Politecnico of Milan

School of Industrial and Information Engineering
Master of Science in Mathematical Engineering

MASTER THESIS

Including covariates and spatial information into (DRPM), a bayesian time-dependent model for clustering

Advisor
Prof. Alessandra Guglielmi

Coadvisor
Prof. Alessandro Carminati

Candidate
Federico Angelo Mor
Matr. 221429

Academic Year 2023–2024

to my cats Otto
and La Micia

Abstract

Sommario

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Chapter 2

Description of the model

> *"Come on, gentlemen, why shouldn't we get rid of all this calm reasonableness with one kick, just so as to send all these logarithms to the devil and be able to live our own lives at our own sweet will?"*
>
> — Fëdor Dostoevskij, *Notes from the Underground*

$$
\begin{aligned}
Y_{it}|Y_{it-1}, \boldsymbol{\mu}_t^\star, \boldsymbol{\sigma}_t^{2\star}, \boldsymbol{\eta}, \boldsymbol{c}_t &\overset{\text{ind}}{\sim} \mathcal{N}(\mu^\star_{c_{it}t} + \eta_{1i} Y_{it-1}, \sigma^{2\star}_{c_{it}t}(1-\eta_{1i}^2)) \\
Y_{i1} &\overset{\text{ind}}{\sim} \mathcal{N}(\mu^\star_{c_{i1}1}, \sigma^{2\star}_{c_{i1}1}) \\
\xi_i = \text{Logit}(\tfrac{1}{2}(\eta_{1i}+1)) &\overset{\text{ind}}{\sim} \text{Laplace}(a, b) \\
(\mu^\star_{jt}, \sigma^\star_{jt}) &\overset{\text{ind}}{\sim} \mathcal{N}(\vartheta_t, \tau_t^2) \times \mathcal{U}(0, A_\sigma) \\
\vartheta_t|\vartheta_{t-1} &\overset{\text{ind}}{\sim} \mathcal{N}((1-\varphi_1)\varphi_0 + \varphi_1\vartheta_{t-1}, \lambda^2(1-\varphi_1^2)) \\
(\vartheta_1, \tau_t) &\overset{\text{iid}}{\sim} \mathcal{N}(\varphi_0, \lambda^2) \times \mathcal{U}(0, A_\tau) \\
(\varphi_0, \varphi_1, \lambda) &\sim \mathcal{N}(m_0, s_0^2) \times \mathcal{U}(-1, 1) \times \mathcal{U}(0, A_\lambda) \\
\{\boldsymbol{c}_t, \dots, \boldsymbol{c}_T\} &\sim \text{tRPM}(\boldsymbol{\alpha}, M) \text{ with } \alpha_t \overset{\text{iid}}{\sim} \text{Beta}(a_\alpha, b_\alpha)
\end{aligned} \tag{2.1}
$$

$$
\begin{aligned}
Y_{it}|Y_{it-1}, \boldsymbol{\mu}_t^\star, \boldsymbol{\sigma}_t^{2\star}, \boldsymbol{\eta}, \boldsymbol{c}_t &\overset{\text{ind}}{\sim} \mathcal{N}(\mu^\star_{c_{it}t} + \eta_{1i} Y_{it-1} + \boldsymbol{x}_{it}^T\boldsymbol{\beta}_t, \sigma^{2\star}_{c_{it}t}(1-\eta_{1i}^2)) \\
Y_{i1} &\overset{\text{ind}}{\sim} \mathcal{N}(\mu^\star_{c_{i1}1} + \boldsymbol{x}_{i1}^T\boldsymbol{\beta}_1, \sigma^{2\star}_{c_{i1}1}) \\
\boldsymbol{\beta}_t &\overset{\text{ind}}{\sim} \mathcal{N}_p(\boldsymbol{b}, s^2 I) \\
\xi_i = \text{Logit}(\tfrac{1}{2}(\eta_{1i}+1)) &\overset{\text{ind}}{\sim} \text{Laplace}(a, b) \\
(\mu^\star_{jt}, \sigma^{2\star}_{jt}) &\overset{\text{ind}}{\sim} \mathcal{N}(\vartheta_t, \tau_t^2) \times \text{invGamma}(a_\sigma, b_\sigma) \\
\vartheta_t|\vartheta_{t-1} &\overset{\text{ind}}{\sim} \mathcal{N}((1-\varphi_1)\varphi_0 + \varphi_1\vartheta_{t-1}, \lambda^2(1-\varphi_1^2)) \\
(\vartheta_1, \tau_t^2) &\overset{\text{iid}}{\sim} \mathcal{N}(\varphi_0, \lambda^2) \times \text{invGamma}(a_\tau, b_\tau) \\
(\varphi_0, \varphi_1, \lambda^2) &\sim \mathcal{N}(m_0, s_0^2) \times \mathcal{U}(-1, 1) \times \text{invGamma}(a_\lambda, b_\lambda) \\
\{\boldsymbol{c}_t, \dots, \boldsymbol{c}_T\} &\sim \text{tRPM}(\boldsymbol{\alpha}, M) \text{ with } \alpha_t \overset{\text{iid}}{\sim} \text{Beta}(a_\alpha, b_\alpha)
\end{aligned} \tag{2.2}
$$

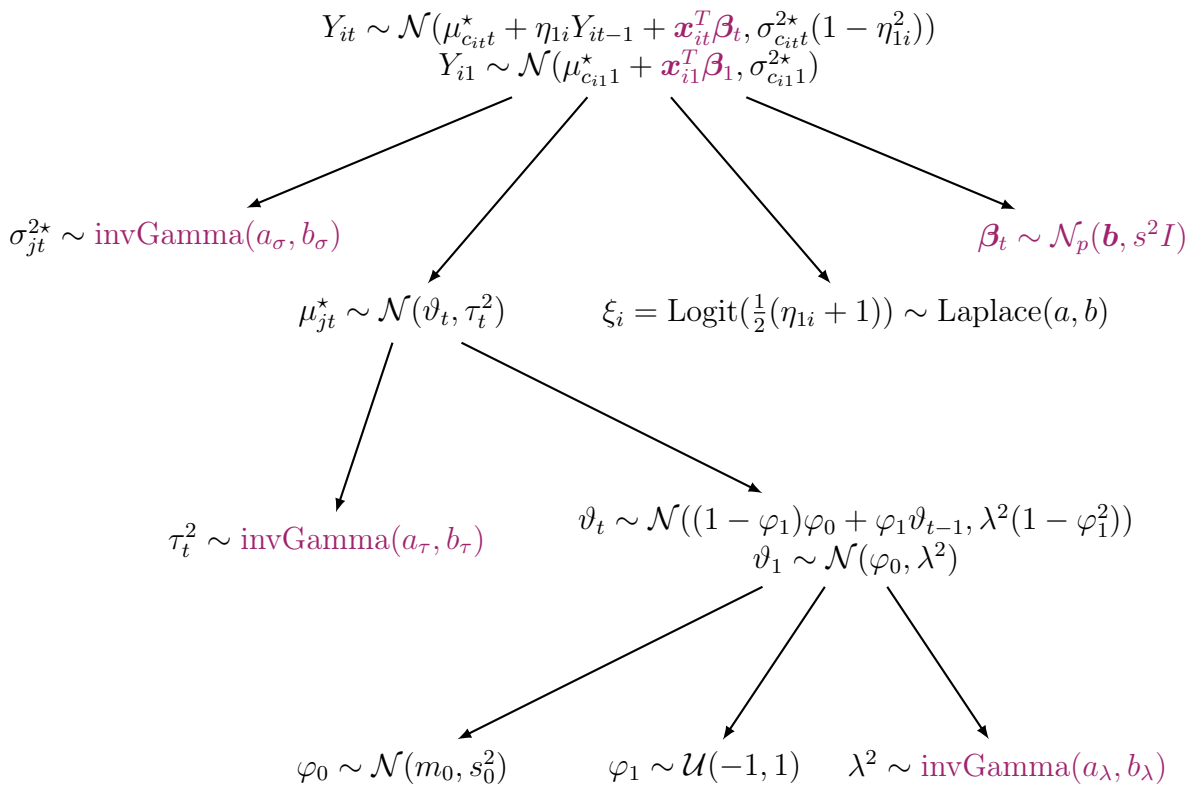


Figure 2.1: Graph visualization of the DRPM model, with highlighted in dark red the changes that we made to the original formulation.

2.1 Update rules derivation

We report here the full conditionals derivation for the parameters which had a conjugacy in the model (for the full computations see Appendix A). The other ones not included here involved instead the classical Metropolis update.

Chapter 3

Implementation and MCMC algorithm

Julia ecc [Bez+17].

One major issue encountered during the development and testing of the fitting function in Julia was controlling the amount of memory and allocations that some functions, structures, or algorithms would require. In fact, at the beginning of the development and testing, where the correctness of the algorithm was the only priority, we saw that most of the execution time was actually spent by the garbage collector of Julia, which had the burden of tracking all the allocated memory and reclaiming the unused one in order to make it available again for new computations.

The two most relevant aspects to focus on for improving performances are ensuring type stability and, most importantly, avoiding useless allocations. Luckily, Julia disposes of several tools to inspect and address these problems.

Regarding the first point, there are packages such as `Cthulhu`, or even the simple `@code_warntype` macro, which allow to ensure that a function is type-stable, i.e. that the types of all variables can be correctly predicted by the compiler and stay consistent throughout the execution. Type stability is crucial for performance as it enables the compiler to generate optimized machine code, removing the load derived from dynamic type checks. In fact, in Julia variables do not have to be necessarily declared with their type, and they could even change it during execution; however for performance purposes these dynamics should be avoided.

Regarding allocations, we relied on profiling tools such as `ProfileCanvas`. This profiler generates a plot, the *flame graph*, which depicts the different sections of code with regions whose size is proportional to the time spent on them during execution. This allows to see if time is spent on actual useful computations or instead on memory management, e.g. garbage collection; indicating therefore the sections of code which could possibly be optimized. All the modelling variables were of course preallocated, but the key has been to refactor the code to make it work more *in place*, passing directly as arguments the variables which would be modified the function, and modify them inside the function, rather than returning some values and then use them on the initial variables.

More in general, `BenchmarkTools` had been also a great ally to quickly analyse and compare even small sections of code, or single instructions, to see which solution among the possible ones that Julia provides will be more performing. For example, in this case

```
using BenchmarkTools
nh_tmp = rand(100)
@btime nclus_temp = sum(nh_tmp .> 0)
# 409.045 ns (3 allocations: 144 bytes)
@btime nclus_temp = count(x->(x>0), nh_tmp)
# 267.433 ns (0 allocations: 0 bytes)
```

or this other one

```
n = 100; rho_tmp = rand((1:5),n); k = 1
@btime findall(j -> rho_tmp[j]==k, 1:n)
# 6.860 μs (11 allocations: 608 bytes)
@btime findall_faster(j -> rho_tmp[j]==k, 1:n)
# a custom implementation of the standard findall
# 5.283 μs (2 allocations: 928 bytes)
@btime findall(rho_tmp .== k)
# 303.175 ns (4 allocations: 336 bytes)
```

we can test different versions of the equivalent instructions to decide which can be the most efficient. A quick comparison which would be instead more complex to replicate in C.

During the finer and final refinements of the code we even used the surgical `--track-allocation` option, which asks Julia to execute the code and annotate the source file to see at which lines (and of which amount) occur allocations.

