POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

A novel spatio-temporal model for Bayesian Source Apportionment for the PM_{10} pollution

Giorgio Aurina, Paolo Bettanin, Alberto Calanchi, Raffaele Scorza, Claudio Strino, Filippo Volpicelli

Tutors:
Michela Frigeri
Alessandra Guglielmi

Academic year:
2024-2025

Abstract: In recent years, environmental pollution has become one of the most critical topics in scientific research, driving numerous efforts to study the sources and impacts of various pollutants. Understanding the sources of air pollution is crucial for effective environmental policies and health risk assessments. This study presents a Bayesian spatio-temporal model for source apportionment of particulate matter pollution, addressing key challenges such as temporal dependence, spatial variability, and the unknown number of pollution sources. By leveraging Bayesian inference, we incorporate prior knowledge and constraints to improve source identification. To validate our approach, we implemented a Monte Carlo Markov Chain (MCMC) method alongside a Turing.jl model, allowing us to compare results and assess model reliability. Our implementation is tested on simulated data and is designed for application to real-world pollution datasets, such as those from ARPA Lombardia.

Key-words: source apportionment, Bayesian functional warping, infinite latent factors model

`https://github.com/GioAurina....`
`https://github.com/Paolofox01....`

1. Introduction

Environmental pollutants frequently exist as complex mixtures of chemical and non-chemical constituents. Using receptors (e.g., ambient monitors), we can measure individual constituents within a mixture. However, we lack direct information on the sources that generated them, nor do we know the contribution of each constituent to the total pollution.
Particulate matter (PM) and other airborne pollutants significantly affect air quality and human health. For example, complex pollutant mixtures like PM are often formed by multiple sources (e.g., biomass burning or traffic). Identifying these sources and quantifying their contributions is crucial for environmental policy and mitigation strategies.
Traditional source apportionment methods, such as Principal Component Analysis (PCA) and Absolute Principal Component Analysis (APCA), have limitations, including violations of non-negativity constraints and unrealistic assumptions about source independence. Most models for source apportionment represent observed constituents at a receptor as a linear combination of the unknown mixture concentrations from each source and the constituent composition of each source. However, these models often lack uniqueness and require additional constraints to provide meaningful solutions.
To address these challenges, we propose a novel spatio-temporal Bayesian model for source apportionment. This Bayesian framework allows for the incorporation of prior knowledge, enforces realistic constraints on source contributions, and accounts for both spatial and temporal dependencies in pollutant concentrations. By modeling the total pollution mass emitted from each source over time and its contribution to different pollutants, our approach provides a more robust and interpretable analysis of pollution dynamics.

The primary objectives of this study are:

- To estimate the contribution of different pollution sources to observed pollutant concentrations.
- To incorporate spatial dependencies, ensuring a more accurate representation of pollution dispersion.
- To model the temporal evolution of pollution sources and identify potential shifts in source contributions over time.
- To improve inference accuracy through a Bayesian hierarchical approach, integrating expert knowledge and observational data.

By addressing these challenges, this work aims to provide a powerful tool for policymakers and environmental agencies to better understand pollution sources and develop targeted mitigation strategies.

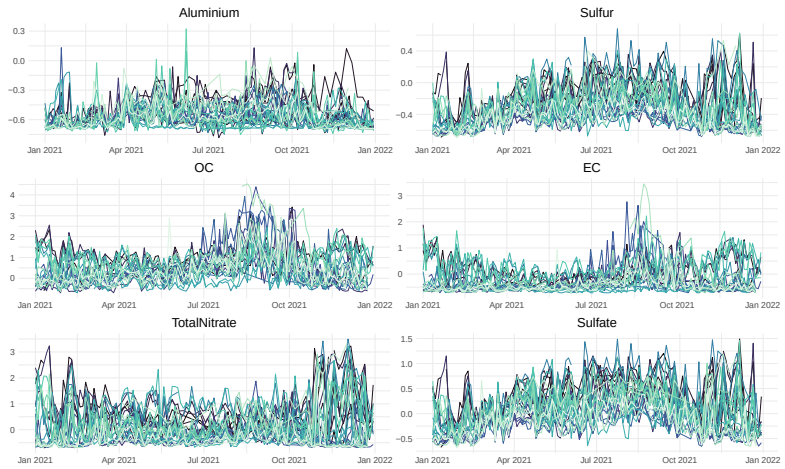

Figure 1: Different pollutants measured over time

2. Modeling

In this section, we present the detailed modeling approach used for source apportionment of PM10 pollution. The proposed model incorporates spatial and temporal dependencies through a Bayesian hierarchical framework. We introduce key components of the model step by step, providing mathematical formulations and explanations.

2.1. Notation in our hierarchical model

We define the following parameters and variables:

- $i = 1, \ldots, N$: indices for the monitoring sites.
- $k = 1, \ldots, K$: indices for the pollution sources.
- $c = 1, \ldots, C$: indices for the different pollutants under analysis.
- $t = 1, \ldots, T$: indices for the time steps (e.g., days).

The observed concentration of pollutant c at site s_i on day t is assumed to follow a Gaussian distribution [1]:

$$y^c(\boldsymbol{s}_i, t) \mid \mu^c(\boldsymbol{s}_i, t), \sigma_c^2 \overset{ind}{\sim} \mathcal{N}\left(\mu^c(\boldsymbol{s}_i, t), \sigma_c^2\right) \quad i = 1, \ldots, N, \quad c = 1, \ldots, C \tag{1}$$

where:

- $\mu^c(\boldsymbol{s}_i, t)$ represents the expected concentration, determined by the contribution of multiple sources.
- σ_c^2 accounts for the residual variance of pollutant C, with a marginal prior:

$$\sigma_c^2 \mid a, b \overset{iid}{\sim} \text{InvGamma}(a, b), \quad c = 1, ..., C \tag{2}$$

2.2. Source Contribution Decomposition

The expected concentration $\mu^c(\boldsymbol{s}_i, t)$ is modeled as a weighted sum of the contributions from the K pollution sources [4]:

$$\mu^c(\boldsymbol{s}_i, t) = \sum_{k=1}^{K} h_k^c g_k(\boldsymbol{s}_i, t) \tag{3}$$

where:

- h_k^c represents the impact of source k on pollutant c, with a marginal prior:

$$\boldsymbol{h}_k \mid \alpha_0 \overset{iid}{\sim} \text{Dirichlet}(\alpha_0, \ldots, \alpha_0), \quad k = 1, ..., K \tag{4}$$

- $g_k(\boldsymbol{s}_i, t)$ is the spatiotemporal intensity function of source k at site $\boldsymbol{s}_i$ and time t.

For $k = 1, \ldots, K$ and $i = 1, \ldots, N$, $t = 1, \ldots, T$ we model $g_k(\boldsymbol{s_i}, t)$ as follows:

$$g_k(\boldsymbol{s_i}, t) = e^{\gamma_{ki}} f(t - \tau_i) \tag{5}$$

reflecting Bayesian functional warping [5].

