Journal of Machine Learning Research 23 (2022) 1-40 Submitted 12/20; Revised 9/21; Published 2/22

Spatial Multivariate Trees for Big Data Bayesian Regression

Michele Peruzzi MICHELE.PERUZZI@DUKE.EDU
David B. Dunson DUNSON@DUKE.EDU
Department of Statistical Science
Duke University
Durham, NC 27708-0251, USA

Editor: John Cunningham

Abstract

High resolution geospatial data are challenging because standard geostatistical models based on Gaussian processes are known to not scale to large data sizes. While progress has been made towards methods that can be computed more efficiently, considerably less attention has been devoted to methods for large scale data that allow the description of complex relationships between several outcomes recorded at high resolutions by different sensors. Our Bayesian multivariate regression models based on spatial multivariate trees (SpamTrees) achieve scalability via conditional independence assumptions on latent random effects following a treed directed acyclic graph. Information-theoretic arguments and considerations on computational efficiency guide the construction of the tree and the related efficient sampling algorithms in imbalanced multivariate settings. In addition to simulated data examples, we illustrate SpamTrees using a large climate data set which combines satellite data with land-based station data. Software and source code are available on CRAN at `https://CRAN.R-project.org/package=spamtree`.

Keywords: Directed acyclic graph, Gaussian process, Geostatistics, Multivariate regression, Markov chain Monte Carlo, Multiscale/multiresolution.

1. Introduction

It is increasingly common in the natural and social sciences to amass large quantities of georeferenced data. Researchers seek to use these data to understand phenomena and make predictions via interpretable models that quantify uncertainty taking into account the spatial and temporal dimensions. Gaussian processes (GP) are flexible tools that can be used to characterize spatial and temporal variability and quantify uncertainty, and considerable attention has been devoted to developing GP-based methods that overcome their notoriously poor scalability to large data. The literature on scaling GPs to large scale is now extensive. We mention low-rank methods (Quiñonero-Candela and Rasmussen, 2005; Snelson and Ghahramani, 2007; Banerjee et al., 2008; Cressie and Johannesson, 2008); their extensions (Low et al., 2015; Ambikasaran et al., 2016; Huang and Sun, 2018; Geoga et al., 2020); methods that exploit special structure or simplify the representation of multidimensional inputs—for instance, a Toeplitz structure of the covariance matrix scales GPs to big time series data, and tensor products of scalable univariate kernels can be used for multidimensional inputs (Gilboa et al., 2015; Moran and Wheeler, 2020; Loper et al., 2020; Wu et al.,

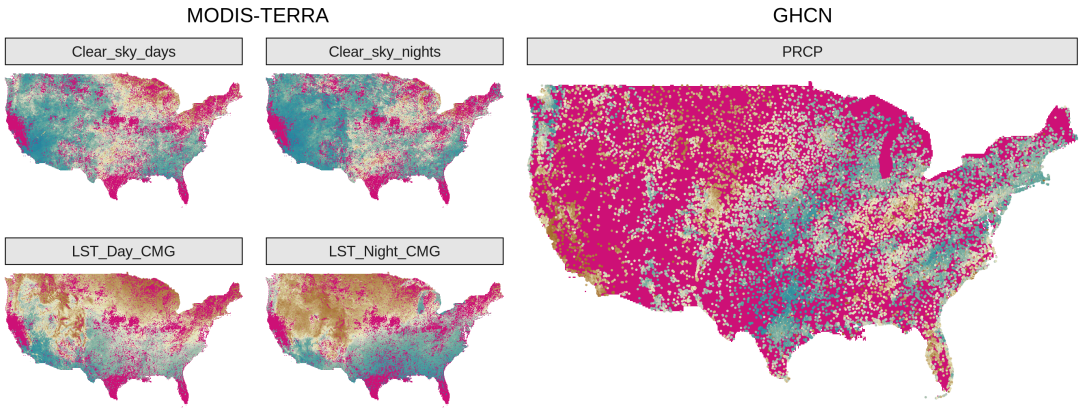

Figure 1: Observed data of Section 4.2. Missing outcomes are in magenta. GHCN data are much more sparsely observed compared to satellite imaging from MODIS.

2021). These methods may be unavailable or perform poorly in geostatistical settings, which focus on small-dimensional inputs, i.e. the spatial coordinates plus time. In these scenarios, low-rank methods oversmooth the spatial surface (Banerjee et al., 2010), Toeplitz-like structures are typically absent, and so-called *separable* covariance functions obtained via tensor products poorly characterize spatial and temporal dependence. To overcome these hurdles, one can use covariance tapering and domain partitioning (Furrer et al., 2006; Kaufman et al., 2008; Sang and Huang, 2012; Stein, 2014; Katzfuss, 2017) or composite likelihood methods and sparse precison matrix approximations (Vecchia, 1988; Rue and Held, 2005; Eidsvik et al., 2014); refer to Sun et al. (2011), Banerjee (2017), Heaton et al. (2019) for reviews of scalable geostatistical methods.

