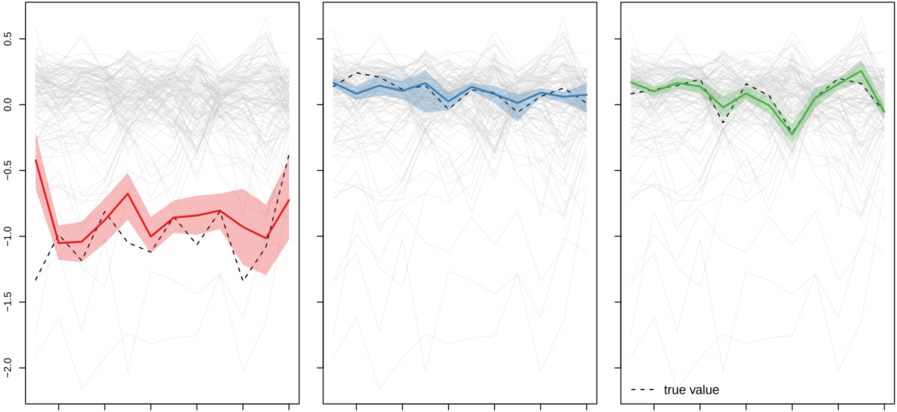


Figure: JDRPM spatially-informed fit.

Inference on a new location

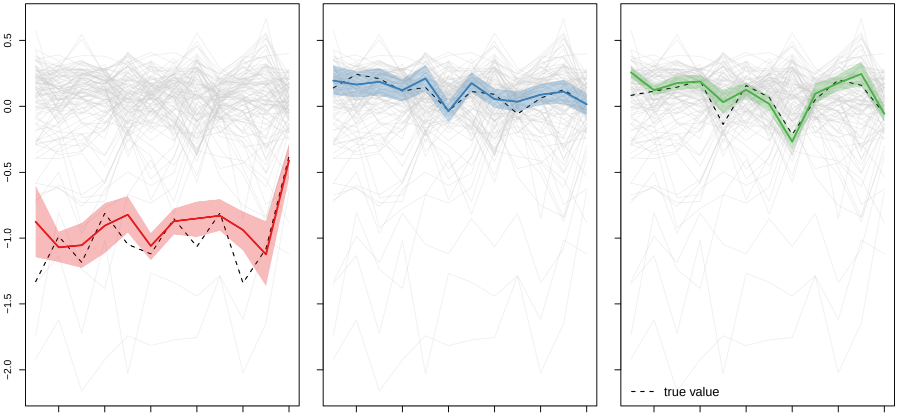


Figure: JDRPM spatially-informed fit with covariates in the likelihood.