3.1 Spatial cohesions optimized computation

The memory allocation problem was especially seen in the spatial cohesion computation. In fact, such computation appears twice in both sections of updating γ and updating ρ, which are inside the outer loops on draws, time, and units, and involve themselves some other loops on clusters. All this implied that those cohesion computations will be executed possibly millions of times during each fit. A simple and quick inspection suggests a value between $\Omega(dTn)$ and $O(dTn^2)$, being d the number of iterations, T the time horizon and n the number of units. We can't provide a Θ estimate due to the variability and randomicity of the inside loops, which depend on the distribution of the clusters. Considering all of this, it was crucial to optimize the performance of the cohesion functions.

Function calls in Julia are very cheap, and so, rather than rethink the algorithm to reduce the function calls, the only optimizing task consisted in carefully designing the cohesion function implementations. Cohesions 1, 2, 5 and 6 didn't exhibit any complication or need for optimization. The natural conversion in code from their mathematical models proved to be already optimally performing. The main problem was instead with cohesions 3 and 4, the auxiliary and double dippery,

which by nature would involve some linear algebra computations with vectors and matrices.

The first implementation of those cohesions turned out to be really slow, due to the overhead generated by allocating and then freeing, at each call, the memory devoted to all vectors and matrices. This in the end meant that most of the execution time was actually spent by the garbage collector, rather than in the real and useful operations. A first solution was then to resort to a scalar implementation, which would remove the overhead of the more complex algebraic structures, but in the end we managed to combine the readability of the first idea with the performance of the second one into a final version, as we can see from Listing 1.

Listing 1: Sections of the Julia code for the three implementation cases of the spatial cohesions 3 and 4. The first one has the classical vector computations derived from the mathematical formulation, the second one is the conversion to only have scalar variables, while the third one is the final version, keeping the vector form but improved using static structures.

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

# scalar-only version
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

# static improved version
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
```

This final version exploits the `StaticArrays` package of Julia, which allows to use vectors and matrices more efficiently if their size is known at compile time; which is the case of the spatial cohesions computation, since working e.g. with (planar) spatial coordinates we will have 2×1 vectors 2×2 matrices. The benefits of this final version are that we maintain the natural mathematical form of the first one, improving the clearness of the code, together with the efficiency of the second one, since now with static structures the compiler is able to optimize all memory allocations as it was doing with the simple scalar variables.

Figure 3.1 shows the comparison of their performances, where we can see how the scalar and static versions indeed perform very similarly.

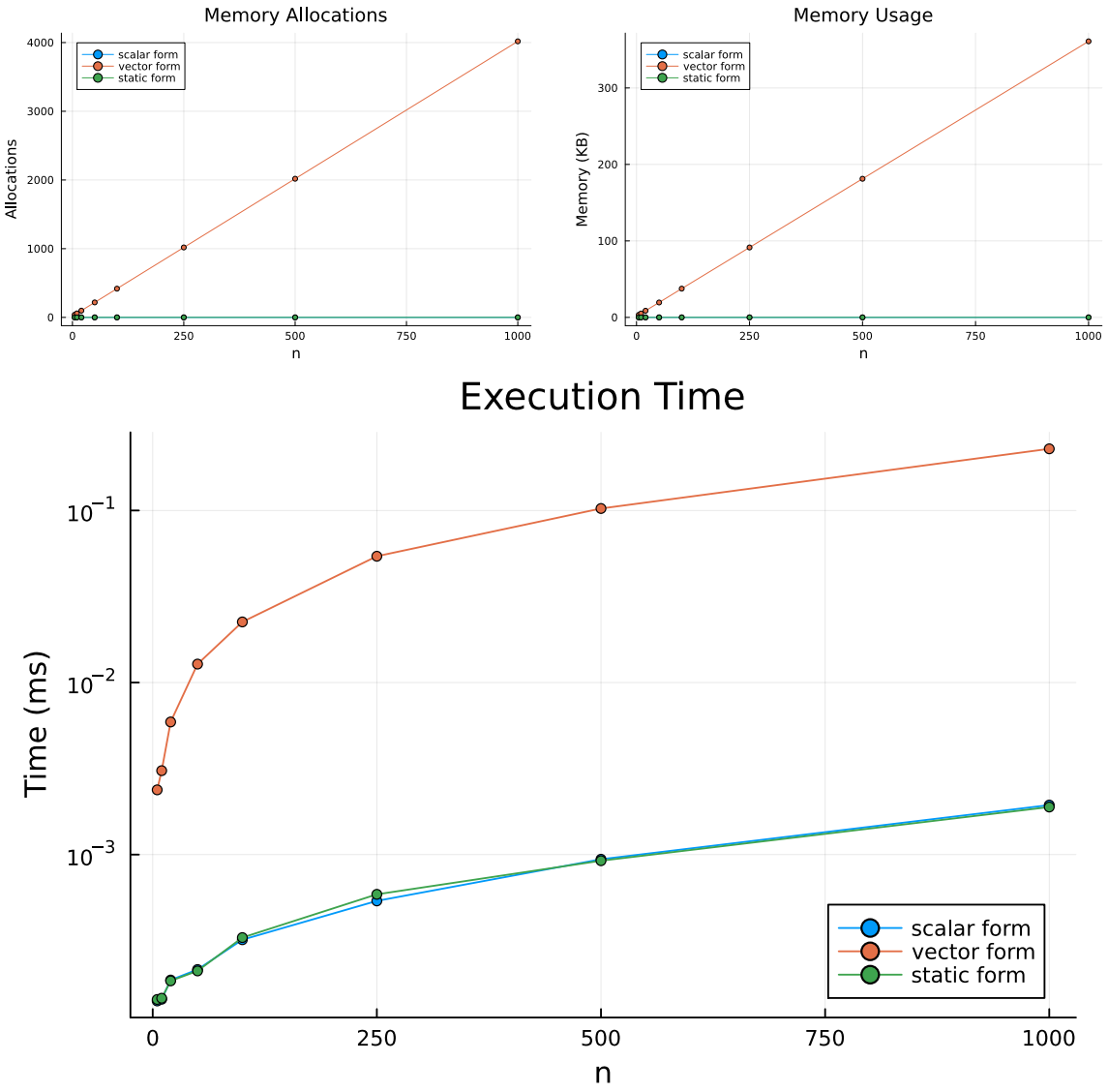


Figure 3.1: Performance comparison between the three versions of the cohesion 4 function. Tests ran through the `BenchmarkTools` package of Julia by randomly generating the spatial coordinates of the various test sizes n, with similar results standing for cohesion 3. The memory allocation and usage plots are constant at zero for both the scalar and vector static cases.

Noticeably, the C implementation of the model didn't have to worry about all this reasoning, since C can't natively nor gracefully work with vectors and matrices.

3.2 Covariates similarities optimized computation

Another problem was how to speed up the computation of the similarity functions, since those would also be called the possibly millions of times as the cohesion ones; or actually even more, considering that we could include more than one covariate

into the clustering process, so there would be an additional loop based on p, the number of covariates decided to be included.

As in the previous case, some of the functions didn't show any special need or room for relevant optimizations. The fourth one instead, the auxiliary similarity function, was essential to be optimized, because not only it is one the most frequent choice among all the similarities, but it also involves a computationally heavy sum of the squares of the covariate values, as we can see in Listing 2.

Listing 2: Final version of the similarity 4 function. The performance analysis will focus just on that inside loop, since the rest is not negotiable.