Additional difficulties arise in multivariate (or multi-output) regression settings. Multivariate geostatistical data are commonly misaligned, i.e. observed at non-overlapping spatial locations (Gelfand et al., 2010). Figure 1 shows several variables measured at non-overlapping locations, with one measurement grid considerably sparser than the others. In these settings, replacing a multi-output regression with separate single-output models is a valid option for predicting outcomes at new locations. While single-output models may sometimes perform equally well or even outperform multi-output models, they fail to characterize and estimate cross-dependences across outputs; testing the existence of such dependences may be scientifically more impactful than making predictions. This issue can be solved by modeling the outputs via latent spatial random effects thought of as a realization of an underlying multivariate GP and embedded in a larger hierarchical model.

Unfortunately, GP approximations that do not correspond to a valid stochastic process may inaccurately characterize uncertainty, as the models used for estimation and interpolation may not coincide. Rather than seeking approximations to the full GP, one can develop valid standalone spatial processes by introducing conditional independence across spatial locations as prescribed by a sparse directed acyclic graph (DAG). These models are advantageous because they lead to scalability by construction; in other words, posterior computing algorithms for these methods can be interpreted not only as approximate algorithms for the full GP, but also as exact algorithms for the standalone process.

This family of methods includes nearest-neighbor Gaussian processes, which limit dependence to a small number of neighboring locations (NNGP; Datta et al. 2016a,b), and block-NNGPs (Quiroz et al., 2019). There is a close relation between DAG structure and computational performance of NNGPs: some orderings may be associated to improved approximations (Guinness, 2018), and graph coloring algorithms (Molloy and Reed, 2002; Lewis, 2016) can be used for parallel Gibbs sampling. Inferring ordering or coloring can be problematic when data are in the millions, but these issues can be circumvented by forcing DAGs with known properties onto the data; in meshed GPs (MGPs; Peruzzi et al., 2020), patterned DAGs associated to domain tiling are associated to more efficient sampling of the latent effects. Alternative so-called multiscale or multiresolution methods correspond to DAGs with hierarchical node structures (trees), which are typically coupled with recursive domain partitioning; in this case, too, efficiencies follow from the properties of the chosen DAG. There is a rich literature on Gaussian processes and recursive partitioning, see e.g Ferreira and Lee (2007); Gramacy and Lee (2008); Fox and Dunson (2012); in geospatial contexts, in addition to the GMRF-based method of Nychka et al. (2015), multi-resolution approximations (MRA; Katzfuss, 2017) replace an orthogonal basis decomposition with approximations based on tapering or domain partitioning and also have a DAG interpretation (Katzfuss and Guinness, 2021).

Considerably less attention has been devoted to process-based methods that ensure scalability in multivariate contexts, with the goal of modeling the spatial and/or temporal variability of several variables jointly via flexible cross-covariance functions (Genton and Kleiber, 2015). When scalability of GP methods is achieved via reductions in the conditioning sets, including more distant locations is thought to aid in the estimation of unknown covariance parameters (Stein et al., 2004). However, the size of such sets may need to be reduced excessively when outcomes are not of very small dimension. One could restrict spatial coverage of the conditioning sets, but this works best when data are not misaligned, in which case all conditioning sets will include outcomes from all margins; this cannot be achieved for misaligned data, leading to pathological behavior. Alternatively, one can model the multivariate outcomes themselves as a DAG; however this may only work on a case-by-case basis. Similarly, recursive domain partitioning strategies work best for data that are measured uniformly in space as this guarantees similarly sized conditioning sets; on the contrary, recursive partitioning struggles in predicting the outcomes at large unobserved areas as they tend to be associated to the small conditioning sets making up the coarser scales or resolutions.

