Back to business

recap dopo la ”pausa” estiva

Politecnico of Milano
Thesis development

September 10, 2024

Come sta il modello?

$$Y_{it} \sim \mathcal{N}(\mu^{\star}_{c_{it}t} + \eta_{1i} Y_{it-1} + \boldsymbol{x}_{it}^T \beta_t, \sigma^{2\star}_{c_{it}t}(1-\eta_{1i}^2))$$
$$Y_{i1} \sim \mathcal{N}(\mu^{\star}_{c_{i1}1} + \boldsymbol{x}_{i1}^T \beta_1, \sigma^{2\star}_{c_{i1}1})$$

$$\sigma^{\star}_{jt} \sim \text{invGamma}(a_\sigma, b_\sigma)$$

$$\beta_t \sim \mathcal{N}_p(\boldsymbol{b}, \kappa^2 I)$$

$$\mu^{\star}_{jt} \sim \mathcal{N}(\theta_t, \tau_t^2)$$

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

$$\tau_t^2 \sim \text{invGamma}(a_\tau, b_\tau)$$

$$\theta_t \sim \mathcal{N}((1-\phi_1)\phi_0 + \phi_1\theta_{t-1}, \lambda^2(1-\phi_1^2))$$
$$\theta_1 \sim \mathcal{N}(\phi_0, \lambda^2)$$

$$\phi_0 \sim \mathcal{N}(m_0, s_0^2)$$

$$\phi_1 \sim \mathcal{U}(-1, 1)$$

$$\lambda^2 \sim \text{invGamma}(a_\lambda, b_\lambda)$$

Come sta la tesi?

Ho impostato un modello iniziale su latex per la tesi, e pensato ad un'idea della divisione in capitoli.

1 Introduction

2 Description of the model

3 Implementation and MCMC algorithm

4 Testing on air pollution data

5 Conclusion

A Computational details

B Algorithm codes

C Further plots

forse meglio fare magari due capitoli, uno dedicato alla teoria/statistica e l'altro all'implementazione/informatica

- Description of the model (teorico)

- Computational details and implementation (pratico)

Funzione di fit in Julia

Potenzialmente finita la funzione di fit su Julia.

- è chiamabile comodamente da R, e infatti su R ho già fatto alcuni test con vari dataset.
- sembra essere più veloce di quella di Page in C, nei casi di fit con solo target o con target + spazio. Con anche le covariate è invece più lenta (inevitabilmente, sono molti calcoli in più).
- le covariate sono gestibili separatamente tra quali includere nella likelihood e quali includere nel processo di clustering (per il problema della non identificabilità, credo).

Argomenti

I parametri con `missing` è perché non sono obbligatori, quindi di default "non ci sono". Questo per permettere di fittare con varie combinazioni di target, spazio, covariate likelihood, covariate clustering.

```
function MCMC_fit(;
    Y::Matrix{Float64},                 # n*T matrix, the observed values
    sp_coords = missing,                # n*2 matrix, the spatial coordinates
    Xlk_covariates = missing,           # n*p*T matrix, the covariates to include in the likelihood
    Xcl_covariates = missing,           # n*p*T matrix, the covariates to include in the clustering process

    M_dp::Float64,                      # Dirichlet mass parameter
    initial_partition = missing,        # Initial partition (if provided)
    # actually not implemented yet the version with the initial partition (which however i think is useless)
    # però vabe dovrebbe essere facile da includere

    starting_alpha::Float64,            # Starting value for alpha
    unit_specific_alpha::Bool,          # Unit-specific alpha values
    time_specific_alpha::Bool,          # Time-specific alpha values
    update_alpha::Bool,                 # Update alpha?

    include_eta1::Bool,                 # Include the autoregressive part of eta1?
    include_phi1::Bool,                 # Include the autoregressive part of phi1?
    update_eta1::Bool,                  # Update the autoregressive part of eta1?
    update_phi1::Bool,                  # Update the autoregressive part of phi1?
```

Argomenti

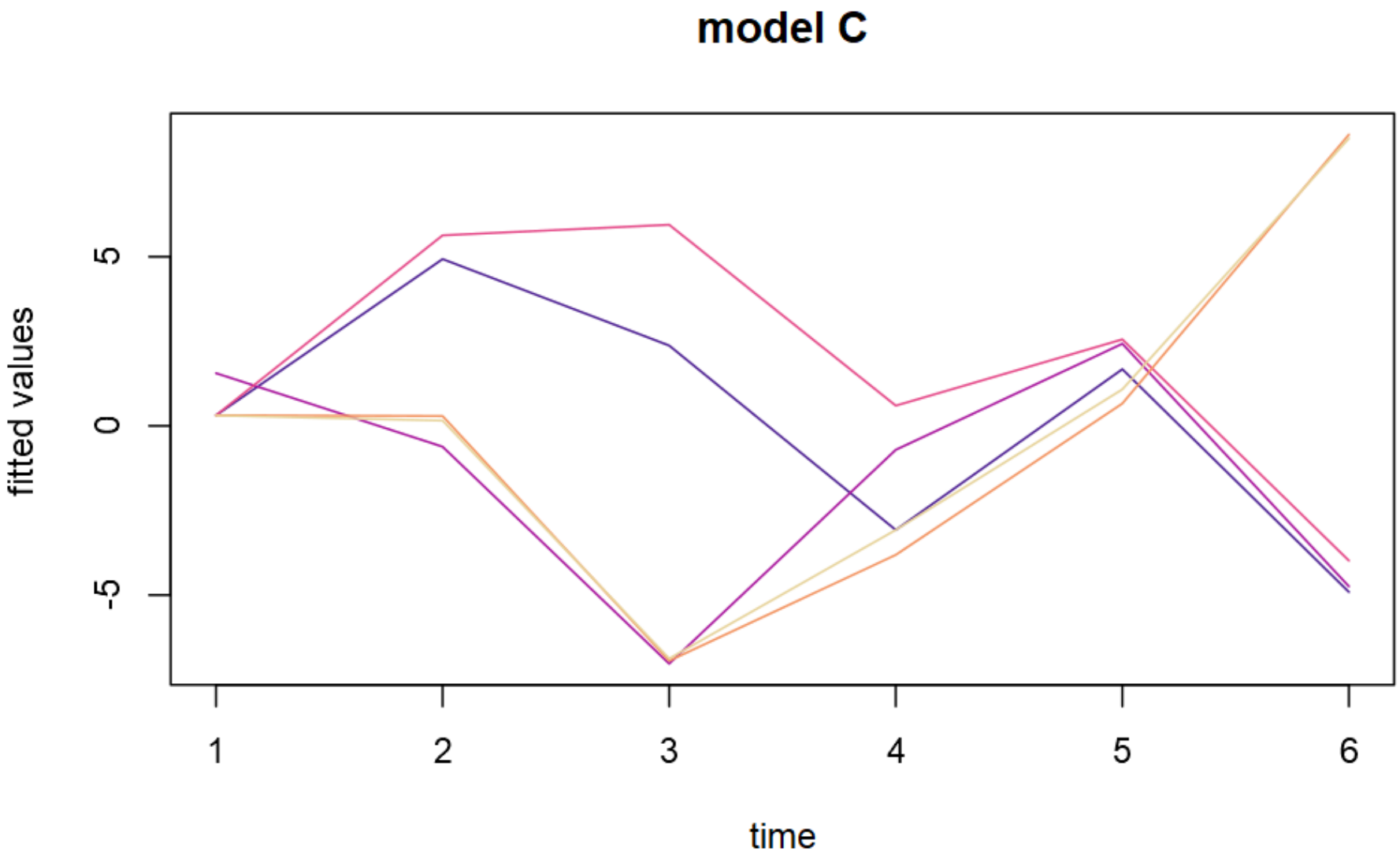
```
sig2h_priors::Vector{Float64},          # Prior parameters for sig2h ~ invGamma(a_sigma,b_sigma)
eta1_priors::Vector{Float64},           # Prior parameters for eta1 ~ Laplace(0,b) so it's the scale parameter b
                                        # plus the std dev for the Metropolis update trough N(eta1_old,mhsig_eta1^2)
beta_priors = missing,                  # Prior parameters for beta ~
tau2_priors::Vector{Float64},           # Prior parameters for tau2 ~ invGamma(a_tau, b_tau)
phi0_priors::Vector{Float64},           # Prior parameters for phi0 ~ N(m0, s0^2)
phi1_priors::Float64,                   # Prior parameters for phi1 ~ U(-1,1)
                                        # so we just need the std dev of the Metropolis update trough N(
                                        phi1_old,mhsig_phi1^2)
lambda2_priors::Vector{Float64},        # Prior parameters for lambda2 ~ invGamma(a_lambda, b_lambda)
alpha_priors::AbstractArray{Float64},   # Prior parameters for alpha ~ Beta(a_alpha, b_alpha)
                                        # but possibly that pair for each unit j, that's why the abstract array

spatial_cohesion_idx = missing,         # cohesion choice
sp_params = missing,                    # Parameters for spatial cohesion functions
covariate_similarity_idx = missing,     # similarity choice
cv_params = missing,                    # Parameters for covariates similarity functions

draws::Float64,                         # Number of MCMC draws
burnin::Float64,                        # Number of burn-in sa
thin::Float64,                          # Thinning interval

logging::Bool,                          # Wheter to save execution infos to log file
seed::Float64                           # Random seed for reproducibility
```

Test 1 con dati fittizi

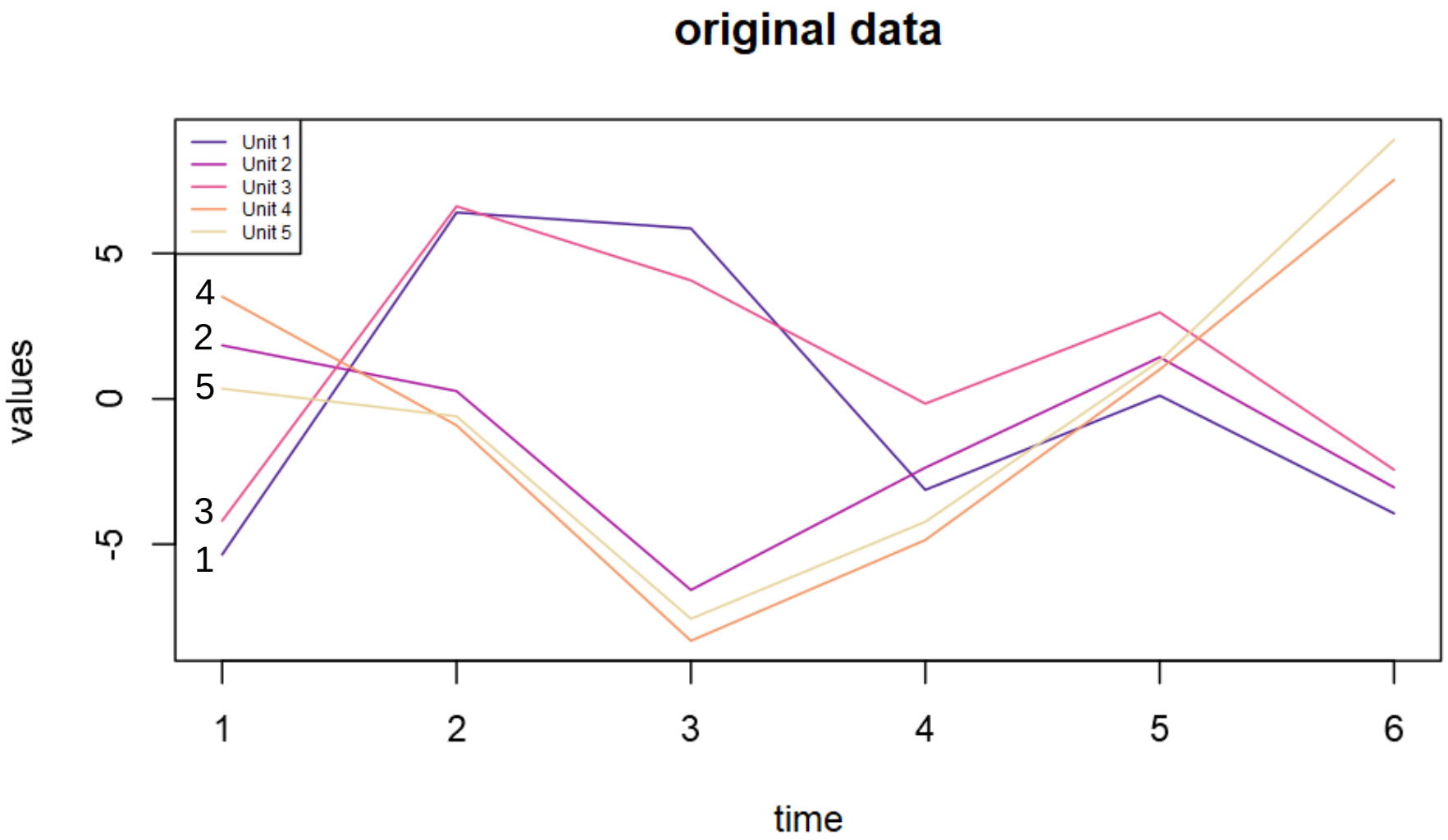
Test per capire intanto se tutto funzionasse.