```
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
```

The idea to optimize it has been to annotate the loop with some macros provided by Julia. They are the following:

- `@inbounds` eliminates the array bounds checking within expressions. This allows to skip the checks, at the cost to guarantee, by code design, that no out-of-bounds or wrong accesses will occur in the code and therefore no undefined behaviour will take place. The above loop is indeed very simple and safe so that assumption is of course satisfied.

- `@fastmath` executes a transformed version of the expression, which calls functions that may violate strict IEEE semantics[1]. For example, its use could make $(a + b) + c \neq a + (b + c)$, but just in very pathological cases. Again, this is not a problem in our case where we are computing $\sum X_i^2$, since it does not have a "right order" in which has to be done.

- `@simd` (single instruction multiple data) annotate a for loop to allow the compiler to take extra liberties to allow loop re-ordering. This is a sort of parallelism technique, but rather than distributing the computational load on more processors we just *vectorize* the loop, i.e. we enable the CPU to

[1]Institute of Electrical and Electronics Engineers. The IEEE-754 standard defines floating-point formats, i.e. the ways to represent real numbers in hardware, and the expected behavior of arithmetic operations on them, including precision, rounding, and handling of special values (e.g. NaN (Not a Number) and infinity).

perform that single instruction (summing the square of the ith component to a reduction variable) on multiple data chunks at once, using vector registers, rather than working on each element of the vector individually.

As we can see from Figure 3.2, the actual performance difference basically derives just from the use of `@simd`, with the other two annotations making not much of a difference.

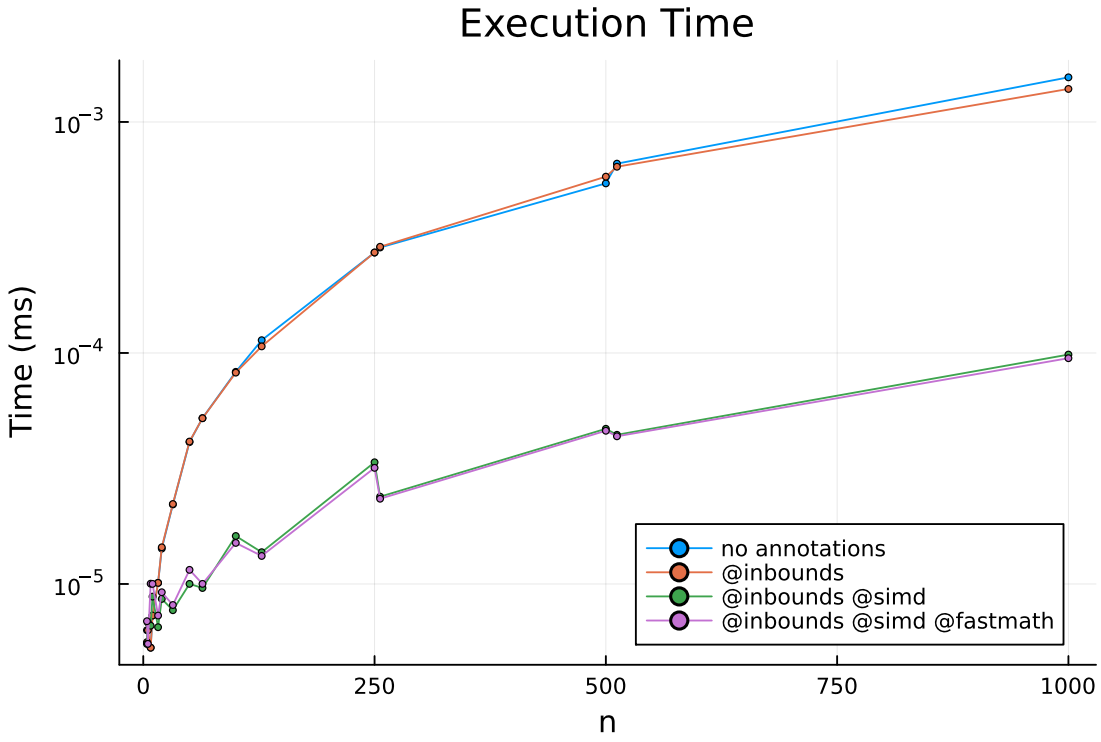


Figure 3.2: Comparison of the performances of the different possible loop annotations in the similarity 4 implementation. Their numerical output results are indeed the same for all cases. There are no memory allocation and usage plots since the analysis has been conducted only to evaluate the performances of the inside loop, which has no memory issues.

We can also notice interestingly how the tests with `@simd` annotation runs quicker in the case of n being a power of two rather than its closest rounded integer (e.g. 256 against 250 or 512 against 500), despite having more data in absolute value. This is a proof of the effectiveness of the SIMD paradigm: according to the different architectures, the CPUs can provide different register sizes (e.g. 64, 128, 256 or 512 bits) and therefore the data subdivision can fit perfectly in them when the total memory occupied by the elements is a multiple of that register size, i.e. the number of data values is a power of two. Otherwise there will be some "leftovers chunks", which will of course be processed, but will also cause, as a consequence of the imperfect fit, a bit of overhead.

Chapter 4

Testing

4.1 Assessing the equivalence of the models

Our model, and the corresponding Julia code, is just an improvement of the original one of Page with his relative C implementation. These improvements, as described in the previous chapters, refer to the insertion of covariates, both at the clustering and likelihood levels, the handling of missing values in the target variable, and the computational efficiency. In this sense, our updates are just add-ons to the original model, and therefore at a common testing level they should perform similarly by agreeing in the clusters that they produce on a given dataset.

To assess this ideally equivalent behaviour we ran two tests: the first one with only the target values, using generated data, while the second with also spatial information, using a real spatio-temporal dataset.

4.1.1 Target variable only

For the first test, we generated a dataset of $n = 10$ units and $T = 12$ time instants, and fitted both models collecting 1000 iterates derived from 10000 total iterations, discarding the first 5000 as burnin, and thinning by 5. Those parameters, as well as the generating function, were the same of [PQD22]. The generating function allowed to create data points with temporal dependence, which could be tuned trough some parameters such as the dispersion or the temporal dependence.

Table 4.1: Summary of the comparison between the two fits. The MSE columns refer to the accuracy between the fitted values generated by the models (estimated by taking the mean and the median of the returned 1000 iterates) and the true values of the target variable. Higher LPML and lower WAIC indicate a better fit.

	MSE (mean)	MSE (median)	LPML	WAIC	execution time
CDRPM	1.673099	1.586079	-249.61	469.69	5.326 s
JDRPM	1.432986	1.401098	-229.20	420.02	2.568 s

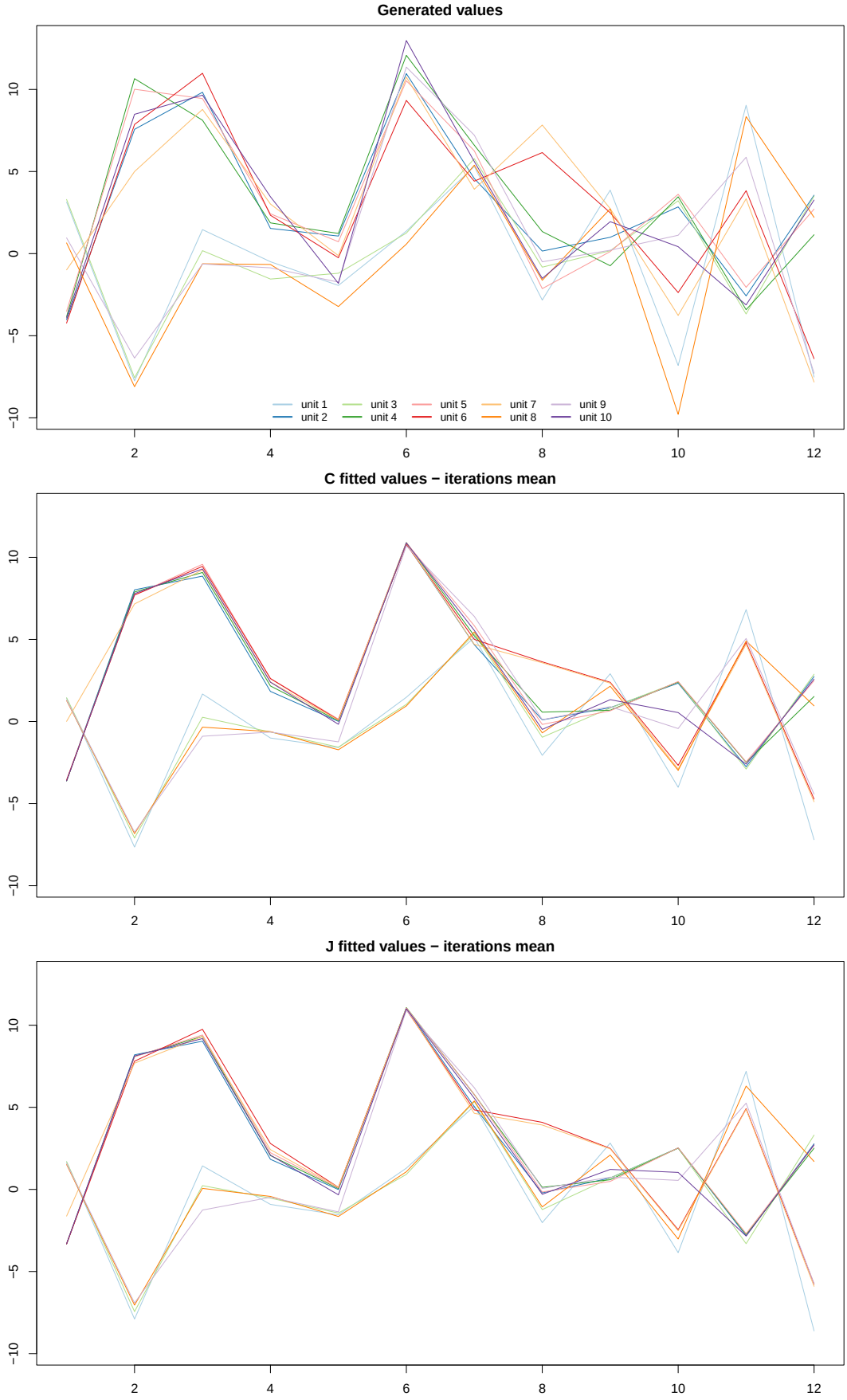


Figure 4.1: Generated target values (top), together with their fitted values estimate trough the mean of the 1000 iterates generated by model CDRPM (middle) and JDRPM (bottom).

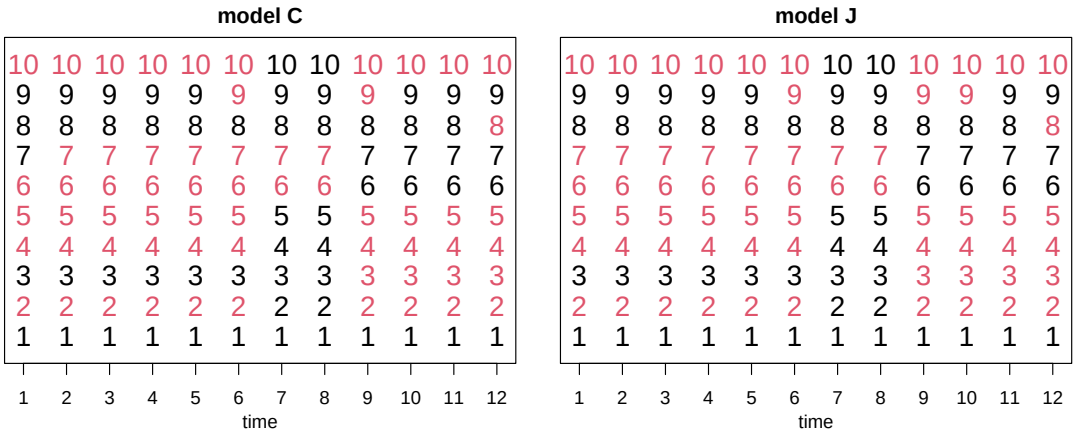


Figure 4.2: Clustering produced by the two models, with time points on the x axis, units indicated vertically by their ID number, and colors representing the cluster label.

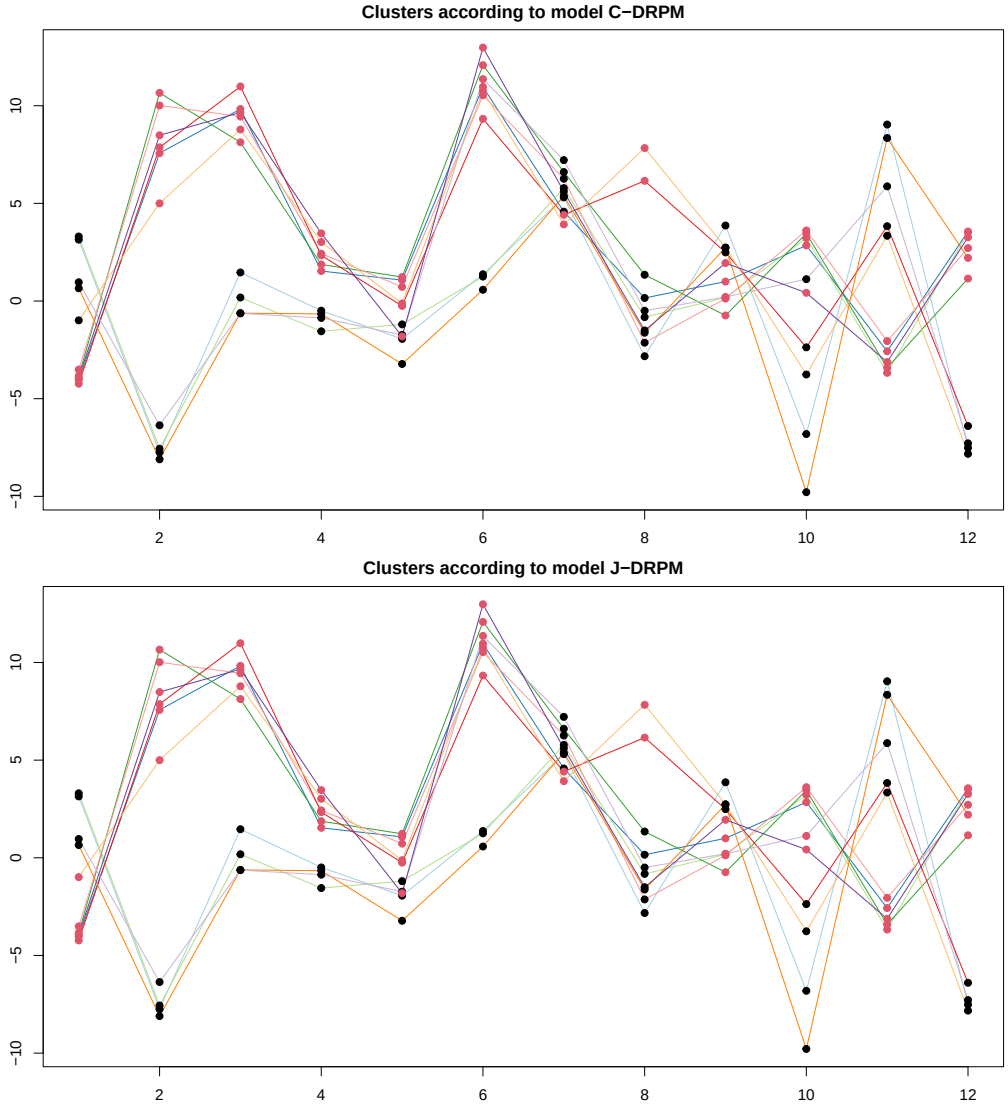


Figure 4.3: Visual representation of the clusters produced by the models, with units' labels represented as colored dots overlaid to the trend of the original generated target variables. The only differences occur at times 1 and 10.

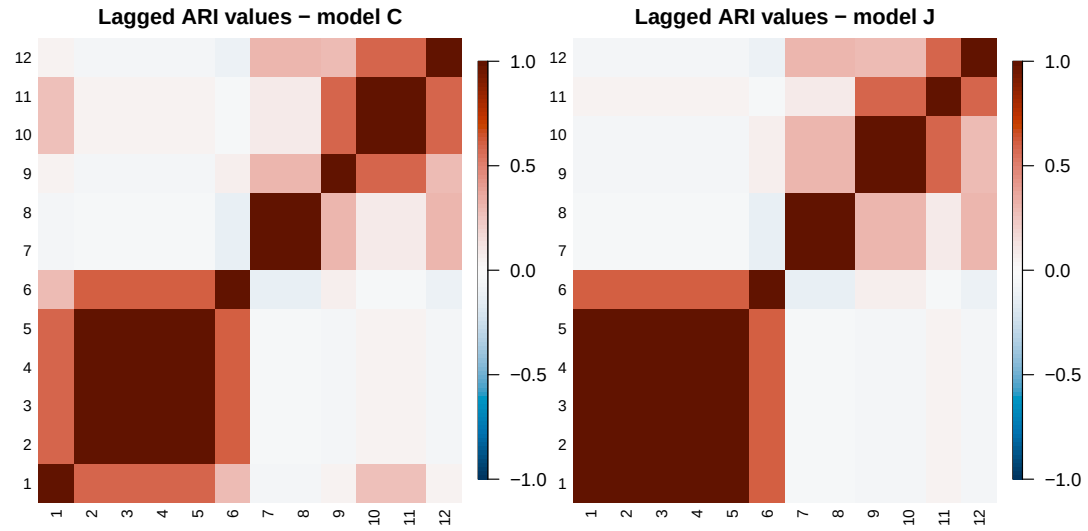


Figure 4.4: Lagged ARI values for the two models. The partitions were estimated using the `salso` function from the corresponding R library.

4.1.2 Target variable plus space

4.2 Performance with missing values

We repeat the tests of the previous section now randomly introducing some NA into the data, to see how the model reacts to them and can still perform well. Exploring some other spatio-temporal dataset, we set the percentage of missing values to 10%.

4.2.1 Target variable only

4.2.2 Target variable plus space

4.3 Effects of the covariates

4.4 Testing on larger time horizons

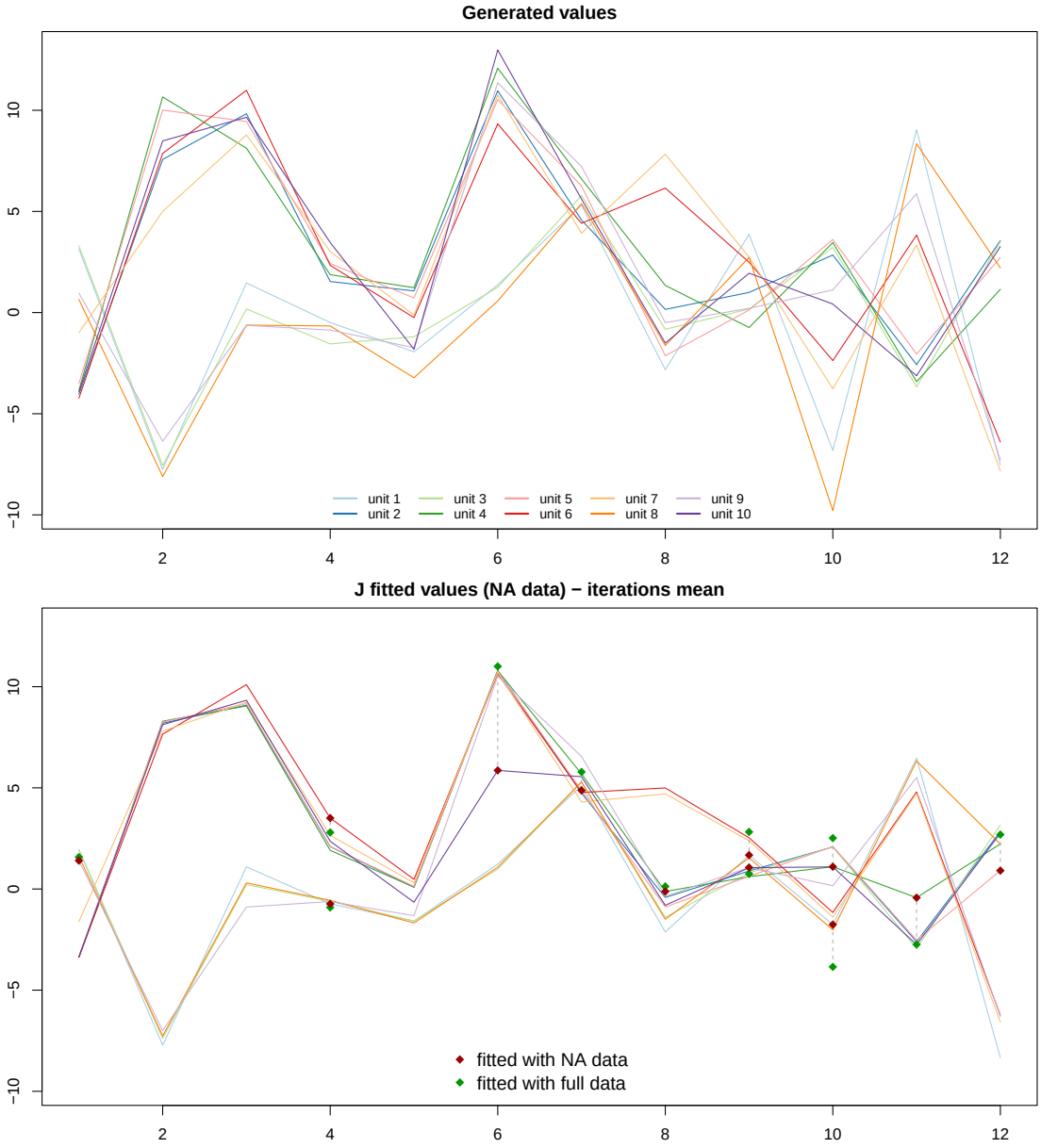