In this article, we solve these issues by introducing a Bayesian regression model that encodes spatial dependence as a latent spatial multivariate tree (SpamTree); conditional independence relations at the *reference* locations are governed by the branches in a treed DAG, whereas a map is used to assign all *non-reference* locations to leaf nodes of the same DAG. This assignment map controls the nature and the size of the conditioning sets at all locations; when severe restrictions on the reference set of locations become necessary due to data size, this map is used to improve estimation and predictions and overcome common issues in standard nearest-neighbor and recursive partition methods while maintaining the desirable recursive properties of treed DAGs. Unlike methods based on defining conditioning sets based solely on spatial proximity, SpamTrees scale to large data sets without excessive reduction of the conditioning sets. Furthermore, SpamTrees are less restrictive

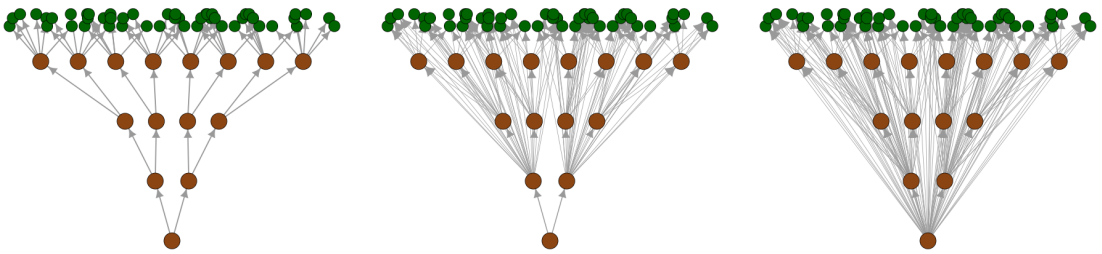

Figure 2: Three SpamTrees on $M = 4$ levels with depths $\delta = 1$ (left), $\delta = 3$ (center), and $\delta = 4$ (right). Nodes are represented by circles, with branches colored in brown and leaves in green.

than methods based on recursive partitioning and can be built to guarantee similarly-sized conditioning sets at all locations.

The present work adds to the growing literature on spatial processes defined on DAGs by developing a method that targets efficient computations of Bayesian multivariate spatial regression models. SpamTrees share similarities with MRAs (Katzfuss, 2017); however, while MRAs are defined as a basis function expansion, they can be represented by a treed graph of a SpamTree with full "depth" as defined later (the DAG on the right of Figure 2), in univariate settings, and "response" models. All these restrictions are relaxed in this article. In considering spatial proximity to add "leaves" to our treed graph, our methodology also borrows from nearest-neighbor methods (Datta et al., 2016a). However, while we use spatial neighbors to populate the conditioning sets for non-reference locations, the same cannot be said about reference locations for which the treed graph is used instead. Our construction of the SpamTree process also borrows from MGPs on tessellated domains (Peruzzi et al., 2020); however, the treed DAG we consider here induces markedly different properties on the resulting spatial process owing to its recursive nature. Finally, a contribution of this article is in developing self-contained sampling algorithms which, based on the graphical model representation of the model, will not require any external libraries.

The article builds SpamTrees as a standalone process based on a DAG representation in Section 2. A Gaussian base process is considered in Section 3 and the resulting properties outlined, along with sampling algorithms. Simulated data and real-world applications are in Section 4; we conclude with a discussion in Section 5. The Appendix provides more in-depth treatment of several topics and additional algorithms.

2. Spatial Multivariate Trees

Consider a spatial or spatiotemporal domain $\mathcal{D}$. With the temporal dimension, we have $\mathcal{D} \subset \Re^d \times [0, \infty)$, otherwise $\mathcal{D} \subset \Re^d$. A q-variate spatial process is defined as an uncountable set of random variables $\{\boldsymbol{w}(\boldsymbol{\ell}) : \boldsymbol{\ell} \in \mathcal{D}\}$, where $\boldsymbol{w}(\boldsymbol{\ell})$ is a $q \times 1$ random vector with elements $w_i(\boldsymbol{\ell})$ for $i = 1, 2, \ldots, q$, paired with a probability law P defining the joint distribution of any finite sample from that set. Let $\{\boldsymbol{\ell}_1, \boldsymbol{\ell}_2, \ldots, \boldsymbol{\ell}_{n_\mathcal{L}}\} = \mathcal{L} \subset \mathcal{D}$ be of size $n_\mathcal{L}$. The $n_\mathcal{L} q \times 1$ random

vector $\boldsymbol{w}_{\mathcal{L}} = (\boldsymbol{w}(\boldsymbol{\ell}_1)^\top, \boldsymbol{w}(\boldsymbol{\ell}_2)^\top, \ldots, \boldsymbol{w}(\boldsymbol{\ell}_{n_{\mathcal{L}}})^\top)^\top$ has joint density $p(\boldsymbol{w}_{\mathcal{L}})$. After choosing an arbitrary order of the locations, $p(\boldsymbol{w}_{\mathcal{L}}) = \prod_{i=1}^{n_{\mathcal{L}}} p(\boldsymbol{w}(\boldsymbol{\ell}_i) \,|\, \boldsymbol{w}(\boldsymbol{\ell}_1), \ldots, \boldsymbol{w}(\boldsymbol{\ell}_{i-1}))$, where the conditioning set for each $\boldsymbol{w}(\boldsymbol{\ell}_i)$ can be interpreted as the set of nodes that have a directed edge towards $\boldsymbol{w}(\boldsymbol{\ell}_i)$ in a DAG. Some scalable spatial processes result from reductions in size of the conditioning sets, following one of several proposed strategies (Vecchia, 1988; Stein et al., 2004; Gramacy and Apley, 2015; Datta et al., 2016a; Katzfuss and Guinness, 2021; Peruzzi et al., 2020). Accordingly,