Inference on a new location

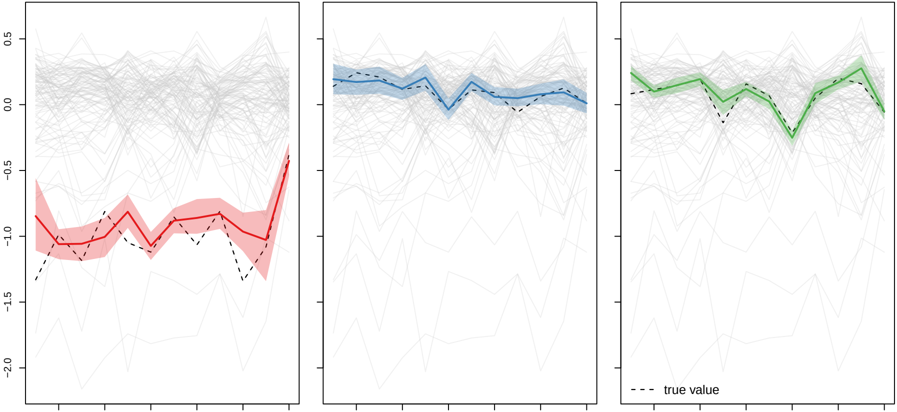
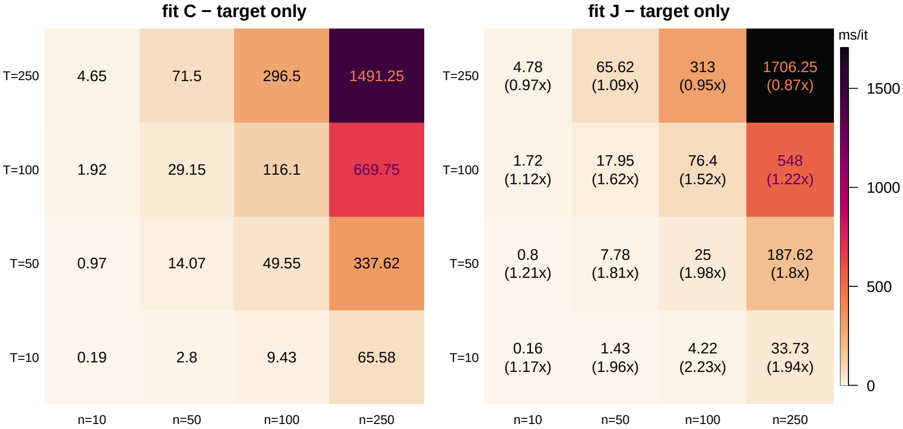


Figure: JDRPM spatially-informed fit with covariates in the likelihood and in the prior.

Performances in the simulated data scenario



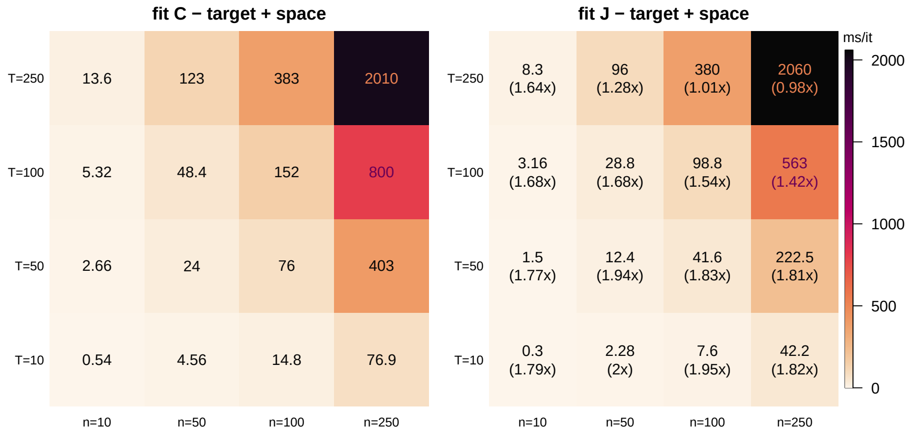
fit C − target only

	n=10	n=50	n=100	n=250
T=250	4.65	71.5	296.5	1491.25
T=100	1.92	29.15	116.1	669.75
T=50	0.97	14.07	49.55	337.62
T=10	0.19	2.8	9.43	65.58

fit J − target only

	n=10	n=50	n=100	n=250
T=250	4.78 (0.97x)	65.62 (1.09x)	313 (0.95x)	1706.25 (0.87x)
T=100	1.72 (1.12x)	17.95 (1.62x)	76.4 (1.52x)	548 (1.22x)
T=50	0.8 (1.21x)	7.78 (1.81x)	25 (1.98x)	187.62 (1.8x)
T=10	0.16 (1.17x)	1.43 (1.96x)	4.22 (2.23x)	33.73 (1.94x)