Figure 4.5: Generated target values (top), together with their fitted values estimate trough the mean of the 1000 iterates generated by model JDRPM (bottom). Special point markers are devoted to the data points corresponding to missing values, to highlight the gaps between the fit on the full dataset and the fit on the NA dataset.

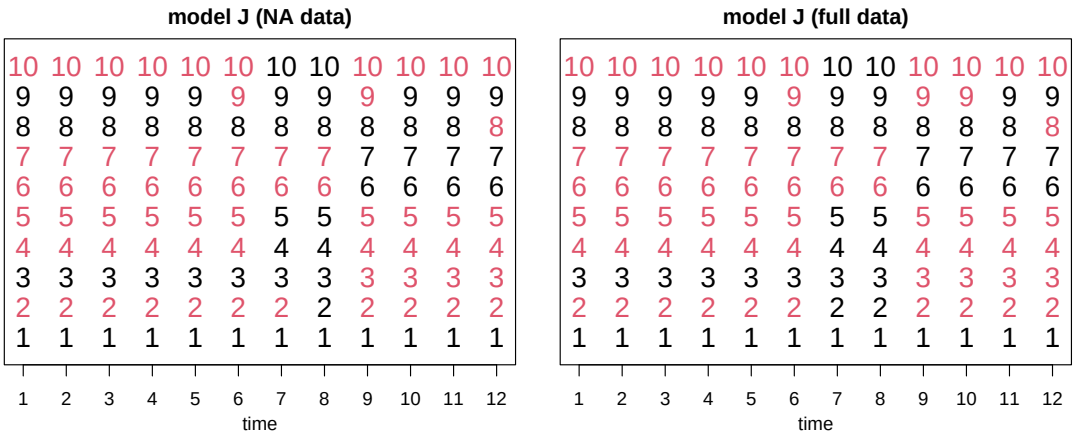


Figure 4.6: Clustering produced by the two tests on the JDRPM model, with time points on the x axis, units indicated vertically by their ID number, and colors representing the cluster label.

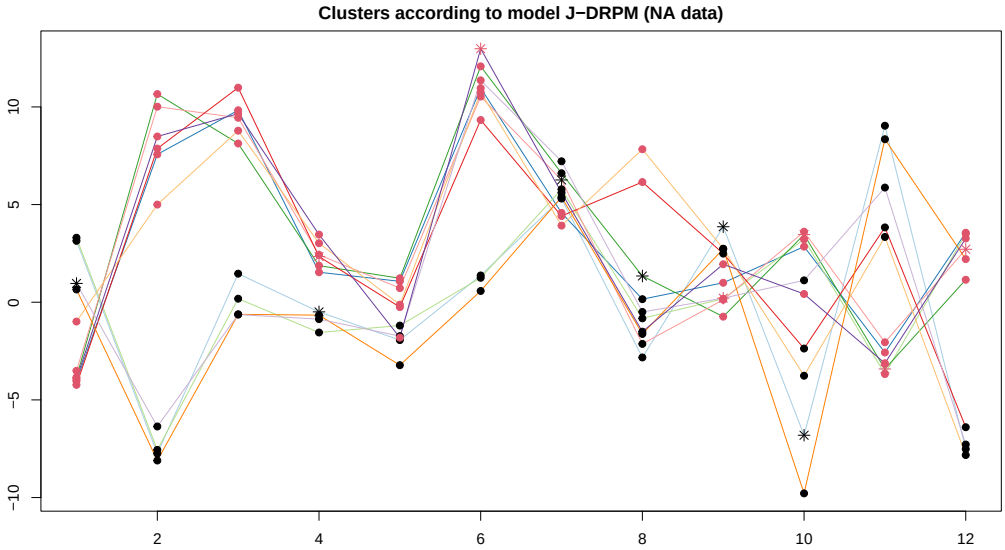


Figure 4.7: Visual representation of the clusters produced by the model, with units' labels represented as colored dots overlaid to the trend of the original generated target variables.

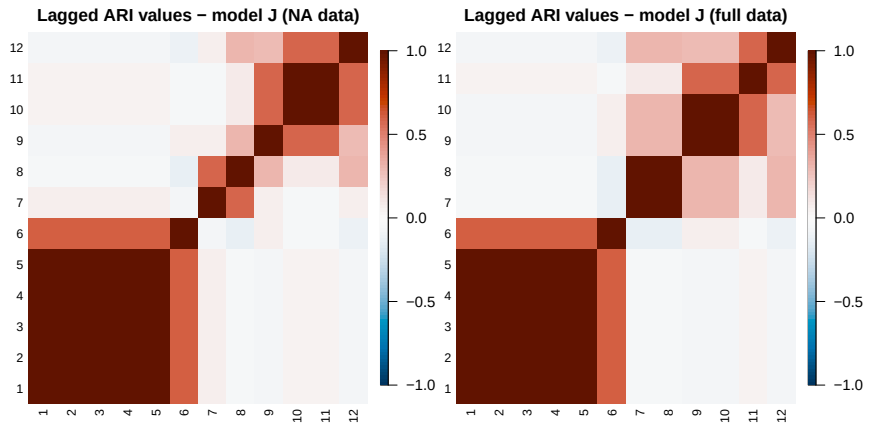


Figure 4.8: Lagged ARI values for the two tests on the JDRPM model. The partitions were estimated using the `salso` function from the corresponding R library.

Chapter 5

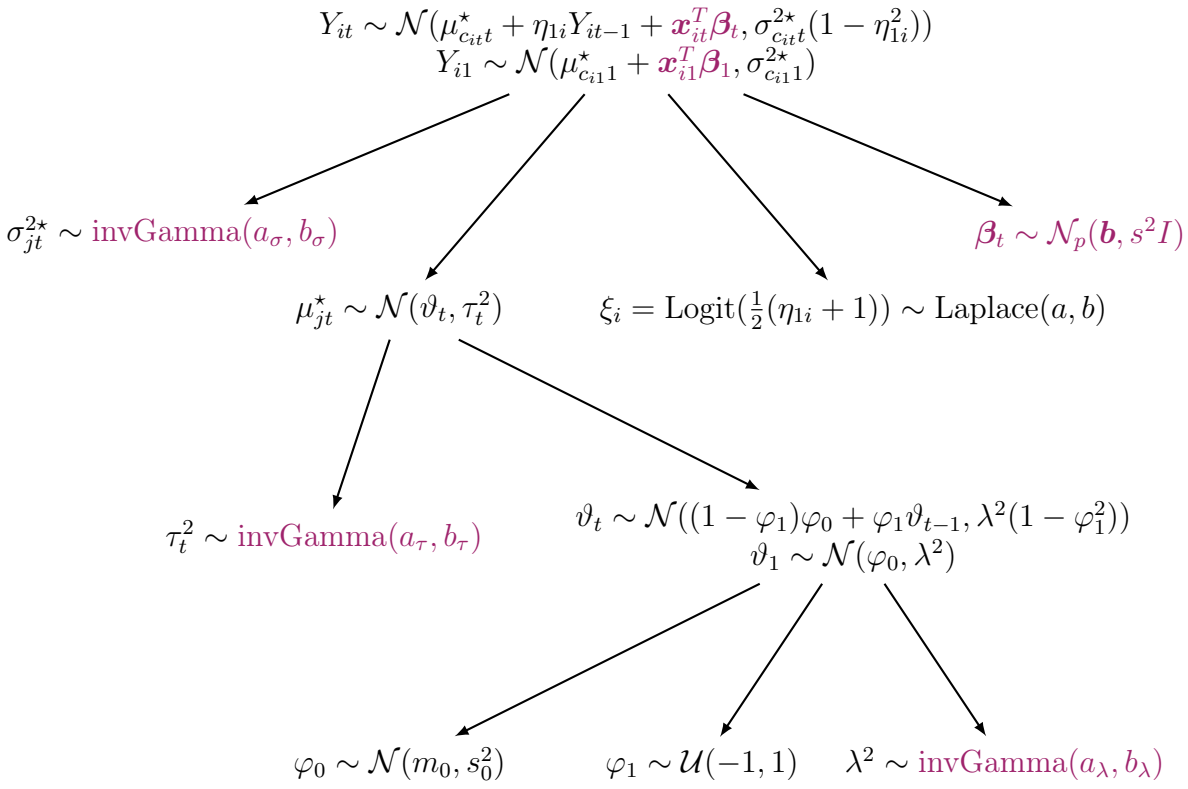
Conclusion

Appendix A

Theoretical details

A.1 Extended computations of the full conditionals

We propose here the extended computations which allowed to extract the full conditionals presented in Chapter 2. We report also the model graph to make it quickly accessible as a reference for the laws involved in the following computations.



- update $\sigma^{2\star}_{jt}$

  for $t = 1$:
  $f(\sigma^{2\star}_{jt}|-) \propto f(\sigma^{2\star}_{jt}) f(\{Y_{it} : c_{it} = j\}|\sigma^{2\star}_{jt}, -)$

$$
\begin{aligned}
&= \mathcal{L}_{\text{invGamma}(a_\sigma,b_\sigma)}(\sigma^{2\star}_{jt}) \prod_{i\in S_{jt}} \mathcal{L}_{\mathcal{N}(\mu^\star_{c_{it}t}+\boldsymbol{x}_{it}^T\boldsymbol{\beta}_t,\sigma^{2\star}_{c_{it}1})}(Y_{it}) \\
&\propto \left[\left(\frac{1}{\sigma^{2\star}_{jt}}\right)^{a_\sigma+1} \exp\left\{-\frac{1}{\sigma^{2\star}_{jt}}b\right\}\right] \\
&\cdot \left[\prod_{i\in S_{jt}} \left(\frac{1}{\sigma^{2\star}_{jt}}\right)^{1/2} \exp\left\{-\frac{1}{2\sigma^{2\star}_{jt}}(Y_{it}-\mu^\star_{jt}-\boldsymbol{x}_{it}^T\boldsymbol{\beta}_t)^2\right\}\right] \\
&\propto \left(\frac{1}{\sigma^{2\star}_{jt}}\right)^{(a_\sigma+|S_{jt}|/2)+1} \exp\left\{-\frac{1}{\sigma^{2\star}_{jt}}\left(b_\sigma+\frac{1}{2}\sum_{i\in S_{jt}}(Y_{it}-\mu^\star_{jt}-\boldsymbol{x}_{it}^T\boldsymbol{\beta}_t)^2\right)\right\}
\end{aligned}
$$

$$
\begin{aligned}
\implies f(\sigma^{2\star}_{jt}|-) &\propto \text{kernel of a invGamma}(a_{\sigma(\text{post})}, b_{\sigma(\text{post})}) \text{ with} \\
a_{\tau(\text{post})} &= a_\sigma + \frac{|S_{jt}|}{2} \\
b_{\tau(\text{post})} &= b_\sigma + \frac{1}{2}\sum_{i\in S_{jt}}(Y_{it}-\mu^\star_{jt}-\boldsymbol{x}_{it}^T\boldsymbol{\beta}_t)^2
\end{aligned} \tag{A.1}
$$

for $t > 1$:

$$
\begin{aligned}
f(\sigma^{2\star}_{jt}|-) &\propto f(\sigma^{2\star}_{jt}) f(\{Y_{it} : c_{it} = j\}|\sigma^{2\star}_{jt}, -) \\
&= \mathcal{L}_{\text{invGamma}(a_\sigma,b_\sigma)}(\sigma^{2\star}_{jt}) \prod_{i\in S_{jt}} \mathcal{L}_{\mathcal{N}(\mu^\star_{c_{it}t}+\eta_{1i}Y_{it-1}+\boldsymbol{x}_{it}^T\boldsymbol{\beta}_t,\sigma^{2\star}_{c_{it}1})}(Y_{it}) \\
&\propto \left[\left(\frac{1}{\sigma^{2\star}_{jt}}\right)^{a_\sigma+1} \exp\left\{-\frac{1}{\sigma^{2\star}_{jt}}b\right\}\right] \\
&\cdot \left[\prod_{i\in S_{jt}} \left(\frac{1}{\sigma^{2\star}_{jt}}\right)^{1/2} \exp\left\{-\frac{1}{2\sigma^{2\star}_{jt}}(Y_{it}-\mu^\star_{jt}-\eta_{1i}Y_{it-1}-\boldsymbol{x}_{it}^T\boldsymbol{\beta}_t)^2\right\}\right] \\
&\propto \left(\frac{1}{\sigma^{2\star}_{jt}}\right)^{(a_\sigma+|S_{jt}|/2)+1} \\
&\cdot \exp\left\{-\frac{1}{\sigma^{2\star}_{jt}}\left(b_\sigma+\frac{1}{2}\sum_{i\in S_{jt}}(Y_{it}-\eta_{1i}Y_{it-1}-\mu^\star_{jt}-\boldsymbol{x}_{it}^T\boldsymbol{\beta}_t)^2\right)\right\}
\end{aligned}
$$

$$
\begin{aligned}
\implies f(\sigma^{2\star}_{jt}|-) &\propto \text{kernel of a invGamma}(a_{\sigma(\text{post})}, b_{\sigma(\text{post})}) \text{ with} \\
a_{\tau(\text{post})} &= a_\sigma + \frac{|S_{jt}|}{2} \\
b_{\tau(\text{post})} &= b_\sigma + \frac{1}{2}\sum_{i\in S_{jt}}(Y_{it}-\mu^\star_{jt}-\eta_{1i}Y_{it-1}-\boldsymbol{x}_{it}^T\boldsymbol{\beta}_t)^2
\end{aligned} \tag{A.2}
$$

- update $\mu^\star_{jt}$

for $t = 1$:

$$
f(\mu^\star_{jt}|-) \propto f(\mu^\star_{jt}) f(\{Y_{it} : c_{it} = j\}|\mu^\star_{jt}, -)
$$

$$
\begin{aligned}
&= \mathcal{L}_{\mathcal{N}(\vartheta_t,\tau_t^2)}(\mu_{jt}^\star) \prod_{i\in S_{jt}} \mathcal{L}_{\mathcal{N}(\mu_{jt}^\star,\sigma_{jt}^{2\star})}(Y_{it}) \\
&\propto \exp\left\{-\frac{1}{2\tau_t^2}(\mu_{jt}^\star-\vartheta_t)^2\right\} \exp\left\{-\frac{1}{2\sigma_{jt}^{2\star}}\left(\sum_{i\in S_{jt}}(\mu_{jt}^\star - Y_{i1})^2\right)\right\} \\
&\propto \exp\left\{-\frac{1}{2\tau_t^2}(\mu_{jt}^\star-\vartheta_t)^2\right\} \exp\left\{-\frac{|S_{jt}|}{2\sigma_{jt}^{2\star}}\left(\mu_{jt}^\star - \frac{\text{SUM}_y}{|S_{jt}|}\right)^2\right\} \\
&\text{where } \text{SUM}_y = \sum_{i\in S_{jt}} Y_{i1}
\end{aligned}
$$

$$
\begin{aligned}
&\implies f(\mu_{jt}^\star|-) \propto \text{kernel of a } \mathcal{N}(\mu_{\mu_{jt}^\star(\text{post})}, \sigma^2_{\mu_{jt}^\star(\text{post})}) \text{ with} \\
&\sigma^2_{\mu_{jt}^\star(\text{post})} = \frac{1}{\frac{1}{\tau_t^2}+\frac{|S_{jt}|}{\sigma_{jt}^{2\star}}} \\
&\mu_{\mu_{jt}^\star(\text{post})} = \sigma^2_{\mu_{jt}^\star(\text{post})}\left(\frac{\vartheta_t}{\tau_t^2}+\frac{\text{SUM}_y}{\sigma_{jt}^{2\star}}\right)
\end{aligned} \tag{A.3}
$$

for $t > 1$:

$$
\begin{aligned}
f(\mu_{jt}^\star|-) &\propto f(\mu_{jt}^\star) f(\{Y_{it} : c_{it} = j\}|\mu_{jt}^\star, -) \\
&= \mathcal{L}_{\mathcal{N}(\vartheta_1,\tau_t^2)}(\mu_{jt}^\star) \prod_{i\in S_{jt}} \mathcal{L}_{\mathcal{N}(\mu_{jt}^\star+\eta_{1i}Y_{i,t-1},\sigma_{jt}^{2\star}(1-\eta_{1i}^2))}(Y_{it}) \\
&\propto \exp\left\{-\frac{1}{2\tau_t^2}(\mu_{jt}^\star-\vartheta_t)^2\right\} \\
&\cdot \exp\left\{-\frac{1}{2\sigma_{jt}^{2\star}}\left(\sum_{i\in S_{jt}}\frac{1}{1-\eta_{i1}^2}\left(\mu_{jt}^\star - (Y_{it}-\eta_{1i}Y_{i,t-1})\right)^2\right)\right\} \\
&\propto \exp\left\{-\frac{1}{2\tau_t^2}(\mu_{jt}^\star-\vartheta_t)^2\right\} \exp\left\{-\frac{\text{SUM}_{e2}}{2\sigma_{jt}^{2\star}}\left(\mu_{jt}^\star - \frac{\text{SUM}_y}{\text{SUM}_{e2}}\right)^2\right\} \\
&\text{where } \text{SUM}_y = \sum_{i\in S_{jt}} \frac{Y_{it}-\eta_{1i}Y_{i,t-1}}{1-\eta_{1i}^2},\ \text{SUM}_{e2} = \sum_{i\in S_{jt}} \frac{1}{1-\eta_{1i}^2}
\end{aligned}
$$

$$
\begin{aligned}