$$p(\boldsymbol{w}_{\mathcal{L}}) = \prod_{i=1}^{n_{\mathcal{L}}} p(\boldsymbol{w}(\boldsymbol{\ell}_i) \,|\, \boldsymbol{w}(\mathrm{Pa}[\boldsymbol{\ell}_i])), \tag{1}$$

where $\mathrm{Pa}[\boldsymbol{\ell}_i]$ is the set of spatial locations that correspond to directed edges pointing to $\boldsymbol{\ell}_i$ in the DAG. If $\mathrm{Pa}[\boldsymbol{\ell}_i]$ is of size J or less for all $i = 1, \ldots, n_{\mathcal{L}}$, then $\boldsymbol{w}(\mathrm{Pa}[\boldsymbol{\ell}_i]))$ is of size Jq. Methods that rely on reducing the size of parent sets are thus negatively impacted by the dimension q of the multivariate outcome; if q is not very small, reducing the number of parent locations J may be insufficient for scalable computations. As an example, an NNGP model has $\mathrm{Pa}[\boldsymbol{\ell}_i] = N(\boldsymbol{\ell}_i)$, where $N(\cdot)$ maps a location in the spatial domain to its neighbor set. It is customary in practice to consider $Jq = m \leq 20$ for accurate and scalable estimation and predictions in univariate settings, but this may be restrictive in some multivariate settings as one must reduce J to maintain similar computing times, possibly harming estimation and prediction accuracy.

We represent the ith component of the $q \times 1$ vector $\boldsymbol{w}(\boldsymbol{\ell})$ as $w(\boldsymbol{\ell}, \xi_i)$, where $\xi_i = (\xi_{i1}, \ldots, \xi_{ik})^\top \in \Xi$ for some k and Ξ serves as the k-dimensional latent spatial domain of variables. The q-variate process $\boldsymbol{w}(\boldsymbol{\ell})$ is thus recast as $\{w(\boldsymbol{\ell}, \xi) : (\boldsymbol{\ell}, \xi) \in \mathcal{D} \times \Xi\}$, with ξ representing the latent location in the domain of variables. We can then write (1) as

$$p(\boldsymbol{w}_{\mathcal{L}^*}) = \prod_{i=1}^{n_{\mathcal{L}^*}} p(w(\boldsymbol{\ell}_i^*) \,|\, w(\mathrm{Pa}[\boldsymbol{\ell}_i^*])), \tag{2}$$

where $\mathcal{L}^* = \{\boldsymbol{\ell}_i^*\}_{i=1}^{n_{\mathcal{L}^*}}$, $\boldsymbol{\ell}_i^* \in \mathcal{D} \times \Xi = \mathcal{D}^*$, and $w(\cdot)$ is a univariate process on the expanded domain $\mathcal{D}^*$. This representation is useful as it provides a clearer accounting of the assumed conditional independence structure of the process in a multivariate context.

2.1 Constructing Spatial Multivariate DAGs

We now introduce the necessary terminology and notation, which are the basis for later detailing of estimation and prediction algorithms involving SpamTrees. The specifics for building treed DAGs with user-specified depth are in Section 2.1.1, whereas Section 2.1.2 gives details on cherry picking and its use when outcomes are imbalanced and misaligned.