- fit su dati generati da una funzione
- solo variabile Y target, niente spazio o covariate per ora
- 30k iterazioni (fin troppe, uscivano già buoni fit con molte meno), 22k burnin, 8 thin
- DRPM su C ci ha messo 14.930 secondi
- Il fit su Julia ci ha messo 4.508 secondi

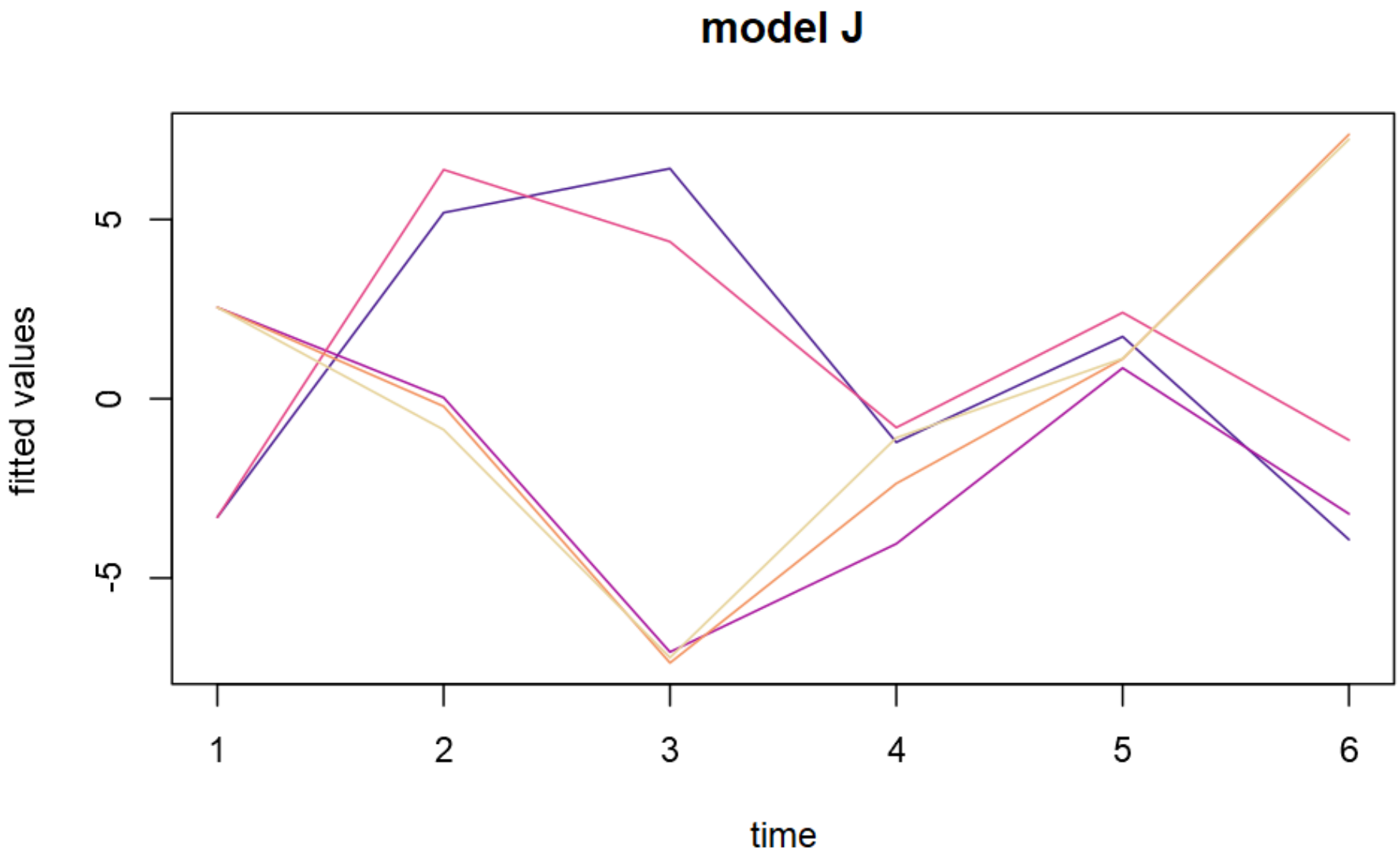
Test 1 con dati fittizi: valori fittati all'ultima iterazione