2.3. Temporal Dependence

To capture the temporal dynamics of the pollution sources, we introduce temporal dependencies through a Gaussian process [5]. Specifically, the temporal dependence of the pollution sources is modeled as follows:

$$\boldsymbol{f}_k \mid \rho_k \overset{ind}{\sim} \mathcal{N}_T(\mathbf{0}, \boldsymbol{\Sigma}_{f_k}) \tag{6}$$

where the temporal covariance is defined by:

$$\boldsymbol{\Sigma}_{f_k}[m,n] = \exp\left\{-\frac{\rho_k^2}{2}(t_m - t_n)^2\right\} \tag{7}$$

and the correlation between the source k and site i over time is modeled by:

$$\boldsymbol{\Sigma}_{ki}[m,n] = \exp\left\{-\frac{\rho_k^2}{2}(t_m - \tau_i - t_n)^2\right\} \tag{8}$$

where ρ_k is a parameter that controls the degree of temporal correlation, and τ_i models the temporal shift for site $\boldsymbol{s}_i$. The marginal prior on ρ_k is given by:

$$\rho_k \mid a_\rho, b_\rho \overset{iid}{\sim} \text{Gamma}(a_\rho, b_\rho) \quad \text{for } k = 1, \ldots, K \tag{9}$$

The temporal shifts τ_i are modeled as:

$$\boldsymbol{\tau} \sim \mathcal{N}_N\left(0, \sigma_\tau^2\left(\boldsymbol{I}_n - \frac{1}{N+1}\mathbf{1}\mathbf{1}'\right)\right), \tag{10}$$

where the marginal variance σ_τ^2 is chosen according to the plausible range of latency of our problem

2.4. Spatial Dependence

To account for spatial correlation, we introduce a Gaussian Process prior over the site-specific scaling factors. The spatial dependence of the sources at different monitoring sites is modeled as follows [2]:

$$\boldsymbol{\gamma}_k | \boldsymbol{\beta}_k, \phi_k \overset{ind}{\sim} \mathcal{N}_N(\boldsymbol{X}\boldsymbol{\beta}_k, \boldsymbol{\Sigma}_{\gamma_k}) \tag{11}$$

where $\boldsymbol{\Sigma}_{\gamma_k}[i,j]$ represents the spatial covariance between sites $\boldsymbol{s}_i$ and $\boldsymbol{s}_j$, which decays with the euclidean distance between the sites:

$$\boldsymbol{\Sigma}_{\gamma_k}[i,j] = \exp\left\{-\frac{\phi_k^2}{2}||\boldsymbol{s}_i - \boldsymbol{s}_j||^2\right\} \tag{12}$$

with ϕ_k controlling the spatial correlation length, modeled as:

$$\phi_k \mid a_\phi, b_\phi \overset{iid}{\sim} \text{Gamma}(a_\phi, b_\phi) \quad \text{for } k = 1, \ldots, K \tag{13}$$

The vector $\boldsymbol{X}_i$ contains the site-specific covariates, and $\boldsymbol{\beta}_k$ represents the regression coefficients, with a marginal prior:

$$\boldsymbol{\beta}_k \overset{iid}{\sim} \mathcal{N}_p(\mathbf{0}, \boldsymbol{I}_p) \tag{14}$$

3. Dataset

We use a dataset that primarily contains categorical information about the sites where pollutant measurements are taken. These categories include site type, location, and other relevant attributes that help contextualize the pollutant levels recorded at each site. This information is crucial for analyzing patterns and variations in air quality across different site characteristics.

FinalStations

Latitude	Longitude	Elevation	Local Site Name	County Name	City Name	LandUse	LandSetting
39.76168	-121.84047	1	Chico-East Avenue	Butte	Chico	IndustrialTraffic	SUBURBAN
41.560952	-124.083964	235	Redwood NP	Del Norte	Redwood National Park	LowPopulation	RURAL
38.9248	-119.97	1935	Lake Tahoe Community College	El Dorado	South Lake Tahoe	LowPopulation	RURAL
36.78538	-119.77321	96	Fresno - Garland	Fresno	Fresno	IndustrialTraffic	URBAN AND CENTER CITY
37.22064	-119.155557	2598	Kaiser Wilderness	Fresno	Not in a City	LowPopulation	RURAL
32.67618	-115.48307	1	Calexico-Ethel Street	Imperial	Calexico	ResidentialAgri	SUBURBAN
37.360684	-118.330783	1257	White Mountain Research Center - Owens Valley Lab	Inyo	Not in a City	ResidentialAgri	RURAL
35.356615	-119.062613	0	Bakersfield-California	Kern	Bakersfield	IndustrialTraffic	URBAN AND CENTER CITY
34.821922	-118.887598	1	Lebec-Peace Valley/Frazier Park Roads	Kern	Lebec	IndustrialTraffic	RURAL
34.06659	-118.22688	87	Los Angeles-North Main Street	Los Angeles	Los Angeles	ResidentialAgri	URBAN AND CENTER CITY
34.813034	-118.884819	1	Lebec	Los Angeles	Not in a City	IndustrialTraffic	RURAL
38.122979	-122.90944	76	Point Reyes NS Ranger Station	Marin	Point Reyes National Seashore	LowPopulation	RURAL
37.71325	-119.7062	1599	Yosemite NP - Turtleback Dome	Mariposa	Not in a City	LowPopulation	RURAL
38.088023	-119.178069	2561	Hoover Wilderness	Mono	Not in a City	LowPopulation	RURAL
33.83062	-117.93845	10	Anaheim	Orange	Anaheim	ResidentialAgri	SUBURBAN
39.81336	-120.47069	8	Portola	Plumas	Portola	ResidentialAgri	SUBURBAN
33.99958	-117.41601	250	Rubidoux	Riverside	Rubidoux	ResidentialAgri	SUBURBAN
33.463644	-116.971457	508	Aqua Tibia Wilderness	Riverside	Not in a City	LowPopulation	RURAL
38.613779	-121.368014	8	Sacramento-Del Paso Manor	Sacramento	Arden-Arcade	ResidentialAgri	SUBURBAN

Figure 2: California's locations

3.1. Simulated Data

To evaluate the proposed approach, we generate a synthetic dataset that mimics real-world PM10 pollution trends. The dataset is created by simulating pollution levels across multiple monitoring sites, taking into account spatial and temporal variations.
The generation process starts by assigning each monitoring station a set of attributes, including location coordinates, elevation, and site classification (e.g., urban or suburban). These characteristics influence the expected pollution levels and their variation over time.
For each site, pollution levels are generated based on a combination of site-specific conditions and general trends observed in air quality data. The simulated pollution values reflect daily and seasonal fluctuations, ensuring that the dataset captures realistic patterns Figure 3.
This synthetic dataset serves as a controlled environment for testing and validating the source apportionment model before applying it to actual observations. By doing so, we ensure that our approach is robust and capable of identifying pollution sources accurately.

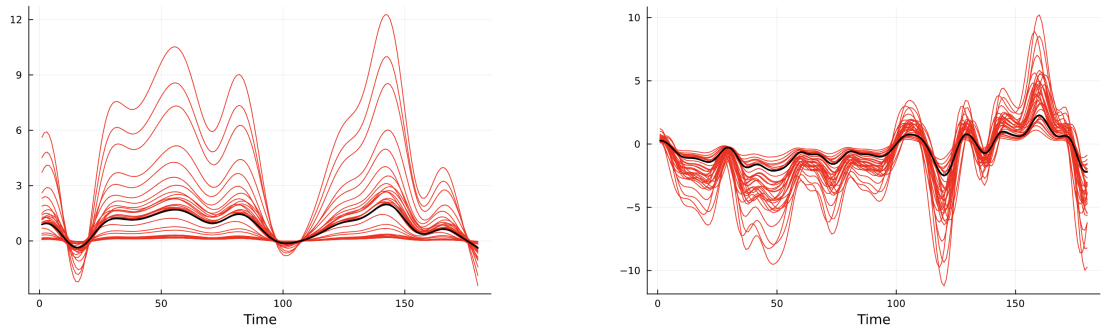

Figure 3: Global (black) and local contributions in each site i of source k, i=1...32, k=1,2