The three key components to build a SpamTree are *(i)* a treed DAG $\mathcal{G}$ with *branches* and *leaves* on M levels and with depth $\delta \leq M$; *(ii)* a reference set of locations $\mathcal{S}$; *(iii)* a *cherry picking* map. The graph is $\mathcal{G} = \{\boldsymbol{V}, \boldsymbol{E}\}$ where the nodes are $\boldsymbol{V} = \{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{m_V}\} = \boldsymbol{A} \cup \boldsymbol{B}$, $\boldsymbol{A} \cap \boldsymbol{B} = \emptyset$. We separate the nodes into *reference* $\boldsymbol{A}$ and *non-reference* $\boldsymbol{B}$ nodes, as this will aid in showing that SpamTrees lead to standalone spatial processes in Section 2.2. The *reference* or *branch* nodes are $\boldsymbol{A} = \{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_{m_A}\} = \boldsymbol{A}_0 \cup \boldsymbol{A}_1 \cup \cdots \cup \boldsymbol{A}_{M-1}$, where $\boldsymbol{A}_i = \{\boldsymbol{a}_{i,1}, \ldots, \boldsymbol{a}_{i,m_i}\}$ for all $i = 0, \ldots, M-1$ and with $\boldsymbol{A}_i \cap \boldsymbol{A}_j = \emptyset$ if $i \neq j$. The *non-reference* or *leaf* nodes are $\boldsymbol{B} = \{\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{m_B}\}$, $\boldsymbol{A} \cap \boldsymbol{B} = \emptyset$. We also denote $\boldsymbol{V}_r = \boldsymbol{A}_r$ for

$r = 0, \ldots, M-1$ and $\boldsymbol{V}_M = \boldsymbol{B}$. The edges are $\boldsymbol{E} = \{\text{Pa}[\boldsymbol{v}] \subset \boldsymbol{V} : \boldsymbol{v} \in \boldsymbol{V}\}$ and similarly $\text{Ch}[\boldsymbol{v}] = \{\boldsymbol{v}' \in \boldsymbol{V} : \boldsymbol{v} \in \text{Pa}[\boldsymbol{v}']\}$. The reference set $\mathcal{S}$ is partitioned in M levels starting from zero, and each level is itself partitioned into reference subsets: $\mathcal{S} = \cup_{r=0}^{M-1}\mathcal{S}_r = \cup_{r=0}^{M-1}\cup_{i=1}^{m_i} S_{ri}$, where $S_{ri} \cap S_{r'i'} = \emptyset$ if $r \neq r'$ or $i \neq i'$ and its complement set of *non-reference* or *other* locations $\mathcal{U} = \mathcal{D}^* \setminus \mathcal{S}$. The *cherry picking* map is $\eta : \mathcal{D}^* \to \boldsymbol{V}$ and assigns a node (and therefore all the edges directed to it in $\mathcal{G}$) to any location in the domain, following a user-specified criterion.

2.1.1 Branches and Leaves

For a given M and a depth $\delta \leq M$, we impose a treed structure on $\mathcal{G}$ by assuming that if $\boldsymbol{v} \in \boldsymbol{A}_i$ and $i > M - \delta = M_\delta$ then there exists a sequence of nodes $\{\boldsymbol{v}_{r_{M_\delta}}, \ldots, \boldsymbol{v}_{r_{i-1}}\}$ such that $\boldsymbol{v}_{r_j} \in \boldsymbol{A}_j$ for $j = M_\delta, \ldots, i-1$ and $\text{Pa}[\boldsymbol{v}] = \{\boldsymbol{v}_{r_{M_\delta}}, \boldsymbol{v}_{r_1}, \ldots, \boldsymbol{v}_{r_{j-1}}\}$. If $i \leq M - \delta = M_\delta$ then $\text{Pa}[\boldsymbol{v}] = \{\boldsymbol{v}_{i-1}\}$ with $\boldsymbol{v}_{i-1} \in \boldsymbol{A}_{i-1}$. $\boldsymbol{A}_0$ is the tree *root* and is such that $\text{Pa}[\boldsymbol{v}_0] = \emptyset$ for all $\boldsymbol{v}_0 \in \boldsymbol{A}_0$. The depth δ determines the number of levels of $\mathcal{G}$ (from the top) across which the parent sets are nested. Choosing $\delta = 1$ implies that all nodes have a single parent; choosing $\delta = M$ implies fully nested parent sets (i.e. if $\boldsymbol{v}_i \in \text{Pa}[\boldsymbol{v}_j]$ then $\text{Pa}[\boldsymbol{v}_i] \subset \text{Pa}[\boldsymbol{v}_j]$ for all $\boldsymbol{v}_i, \boldsymbol{v}_j \in \boldsymbol{V}$). The m_i elements of $\boldsymbol{A}_i$ are the branches at level i of $\mathcal{G}$ and they have $i - M_\delta$ parents if the current level i is above the depth level M_δ and 1 parent otherwise. We refer to *terminal branches* as nodes $\boldsymbol{v} \in \boldsymbol{A}$ such that $\text{Ch}[\boldsymbol{v}] \subset \boldsymbol{B}$. For all choices of δ, $\boldsymbol{v} \in \boldsymbol{A}_i, \boldsymbol{v}' \in \boldsymbol{A}_j$ and $\boldsymbol{v} \in \text{Pa}[\boldsymbol{v}']$ implies $i < j$; this guarantees acyclicity.