Test 1 con dati fittizi: valori originali

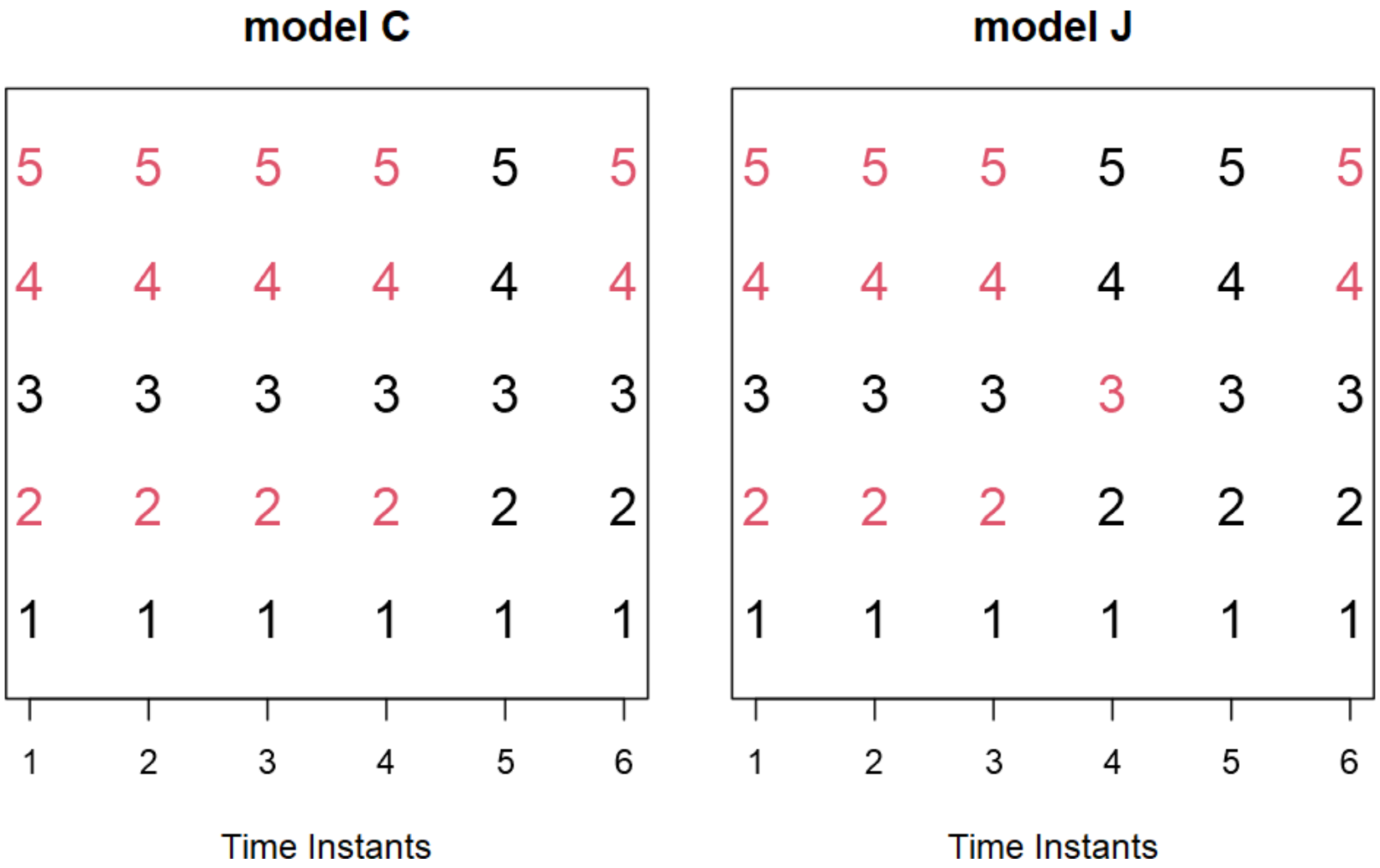


Test 1 con dati fittizi: valori fittati all'ultima iterazione

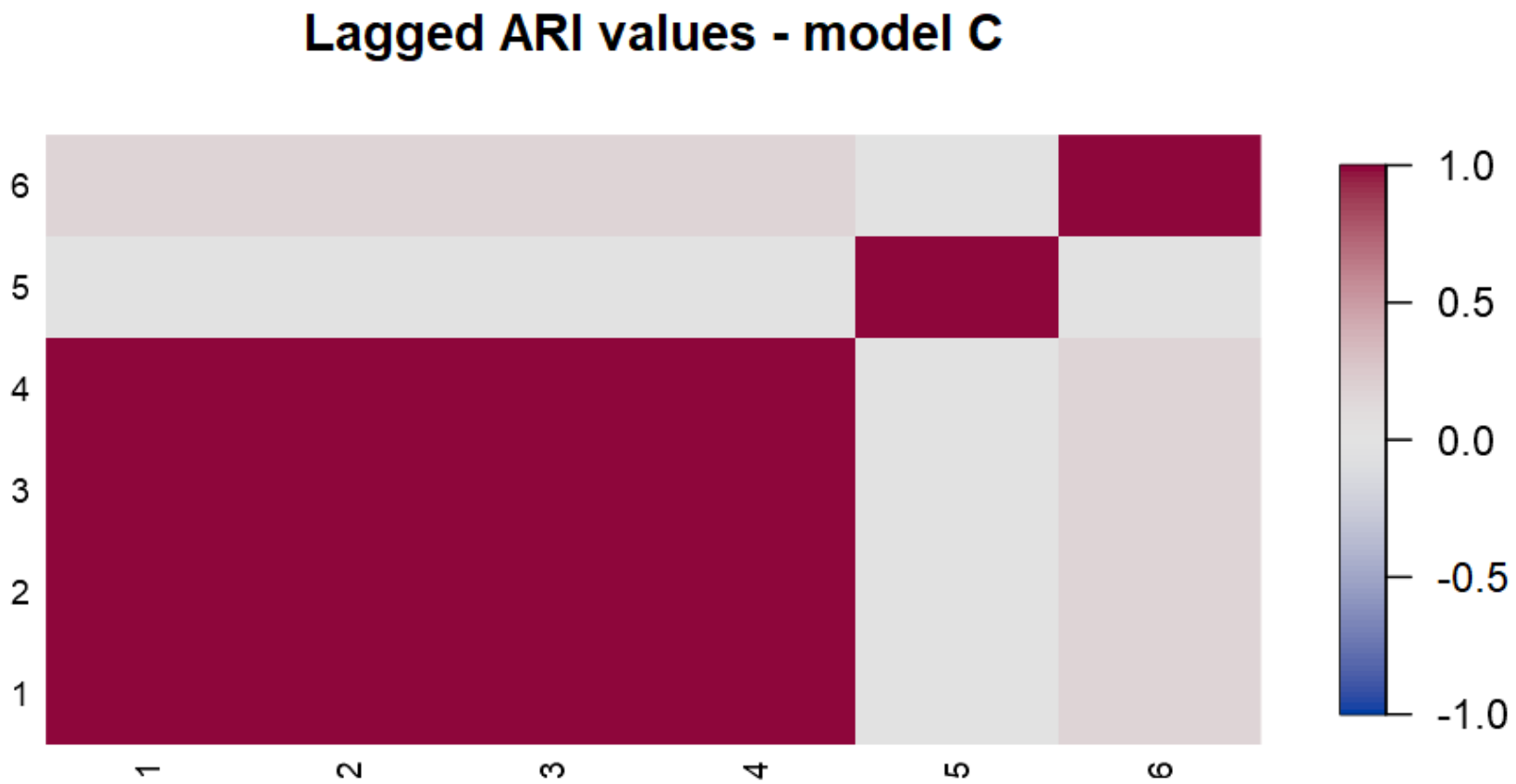


Test 1 con dati fittizi

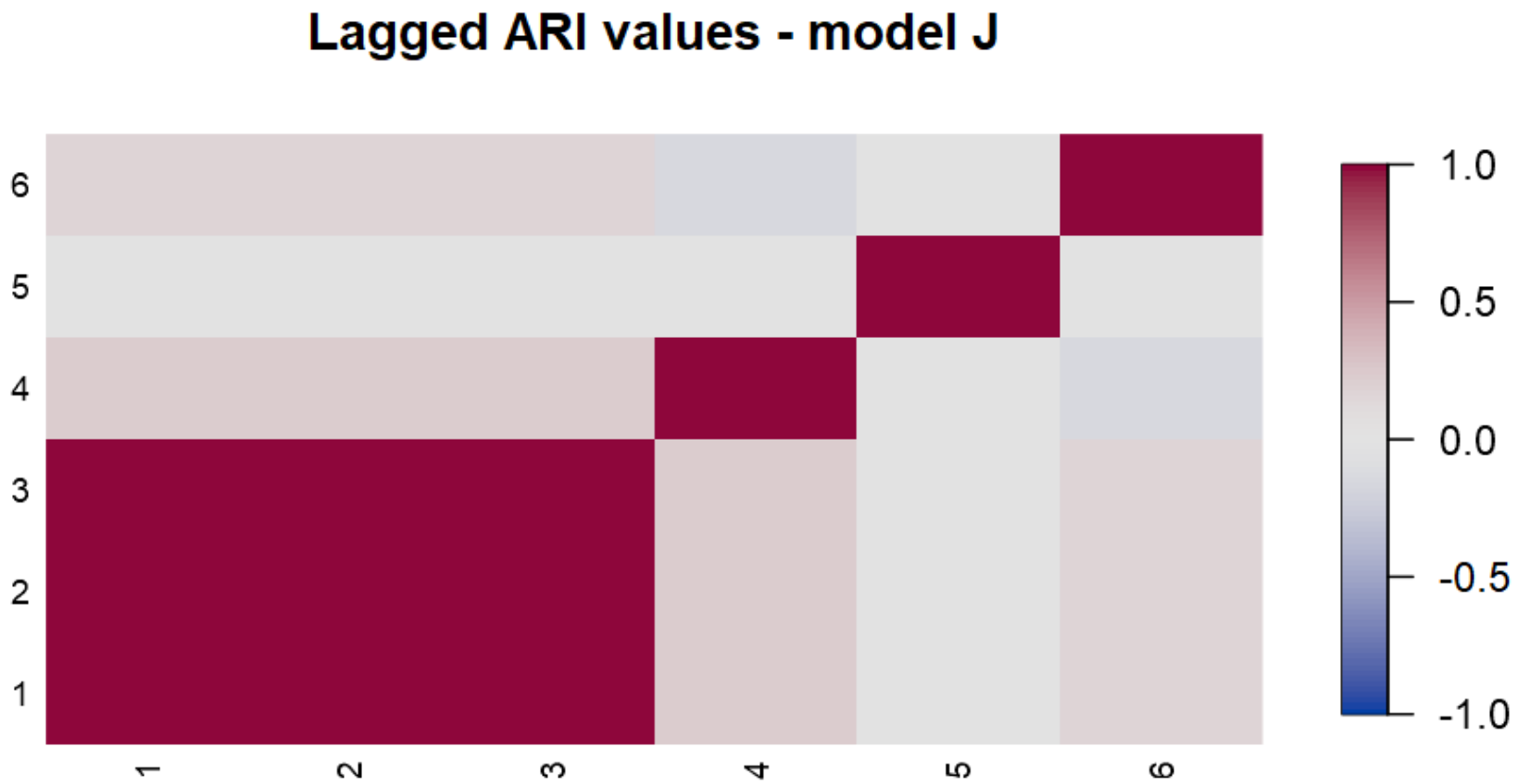
Cluster prodotti. Unica differenza al tempo 4.



Test 1 con dati fittizi



Test 1 con dati fittizi



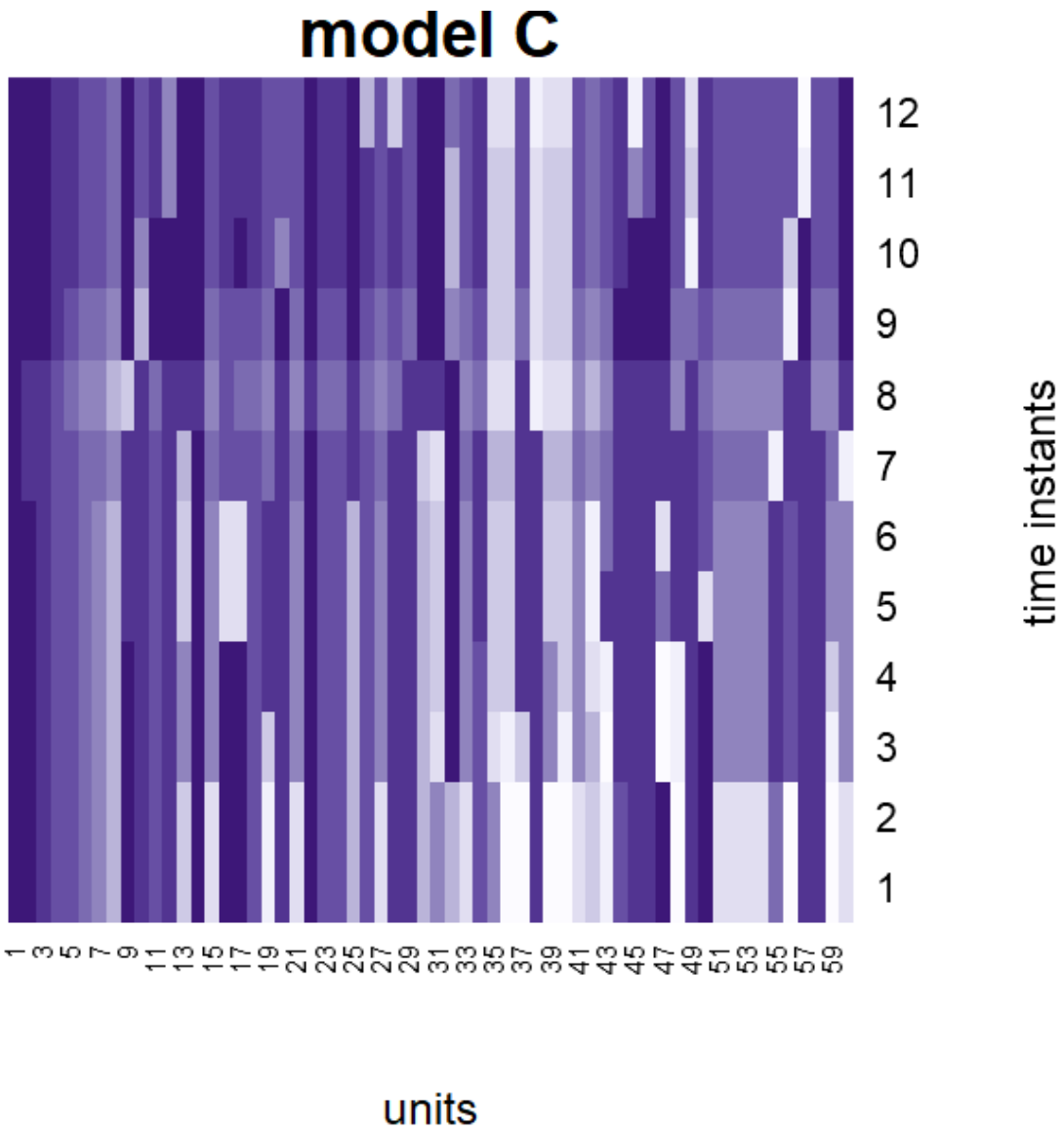
Test 2 con dati spaziali

Test con anche la componente spaziale. Dati dal pacchetto `gstat` di R.

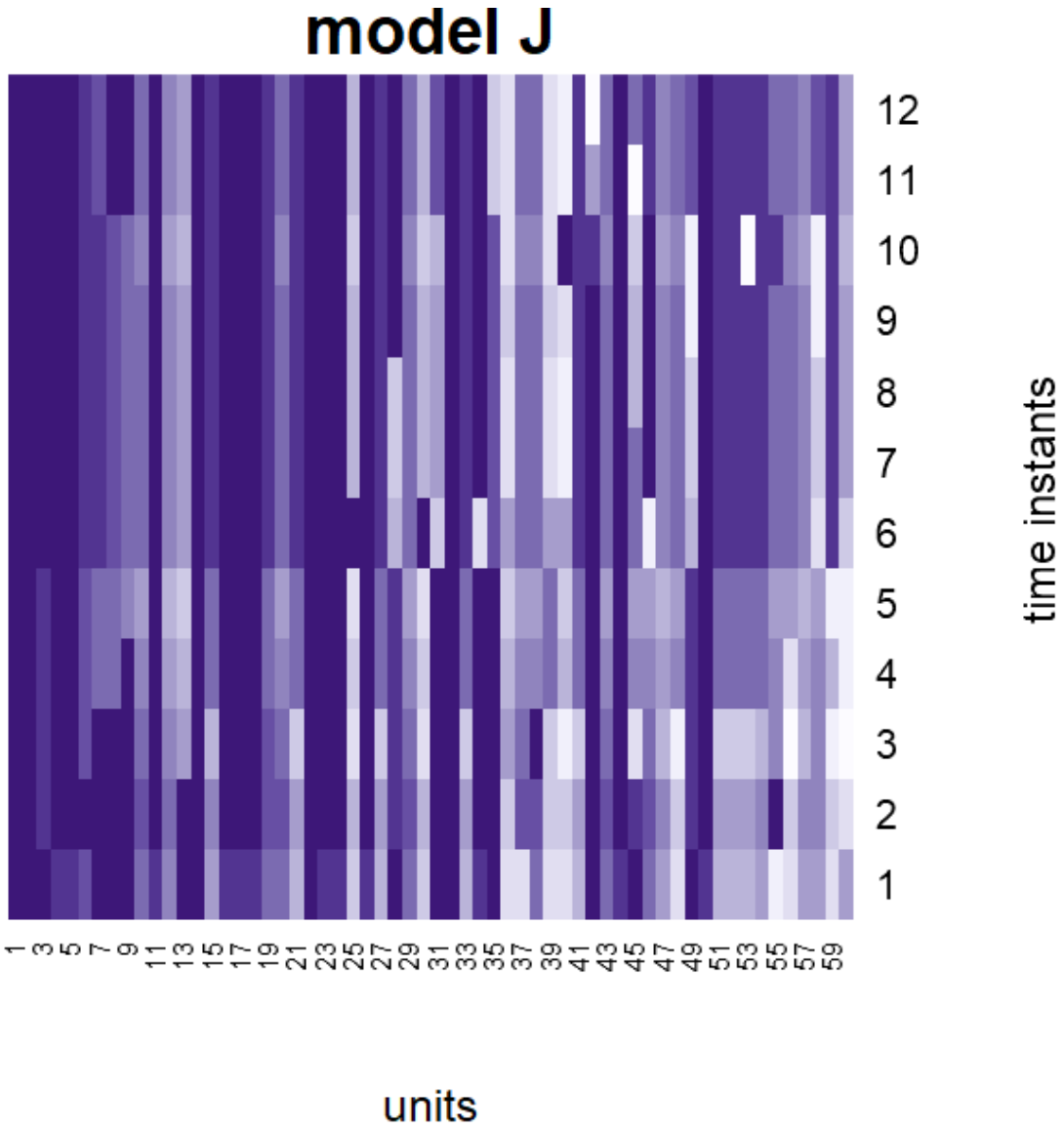
- $n = 60$ unità, $T = 12$ istanti temporali
- 30k iterazioni, 22k burnin, 8 thin
- DRPM su C ci ha messo 36 minuti
  - LPML = -1621.91299023557
  - WAIC = 2961.57613389996
- Il fit su Julia ci ha messo 13 minuti
  - LPML: -1595.8684821766847 (the higher the better)
  - WAIC: 2692.95647704905 (the lower the better)

Update: in realtà nei primi test il fit anche spaziali su julia erano molto più lenti di C, ma poi pian piano ho capito dove ottimizzare il codice per renderlo più veloce. Ho già scritto qualcosa nell'appendice "Computational details".