&\implies f(\mu_{jt}^\star|-) \propto \text{kernel of a } \mathcal{N}(\mu_{\mu_{jt}^\star(\text{post})}, \sigma^2_{\mu_{jt}^\star(\text{post})}) \text{ with} \\
&\sigma^2_{\mu_{jt}^\star(\text{post})} = \frac{1}{\frac{1}{\tau_t^2}+\frac{\text{SUM}_{e2}}{\sigma_{jt}^{2\star}}} \\
&\mu_{\mu_{jt}^\star(\text{post})} = \sigma^2_{\mu_{jt}^\star(\text{post})}\left(\frac{\vartheta_t}{\tau_t^2}+\frac{\text{SUM}_y}{\sigma_{jt}^{2\star}}\right)
\end{aligned} \tag{A.4}
$$

- update $\boldsymbol{\beta}_t$

for $t = 1$:

$$
f(\boldsymbol{\beta}_t|-) \propto f(\boldsymbol{\beta}_t) f(\{Y_{1t}, \ldots, Y_{nt}\}|\boldsymbol{\beta}_t, -)
$$

$$
\begin{aligned}
&= \mathcal{L}_{\mathcal{N}(\boldsymbol{b},s^2I)}(\boldsymbol{\beta}_t) \prod_{i=1}^{n} \mathcal{L}_{\mathcal{N}(\mu^\star_{c_{it}t}+\boldsymbol{x}_{it}^T\boldsymbol{\beta}_t,\sigma^{2\star}_{c_{it}t})}(Y_{it}) \\
&\propto \exp\left\{-\frac{1}{2}\left[(\boldsymbol{\beta}_t^T-\boldsymbol{b})^T\frac{1}{s^2}(\boldsymbol{\beta}_t-\boldsymbol{b})\right]\right\} \exp\left\{-\frac{1}{2}\sum_{i=1}^{n}(Y_{it}-\mu^\star_{c_{it}t}-\boldsymbol{x}_{it}^T\boldsymbol{\beta}_t)^2\right\} \\
&\propto \exp\left\{-\frac{1}{2}\left[\boldsymbol{\beta}_t^T\left(\frac{1}{s^2}I+\sum_{i=1}^{n}\frac{\boldsymbol{x}_{it}\boldsymbol{x}_{it}^T}{\sigma^{2\star}_{c_{it}t}}\right)\boldsymbol{\beta}_t\right.\right. \\
&\quad \left.\left.-\ 2\left(\frac{\boldsymbol{b}}{s^2}+\sum_{i=1}^{n}\frac{(Y_{it}-\mu^\star_{c_{it}t})\boldsymbol{x}_{it}}{\sigma^{2\star}_{c_{it}t}}\right)\cdot\boldsymbol{\beta}_t\right]\right\} \\
\implies\ & f(\boldsymbol{\beta}_t|-) \propto \text{kernel of a } \mathcal{N}(\boldsymbol{b}_{(\text{post})}, A_{(\text{post})}) \text{ with} \\
& A_{(\text{post})} = \left(\frac{1}{s^2}I+\sum_{i=1}^{n}\frac{\boldsymbol{x}_{it}\boldsymbol{x}_{it}^T}{\sigma^{2\star}_{c_{it}t}}\right)^{-1} \\
& \boldsymbol{b}_{(\text{post})} = A_{(\text{post})}\left(\frac{\boldsymbol{b}}{s^2}+\sum_{i=1}^{n}\frac{(Y_{it}-\mu^\star_{c_{it}t})\boldsymbol{x}_{it}}{\sigma^{2\star}_{c_{it}t}}\right) \\
\iff\ & f(\boldsymbol{\beta}_t|-) \propto \text{kernel of a } \mathcal{N}\text{Canon}(\boldsymbol{h}_{(\text{post})}, J_{(\text{post})}) \text{ with} \\
& \boldsymbol{h}_{(\text{post})} = \left(\frac{\boldsymbol{b}}{s^2}+\sum_{i=1}^{n}\frac{(Y_{it}-\mu^\star_{c_{it}t})\boldsymbol{x}_{it}}{\sigma^{2\star}_{c_{it}t}}\right) \\
& J_{(\text{post})} = \left(\frac{1}{s^2}I+\sum_{i=1}^{n}\frac{\boldsymbol{x}_{it}\boldsymbol{x}_{it}^T}{\sigma^{2\star}_{c_{it}t}}\right)
\end{aligned}
\tag{A.5}
$$

for $t > 1$:

$$
\begin{aligned}
f(\boldsymbol{\beta}_t|-) &\propto f(\boldsymbol{\beta}_t) f(\{Y_{1t},\ldots,Y_{nt}\}|\boldsymbol{\beta}_t,-) \\
&= \mathcal{L}_{\mathcal{N}(\boldsymbol{b},s^2I)}(\boldsymbol{\beta}_t) \prod_{i=1}^{n} \mathcal{L}_{\mathcal{N}(\mu^\star_{c_{it}t}+\eta_{1i}Y_{it-1}+\boldsymbol{x}_{it}^T\boldsymbol{\beta}_t,\sigma^{2\star}_{c_{it}t})}(Y_{it}) \\
&\propto \exp\left\{-\frac{1}{2}\left[(\boldsymbol{\beta}_t^T-\boldsymbol{b})^T\frac{1}{s^2}(\boldsymbol{\beta}_t-\boldsymbol{b})\right]\right\} \\
&\quad \cdot \exp\left\{-\frac{1}{2}\sum_{i=1}^{n}(Y_{it}-\mu^\star_{c_{it}t}-\eta_{1i}Y_{it-1}-\boldsymbol{x}_{it}^T\boldsymbol{\beta}_t)^2\right\} \\
&\propto \exp\left\{-\frac{1}{2}\left[\boldsymbol{\beta}_t^T\left(\frac{1}{s^2}I+\sum_{i=1}^{n}\frac{\boldsymbol{x}_{it}\boldsymbol{x}_{it}^T}{\sigma^{2\star}_{c_{it}t}}\right)\boldsymbol{\beta}_t\right.\right. \\
&\quad \left.\left.-2\left(\frac{\boldsymbol{b}}{s^2}+\sum_{i=1}^{n}\frac{(Y_{it}-\mu^\star_{c_{it}t}-\eta_{1i}Y_{it-1})\boldsymbol{x}_{it}}{\sigma^{2\star}_{c_{it}t}}\right)\cdot\boldsymbol{\beta}_t\right]\right\} \\
\implies\ & f(\boldsymbol{\beta}_t|-) \propto \text{kernel of a } \mathcal{N}(\boldsymbol{b}_{(\text{post})}, A_{(\text{post})}) \text{ with} \\
& A_{(\text{post})} = \left(\frac{1}{s^2}I+\sum_{i=1}^{n}\frac{\boldsymbol{x}_{it}\boldsymbol{x}_{it}^T}{\sigma^{2\star}_{c_{it}t}}\right)^{-1}
\end{aligned}
$$

$$
\begin{aligned}
\boldsymbol{b}_{(\text{post})} &= A_{(\text{post})} \left(\frac{\boldsymbol{b}}{s^2} + \sum_{i=1}^{n} \frac{(Y_{it} - \mu^{\star}_{c_{it}t} - \eta_{1i} Y_{it-1}) \boldsymbol{x}_{it}}{\sigma^{2\star}_{c_{it}t}} \right) \\
\iff f(\boldsymbol{\beta}_t|-) &\propto \text{kernel of a } \mathcal{N}\text{Canon}(\boldsymbol{h}_{(\text{post})}, J_{(\text{post})}) \text{ with} \\
\boldsymbol{h}_{(\text{post})} &= \left(\frac{\boldsymbol{b}}{s^2} + \sum_{i=1}^{n} \frac{(Y_{it} - \mu^{\star}_{c_{it}t} - \eta_{1i} Y_{it-1}) \boldsymbol{x}_{it}}{\sigma^{2\star}_{c_{it}t}} \right) \\
J_{(\text{post})} &= \left(\frac{1}{s^2} I + \sum_{i=1}^{n} \frac{\boldsymbol{x}_{it} \boldsymbol{x}_{it}^T}{\sigma^{2\star}_{c_{it}t}} \right)
\end{aligned} \tag{A.6}
$$

- update τ_t^2

$$
\begin{aligned}
f(\tau_t^2|-) &\propto f(\tau_t^2) f((\mu^{\star}_{1t}, \ldots, \mu^{\star}_{k_t t})|\tau_t^2, -) \\
&= \mathcal{L}_{\text{invGamma}(a_\tau, b_\tau)}(\tau_t^2) \prod_{j=1}^{k_t} \mathcal{L}_{\mathcal{N}(\vartheta_t, \tau_t^2)}(\mu^{\star}_{jt}) \\
&\propto \left[\left(\frac{1}{\tau_t^2} \right)^{a_\tau + 1} \exp\left\{ -\frac{b_\tau}{\tau_t^2} \right\} \right] \left[\prod_{j=1}^{k_t} \left(\frac{1}{\tau_t^2} \right)^{1/2} \exp\left\{ -\frac{1}{2\tau_t^2} (\mu^{\star}_{jt} - \vartheta_t)^2 \right\} \right] \\
&\propto \left(\frac{1}{\tau_t^2} \right)^{\left(\frac{k_t}{2} + a_\tau\right) + 1} \exp\left\{ -\frac{1}{\tau_t^2} \left(\frac{\sum_{j=1}^{k_t} (\mu^{\star}_{jt} - \vartheta_t)^2}{2} + b_\tau \right) \right\} \\
\implies f(\tau_t^2|-) &\propto \text{kernel of a invGamma}(a_{\tau(\text{post})}, b_{\tau(\text{post})}) \text{ with} \\
a_{\tau(\text{post})} &= \frac{k_t}{2} + a_\tau \\
b_{\tau(\text{post})} &= \frac{\sum_{j=1}^{k_t} (\mu^{\star}_{jt} - \vartheta_t)^2}{2} + b_\tau
\end{aligned} \tag{A.7}
$$

- update ϑ_t

for $t = T$:

$$
\begin{aligned}
f(\vartheta_t|-) &\propto f(\vartheta_t) f((\mu^{\star}_{1t}, \ldots, \mu^{\star}_{k_t t})|\vartheta_t, -) \\
&= \mathcal{L}_{\mathcal{N}((1-\varphi_1)\varphi_0 + \varphi_1 \vartheta_{t-1}, \lambda^2(1-\varphi_1^2))}(\vartheta_t) \prod_{j=1}^{k_t} \mathcal{L}_{\mathcal{N}(\vartheta_t, \tau_t^2)}(\mu^{\star}_{jt}) \\
&\propto \exp\left\{ -\frac{1}{2(\lambda^2(1-\varphi_1^2))} \Big(\vartheta_t - ((1-\varphi_1)\varphi_0 + \varphi_1 \vartheta_{t-1}) \Big)^2 \right\} \\
&\cdot \exp\left\{ -\frac{k_t}{2\tau_t^2} \left(\vartheta_t - \frac{\sum_{j=1}^{k_t} \mu^{\star}_{jt}}{k_t} \right) \right\} \\
\implies f(\vartheta_t|-) &\propto \text{kernel of a } \mathcal{N}(\mu_{\vartheta_t(\text{post})}, \sigma^2_{\vartheta_t(\text{post})}) \text{ with} \\