As for the leaves, for all $\boldsymbol{v} \in \boldsymbol{B}$ we assume $\text{Pa}[\boldsymbol{v}] = \{\boldsymbol{v}_{r_{M_\delta}}, \ldots, \boldsymbol{v}_{r_k}\}$ for some integer sequence $\{r_{M_\delta}, \ldots, r_k\}$ and $\boldsymbol{v}_{r_i} \in \boldsymbol{A}_i$ with $i \geq M_\delta$. We allow the existence of multiple leaves with the same parent set, i.e. there can be k and $\boldsymbol{b}_{i_1}, \ldots, \boldsymbol{b}_{i_k}$ such that for all $i_2, \ldots, i_k$, $\text{Pa}[\boldsymbol{b}_{i_h}] = \text{Pa}[\boldsymbol{b}_{i_1}]$. Acyclicity of $\mathcal{G}$ is maintained as leaves are assumed to have no children. Figure 2 represents the graph associated to SpamTrees with different depths.

2.1.2 Cherry Picking via $\eta(\cdot)$

The link between $\mathcal{G}$, $\mathcal{S}$ and $\mathcal{U}$ is established via the map $\eta : \mathcal{D}^* \to \boldsymbol{V}$ which associates a node in $\mathcal{G}$ to any location $\boldsymbol{\ell}^*$ in the expanded domain $\mathcal{D}^*$:

$$\eta(\boldsymbol{\ell}^*) = \begin{cases} \eta_A(\boldsymbol{\ell}^*) = \boldsymbol{a}_{ri} \in \boldsymbol{A}_r \text{ if } \boldsymbol{\ell}^* \in S_{ri}, \\ \eta_B(\boldsymbol{\ell}^*) = \boldsymbol{b} \in \boldsymbol{B} \text{ if } \boldsymbol{\ell}^* \in \mathcal{U}. \end{cases} \tag{3}$$

This is a many-to-one map; note however that all locations in S_{ij} are mapped to $\boldsymbol{a}_{ij}$: by calling $\eta(X) = \{\eta(\boldsymbol{\ell}^*) : \boldsymbol{\ell}^* \in X\}$ then for any $i = 0, \ldots, M-1$ and any $j = 1, \ldots, m_i$ we have $\eta(S_{ij}) = \eta_A(S_{ij}) = \boldsymbol{a}_{ij}$. SpamTrees introduce flexibility by cherry picking the leaves, i.e. using $\eta_B : \mathcal{U} \to \boldsymbol{B}$, the restriction of η to $\mathcal{U}$. Since each leaf node $\boldsymbol{b}_j$ determines a unique path in $\mathcal{G}$ ending in $\boldsymbol{b}_j$, we use η_B to assign a convenient parent set to $w(\boldsymbol{u})$, $\boldsymbol{u} \in \mathcal{U}$, following some criterion.

For example, suppose that $\boldsymbol{u} = (\boldsymbol{\ell}, \xi_s)$ meaning that $w(\boldsymbol{u}) = w(\boldsymbol{\ell}, \xi_s)$ is the realization of the s-th variable at the spatial location $\boldsymbol{\ell}$, and we wish to ensure that $\text{Pa}[w(\boldsymbol{u})]$ includes realizations of the same variable. Denote $\boldsymbol{T} = \{\boldsymbol{v} \in \boldsymbol{A} : \text{Ch}[\boldsymbol{v}] \subset \boldsymbol{B}\}$ as the set of terminal branches of $\mathcal{G}$. Then we find $(\boldsymbol{\ell}, \xi_s)_{\text{opt}} = \arg\min_{(\boldsymbol{\ell}', \xi' = \xi_s) \in \eta_A^{-1}(\boldsymbol{T})} d(\boldsymbol{\ell}', \boldsymbol{\ell})$ where $d(\cdot, \cdot)$ is the Euclidean distance. Since $(\boldsymbol{\ell}, \xi_s)_{\text{opt}} \in S_{ij}$ for some i, j we have $\eta_A((\boldsymbol{\ell}, \xi_s)_{\text{opt}}) = \boldsymbol{a}_{ij}$. We then set $\eta_B(\boldsymbol{u}) = \boldsymbol{b}_k$ where $\text{Pa}[\boldsymbol{b}_k] = \{\boldsymbol{a}_{ij}\}$. In a sense $\boldsymbol{a}_{ij}$ is the terminal node nearest