3.2. Generating data in Julia

```
function simulate_data(df, seed, K, n, n_time)
    Random.seed!(seed)

    # Parameters
    # K = 2 #number of sources
    # n = 32 #number of sites
    # n_time = 180

    intercept = ones(n)
    x1 = zeros(n)
    x2 = zeros(n)

    for cont in 1:n
        if df[cont, 8] == "SUBURBAN"
            x1[cont] = 1
        end

        if df[cont, 8] == "URBAN_AND_CENTER_CITY"
            x2[cont] = 1
        end

    end

    sites = Matrix(DataFrame(Latitude = df[:, 1], Longitude = df[:, 2],
       intercept = intercept, x1 = x1, x2=x2, Elevation = df[:, 3]))
    maximum(euclid_dist(sites[: , 1], sites[: , 2], n))

    theta = Dict{Int64, Dict{Any,Any}}()
    theta[1] = Dict(
        :rho => 0.1,
        :phi => 1/300.0,
        :gamma => zeros(32),
        :beta => [-0.5, 0.5, 0.8, 0.1],
        :tau => rand(Normal(0, 1), 32)
    )

    theta[2] = Dict(
        :rho => 0.2,
        :phi => 1/400.0,
        :gamma => zeros(32),
        :beta => [0.3, -0.4, -0.7, -0.1],
        :tau => rand(Normal(0, 1), 32)
    )

    dat = generate_data(sites, n, K, n_time, theta)

    theta_true = copy(theta)
    df_new = Dict{Int64, DataFrame}()
    dat_trials = Dict{Int64, DataFrame}()
    for k in 1:K
        theta_true[k][:gamma] = dat[:gamma][:,k]
        df_new[k] = DataFrame(dat[:g][:, k, :], :auto)
        dat_trials[k] = stack(df_new[k], variable_name = "time",
            value_name = "value")
        dat_trials[k].trial = repeat(1:n,n_time)
    end
```

```
#' Generate data.
#'
#' @param n Number of trials.
#' @param n_time Number of time points.
#' @param theta Named list of parameter values.
function generate_data(sites, n, K, n_time, theta)

    X = sites[:, 3:end] # Design matrix
    coords = sites[:, 1:2]

    t = range(1, stop=n_time, length=n_time)
    f = zeros(Float64, K, n_time)
    g = zeros(Float64, n, K, n_time)
    gamma = zeros(Float64, n, K)

    dist = euclid_dist(coords[:, 1], coords[:, 2], n)

    for k in 1:K
        theta_k = theta[k]
        Sigma_f = sq_exp_kernel(t, theta_k[:rho]; nugget=1e-6)
        Sigma_f_inv = inv(Sigma_f)
        f[k, :] = rand(MvNormal(zeros(n_time), Sigma_f))

        Sigma_gamma = get_Sigma_gamma(dist, theta_k[:phi])
        gamma[:, k] = rand(MvNormal(X*theta_k[:beta], Sigma_gamma))
        theta_k[:gamma] = gamma[:, k]
        for i in 1:n
            Sigma_i = get_Sigma_i(i, t, theta_k)

            g[i, k, :] = get_mu_g(i, t, f[k,:], theta_k, Sigma_f_inv)
        end

    end

    return Dict(:g => g, :f => f, :gamma => gamma)
end
```

4. Sample from the Joint Posterior Distribution

4.1. Full Conditional of g_k

We have the following joint distribution:

$$\mathcal{L}\left(\boldsymbol{f}_k, \boldsymbol{g}_k(\boldsymbol{s}_i)\right) = \begin{pmatrix} f_k(1) \\ \vdots \\ f_k(T) \\ a_i f_k(1-\tau_i) \\ \vdots \\ a_i f_k(T-\tau_i) \end{pmatrix} \sim \mathcal{N}_{2T}\left(\begin{pmatrix}\mathbf{0}\\ \mathbf{0}\end{pmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_{f_k} & a_i\boldsymbol{\Sigma}_{ki} \\ a_i\boldsymbol{\Sigma}_{ki}^T & a_i^2\boldsymbol{\Sigma}_{f_k} \end{bmatrix}\right),$$

where $a_i = e^{\gamma_{ki}}$.
This representation shows that $\boldsymbol{g}_k(\boldsymbol{s}_i)$ is the same Gaussian process of $\boldsymbol{f}_k$ but evaluated at new points, effectively performing temporal kriging over τ_i, i = 1,..., 32. This allows us to derive the conditional distribution:

$$\boldsymbol{g}_k(\boldsymbol{s}_i)|\boldsymbol{f}_k, \gamma_{k_i}, \tau_i \overset{ind}{\sim} \mathcal{N}_T(\boldsymbol{\mu}_{g_{ki}}, \boldsymbol{\Sigma}_{g_{ki}}), \quad i = 1, \ldots, N, \ k = 1, 2$$

where the mean and covariance are given by:

$$\boldsymbol{\mu}_{g_{ki}} = e^{\gamma_{k_i}} \boldsymbol{\Sigma}_{k_i} \boldsymbol{\Sigma}_{f_k}^{-1} \boldsymbol{f}_k$$

$$\boldsymbol{\Sigma}_{g_{ki}} = e^{2\gamma_{k_i}} (\boldsymbol{\Sigma}_{f_k} - \boldsymbol{\Sigma}_{k_i} \boldsymbol{\Sigma}_{f_k}^{-1} \boldsymbol{\Sigma}_{k_i}^T + \epsilon \boldsymbol{I})$$

This result follows from standard Gaussian conditioning, where $\boldsymbol{g}_k(\boldsymbol{s}_i)$ inherits the properties of the underlying process $\boldsymbol{f}_k$, but is adjusted according to the transformation $a_i f(t - \tau_i)$.
Note about the Matrices $\boldsymbol{\Sigma}_{f_k}$ and $\boldsymbol{\Sigma}_{k_i}$:
The temporal covariance $\boldsymbol{\Sigma}_{f_k}$ is defined by:

$$\boldsymbol{\Sigma}_{f_k}[m, n] = \exp\left\{-\frac{\rho_k^2}{2}(t_m - t_n)^2\right\}$$

where t_m and t_n are time indices, and ρ_k is the length-scale parameter for source k.
The correlation between the source k and site i over time is modeled by:

$$\boldsymbol{\Sigma}_{k_i}[m, n] = \exp\left\{-\frac{\rho_k^2}{2}(t_m - \tau_i - t_n)^2\right\}$$

where τ_i represents the shift for site i.
On the Nugget ϵ:
If $\tau_i = 0$, the variance of the posterior would approach zero, making $\boldsymbol{g}_k(\boldsymbol{s}_i)$ deterministic rather than stochastic. To preserve the stochastic nature of $\boldsymbol{g}_k(\boldsymbol{s}_i)$ even when $\tau_i = 0$, we introduce a nugget term ϵ. This nugget is modeled as a small noise term, drawn from a normal distribution with mean zero and a small variance, i.e., $\epsilon \sim \mathcal{N}(0, b_\epsilon)$, where b_ϵ is a fixed parameter.

4.2. Full conditional of β_k

We assume the following model:

$$\boldsymbol{\gamma}_k \mid \boldsymbol{\beta}_k, \phi_k \overset{ind}{\sim} \mathcal{N}_N\left(\boldsymbol{X}\boldsymbol{\beta}_k, \boldsymbol{\Sigma}_{\gamma_k}\right)$$

$$\boldsymbol{\Sigma}_{\gamma_k}[i, j] = \exp\left(-\frac{\phi_k^2}{2}\|\boldsymbol{s}_i - \boldsymbol{s}_j\|^2\right)$$

The marginal prior on $\boldsymbol{\beta}_k$ is:

$$\boldsymbol{\beta}_k \overset{iid}{\sim} \mathcal{N}_P\left(\mathbf{0}, \boldsymbol{I}_p\right)$$

where P represents the number of covariate
Applying Bayes' theorem, the full conditional is proportional to:

$$\boldsymbol{\beta}_k \mid \boldsymbol{\gamma}_k, \phi_k \propto \mathcal{L}(\boldsymbol{\gamma}_k \mid \boldsymbol{\beta}_k, \phi_k)\pi(\boldsymbol{\beta}_k)$$