\sigma^2_{\vartheta_t(\text{post})} &= \frac{1}{\frac{1}{\lambda^2(1-\varphi_1^2)} + \frac{k_t}{\tau_t^2}} \\
\mu_{\vartheta_t(\text{post})} &= \sigma^2_{\vartheta_t(\text{post})} \left(\frac{\sum_{j=1}^{k_t} \mu^{\star}_{jt}}{\tau_t^2} + \frac{(1-\varphi_1)\varphi_0 + \varphi_1 \vartheta_{t-1}}{\lambda^2(1-\varphi_1^2)} \right)
\end{aligned} \tag{A.8}
$$

for $1 < t < T$:

$$
\begin{aligned}
f(\vartheta_t|-) &\propto \underbrace{f(\vartheta_t) f((\mu^\star_{1t}, \dots, \mu^\star_{k_t t})|\vartheta_t, -)}_{\text{as in the case } t = T} f(\vartheta_{t+1}|\vartheta_t, -) \\
&= \mathcal{L}_{\mathcal{N}(\mu_{\vartheta_t(\text{post})}, \sigma^2_{\vartheta_t(\text{post})})}(\vartheta_t) \mathcal{L}_{\mathcal{N}((1-\varphi_1)\varphi_0 + \varphi_1\vartheta_t, \lambda^2(1-\varphi_1^2))}(\vartheta_{t+1}) \\
&\propto \exp\left\{ -\frac{1}{2\sigma^2_{\vartheta_t(\text{post})}} \left(\vartheta_t - \mu_{\vartheta_t(\text{post})}\right)^2 \right\} \\
&\quad \cdot \exp\left\{ -\frac{1}{2\frac{\lambda^2(1-\varphi_1^2)}{\varphi_1^2}} \left(\vartheta_t - \frac{\vartheta_{t+1} - (1-\varphi_1)\varphi_0}{\varphi_1} \right)^2 \right\} \\
\implies f(\vartheta_t|-) &\propto \text{kernel of a } \mathcal{N}(\mu_{\vartheta_t(\text{post})}, \sigma^2_{\vartheta_t(\text{post})}) \text{ with} \\
\sigma^2_{\vartheta_t(\text{post})} &= \frac{1}{\frac{1+\varphi_1^2}{\lambda^2(1-\varphi_1^2)} + \frac{k_t}{\tau_t^2}} \\
\mu_{\vartheta_t(\text{post})} &= \sigma^2_{\vartheta_t(\text{post})} \left(\frac{\sum_{j=1}^{k_t} \mu^\star_{jt}}{\tau_t^2} + \frac{\varphi_1(\vartheta_{t-1} + \vartheta_{t+1}) + \varphi_0(1-\varphi_1)^2}{\lambda^2(1-\varphi_1^2)} \right) \qquad \text{(A.9)}
\end{aligned}
$$

for $t = 1$:

$$
\begin{aligned}
f(\vartheta_t|-) &\propto f(\vartheta_t) f(\vartheta_{t+1}|\vartheta_t, -) f(\boldsymbol{\mu}^\star_t|\vartheta_t, -) \\
&= \mathcal{L}_{\mathcal{N}(\varphi_0, \lambda^2)}(\vartheta_t) \mathcal{L}_{\mathcal{N}((1-\varphi_1)\varphi_0 + \varphi_1\vartheta_{t-1}, \lambda^2(1-\varphi_1^2))}(\vartheta_{t+1}) \prod_{j=1}^{k_t} \mathcal{L}_{\mathcal{N}(\vartheta_t, \tau_t^2)}(\mu^\star_{jt}) \\
&\propto \exp\left\{ -\frac{1}{2\lambda^2} (\vartheta_t - \varphi_0)^2 \right\} \\
&\quad \cdot \exp\left\{ -\frac{1}{2\frac{\lambda^2(1-\varphi_1^2)}{\varphi_1^2}} \left(\vartheta_t - \frac{\vartheta_{t+1} - (1-\varphi_1)\varphi_0}{\varphi_1} \right)^2 \right\} \\
&\quad \cdot \exp\left\{ -\frac{k_t}{2\tau_t^2} \left(\vartheta_t - \frac{\sum_{j=1}^{k_t} \mu^\star_{jt}}{k_t} \right) \right\} \\
\implies f(\vartheta_t|-) &\propto \text{kernel of a } \mathcal{N}(\mu_{\vartheta_t(\text{post})}, \sigma^2_{\vartheta_t(\text{post})}) \text{ with} \\
\sigma^2_{\vartheta_t(\text{post})} &= \frac{1}{\frac{1}{\lambda^2} + \frac{\varphi_1^2}{\lambda^2(1-\varphi_1^2)} + \frac{k_t}{\tau_t^2}} \\
\mu_{\vartheta_t(\text{post})} &= \sigma^2_{\vartheta_t(\text{post})} \left(\frac{\varphi_0}{\lambda^2} + \frac{\varphi_1(\vartheta_{t+1} - (1-\varphi_1)\varphi_0)}{\lambda^2(1-\varphi_1^2)} + \frac{\sum_{j=1}^{k_t} \mu^\star_{jt}}{\tau_t^2} \right) \qquad \text{(A.10)}
\end{aligned}
$$

- update φ_0

$$
\begin{aligned}
f(\varphi_0|-) &\propto f(\varphi_0) f((\vartheta_1, \dots, \vartheta_T)|\varphi_0, -) \\
&= \mathcal{L}_{\mathcal{N}(m_0, s_0^2)}(\varphi_0) \mathcal{L}_{\mathcal{N}(\varphi_0, \lambda^2)}(\vartheta_1) \prod_{t=2}^{T} \mathcal{L}_{\mathcal{N}((1-\varphi_1)\varphi_0 + \varphi_1\vartheta_{t-1}, \lambda^2(1-\varphi_1^2))}(\vartheta_t) \\
&\propto \exp\left\{ -\frac{1}{2s_0^2} (\varphi_0 - m_o)^2 \right\} \exp\left\{ -\frac{1}{2\lambda^2} (\varphi_0 - \vartheta_1)^2 \right\}
\end{aligned}
$$

$$\cdot \exp\left\{-\frac{1}{2\frac{\lambda^2(1-\varphi_1^2)}{(T-1)(1-\varphi_1)^2}}\left(\varphi_0 - \frac{(1-\varphi_1)(\text{SUM}_t)}{(T-1)(1-\varphi_1)^2}\right)^2\right\}$$

$$\text{where SUM}_t = \sum_{t=2}^{T}(\vartheta_t - \varphi_1\vartheta_{t-1})$$

$$\implies f(\varphi_0|-) \propto \text{kernel of a } \mathcal{N}(\mu_{\varphi_0(\text{post})}, \sigma^2_{\varphi_0(\text{post})}) \text{ with}$$

$$\sigma^2_{\varphi_0(\text{post})} = \frac{1}{\frac{1}{s_0^2} + \frac{1}{\lambda^2} + \frac{(T-1)(1-\varphi_1)^2}{\lambda^2(1-\varphi_1^2)}}$$

$$\mu_{\varphi_0(\text{post})} = \sigma^2_{\varphi_0(\text{post})}\left(\frac{m_0}{s_0^2} + \frac{\vartheta_1}{\lambda^2} + \frac{1-\varphi_1}{\lambda^2(1-\varphi_1^2)}\text{SUM}_t\right) \tag{A.11}$$

- update λ^2

$$\begin{aligned}
f(\lambda^2|-) &\propto f(\lambda^2)f(\vartheta_1,\ldots,\vartheta_T|\lambda^2,-) \\
&= \mathcal{L}_{\text{invGamma}(a_\lambda,b_\lambda)}(\lambda^2)\mathcal{L}_{\mathcal{N}(\varphi_0,\lambda^2)}(\vartheta_1)\prod_{t=2}^{T}\mathcal{L}_{\mathcal{N}((1-\varphi_1)\varphi_0+\varphi_1\vartheta_{t-1},\lambda^2(1-\varphi_1^2))}(\vartheta_t) \\
&\propto \left[\left(\frac{1}{\lambda_t^2}\right)^{a_\lambda+1}\exp\left\{-\frac{b_\lambda}{\lambda^2}\right\}\right]\left[\left(\frac{1}{\lambda^2}\right)^{1/2}\exp\left\{-\frac{1}{2\lambda^2}(\vartheta_1-\varphi_0)^2\right\}\right] \\
&\cdot \left[\prod_{t=2}^{T}\left(\frac{1}{\lambda^2}\right)^{1/2}\exp\left\{-\frac{1}{2\lambda^2}\left(\vartheta_t-(1-\varphi_1)\varphi_0-\varphi_1\vartheta_{t-1}\right)^2\right\}\right] \\
&\propto \left(\frac{1}{\lambda^2}\right)^{\left(\frac{T}{2}+a_\lambda\right)+1} \\
&\cdot \exp\left\{-\frac{1}{\lambda^2}\left(\frac{(\vartheta_1-\varphi_0)^2}{2} + \sum_{t=2}^{T}\frac{\left(\vartheta_t-(1-\varphi_1)\varphi_0-\varphi_1\vartheta_{t-1}\right)^2}{2} + b_\lambda\right)\right\}
\end{aligned}$$

$$\implies f(\lambda^2|-) \propto \text{kernel of a invGamma}(a_{\lambda(\text{post})}, b_{\lambda(\text{post})}) \text{ with}$$

$$a_{\lambda(\text{post})} = \frac{T}{2} + a_\lambda$$

$$b_{\lambda(\text{post})} = \frac{(\vartheta_1-\varphi_0)^2}{2} + \sum_{t=2}^{T}\frac{\left(\vartheta_t-(1-\varphi_1)\varphi_0-\varphi_1\vartheta_{t-1}\right)^2}{2} + b_\lambda \tag{A.12}$$

- a missing observation Y_{it}

for $t = 1$:

$$\begin{aligned}
f(Y_{it}|-) &\propto f(Y_{it})f(Y_{it+1}|Y_{it},-) \\
&= \mathcal{L}_{\mathcal{N}(\mu^\star_{c_{it}t}+\boldsymbol{x}_{it}^T\boldsymbol{\beta}_t,\sigma^{2\star}_{c_{it}t})}(Y_{it}) \\
&\cdot \mathcal{L}_{\mathcal{N}(\mu^\star_{c_{it+1}t+1}+\eta_{1i}Y_{it}+\boldsymbol{x}_{it+1}^T\boldsymbol{\beta}_{t+1},\sigma^{2\star}_{c_{it+1}t+1}(1-\eta_{1i}^2))}(Y_{it+1}) \\
&\propto \exp\left\{-\frac{1}{2\sigma^{2\star}_{c_{it}t}}(Y_{it}-\mu^\star_{c_{it}t}-\boldsymbol{x}_{it}^T\boldsymbol{\beta}_t)^2\right\} \\