Performances in the real-world scenario



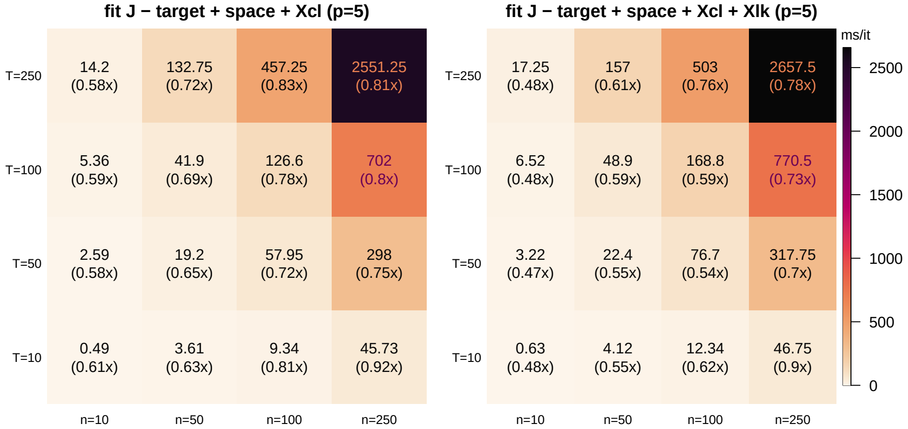
fit C − target + space

	n=10	n=50	n=100	n=250
T=250	13.6	123	383	2010
T=100	5.32	48.4	152	800
T=50	2.66	24	76	403
T=10	0.54	4.56	14.8	76.9

fit J − target + space

	n=10	n=50	n=100	n=250
T=250	8.3 (1.64x)	96 (1.28x)	380 (1.01x)	2060 (0.98x)
T=100	3.16 (1.68x)	28.8 (1.68x)	98.8 (1.54x)	563 (1.42x)
T=50	1.5 (1.77x)	12.4 (1.94x)	41.6 (1.83x)	222.5 (1.81x)
T=10	0.3 (1.79x)	2.28 (2x)	7.6 (1.95x)	42.2 (1.82x)

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



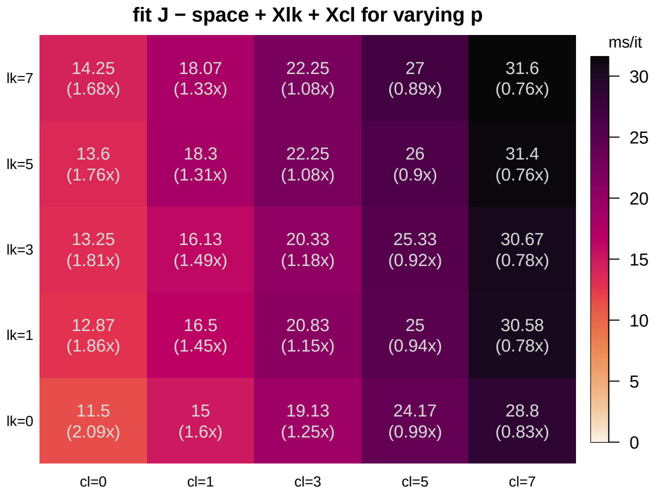
fit J − target + space + Xcl (p=5)

	n=10	n=50	n=100	n=250
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)

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

	n=10	n=50	n=100	n=250
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)

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



fit J − space + Xlk + Xcl for varying p

	cl=0	cl=1	cl=3	cl=5	cl=7
lk=7	14.25 (1.68x)	18.07 (1.33x)	22.25 (1.08x)	27 (0.89x)	31.6 (0.76x)
lk=5	13.6 (1.76x)	18.3 (1.31x)	22.25 (1.08x)	26 (0.9x)	31.4 (0.76x)
lk=3	13.25 (1.81x)	16.13 (1.49x)	20.33 (1.18x)	25.33 (0.92x)	30.67 (0.78x)
lk=1	12.87 (1.86x)	16.5 (1.45x)	20.83 (1.15x)	25 (0.94x)	30.58 (0.78x)
lk=0	11.5 (2.09x)	15 (1.6x)	19.13 (1.25x)	24.17 (0.99x)	28.8 (0.83x)