Expanding the law of $\boldsymbol{\gamma}_k$ given the rest:

$$\mathcal{L}(\boldsymbol{\gamma}_k \mid \boldsymbol{\beta}_k, \phi_k) \propto \exp\left(-\frac{1}{2}(\boldsymbol{\gamma}_k - \boldsymbol{X}\boldsymbol{\beta}_k)^T \boldsymbol{\Sigma}_{\gamma_k}^{-1}(\boldsymbol{\gamma}_k - \boldsymbol{X}\boldsymbol{\beta}_k)\right)$$

Expanding the marginal prior:

$$\pi(\boldsymbol{\beta}_k) \propto \exp\left(-\frac{1}{2}\boldsymbol{\beta}_k^T \boldsymbol{I}_p \boldsymbol{\beta}_k\right)$$

Rearranging terms:

$$\propto \exp\left[-\frac{1}{2}\left((\boldsymbol{\gamma}_k - \boldsymbol{X}\boldsymbol{\beta}_k)^T \boldsymbol{\Sigma}_{\gamma_k}^{-1}(\boldsymbol{\gamma}_k - \boldsymbol{X}\boldsymbol{\beta}_k) + \boldsymbol{\beta}_k^T \boldsymbol{\beta}_k\right)\right]$$

$$\propto \exp\left[-\frac{1}{2}\left(-2\boldsymbol{\gamma}_k^T \boldsymbol{\Sigma}_{\gamma_k}^{-1} X \boldsymbol{\beta}_{\boldsymbol{k}} + \boldsymbol{\beta}_k^T \boldsymbol{X}^T \boldsymbol{\Sigma}_k^{-1} \boldsymbol{X}\boldsymbol{\beta}_k + \boldsymbol{\beta}_k^T \boldsymbol{\beta}_k\right)\right]$$

$$\propto \exp\left[-\frac{1}{2}\left(-2\boldsymbol{\gamma}_k^T \boldsymbol{\Sigma}_{\gamma_k}^{-1} X \boldsymbol{\beta}_{\boldsymbol{k}} + +\boldsymbol{\beta}_k^T (\boldsymbol{X}^T \boldsymbol{\Sigma}_k^{-1} \boldsymbol{X} + \boldsymbol{I}_p)\boldsymbol{\beta}_k\right)\right]$$

Defining:

$$\boldsymbol{S} = \boldsymbol{X}^T \boldsymbol{\Sigma}_{\gamma_k}^{-1} \boldsymbol{X} + \boldsymbol{I}_p,$$

$$\propto \exp\left[-\frac{1}{2}\left(-2\boldsymbol{\gamma}_k^T \boldsymbol{\Sigma}_{\gamma_k}^{-1} X \boldsymbol{S}^{-1} \boldsymbol{S}\boldsymbol{\beta}_{\boldsymbol{k}} + +\boldsymbol{\beta}_k^T \boldsymbol{S}\boldsymbol{\beta}_k\right)\right]$$

Defining:

$$\boldsymbol{m}^T = \boldsymbol{\gamma}_{\boldsymbol{k}}^T \boldsymbol{\Sigma}_{\gamma_k}^{-1} \boldsymbol{X}\boldsymbol{S}^{-1}$$

$$\propto \exp\left[-\frac{1}{2}\left(-2\boldsymbol{m}^T \boldsymbol{S}\boldsymbol{\beta}_{\boldsymbol{k}} + +\boldsymbol{\beta}_k^T \boldsymbol{S}\boldsymbol{\beta}_k\right)\right]$$

Now we look for a quadratic form, getting rid of terms which do not depend on $\boldsymbol{\beta}_k$. We retrieve:

$$\propto \exp\left[-\frac{1}{2}(\boldsymbol{\beta}_k - \boldsymbol{m})^T \boldsymbol{S}(\boldsymbol{\beta}_k - \boldsymbol{m})\right]$$

Thus, the full conditional follows in close form:

$$\boldsymbol{\beta}_k \mid \boldsymbol{\gamma}_k, \phi_k \overset{ind}{\sim} \mathcal{N}_P(\boldsymbol{m}, \boldsymbol{S}^{-1})$$

and the final Expression follows:

$$\boldsymbol{\beta}_k \mid \boldsymbol{\gamma}_k, \phi_k \overset{ind}{\sim} \mathcal{N}_P\left((\boldsymbol{X}^T \boldsymbol{\Sigma}_{\gamma_k}^{-1} \boldsymbol{X} + \boldsymbol{I}_p)^{-T} \boldsymbol{X}^T \boldsymbol{\Sigma}_{\gamma_k}^{-T} \boldsymbol{\gamma}_k, (\boldsymbol{X}^T \boldsymbol{\Sigma}_{\gamma_k}^{-1} \boldsymbol{X} + \boldsymbol{I}_p)^{-1}\right) \tag{15}$$

4.3. Full conditional of ϕ_k

Since in the Metropolis Hasting steps the proposal must be positive, we work with a transformed variable:

$$\theta_k = \log(\phi_k) \tag{16}$$

s.t $\phi_k = e^{\theta_k}$

The Gaussian proposal is then:

$$q(\theta^*|\theta^{\text{old}}) = \mathcal{N}(\theta^{\text{old}}, \varepsilon) \tag{17}$$

and the marginal prior transforms as:

$$\pi_{\theta_k}(\theta_k) = \pi_{\phi_k}(e^{\theta_k})e^{\theta_k} \tag{18}$$

Thus, the MH target function is:

$$\mathcal{L}(\boldsymbol{\gamma}_k|\boldsymbol{\beta}_k, e^{\theta_k})\pi_{\phi_k}(e^{\theta_k})e^{\theta_k} \tag{19}$$

4.4. Full conditional of τ

The MH target function is:

$$\mathcal{L}(\boldsymbol{\tau}|\boldsymbol{g_k}(\boldsymbol{s_i}), \boldsymbol{\gamma}_k, \boldsymbol{f}_k) \propto \prod_{i=1}^{N} \mathcal{L}(\boldsymbol{g_k}(\boldsymbol{s_i})|\boldsymbol{\gamma}_{ki}, \boldsymbol{\tau}_i, \boldsymbol{f}_k)\pi(\boldsymbol{\tau}) \tag{20}$$

4.5. Full conditional of ρ_k

Defining $\theta_k = \log(\rho_k)$, we get:

$$\mathcal{L}(\rho_k|\boldsymbol{g_k}(\boldsymbol{s_i})\boldsymbol{\gamma}_k, \boldsymbol{\tau}, \boldsymbol{f}_k) \propto \prod_{i=1}^{N} \mathcal{L}(\boldsymbol{g_k}(\boldsymbol{s_i})|\boldsymbol{\gamma}_{ki}, \boldsymbol{\tau}_i, \boldsymbol{f}_k)\mathcal{L}(\boldsymbol{f_k}|\rho_k)\pi_{\rho_k}(e^{\theta_k})e^{\theta_k} \tag{21}$$

4.6. Full conditional of f_k

It is trivial to show that it follows a closed-form Gaussian distribution [5]:

$$\mathcal{L}(\boldsymbol{f}_k|\boldsymbol{G}_k, \boldsymbol{\gamma}_k, \boldsymbol{\tau}, \rho_k) \propto \mathcal{L}(\boldsymbol{G}_k|\boldsymbol{\gamma}_k, \boldsymbol{\tau}, \boldsymbol{f}_k)\mathcal{L}(\boldsymbol{f}_k|\rho_k) \tag{22}$$

and so we get:

$$\boldsymbol{f}_k|\boldsymbol{G}_k, \boldsymbol{\gamma}_k, \boldsymbol{\tau}, \rho_k \sim \mathcal{N}_T(\boldsymbol{\mu}_{f_k\text{-post}}, \boldsymbol{\Sigma}_{f_k\text{-post}}) \tag{23}$$

$$\boldsymbol{\Sigma}_{f_k\text{-post}} = (\boldsymbol{\Sigma}_{f_k}^{-1} + \sum_{i=1}^{n}(\boldsymbol{\Sigma}_{k_i}\boldsymbol{\Sigma}_{f_k}^{-1})^T\boldsymbol{\Sigma}_{g_{ki}}^{-1}\boldsymbol{\Sigma}_{k_i}\boldsymbol{\Sigma}_{f_k}^{-1})^{-1} \tag{24}$$

$$\boldsymbol{\mu}_{f_k\text{-post}} = \left(\boldsymbol{\Sigma}_{f_k}^{-1} + \sum_{i=1}^{n}\left(\boldsymbol{\Sigma}_{k_i}\boldsymbol{\Sigma}_{f_k}^{-1}\right)^T\boldsymbol{\Sigma}_{g_{ki}}^{-1}\boldsymbol{\Sigma}_{k_i}\boldsymbol{\Sigma}_{f_k}^{-1}\right)^{-1}\sum_{i=1}^{n}\boldsymbol{G}_k[i,:]^T\boldsymbol{\Sigma}_{g_{ki}}^{-1}\boldsymbol{\Sigma}_{k_i}\boldsymbol{\Sigma}_{f_k}^{-1} \tag{25}$$