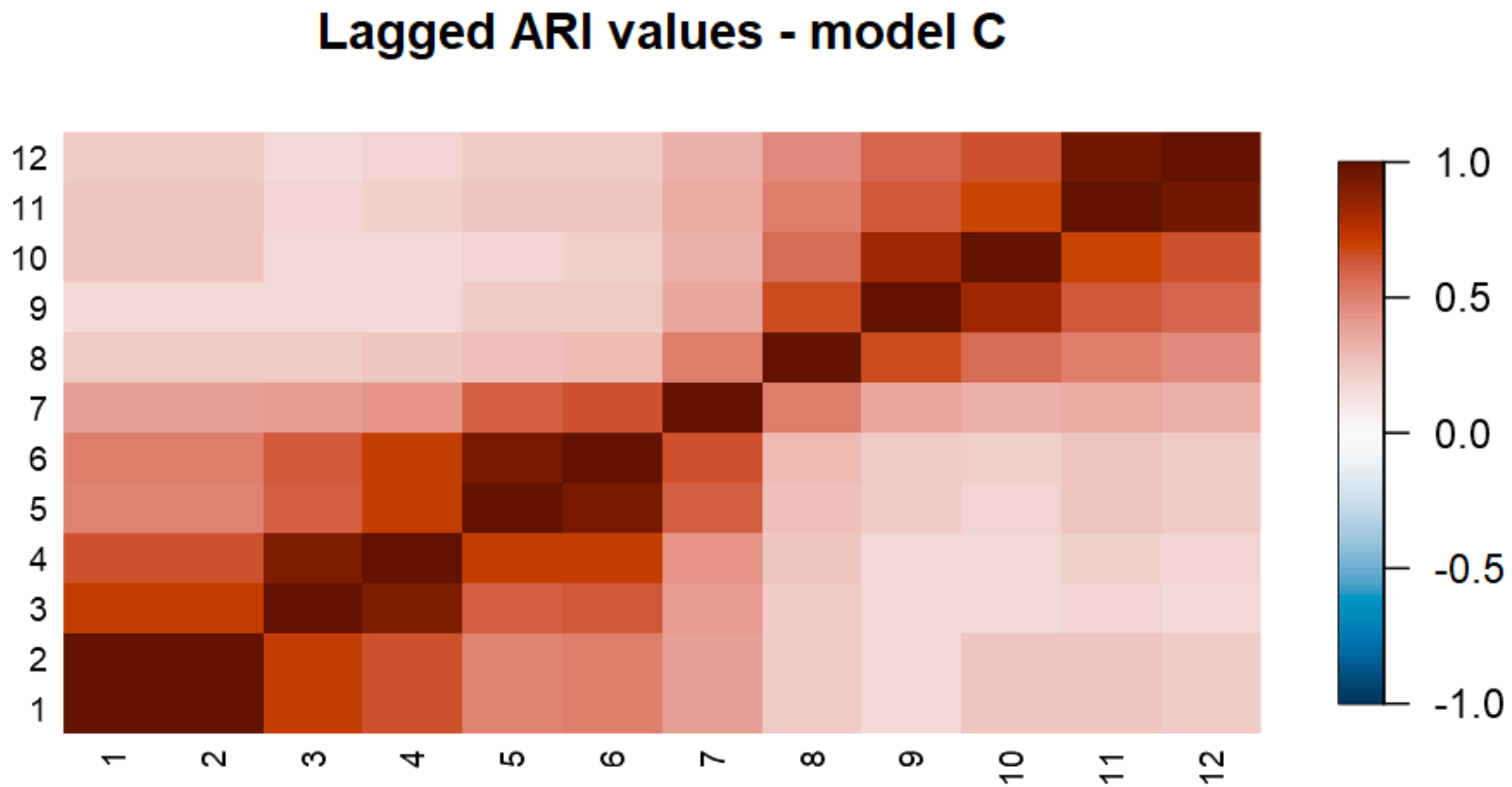
Test 2 con dati spaziali



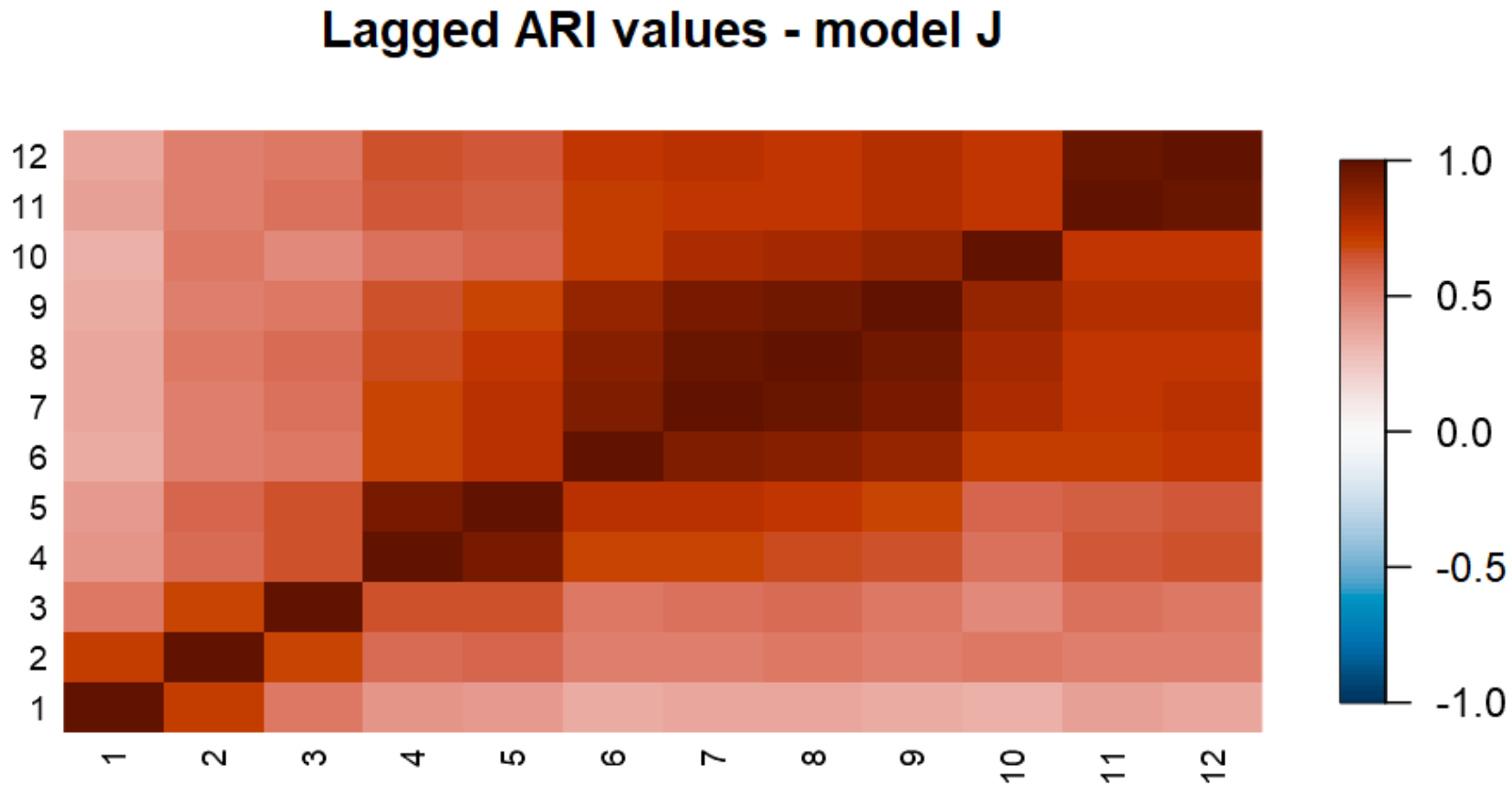
Test 2 con dati spaziali



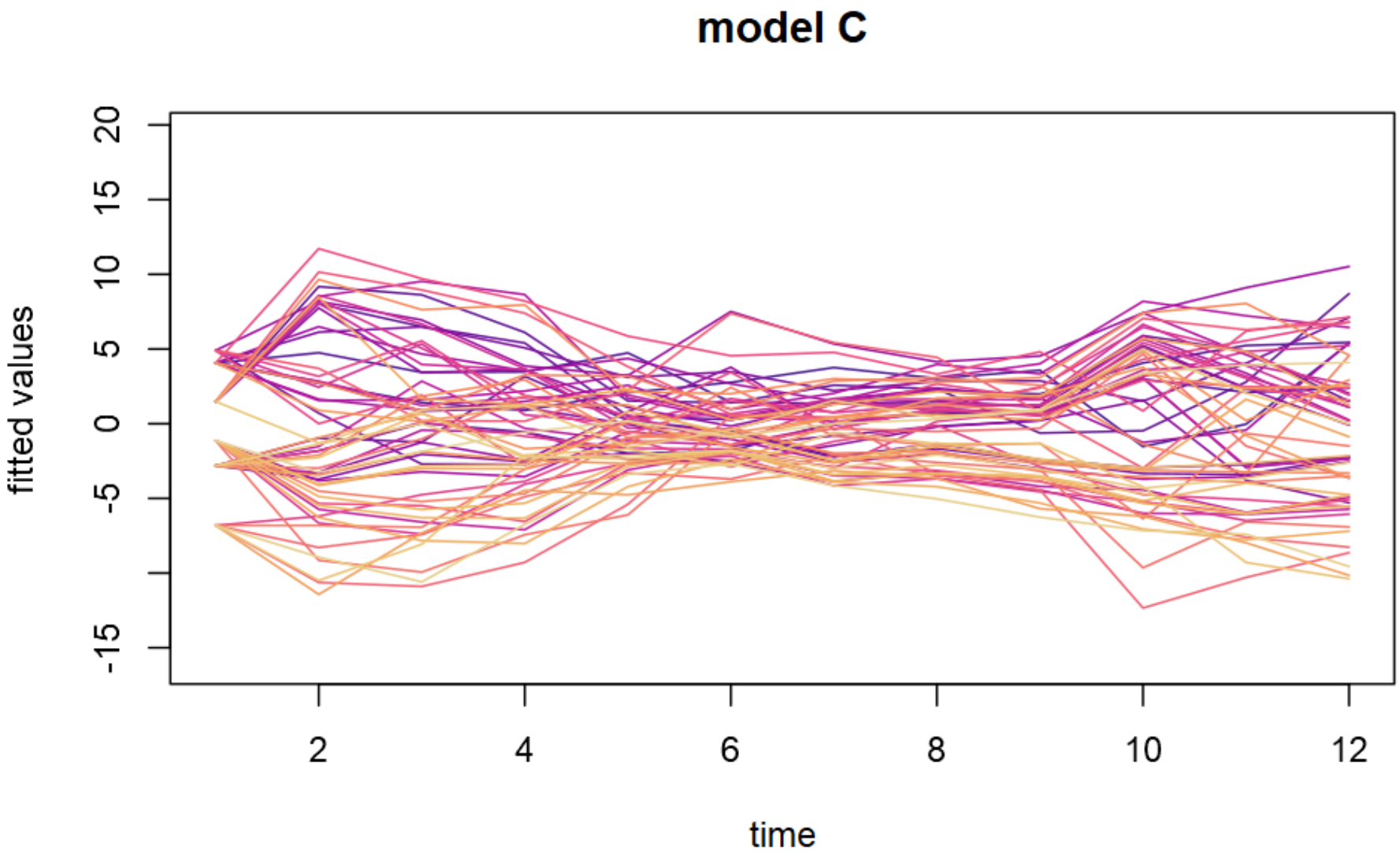
Test 2 con dati spaziali



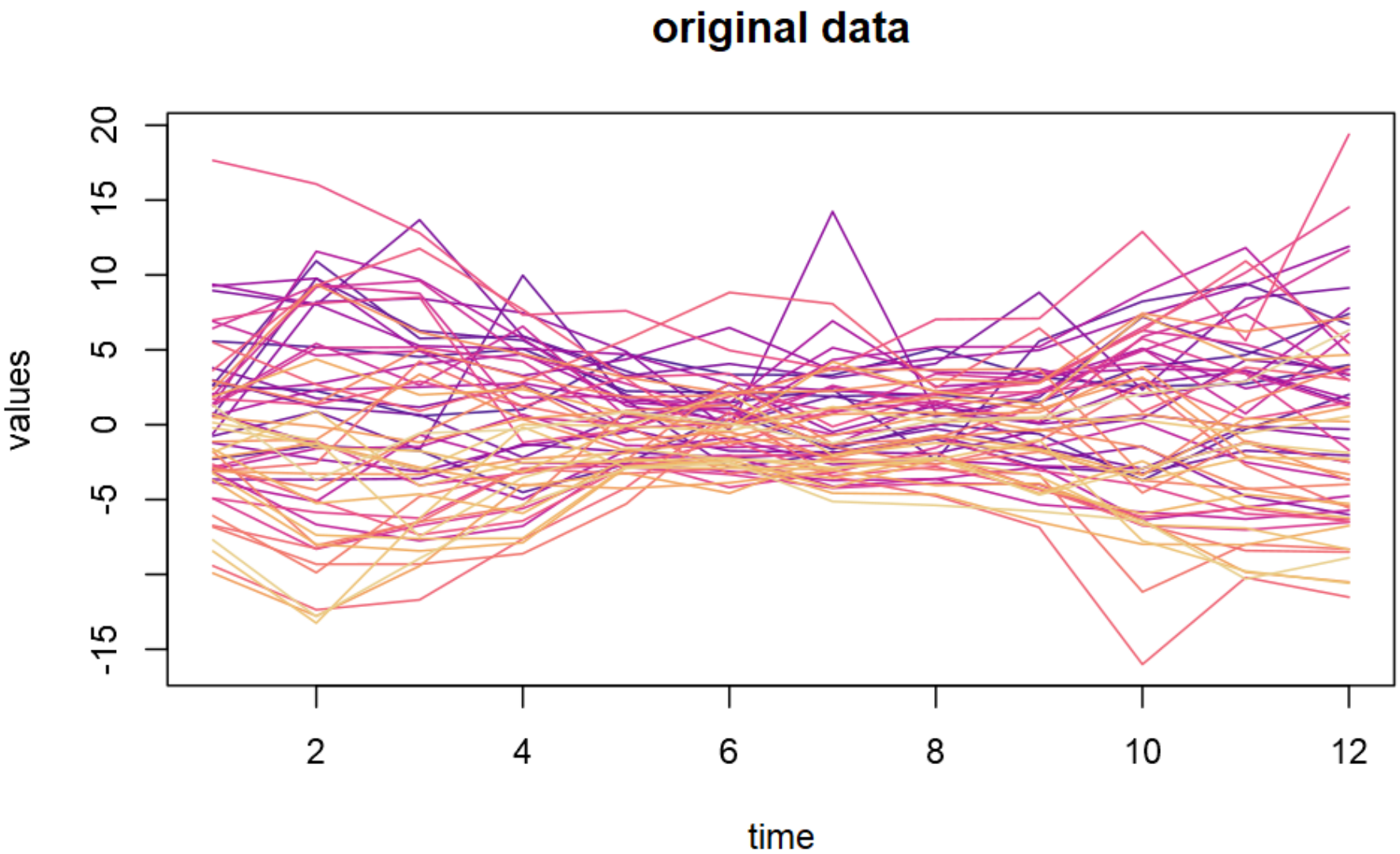
Test 2 con dati spaziali