to $\boldsymbol{u}$; having defined η_B in such a way forces the parent set of any location to include at least one realization of the process from the same variable. There is no penalty in using $\mathcal{D}^* = \mathcal{D} \times \Xi$ as we can write $p(\boldsymbol{w}(\boldsymbol{u}) \,|\, \mathrm{Pa}[\boldsymbol{w}(\boldsymbol{u})]) = p(\boldsymbol{w}((\boldsymbol{\ell}, \xi_1), \ldots, (\boldsymbol{\ell}, \xi_q)) \,|\, \mathrm{Pa}[\boldsymbol{w}(\boldsymbol{u})]) = \prod_{s=1}^{q} p(w(\boldsymbol{\ell}, \xi_s) \,|\, w(\boldsymbol{\ell}, \xi_1), \ldots, w(\boldsymbol{\ell}, \xi_{s-1}), \mathrm{Pa}[\boldsymbol{w}(\boldsymbol{\ell})]))$, which also implies that the size of the parent set may depend on the variable index. Assumptions of conditional independence across variables can be encoded similarly. Also note that any specific choice of η_B induces a partition on $\mathcal{U}$; let $U_j = \{\boldsymbol{u} \in \mathcal{U} : \eta_B(\boldsymbol{u}) = \boldsymbol{b}_j\}$, then clearly $\mathcal{U} = \cup_{j=1}^{m_U} U_j$ with $U_i \cap U_j = \emptyset$ if $i \neq j$. This partition does not necessarily correspond to the partitioning scheme used on $\mathcal{S}$. η_B may by designed to ignore part of the tree and result in $m_U < m_B$. However, we can just drop the unused leaves from $\mathcal{G}$ and set $\mathrm{Ch}[\boldsymbol{a}] = \emptyset$ for terminal nodes whose leaf is inactive, resulting in $m_U = m_B$. We will thus henceforth assume that $m_U = m_B$ without loss of generality.

2.2 SpamTrees as a Standalone Spatial Process

We define a valid joint density for any finite set of locations in $\mathcal{D}^*$ satisfying the Kolmogorov consistency conditions in order to define a valid process. We approach this problem analogously to Datta et al. (2016a) and Peruzzi et al. (2020). Enumerate each of the m_S reference subsets as $S_i = \{\boldsymbol{s}_{i_1}, \ldots, \boldsymbol{s}_{i_{n_i}}\}$ where $\{i_1, \ldots, i_{n_i}\} \subset \{1, \ldots, n_{\mathcal{S}}\}$, and each of the m_U non-reference subsets as $U_i = \{\boldsymbol{u}_{i_1}, \ldots, \boldsymbol{u}_{i_{n_i}}\}$ where $\{i_1, \ldots, i_{n_i}\} \subset \{1, \ldots, n_{\mathcal{U}}\}$. Then introduce $\mathcal{V} = \{V_1, \ldots, V_{m_V}\}$ where $m_V = m_S + m_U$ and $V_i = S_i$ for $i = 1, \ldots, m_S$, $V_{m_S+i} = U_i$ for $i = 1, \ldots, m_U$. Then take $\boldsymbol{w}_i = (w(\boldsymbol{\ell}_{i_1}), \ldots, w(\boldsymbol{\ell}_{i_{n_i}}))^\top$ as the $n_i \times 1$ random vector with elements of $w(\boldsymbol{\ell})$ for each $\boldsymbol{\ell} \in V_i$. Denote $\boldsymbol{w}_{[i]} = \boldsymbol{w}(\eta^{-1}(\mathrm{Pa}[\boldsymbol{v}_i]))$. Then

$$\widetilde{p}(\boldsymbol{w}_{\mathcal{S}}) = \widetilde{p}(\boldsymbol{w}_1, \ldots, \boldsymbol{w}_{m_S}) = \prod_{r=0}^{M-1} \prod_{i:\{\boldsymbol{v}_i \in \boldsymbol{A}_r\}} p(\boldsymbol{w}_i \,|\, \boldsymbol{w}_{[i]}) \qquad \widetilde{p}(\boldsymbol{w}_{\mathcal{U}} \mid \boldsymbol{w}_{\mathcal{S}}) = \prod_{i:\{\boldsymbol{v}_i \in \boldsymbol{B}\}} p(\boldsymbol{w}_i \,|\, \boldsymbol{w}_{[i]})$$

$$\widetilde{p}(\boldsymbol{w}_{\mathcal{S}})\widetilde{p}(\boldsymbol{w}_{\mathcal{U}} \mid \boldsymbol{w}_{\mathcal{S}}) = \prod_{r=0}^{M-1} \prod_{i:\{\boldsymbol{v}_i \in \boldsymbol{A}_r\}} p(\boldsymbol{w}_i \,|\, \boldsymbol{w}_{[i]}) \prod_{i:\{\boldsymbol{v}_i \in \boldsymbol{B}\}} p(\boldsymbol{w}_i \,|\, \boldsymbol{w}_{[i]}) \tag{4}$$