This result follows from standard Gaussian conditioning, a more general case is shown in Vannucci's paper. To understand the updating formulas for $\boldsymbol{\mu}_{f\text{-post}}, \boldsymbol{\Sigma}_{f\text{-post}}$ we invite the reader to check the code on Github.

4.7. Full conditional of γ_k

The full conditional distribution of $\boldsymbol{\gamma}_k$ is not available in closed form. Moreover, Metropolis-Hastings methods tend to perform poorly in high-dimensional or highly correlated multivariate settings. To mitigate this issue, we propose an alternative approach: instead of sampling $\boldsymbol{\gamma}_k$ jointly, we marginalize over $\boldsymbol{\gamma}_k$ and work directly with its conditional components γ_{ki}.

To illustrate this, we start by considering the distribution of $\boldsymbol{\gamma}_k$, that here will be denoted ad $\boldsymbol{\gamma}$ to make the notation easier:

$$\boldsymbol{\gamma} = \begin{pmatrix} \gamma_1 \\ \vdots \\ \gamma_N \end{pmatrix} = \begin{pmatrix} \boldsymbol{\gamma}_{-N} \\ \gamma_N \end{pmatrix} \sim \mathcal{N}_N \left(\begin{pmatrix} X_{[-N,:]}\boldsymbol{\beta} \\ X_{[N,:]}\boldsymbol{\beta} \end{pmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_{[-N,-N]} & \boldsymbol{\Sigma}_{[-N,N]} \\ \boldsymbol{\Sigma}_{(N,-N)} & \boldsymbol{\Sigma}_{(N,N)} \end{bmatrix} \right).$$

From this, we derive the marginal distribution of γ_N given $\boldsymbol{\gamma}_{-N}$:

$$\gamma_N \mid \boldsymbol{\gamma}_{-N} \overset{ind}{\sim} \mathcal{N}\left(\mu_{N_{\text{marginal}}}, \Sigma_{N_{\text{marginal}}}\right).$$

We now focus on the conditional structure of $\boldsymbol{\gamma}_k$. Specifically, we consider:

$$\gamma_{ki} \mid \boldsymbol{\gamma}_k^{(-i)}, \boldsymbol{\tau}, \boldsymbol{f}_k \propto \mathcal{L}(\boldsymbol{g}_k(s_i) \mid \gamma_{ki}, \tau, \boldsymbol{f}_k)\mathcal{L}(\gamma_{ki} \mid \boldsymbol{\gamma}_k^{(-i)}). \tag{26}$$

Using the properties of the multivariate normal distribution, the conditional distribution of γ_{k_i} given $\boldsymbol{\gamma}_k^{(-i)}$ follows:

$$\begin{aligned} \gamma_{ki} \mid \boldsymbol{\gamma}_k^{(-i)} \overset{ind}{\sim} \mathcal{N}\Big(& \boldsymbol{X}_i^\top \beta_k + \boldsymbol{\Sigma}_{\gamma_k}[i,-i]\left(\boldsymbol{\Sigma}_{\gamma_k}[-i,-i]\right)^{-1}\left(\boldsymbol{\gamma}_k^{(-i)} - \boldsymbol{X}_{[-i,:]}\beta_k\right), \\ & \boldsymbol{\Sigma}_{\gamma_k}[i,i] - \boldsymbol{\Sigma}_{\gamma_k}[i,-i]\left(\boldsymbol{\Sigma}_{\gamma_k}[-i,-i]\right)^{-1}\boldsymbol{\Sigma}_{\gamma_k}[-i,i]\Big). \end{aligned} \tag{27}$$

$$\boldsymbol{\Sigma}_{\gamma_k} = \begin{bmatrix} \boldsymbol{\Sigma}_{[-i,-i]} & \boldsymbol{\Sigma}_{[-i,i]} \\ \boldsymbol{\Sigma}_{[i,-i]} & \boldsymbol{\Sigma}_{[i,i]} \end{bmatrix}$$

NB **:** Following the R language alike notation, $\boldsymbol{\Sigma}_{\gamma_k}[-i,-i]$ indicates the matrix $\boldsymbol{\Sigma}_{\gamma_k}$ without the i-th row and the i-th column (Dim = (number of sites -1) x (number of sites -1)), $\boldsymbol{\Sigma}_{\gamma_k}[i,-i]$ indicates only the i-th row of $\boldsymbol{\Sigma}_{\gamma_k}$, without the only element in the i-th column of the matrix (Dim = 1 x (number of sites -1)), $\boldsymbol{\Sigma}_{\gamma_k}[-i,i]$ indicates only the i-th column of $\boldsymbol{\Sigma}_{\gamma_k}$, without the only element in the i-th row of the matrix (Dim = (number of sites -1) x 1), and $\boldsymbol{\Sigma}_{\gamma_k}[i,i]$ is the element in the i-th row and i-th column of $\boldsymbol{\Sigma}_{\gamma_k}$.

Thus, the Metropolis-Hastings target function is:

$$\mathcal{L}(\boldsymbol{g}_k(s_i) \mid \gamma_{ki}, \tau_i, \boldsymbol{f}_k)\mathcal{L}(\gamma_{ki} \mid \boldsymbol{\gamma}_k^{(-i)}). \tag{28}$$

Computational Advantages:
This approach simplifies the sampling process by reducing the dimensionality of the sampling space. Instead of handling the full joint posterior of $\boldsymbol{\gamma}_k$, we iteratively sample its components γ_{ki}. This improves computational efficiency and convergence properties.

5. MCMC Algorithm

Given the full conditionals derived in the previous section, we can now proceed to implement the following MCMC algorithm in order to sample from the joint posterior distribution, with a Gibbs sampler for parameters for which we have a known closed form, and Metropolis Hasting steps when the closed form is not available.