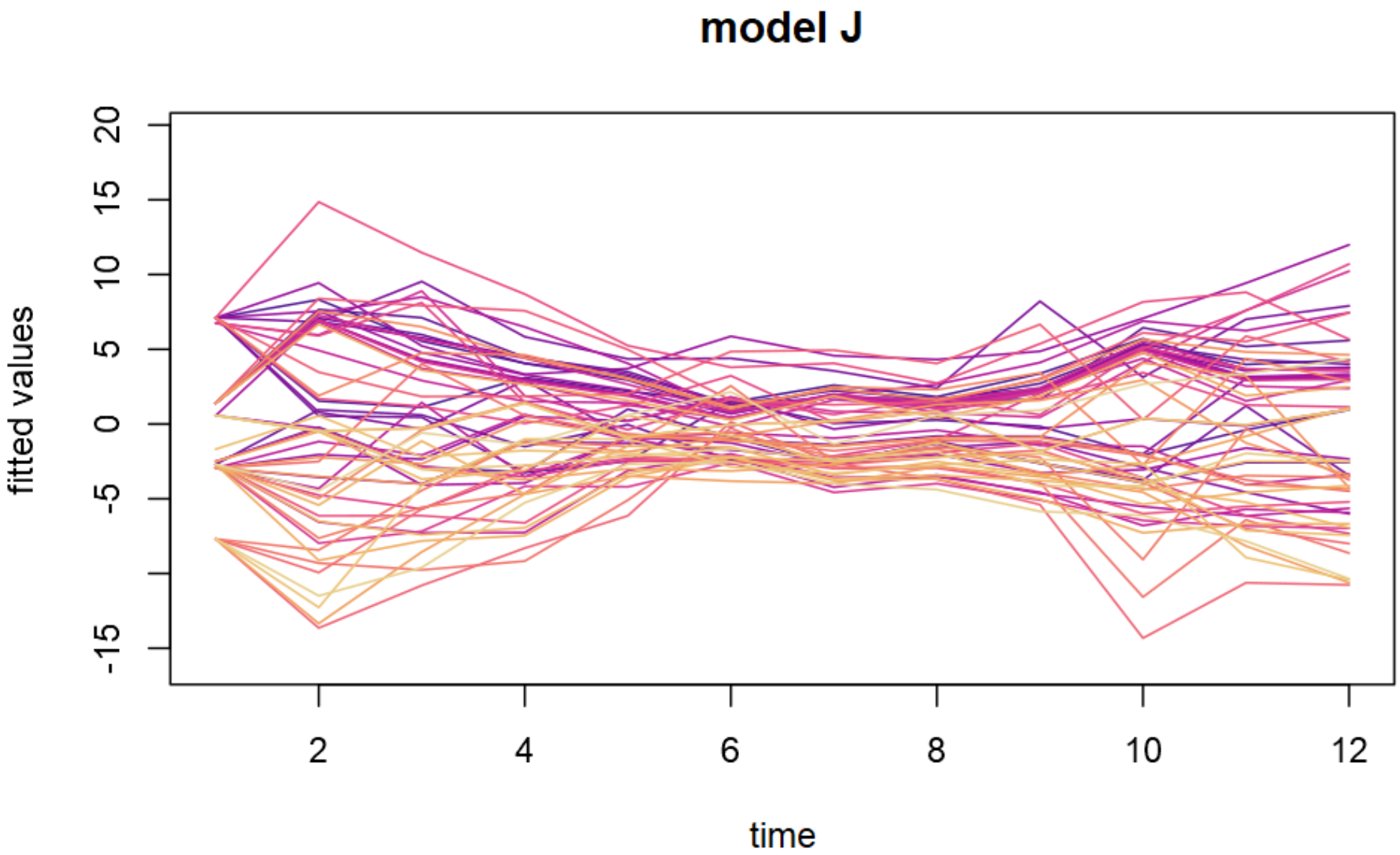
Test 2 con dati spaziali: valori fittati all'ultima iterazione



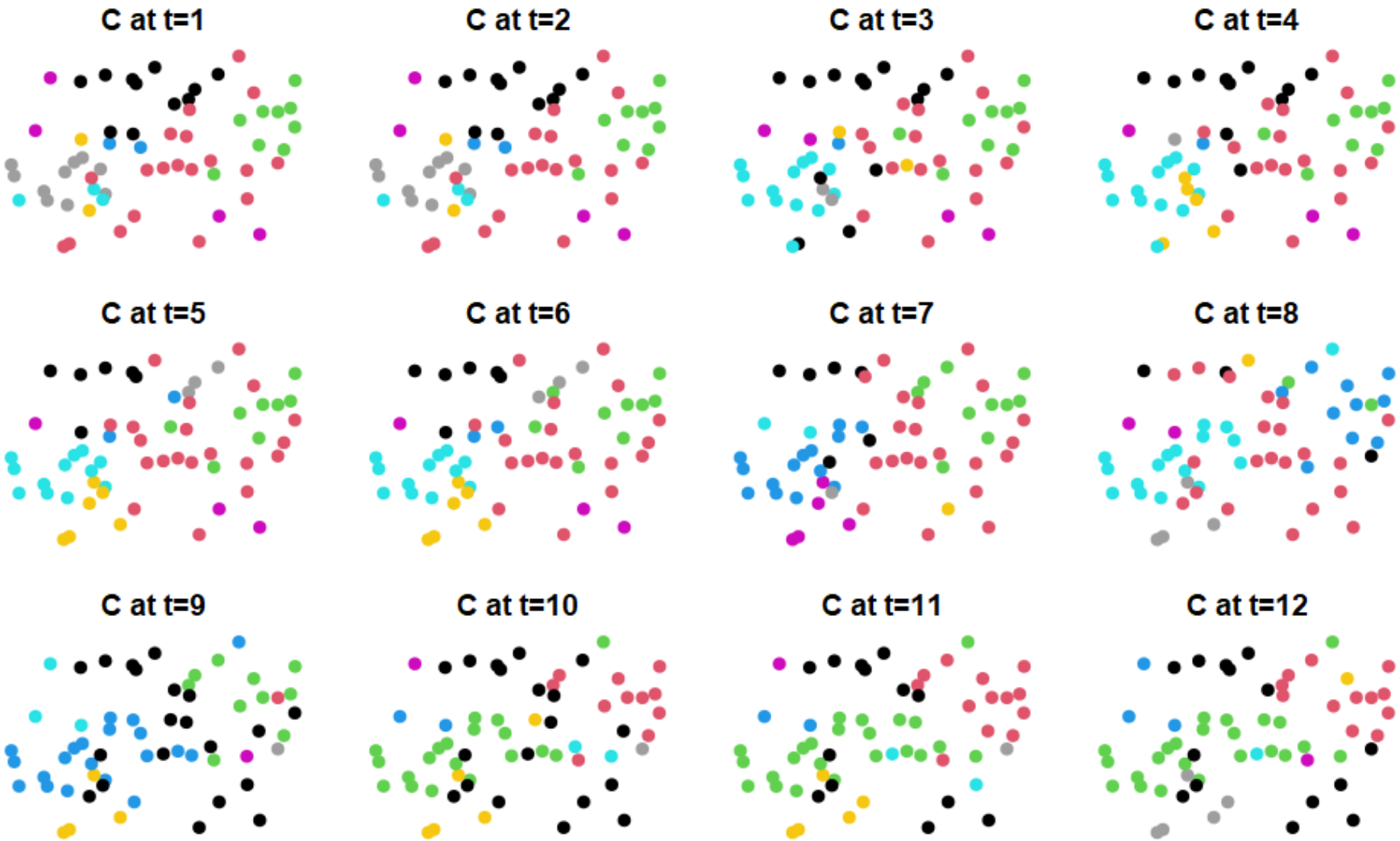
Test 2 con dati spaziali: valori originali



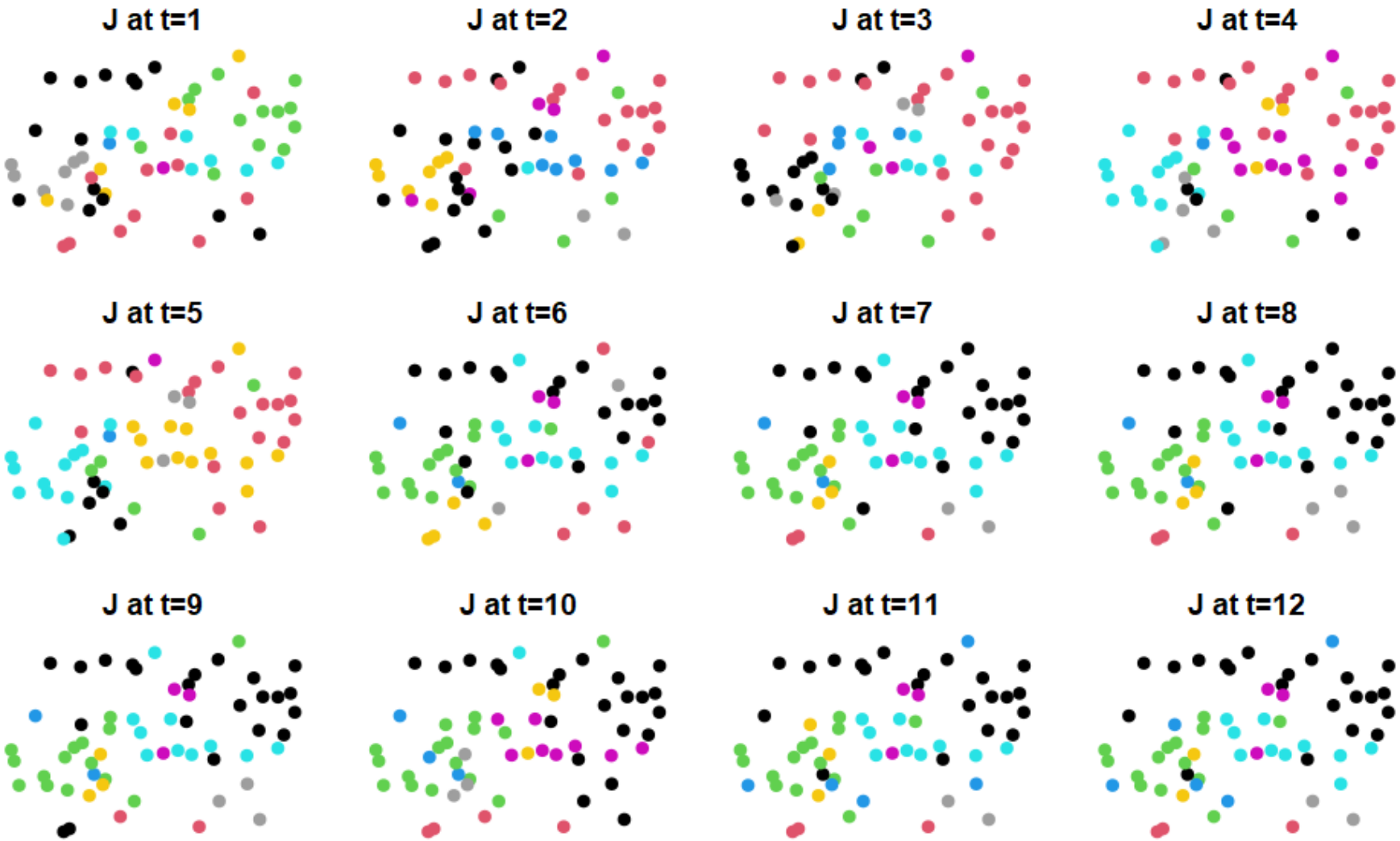
Test 2 con dati spaziali: valori fittati all'ultima iterazione