&\cdot \exp\left\{-\frac{1}{2\sigma^{2\star}_{c_{it+1}t+1}(1-\eta_{1i}^2)}(Y_{it+1}-\mu^\star_{c_{it+1}t+1}-\eta_{1i}Y_{it}-\boldsymbol{x}_{it+1}^T\boldsymbol{\beta}_{t+1})^2\right\}
\end{aligned}$$

$$
\begin{aligned}
&= \exp\left\{-\frac{1}{2\sigma^{2\star}_{c_{it}t}}\left(Y_{it} - (\mu^{\star}_{c_{it}t} + \boldsymbol{x}_{it}^T\boldsymbol{\beta}_t)\right)^2\right\} \\
&\cdot \exp\left\{-\frac{\eta_{1i}^2}{2\sigma^{2\star}_{c_{it+1}t+1}(1-\eta_{1i}^2)}\left(Y_{it} - \frac{Y_{it+1} - \mu^{\star}_{c_{it+1}t+1} - \boldsymbol{x}_{it+1}^T\boldsymbol{\beta}_{t+1}}{\eta_{1i}}\right)^2\right\} \\
\implies & f(Y_{it}|-) \propto \text{kernel of a } \mathcal{N}(\mu_{Y_{it}(\text{post})}, \sigma^2_{Y_{it}(\text{post})}) \text{ with} \\
\sigma^2_{Y_{it}(\text{post})} &= \frac{1}{\frac{1}{\sigma^{2\star}_{c_{it}t}} + \frac{\eta_{1i}^2}{2\sigma^{2\star}_{c_{it+1}t+1}(1-\eta_{1i}^2)}} \\
\mu_{Y_{it}(\text{post})} &= \sigma^2_{Y_{it}(\text{post})}\left(\frac{\mu^{\star}_{c_{it}t} + \boldsymbol{x}_{it}^T\boldsymbol{\beta}_t}{\sigma^{2\star}_{c_{it}t}} + \frac{\eta_{1i}(Y_{it+1} - \mu^{\star}_{c_{it+1}t+1} - \boldsymbol{x}_{it+1}^T\boldsymbol{\beta}_{t+1})}{\sigma^{2\star}_{c_{it+1}t+1}(1-\eta_{1i}^2)}\right)
\end{aligned}
\tag{A.13}
$$

$$
\begin{aligned}
&\text{for } 1 < t < T\text{:} \\
&f(Y_{it}|-) \propto f(Y_{it})f(Y_{it+1}|Y_{it},-) \\
&\quad = \mathcal{L}_{\mathcal{N}(\mu^{\star}_{c_{it}t} + \eta_{1i}Y_{it-1} + \boldsymbol{x}_{it}^T\boldsymbol{\beta}_t, \sigma^{2\star}_{c_{it}t})}(Y_{it}) \\
&\quad \cdot \mathcal{L}_{\mathcal{N}(\mu^{\star}_{c_{it+1}t+1} + \eta_{1i}Y_{it} + \boldsymbol{x}_{it+1}^T\boldsymbol{\beta}_{t+1}, \sigma^{2\star}_{c_{it+1}t+1}(1-\eta_{1i}^2))}(Y_{it+1}) \\
&\quad \propto \exp\left\{-\frac{1}{2\sigma^{2\star}_{c_{it}t}(1-\eta_{1i}^2)}(Y_{it} - \mu^{\star}_{c_{it}t} - \eta_{1i}Y_{it-1} - \boldsymbol{x}_{it}^T\boldsymbol{\beta}_t)^2\right\} \\
&\quad \cdot \exp\left\{-\frac{1}{2\sigma^{2\star}_{c_{it+1}t+1}(1-\eta_{1i}^2)}(Y_{it+1} - \mu^{\star}_{c_{it+1}t+1} - \eta_{1i}Y_{it} - \boldsymbol{x}_{it+1}^T\boldsymbol{\beta}_{t+1})^2\right\} \\
&\quad = \exp\left\{-\frac{1}{2\sigma^{2\star}_{c_{it}t}(1-\eta_{1i}^2)}\left(Y_{it} - (\mu^{\star}_{c_{it}t} + \eta_{1i}Y_{it-1} + \boldsymbol{x}_{it}^T\boldsymbol{\beta}_t)\right)^2\right\} \\
&\quad \cdot \exp\left\{-\frac{\eta_{1i}^2}{2\sigma^{2\star}_{c_{it+1}t+1}(1-\eta_{1i}^2)}\left(Y_{it} - \frac{Y_{it+1} - \mu^{\star}_{c_{it+1}t+1} - \boldsymbol{x}_{it+1}^T\boldsymbol{\beta}_{t+1}}{\eta_{1i}}\right)^2\right\} \\
&\implies f(Y_{it}|-) \propto \text{kernel of a } \mathcal{N}(\mu_{Y_{it}(\text{post})}, \sigma^2_{Y_{it}(\text{post})}) \text{ with} \\
&\sigma^2_{Y_{it}(\text{post})} = \frac{1-\eta_{1i}^2}{\frac{1}{\sigma^{2\star}_{c_{it}t}} + \frac{\eta_{1i}^2}{\sigma^{2\star}_{c_{it+1}t+1}}} \\
&\mu_{Y_{it}(\text{post})} = \sigma^2_{Y_{it}(\text{post})}\left(\frac{\mu^{\star}_{c_{it}t} + \eta_{1i}Y_{it-1} + \boldsymbol{x}_{it}^T\boldsymbol{\beta}_t}{\sigma^{2\star}_{c_{it}t}(1-\eta_{1i}^2)} + \frac{\eta_{1i}(Y_{it+1} - \mu^{\star}_{c_{it+1}t+1} - \boldsymbol{x}_{it+1}^T\boldsymbol{\beta}_{t+1})}{\sigma^{2\star}_{c_{it+1}t+1}(1-\eta_{1i}^2)}\right)
\end{aligned}
\tag{A.14}
$$

$$
\begin{aligned}
&\text{for } t = T\text{:} \\
&f(Y_{it}|-) \propto f(Y_{it}) \\
&\quad = \mathcal{L}_{\mathcal{N}(\mu^{\star}_{c_{it}t} + \eta_{1i}Y_{it-1} + \boldsymbol{x}_{it}^T\boldsymbol{\beta}_t, \sigma^{2\star}_{c_{it}t}(1-\eta_{1i}^2))}(Y_{it}) \\
&\implies f(Y_{it}|-) \text{ is just the likelihood of } Y_{it}
\end{aligned}
\tag{A.15}
$$

Appendix B

Computational details

B.1 Fitting algorithm code

We report here the full code of the `MCMC_fit` function, which implements the updated DRPM model described in this work. We include it to let the readers appreciate the ease, clarity and even elegance of the Julia language in translating the mathematical formulation into code. Especially compared to the original C implementation, Julia is not only more readable and efficient, being still a compiled language, but also faster in the testing and development phase, thanks to the possibility of her interactive use trough the Julia `REPL`, similar to the one of the R console. Moreover, Julia's extensive ecosystem provides users flexibility and all the supports they could need, thanks to the vast documentation or even an online active forum[1] (where I wrote myself some questions during the development of this work). For example, packages like `Distributions` [Bes+21] [Lin+19] and `Statistics` came in very handy for this work; while the C implementation had to write all the statistical functionalities from scratch.

All this with the the hope for Julia to become the new natural choice in the statistical, or in general scientific, computing field, giving to it a refreshing and tidier approach.

B.2 Interface

Now some more technical details about the whole implementation design. The fitting code was written in Julia, but its intended use is from R. This choice was made because R is currently the best language for statistical purposes; in fact all the other models "competitors" to the DRPM are available trough some R package, and therefore we thought that letting our work accessible from R would have eased the possibilities of testing and comparisons with other models or datasets.

To do so, we relied on the `JuliaConnectoR` library [LHB22] on R, which allows the interaction between the two languages. In this sense, we can load in R the Julia

[1] `https://discourse.julialang.org/`

package `JDRPM`, which stores the improved version of the DRPM model, and call it using data and arguments from R. Then the output produced by the Julia function has then to be converted back into R structures, and this can be done trough an appropriate function.

Regarding instead the "visual" interface, in the sense of the feedback provided by the function, we decided to make it more user-friendly, and possibly informative, than the original C code. The most relevant add-on is probably the progress monitoring, indicating and updating in real-time the estimated remaining time to complete the fit (i.e. the ETA, estimated time of arrival). As a further proof of the ease of this language, this feature was a simple insertion of few lines of code thanks to the Julia package `ProgressMeter`.

Listing 3: Feedback from the Julia implementation of the updated DRPM.