Initialize $\gamma_k^{(0)}, f_k^{(0)}, \beta_k^{(0)}, \Phi_k^{(0)}, \tau^{(0)}, \rho_k^{(0)}$
for *iter* $\in 1 : num.iterations$ **do**
  Update $f_k^{(\text{iter})} | \gamma_k^{(\text{iter}-1)}, \tau^{(\text{iter}-1)}, \rho_k^{(\text{iter}-1)}, G_k^{(\text{iter-1})}$ **(closed-form)**
  Update $\beta_k^{(\text{iter})} | \gamma_k^{(\text{iter}-1)}, \Phi_k^{(\text{iter}-1)}$ **(closed-form)**
  Update $\Phi_k^{(\text{iter})} | \gamma_k^{(\text{iter}-1)}, \beta_k^{(\text{iter})}, g_k^{(\text{iter-1})}$
  for $i \in 1 : N$ **do**
    Update $\gamma_{ki}^{(\text{iter})} | \gamma_{k(-i)}^{(\text{iter})}, \tau_i^{(\text{iter}-1)}, f_k^{(\text{iter})}, \beta_k^{(\text{iter})}, \Phi_k^{(\text{iter})}$ **(MH on single marginal)**
  end
  Update $\tau^{(\text{iter})} | \gamma_k^{(\text{iter})}, f_k^{(\text{iter})}, g_k^{(\text{iter-1})}$
  Update $\rho_k^{(\text{iter})} | \gamma_k^{(\text{iter})}, \tau^{(\text{iter})}, f_k^{(\text{iter})}, g_k^{(\text{iter-1})}$
end

6. MCMC structure in Julia

```
function fit_model(sites, g, n_iter, theta0, hyperparam)
    chain = Vector{Any}(undef, n_iter)
    chain_f = Vector{Any}(undef, n_iter)
    chain_g = Vector{Any}(undef, n_iter)
    chain_z = Vector{Any}(undef, n_iter)
    chain_beta = Vector{Any}(undef, n_iter)
    chain_gamma = Vector{Any}(undef, n_iter)
    chain_tau = Vector{Any}(undef, n_iter)
    chain_rho = Vector{Any}(undef, n_iter)
    chain_phi = Vector{Any}(undef, n_iter)
    n_time = size(g, 2)
    n = size(g, 1)

    t = range(1, stop=n_time, length=n_time)
    dist = euclid_dist(sites[:, 1], sites[:, 2], n)
    X = sites[:, 3:6]

    chain[1] = copy(theta0)

    chain_beta[1] = theta0[:beta]
    chain_tau[1] = theta0[:tau]
    chain_gamma[1] = theta0[:gamma]
    chain_rho[1] = theta0[:rho]
    chain_phi[1] = theta0[:phi]
    chain_f[1] = sample_f(g, chain[1], 1)
    Sigma_f = sq_exp_kernel(t, chain[1][:rho], nugget = 1e-9)
    Sigma_f_inv = inv(Sigma_f)

    chain_g[1] = get_mu_g_matrix(g, chain_f[1], t, chain[1], Sigma_f_inv)
    chain_z[1] = g - chain_g[1]

```

```
    start = time()
    Sigma_gamma = get_Sigma_gamma(dist, chain[1][:phi])

    for iter in 2:n_iter
        if (iter/n_iter*100) % 10 == 0.0
            println("...", floor(Int, (iter / n_iter) * 100), "%")
        end

        curr = copy(chain[iter - 1])
        f = sample_f(g, curr, 1)

        curr[:beta] = sample_beta(curr, hyperparam, Sigma_gamma, X)
        curr[:phi], Sigma_gamma = sample_phi(dist, X, curr, hyperparam)
        curr[:gamma] = sample_gamma(t, g, f, curr, Sigma_f, Sigma_f_inv,
            Sigma_gamma, X, hyperparam)
        curr[:tau] = sample_tau(t, g, f, curr, hyperparam, Sigma_f,
            Sigma_f_inv)
        curr[:rho] = sample_rho(t, g, f, curr, hyperparam)

        Sigma_f = sq_exp_kernel(t, curr[:rho], nugget = 1e-9)
        Sigma_f_inv = inv(Sigma_f)

        g_hat = get_mu_g_matrix(g, f, t, curr, Sigma_f_inv)
        z = g - g_hat

        chain_f[iter] = copy(f)
        chain[iter] = copy(curr)
        chain_g[iter] = copy(g_hat)
        chain_z[iter] = copy(z)
        chain_beta[iter] = copy(curr[:beta])
        chain_gamma[iter] = copy(curr[:gamma])
        chain_tau[iter] = copy(curr[:tau])
        chain_rho[iter] = copy(curr[:rho])
        chain_phi[iter] = copy(curr[:phi])
    end

    fine = time()
    runtime = fine - start
    println("Execution_Time:", runtime)

    return Dict(
        :chain => chain,
        :chain_f => chain_f,
        :chain_g => chain_g,
        :chain_z => chain_z,
        :chain_beta => chain_beta,
        :chain_gamma => chain_gamma,
        :chain_tau => chain_tau,
        :chain_phi => chain_phi,
        :chain_rho => chain_rho,
        :runtime => runtime
    )
end
```

7. Turing: Model Implementation in Julia

In this section, we present a complementary approach explored during the development of the MCMC. Specifically, we implemented a probabilistic model using Turing.jl to obtain preliminary results more efficiently, without manually constructing a custom MCMC algorithm.

7.1. Overview of Turing.jl

Turing.jl is a general-purpose probabilistic programming language (PPL) written entirely in Julia. It provides a streamlined approach to Bayesian inference, requiring the user to define only the probabilistic model—including priors and likelihood—within a function. Once the model is specified, Turing.jl automatically performs MCMC sampling through a dedicated command. This not only simplifies the implementation process but also enhances the readability and interpretability of the analysis.

7.2. Model Construction

The model follows a structure similar to the custom MCMC approach, iterating through the definition of priors and likelihoods up to the retrieval of the latent variables g. The final component of the Turing.jl model introduces an additional step for modeling the observed data y, as outlined below.
Observed data y is modeled as follows:

$$h_k \sim \text{Dirichlet}(\mathbf{1}) \tag{29}$$

$$\sigma_y \sim \text{InverseGamma}(3, 2) \tag{30}$$

$$\mu_{y,i,t,c} = \sum_{k=1}^{K} h_{k,c} g_{i,t,k} \tag{31}$$

$$y_{i,t,c} \sim \mathcal{N}(\mu_{y,i,t,c}, \sigma_y) \tag{32}$$

Here, y follows a normal distribution with mean μ_y and standard deviation σ_y, where μ_y is computed as a weighted sum of the latent variables g using the Dirichlet-distributed weights h.

8. Convergence Diagnostics and Posterior Inference

In this section, we begin by discussing the convergence diagnostics used to validate the MCMC chains, followed by detailed posterior inferences for the parameters $\mathbf{g}_k$, $\mathbf{f}_k$, γ_{ki}, and $\boldsymbol{\beta}_k$. Finally, we revisit convergence diagnostics by examining the autocorrelation behavior.
(*Note:* all results concern source K=2 and because of computational issues, we run a MCMC simulation with T=180.)

8.1. Convergence Diagnostic

Model validation and performance metrics.

- *Trace plots*: The MCMC trajectories should eventually look like white noise without a discernible trend, indicating stationarity. Starting chains at different initial values is a practical way to confirm that all chains converge to the same posterior region.
- *Markov chain standard error*: We computed the chain-specific standard error for each parameter. As the number of iterations increases, the chains mix well, and the standard errors stabilize.

8.2. Posterior Inference: $\mathrm{g}_k(\mathrm{s}_i, t)$

Retrieving the simulated data. Our MCMC of 11,000 iterations successfully recovers the pollutant levels $\mathbf{g}_k(\mathbf{s}_i, t)$ across different sites and time points. Figure 4 illustrates how the posterior mean (and associated credible intervals) align closely with the simulated data.

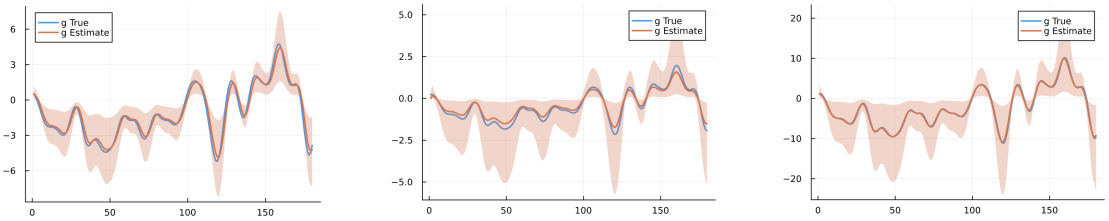

Figure 4: Retrieved $\boldsymbol{g}_k(\boldsymbol{s}_i, t)$ for sites $\{1, 2, 3\}$ with different scale of magnitude. Comparison with the "true" $\boldsymbol{g}_k$ along with its 0.95 credible interval

Across all sites, the model captures the temporal trends and magnitude of $\mathbf{g}_k$. The slight variations around the true values reflect the model's uncertainty quantification.