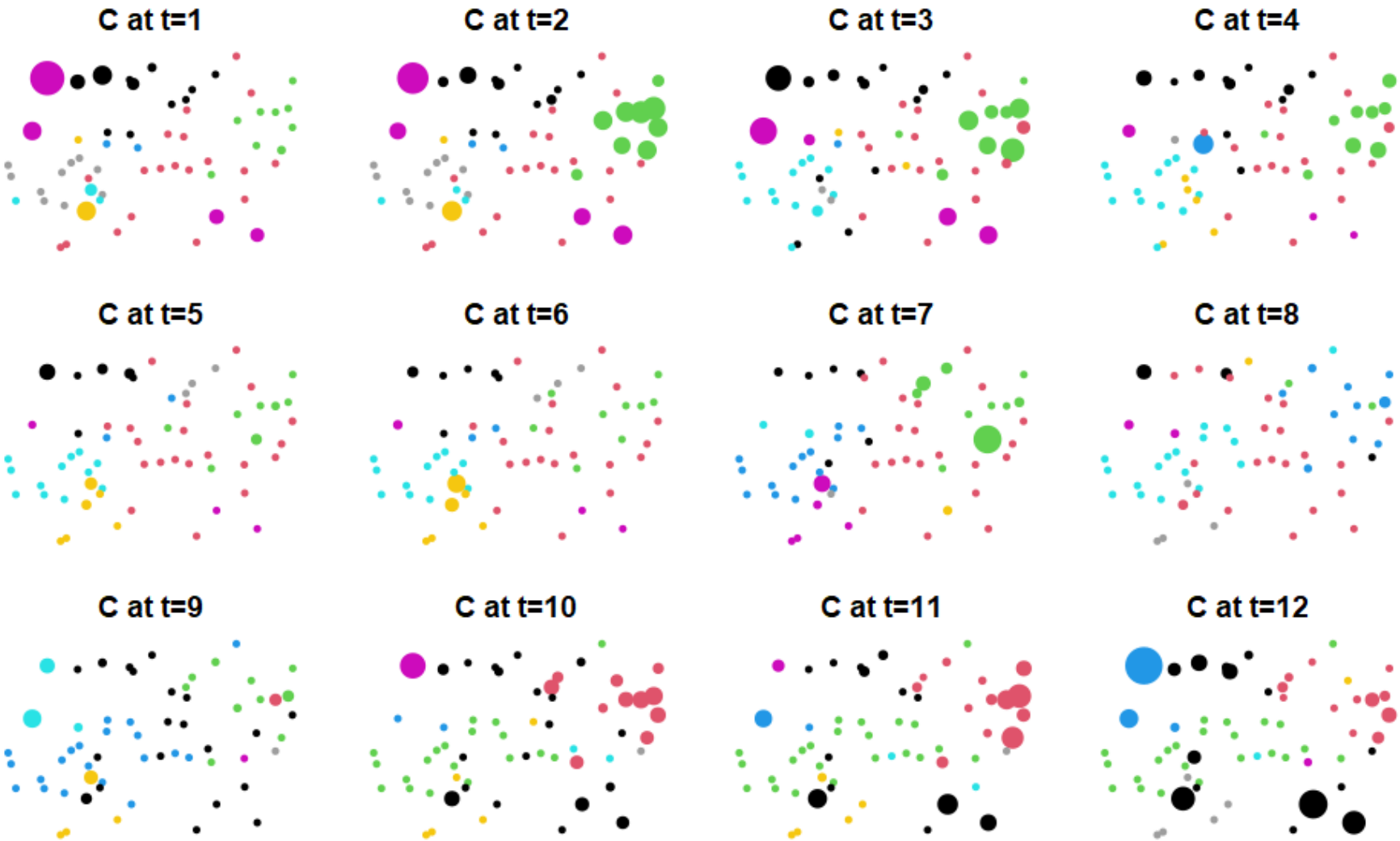
Test 2 con dati spaziali



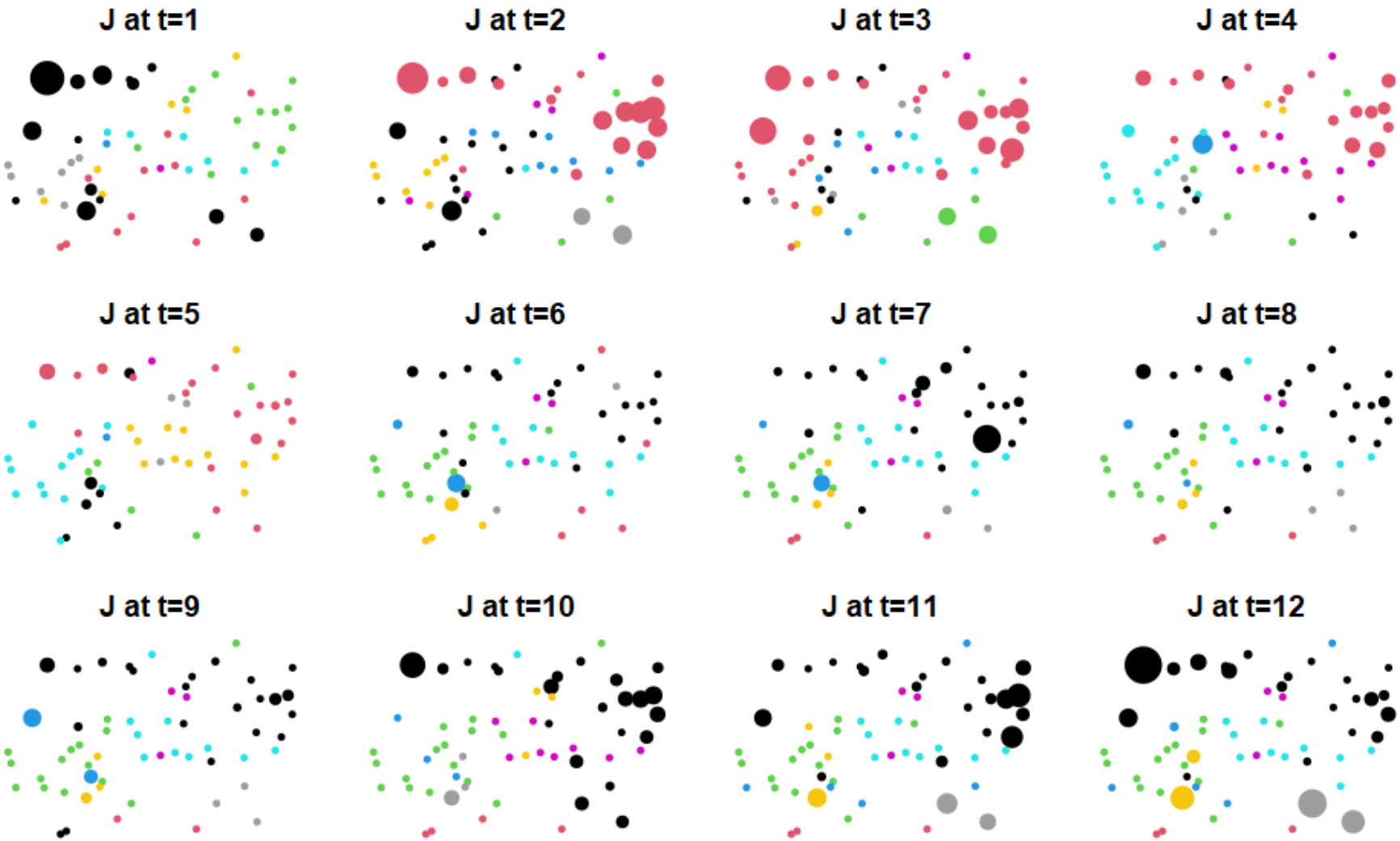
Test 2 con dati spaziali



Test 2 con dati spaziali



Test 2 con dati spaziali



Test 3 con covariate nella likelihood

Dati del progetto di bayesian statistics sul PM10.
Fit su $n = 105$, $T = 6$; con 40k iterazioni, 28k burnin, 8 thin.

senza covariate nella likelihood:

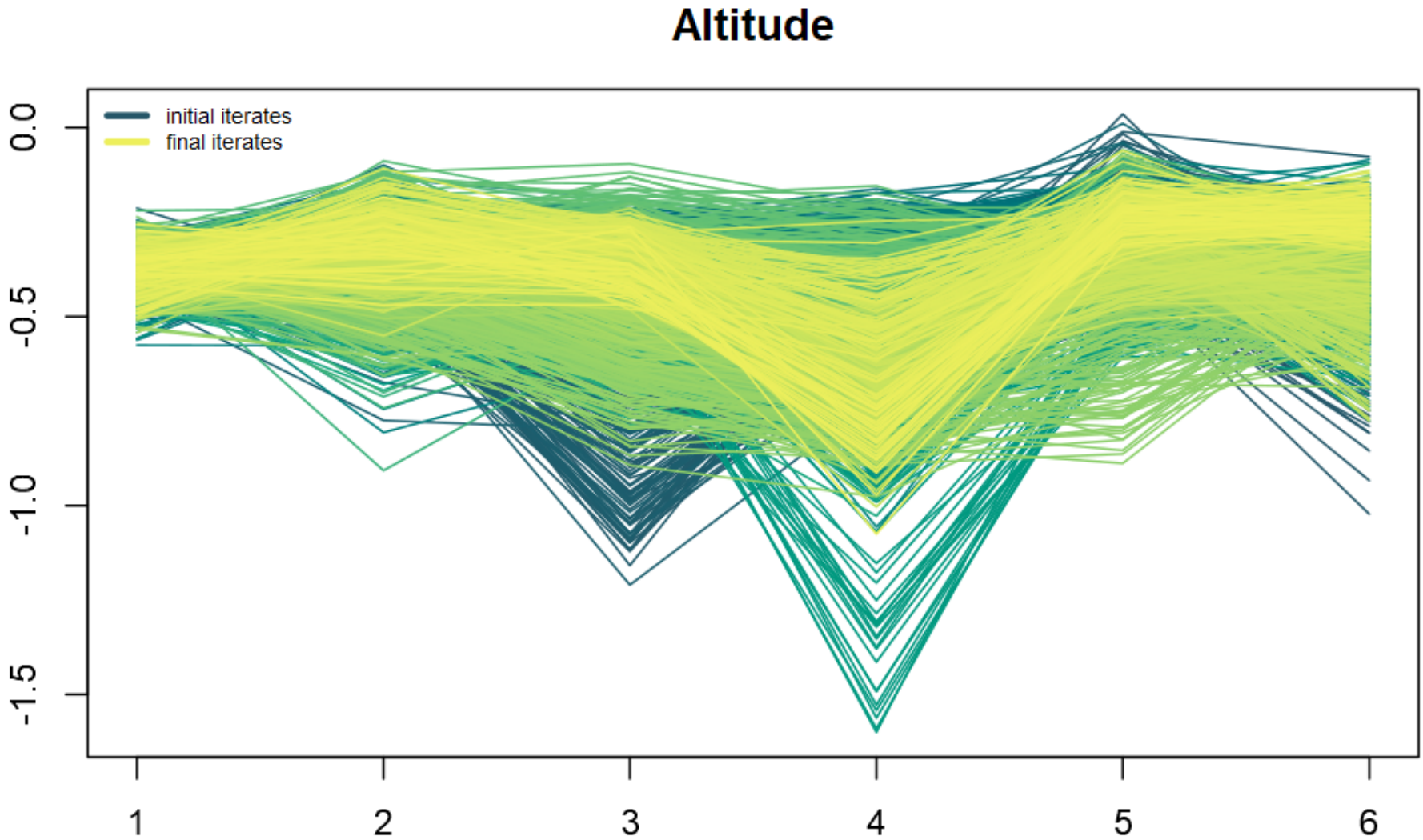
- LPML: -2238.0127582030173 (the higher the better)
- WAIC: 18.864069686401773 (the lower the better)

con covariate nella likelihood:

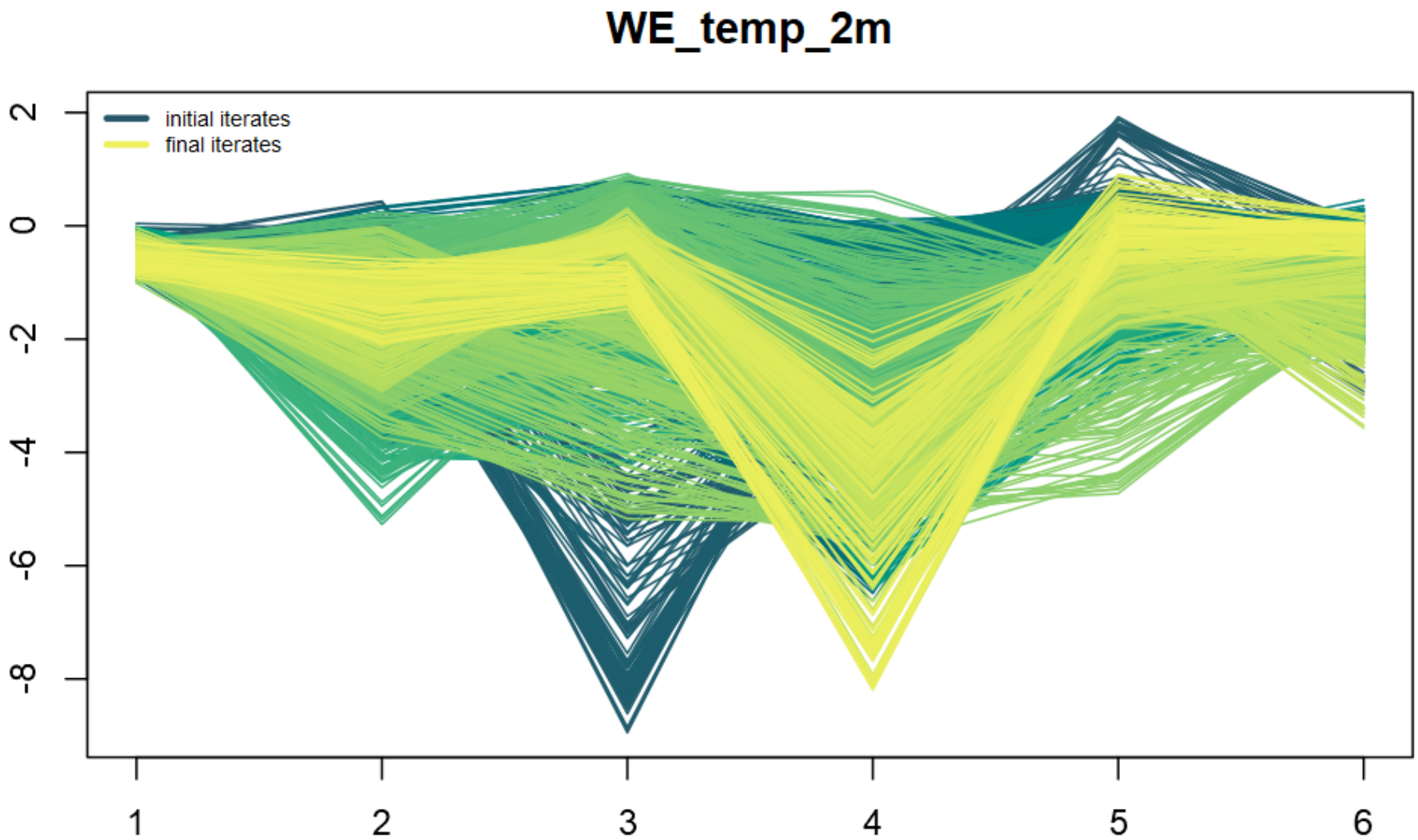
- LPML: -3236.4118577338922 (the higher the better)
- WAIC: 352.86782815050054 (the lower the better)

Sembra peggiore con le covariate. Forse le ho scelte male e/o c'erano troppe poche iterazioni (in effetti per il progetto avevamo fittato 100k iterazioni per avere buoni fit).