```
- using seed 314.0 -
fitting 10000 total iterates (with burnin=2000, thinning=8)
thus producing 1000 valid iterates in the end

on n=10 subjects
for T=12 time instants

with space? false
with covariates in the likelihood? false
with covariates in the clustering process? false
are there missing data in Y? false

Starting MCMC algorithm
Progress: 100% Time: 0:00:02 ( 0.24 ms/it)

done!
Elapsed time: 2 seconds, 369 milliseconds
LPML: -239.00521650726978 (the higher the better)
WAIC: 422.206629519383 (the lower the better)
acceptance ratio eta1: 56.26%
acceptance ratio phi1: 81.70%
```

Appendix C

Further plots

Bibliography

[Bez+17] Jeff Bezanson et al. "Julia: A fresh approach to numerical computing". In: *SIAM Review* 59.1 (2017), pp. 65–98. DOI: 10.1137/141000671. URL: https://epubs.siam.org/doi/10.1137/141000671 (cit. on p. 5).

[PQD22] Garrit L. Page, Fernando A. Quintana, and David B. Dahl. "Dependent Modeling of Temporal Sequences of Random Partitions". In: *Journal of Computational and Graphical Statistics* 31.2 (2022), pp. 614–627. DOI: 10.1080/10618600.2021.1987255. eprint: https://doi.org/10.1080/10618600.2021.1987255. URL: https://doi.org/10.1080/10618600.2021.1987255 (cit. on p. 11).

[Bes+21] Mathieu Besançon et al. "Distributions.jl: Definition and Modeling of Probability Distributions in the JuliaStats Ecosystem". In: *Journal of Statistical Software* 98.16 (2021), pp. 1–30. ISSN: 1548-7660. DOI: 10.18637/jss.v098.i16. URL: https://www.jstatsoft.org/v098/i16 (cit. on p. 27).

[Lin+19] Dahua Lin et al. *JuliaStats/Distributions.jl: a Julia package for probability distributions and associated functions*. July 2019. DOI: 10.5281/zenodo.2647458. URL: https://doi.org/10.5281/zenodo.2647458 (cit. on p. 27).

[LHB22] Stefan Lenz, Maren Hackenberg, and Harald Binder. "The JuliaConnectoR: A Functionally-Oriented Interface for Integrating Julia in R". In: *Journal of Statistical Software* 101.6 (2022), pp. 1–24. DOI: 10.18637/jss.v101.i06 (cit. on p. 27).

Acknowledgements

Ringraziamenti

"Ecco il mio ponte d'approdo per molti lunghi anni, il mio angoletto, nel quale faccio il mio ingresso con una sensazione così diffidente, così morbosa... Ma chi lo sa? Forse, quando tra molti anni mi toccherà abbandonarlo, magari potrei anche rimpiangerlo!"

— Fëdor Dostoevskij, *Memorie da una casa di morti*