8.3. Posterior Inference: f_k

Recovering the second hierarchical level. We partially succeed in retrieving $\mathbf{f}_k$, subject to the normalization constraints. Up to a constant, we successfully recover the true function f_{true} (Figure 5). By standardizing both f_{true} and f_{estimate}, we observe a clear alignment between the two (Figure 6).
This agreement indicates that the Bayesian framework appropriately infers the underlying source composition while accounting for potential scaling ambiguities inherent in hierarchical models.

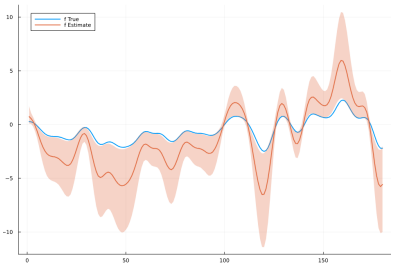

Figure 5: Retrieved original f

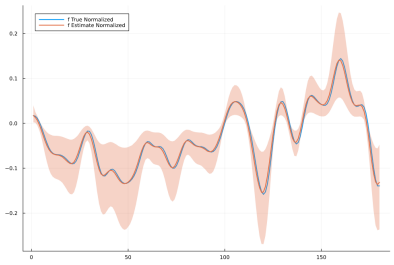

Figure 6: Retrieved adjusted f

8.4. Posterior Inference: γ_k

Normalization adjustment. Due to the normalization applied to $\mathbf{f}_k$, we adjust the retrieved γ ((Figure 7) accordingly. Specifically, we use:

$$\gamma_{\text{adjusted}} = \ln(\|\mathbf{f}\|) + \gamma,$$

which ensures consistency between the model's inference and the generative process (Figure 8).

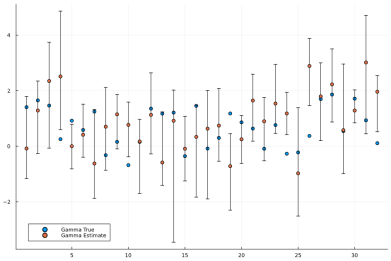

Figure 7: Retrieved γ_{ki}

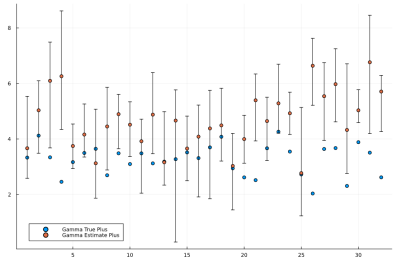

Figure 8: Adjusted γ_{ki}

The adjusted γ values align more closely with the physical interpretation of the model, correcting for the shift introduced by normalization.

8.5. Convergence Diagnostic: β_k

Trace plots and credible intervals. For the regression coefficients $\boldsymbol{\beta}_k$, we have a closed-form solution in each MCMC iteration, leading to well-mixed "fat caterpillar" trace plots (Figures 9). This visual pattern suggests efficient sampling. Moreover, the recovered values of $\boldsymbol{\beta}_k$ lie within the 0.95 credible intervals of their true values (except for β_0), indicating that the model is capturing these parameters accurately (Figure 10).

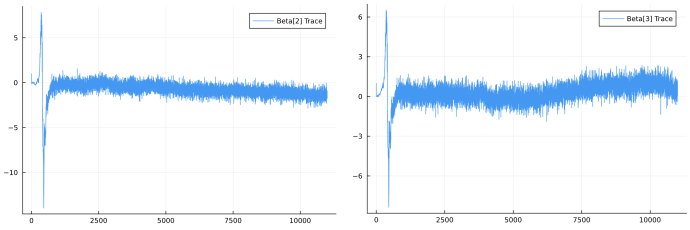

Figure 9: Traceplot of $\boldsymbol{\beta}_k[1]$ and $\boldsymbol{\beta}_k[2]$

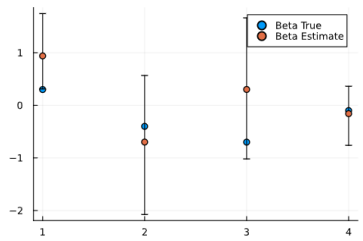

Figure 10: Estimated $\boldsymbol{\beta}_k$'s

The diagnostics confirm that the sampling scheme for $\boldsymbol{\beta}_k$ converges and yields estimates that reflect the true underlying parameter values.

8.6. Convergence Diagnostic: Autocorrelation

Assessing mixing quality. To further evaluate chain mixing, we examine the autocorrelation plots of key parameters. Ideally, these plots should show a decreasing pattern as the lag increases, confirming that each subsequent sample is becoming less dependent on its predecessor.

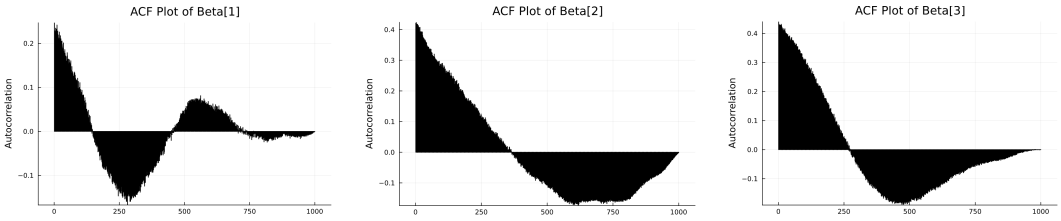

Figure 11: Autocorrelation of $\beta_0, \beta_1, \beta_2$

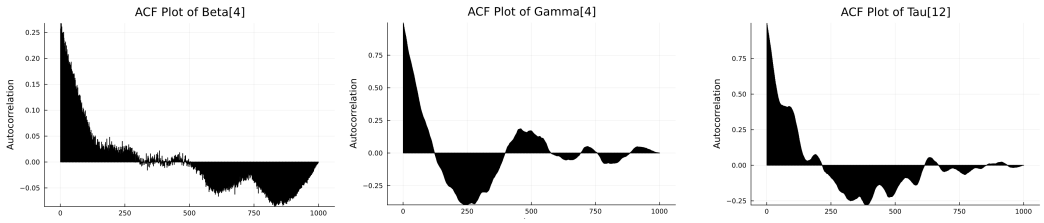

Figure 12: Autocorrelation of $\beta_3, \gamma_4, \tau_{23}$

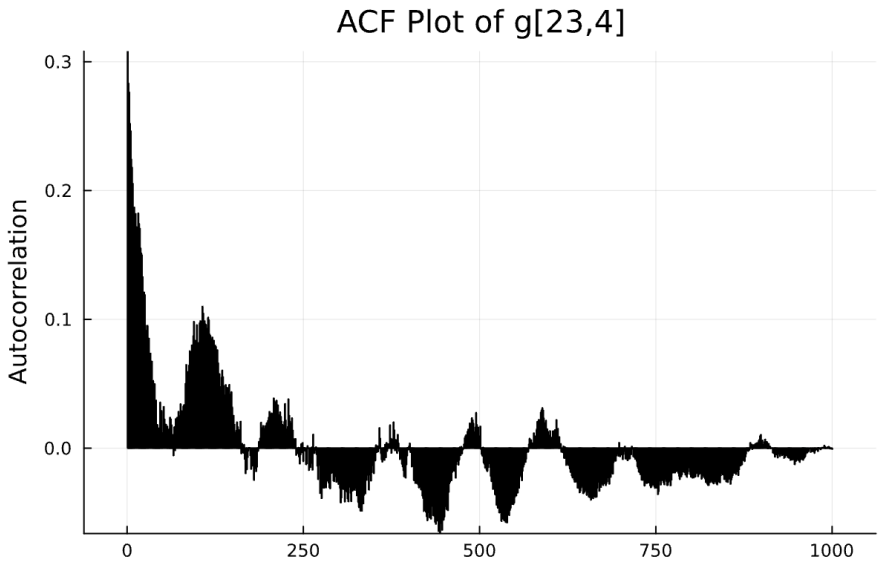

Figure 13: Autocorrelation of $g_{23,4}$

In practice, we apply thinning since the autocorrelation is high, in order to reduce bias and improve effective sample size. Here, the autocorrelation decreases rapidly with increasing lag, confirming that the chains mix efficiently.