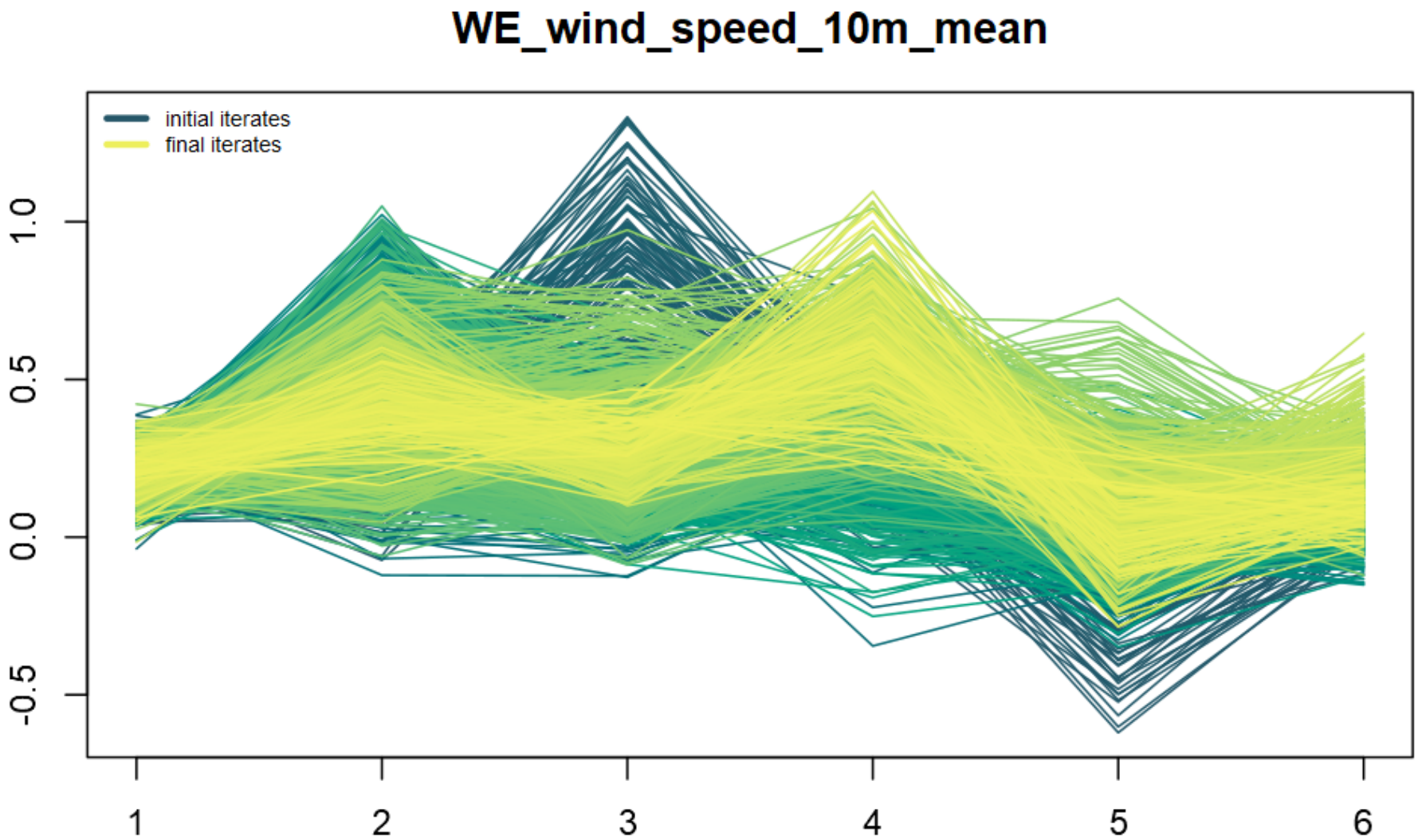
Test 3 con covariate nella likelihood: trace plots β_t