Note: to see more convergence diagnostic insights, along with few issues that often appear with infinite latent factors model, see Appendix A and B.

9. Conclusions

In this study, we have successfully implemented a comprehensive Bayesian hierarchical framework to address complex spatial-temporal modeling challenges. The key findings of our work can be summarized as follows:

- **Implementation Pipeline:** Starting from Vannucci's model for random phase amplitude Gaussian processes in R, we developed a complex data generation pipeline in Julia. This pipeline is capable of producing synthetic data that closely mimics real observations by leveraging a rigorous Bayesian hierarchical model.
- **Customized MCMC Algorithm:** After extensive derivations of full conditional distributions—including efforts to identify closed-form solutions—we devised a customized Markov Chain Monte Carlo (MCMC) algorithm.
- **Parameter Retrieval:** Our approach successfully retrieves the target functions g_k that represent the first level of the model. In addition, key second-level parameters, such as β_k and f, are accurately estimated. These results demonstrate the model's capability to capture both the primary and secondary structures of the data.

Overall, the methodology proves to be effective in efficiently estimating parameters within intricate spatial-temporal frameworks, highlighting its potential for broader applications in environmental and other complex hierarchical models.

10. Possible Improvements

10.1. Extending MCMC to the Variable y

The next step is to extend the MCMC framework to incorporate the observed pollutant levels y, incorporating the positivity constraints.

10.2. Shrinkage in the RPAGP Model

In the initial formulation of the model, the number of pollutant sources k is fixed. However, in the context of the proposed shrinkage method [3], we introduce η_k, a parameter that controls the degree of shrinkage for each pollutant source. Specifically, the formulation becomes:

$$g_k(\boldsymbol{s}_i, t) = \gamma_{k,i} \cdot f_k(t - \tau_i)$$

for $k = 1, \ldots, K$, $i = 1, \ldots, N$, and $t = 1, \ldots, T$, where:

- $f_k \overset{\text{ind}}{\sim} \mathcal{N}_T^+\left(\mathbf{0}, \frac{1}{\eta_k}\Sigma_f\right)$, representing a Gaussian process with positive values, where the covariance structure is governed by $\Sigma_f(t', t) = \exp\left(-\frac{\rho^2}{2}(t' - t)^2\right)$, the squared exponential kernel.
- The parameter η_k is defined as:

  $$\eta_k = \prod_{l=1}^{k} \delta_l$$

  where $\delta_1 \sim \text{Gamma}(a_1, 1)$ and $\delta_l \sim \text{Gamma}(a_2, 1)$ for $l \geq 2$, introducing a hierarchical shrinkage effect.
- The length-scale parameter ρ governs the smoothness of the source functions and is assumed to follow a Gamma distribution:

  $$\rho \overset{\text{iid}}{\sim} \text{Gamma}(a_\rho, b_\rho)$$

  where a_ρ and b_ρ are hyperparameters selected based on a plausible range of values.

The introduction of η_k ensures that only those pollutant sources whose variance is sufficiently large are kept in the model, while redundant or insignificant sources are pruned away. This mechanism enables the model to adaptively select the appropriate number of sources.

10.3. Testing the Shrinkage Model on Real Data from ARPA

The next phase involves applying the model to real-world pollution data collected by ARPA in Lombardy. This data provides actual measurements of PM10 across various locations and times. Testing the model on this real data will allow us to assess its ability to identify the relevant pollutant sources in a practical, real-world setting. The real-world validation is crucial for confirming that the dynamic source selection and shrinkage mechanism can adapt to complex, real-world pollution scenarios.

References

[1] Oliver Baerenbold et al. *A dependent Bayesian Dirichlet process model for source apportionment of particle number size distribution*. 2022.

[2] Sudipto Banerjee, Bradley P Carlin, and Alan E Gelfand. *Hierarchical modeling and analysis for spatial data*. Chapman and Hall/CRC, 2003.

[3] A. Bhattacharya and D. B. Dunson. *Sparse Bayesian infinite factor models*. 2011.

[4] Jenna R. Krall and Howard H. Chang. *Statistical methods for source apportionment*. 2019.

[5] Dustin Pluta and Marina Vannucci. *Improved data quality and statistical power of trial-level event-related potentials with Bayesian random-shift Gaussian processes*. 2024.

Appendix A: Trace plots, nonidentifiability issues for latent parameters

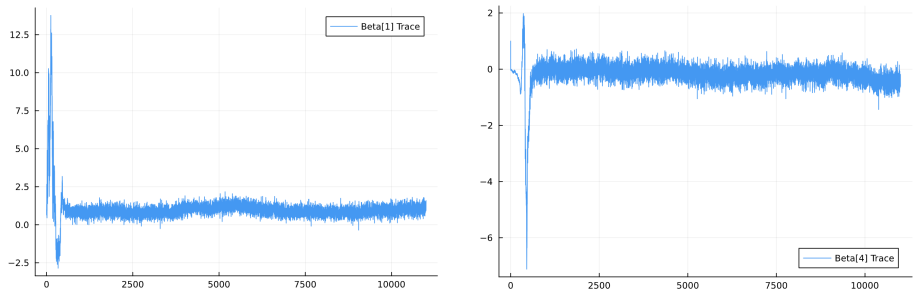

Figure 14: Trace plots of β_1, β_2

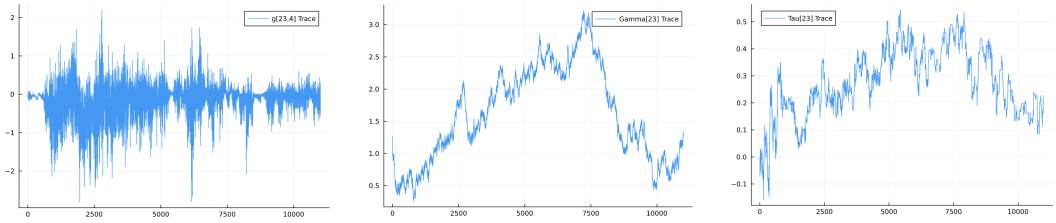

Figure 15: Trace plots of $g_{23,4}, \gamma_{23}, \tau_{23}$

Appendix B: Autocorrelation plots

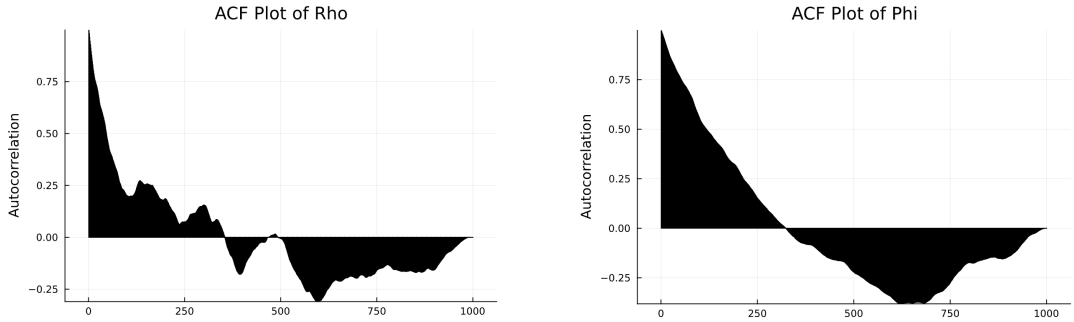

Figure 16: Autocorrelation of ρ_k and ϕ_k