Test 3 con covariate nella likelihood: trace plots β_t

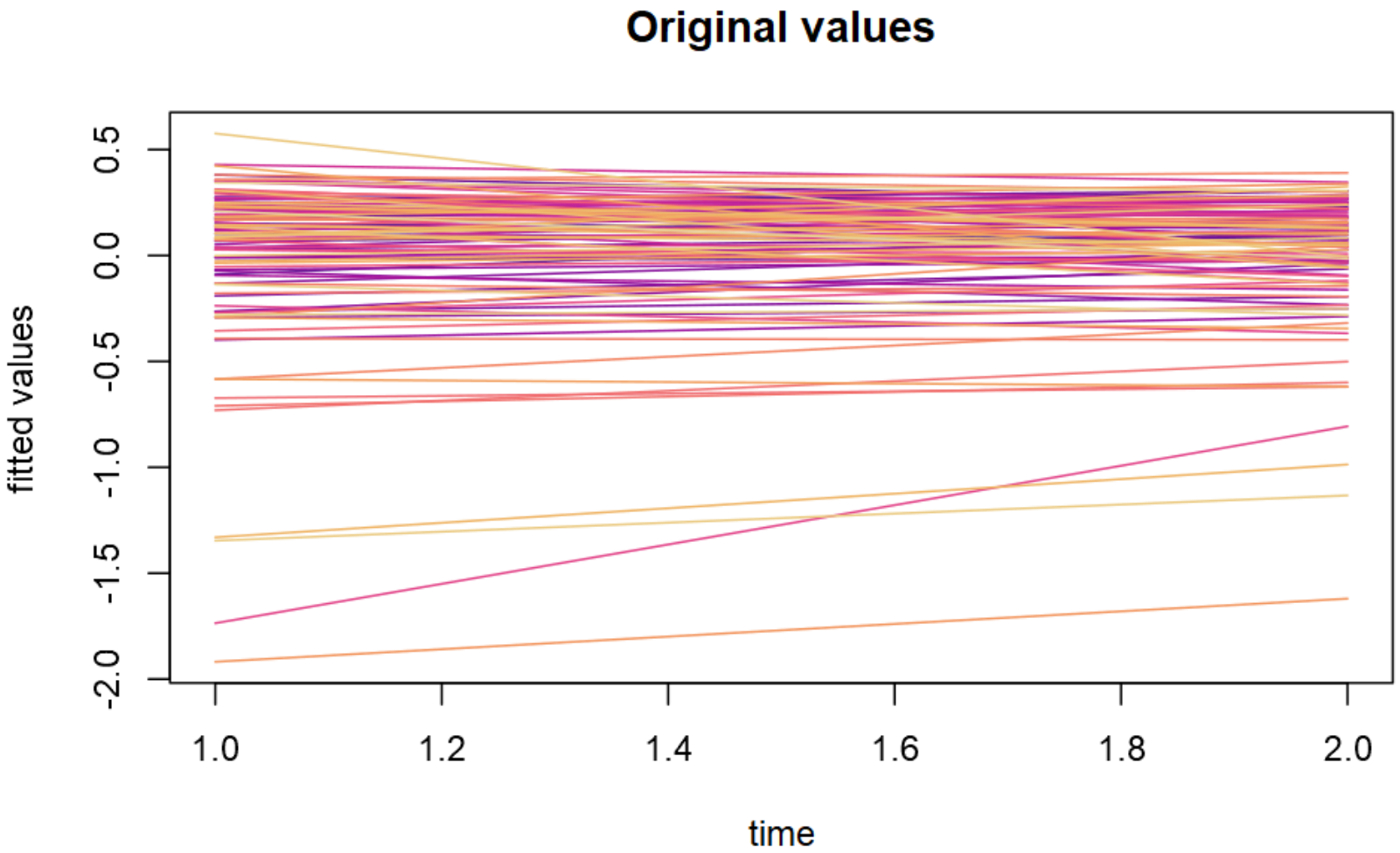


Test 3 con covariate nella likelihood: trace plots β_t

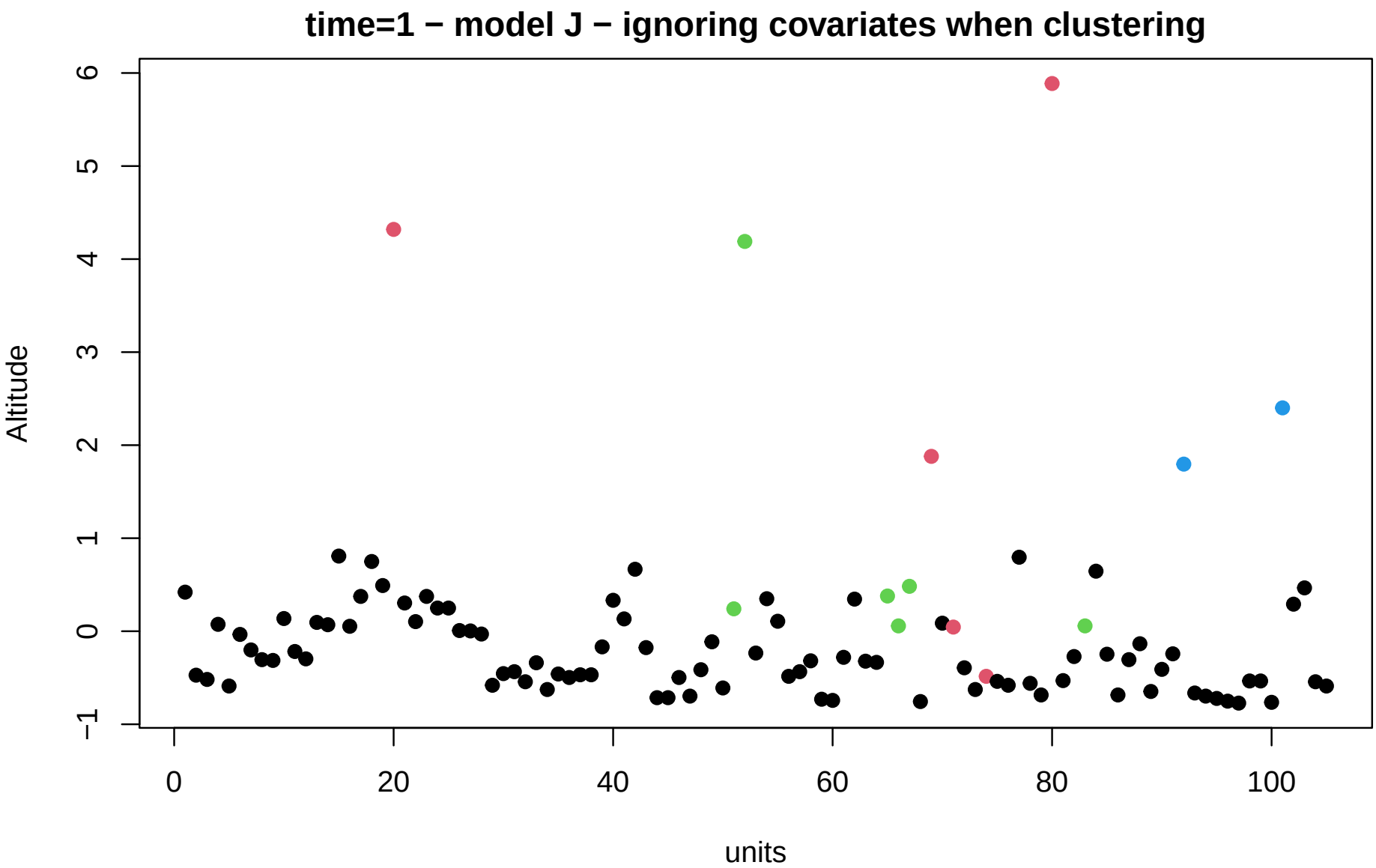


Test 4 con covariate nel clustering

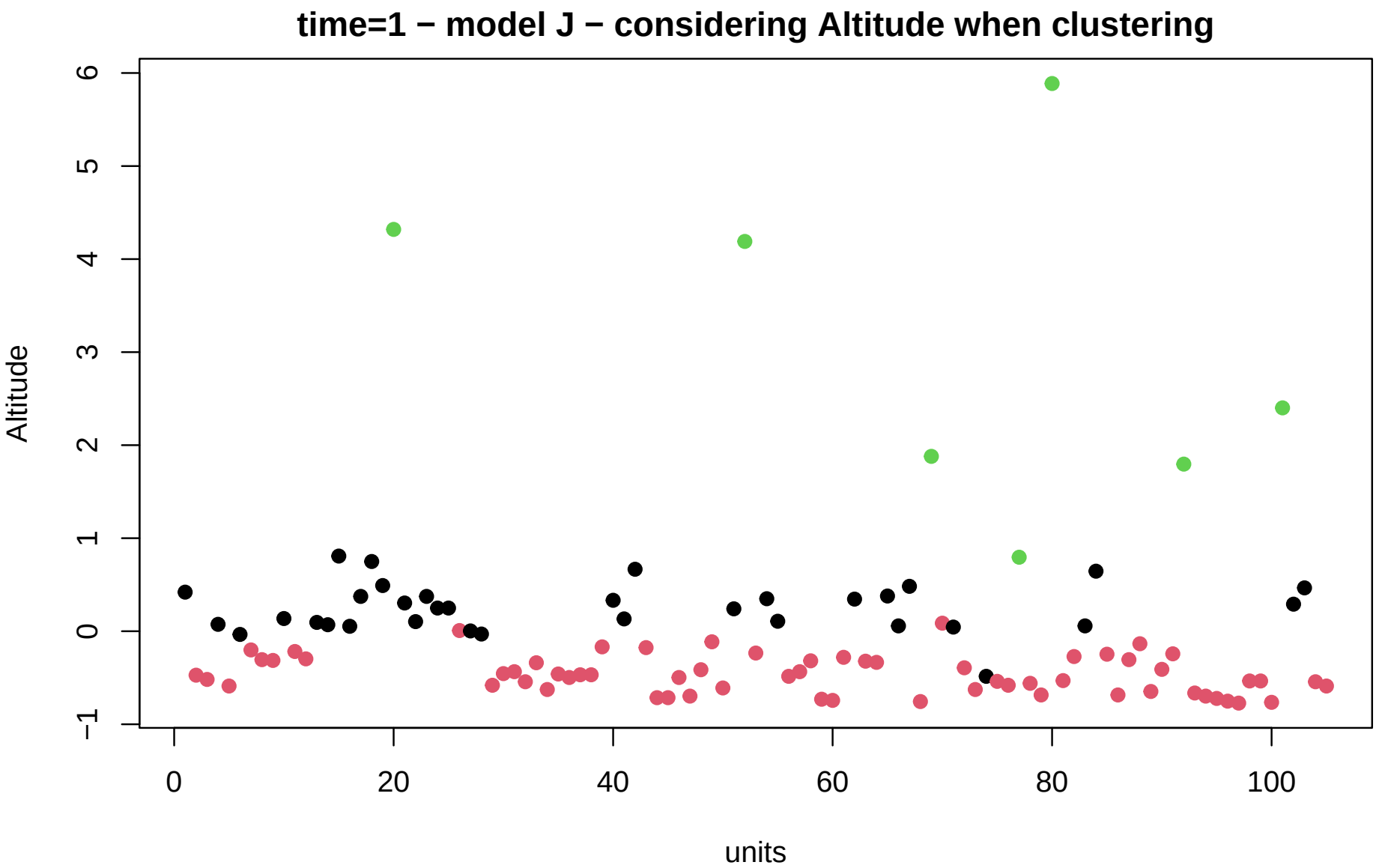
Sempre PM10 ma ora dati spaziali + covariata Altitude. $n = 105$, $T = 2$ (T basso giusto per testing), 50k iterazioni, 40k burnin, 10 thin.



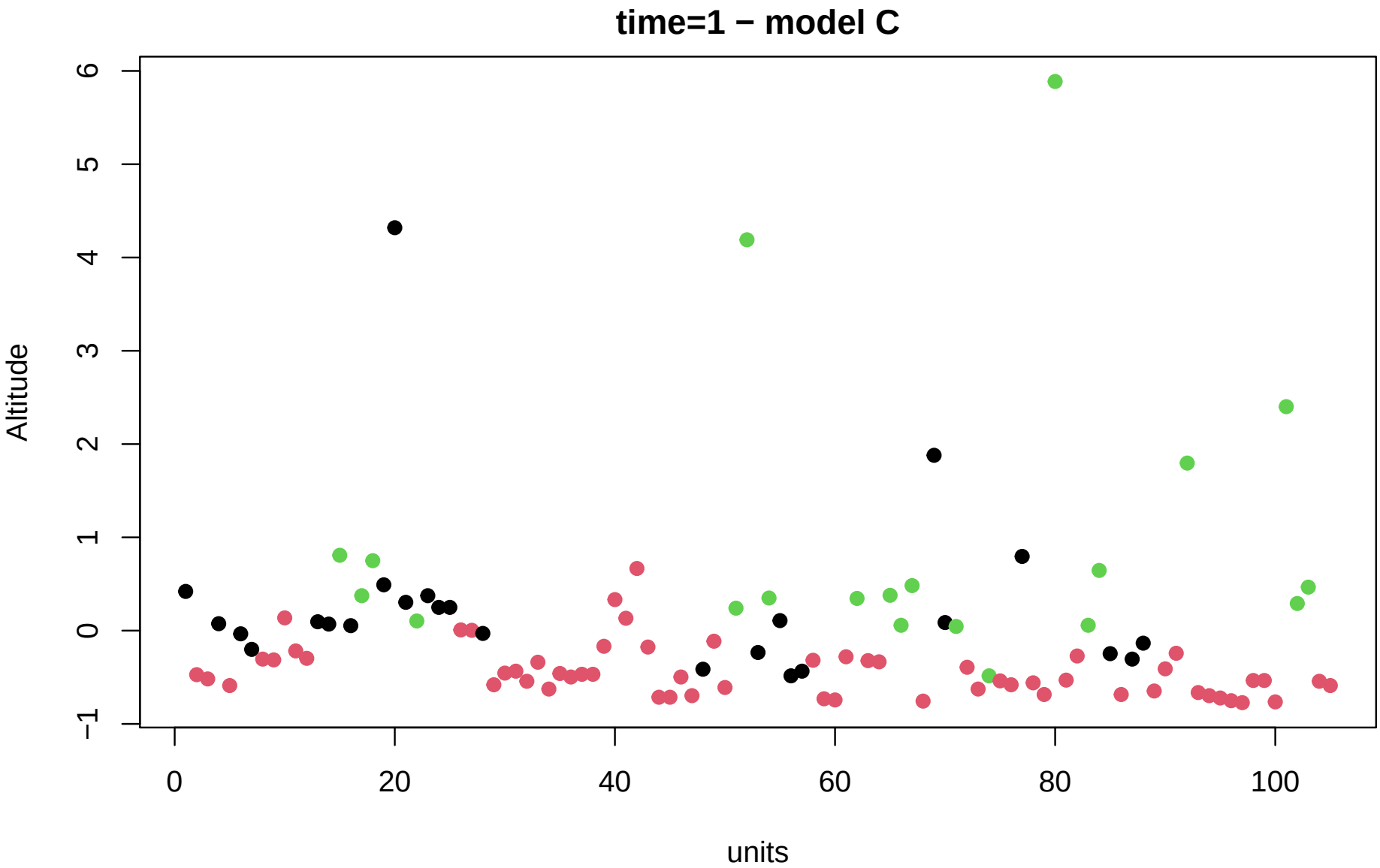
Test 4 con covariate nel clustering



Test 4 con covariate nel clustering



Test 4 con covariate nel clustering



Dubbio inizializzazione parametri

I parametri del modello devo inizializzarli con

- dei random sample dalle loro prior? (come per ora è implementato)
- o magari valori puntuali come media/mediana della loro prior?
- o altro?

```
# hierarchy level 3
phi0_iter = rand(Normal(phi0_priors[1], sqrt(phi0_priors[2])))
phi1_iter = 0.0
if include_phi1
    phi1_iter = rand(Uniform(-1,1))
end
lambda2_iter = rand(InverseGamma(lambda2_priors...))

# hierarchy level 2
theta_iter = zeros(T_star)
theta_iter[1] = rand(Normal(phi0_iter,sqrt(lambda2_iter)))
for t in 2:T_star
    theta_iter[t] = rand(Normal((1-phi1_iter)*phi0_iter + phi1_iter*theta_iter[t-1],sqrt(lambda2_iter*(1-phi1_iter^2))))
end
tau2_iter = zeros(T_star)
for t in 1:T_star
    tau2_iter[t] = rand(InverseGamma(tau2_priors...))
end
```

Tenere le inverse gamma?

Mi sembra che le inverse gamma a volte facessero fatica ad entrare nella "giusta" finestra di valori:

- rimanevano alte quando la varianza in realtà era praticamente zero
- oppure invece non riuscivano a sbilanciarsi bene verso i valori più alti

Update: forse no basta solo un attimo sperimentare coi parametri e penso funzioni tutto (visti anche i fit ottenuti sopra per esempio, che mi sembravano incoraggianti).

Una coesione spaziale complicata

I paper sulle coesioni spaziali secondo me spiegano poco su come passare dalla teoria raccontata all'implementazione. Cioè il passaggio da qui

$$C_3(S_h, \boldsymbol{s}_h^\star) = M \cdot \Gamma(|S_h|) \cdot \int \prod_{i \in S_h} q(\boldsymbol{s}_i|\boldsymbol{\xi}_h) q(\boldsymbol{\xi}_h)\, d\boldsymbol{\xi}_h$$

a qui $\longrightarrow$

```
function cohesion3(s1::AbstractVector{Float64}, s2::AbstractVector{Float64}, mu_0::AbstractVector{Float64}, k0::Real,
     v0::Real, Psi::Matrix{Float64}; lg::Bool, M::Real=1.0)
    sdim = length(s1)
    sp = [s1 s2]
    sbar = [mean(s1), mean(s2)]
    S = zeros(2,2)
    for i in 1:sdim
        vtemp1 = sp[i,:] - sbar
        S += (vtemp1)*(vtemp1)'
    end

    # compute updated parameters
    kn = k0+sdim
    vn = v0+sdim
    vtemp2 = sbar-mu_0
    Psi_n = Psi + S + (k0*sdim)/(k0+sdim)*vtemp2*vtemp2'

    out = -sdim * logpi + G2a(0.5 * vn, true) - G2a(0.5 * v0, true) + 0.5 * v0 * logdet(Psi) - 0.5 * vn * logdet(
        Psi_n) + log(k0) - log(kn)
    return lg ? out : exp(out)
end

function G2a(a::Real, lg::Bool)
    out = logpi + lgamma(a) + lgamma(a - 0.5)
    return lg ? out : exp(out)
end
```

Una coesione spaziale complicata

Update: forse sto capendo, ma sono molti calcoli. Sarebbe l'estensione multivariata dell'auxiliary normal-inverse-gamma similarity function (di cui Alessandro mi aveva inviato i calcoli).

We consider an "Auxiliary Normal-Inverse Gamma" similarity function:

$$\begin{aligned} \xi_h &= (\mu; \sigma_2) \\ \mu &\sim \mathcal{N}\left(\mu_c, \frac{\sigma^2}{\lambda_c}\right) \\ \sigma^2 &\sim IG(a_c, b_c) \\ x \mid \xi_h &\sim \mathcal{N}(\mu, \sigma^2) \end{aligned}$$

We compute the similarity function, i.e., $\int_{\mathbb{R}\times\mathbb{R}_+} \prod_{i=1}^{n} q(x_i \mid \xi_h) q(\xi_h) d\xi_h$.

$$\int_{\mathbb{R}\times\mathbb{R}_+} \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{1}{2\sigma^2}(x_i-\mu)^2\right\} \frac{1}{\sqrt{2\pi}}\sqrt{\frac{\lambda_c}{\sigma^2}} \exp\left\{-\frac{\lambda_c}{2\sigma^2}(\mu-\mu_c)^2\right\} \frac{b_c^{a_c}}{\Gamma(a_c)} (\sigma^2)^{-a_c-1} \exp\left\{-\frac{b_c}{\sigma^2}\right\} d\mu d\sigma^2$$

$$\frac{1}{(2\pi)^{\frac{n}{2}}} \int_{\mathbb{R}\times\mathbb{R}_+} \exp\left\{-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(x_i-\mu)^2\right\} \frac{1}{\sqrt{2\pi}}\sqrt{\frac{\lambda_c}{\sigma^2}} \exp\left\{-\frac{\lambda_c}{2\sigma^2}(\mu-\mu_c)^2\right\} \frac{b_c^{a_c}}{\Gamma(a_c)} (\sigma^2)^{-a_c-\frac{n}{2}-1} \exp\left\{-\frac{b_c}{\sigma^2}\right\} d\mu d\sigma^2$$

We can rewrite the first exponent:

Prossimi passi e problemi riscontrati

- Decidere meglio/sperimentare con gli iperparametri di input (tipo per ora ho messo delle invGamma$(2, 2)$, Beta$(2, 2)$, 0.1 come std dev degli update con Metropolis, ma magari non sono ottimali)
- Capire quali tipi di test fare, su quali dati, contro quali altri modelli. Potenzialmente i possibili fit in julia sarebbero 8 o più, perché ci sono
  - valori target (sempre richiesti)
  - sì/no per spazio
  - sì/no per covariate LK
  - sì/no per covariate CL

  E in più:
  - la scelta di quali covariate
  - escludere/includere alcuni parametri, ovvero i due di autoregressione (come faceva Page nei suoi test)
  - le varie scelte delle funzioni di coesione e similarità