which is a proper multivariate joint density since $\mathcal{G}$ is acyclic (Lauritzen, 1996). All locations inside U_j always share the same parent set, but a parent set is not necessarily unique to a single U_j. This includes as a special case a scenario in which one can assume

$$\widetilde{p}(\boldsymbol{w}_{\mathcal{U}} \mid \boldsymbol{w}_{\mathcal{S}}) = \prod_{j=1}^{m_U} \prod_{i=1}^{|U_j|} p(w(\boldsymbol{u}_i) \,|\, \boldsymbol{w}(\eta^{-1}(\mathrm{Pa}[\boldsymbol{b}_j]))); \tag{5}$$

in this case each location corresponds to a leaf node. To conclude the construction, for any finite subset of spatial locations $\mathcal{L} \subset \mathcal{D}$ we can let $\mathcal{U} = \mathcal{L} \setminus \mathcal{S}$ and obtain

$$\widetilde{p}(\boldsymbol{w}_{\mathcal{L}}) = \int \widetilde{p}(\boldsymbol{w}_{\mathcal{U}} \mid \boldsymbol{w}_{\mathcal{S}})\widetilde{p}(\boldsymbol{w}_{\mathcal{S}}) \prod_{\boldsymbol{s}_i \in \mathcal{S} \setminus \mathcal{L}} d(\boldsymbol{w}(\boldsymbol{s}_i)),$$

leading to a well-defined process satisfying the Kolmogorov conditions (see Appendix A).

2.2.1 Positioning of Spatial Locations in Conditioning Sets

In spatial models based on sparse DAGs, larger conditioning sets yield processes that are closer to the base process p in terms of Kullback-Leibler divergence (Banerjee, 2020; Peruzzi et al., 2020), denoted as $KL(p\|\cdot)$. The same results cannot be applied directly to SpamTrees given the treed structure of the DAG. For a given $\mathcal{S}$, we consider the distinct but related issues of placing individual locations into reference subsets (1) at different levels of the treed hierarchy; (2) within the same level of the hierarchy.

Proposition 1 *Suppose $\mathcal{S} = \mathcal{S}_0 \cup \mathcal{S}_1$ where $S_0 \cap S_1 = \emptyset$ and $\mathcal{S}_1 = S_{11} \cup S_{12}$, $S_{11} \cap S_{12} = \emptyset$. Take $\boldsymbol{s}^* \notin \mathcal{S}$. Consider the graph $\mathcal{G} = \{\boldsymbol{V} = \{\boldsymbol{v}_0, \boldsymbol{v}_1, \boldsymbol{v}_2\}, \boldsymbol{E} = \{\boldsymbol{v}_0 \to \boldsymbol{v}_1, \boldsymbol{v}_0 \to \boldsymbol{v}_2\}\}$; denote as p_0 the density of a SpamTree using $\eta(\mathcal{S}_0 \cup \{\boldsymbol{s}^*\}) = \boldsymbol{v}_0$, $\eta(S_{11}) = \boldsymbol{v}_1$ and $\eta(S_{12}) = \boldsymbol{v}_2$, whereas let p_1 be the density of a SpamTree with $\eta(\mathcal{S}_0) = \boldsymbol{v}_0$, $\eta(S_{11} \cup \{\boldsymbol{s}^*\}) = \boldsymbol{v}_1$ and $\eta(S_{12}) = \boldsymbol{v}_2$. Then $KL(p\|p_1) - KL(p\|p_0) > 0$.*

The proof proceeds by an "information never hurts" argument (Cover and Thomas, 1991). Denote $\mathcal{S}^* = \mathcal{S} \cup \{\boldsymbol{s}^*\}$, $\boldsymbol{w}^* = \boldsymbol{w}_{\mathcal{S}^*}$, $w^* = w(\boldsymbol{s}^*)$ and $\boldsymbol{w}_j^* = (\boldsymbol{w}_j^\top, w^*)^\top$. Then

$$
\begin{aligned}
p_0(\boldsymbol{w}^*) &= p(\boldsymbol{w}_0^*)p(\boldsymbol{w}_1 \,|\, \boldsymbol{w}_0^*)p(\boldsymbol{w}_2 \,|\, \boldsymbol{w}_0^*) = p(\boldsymbol{w}_0)p(w^* \,|\, \boldsymbol{w}_0)p(\boldsymbol{w}_1 \,|\, \boldsymbol{w}_0, w^*)p(\boldsymbol{w}_2 \,|\, \boldsymbol{w}_0^*) \\
p_1(\boldsymbol{w}^*) &= p(\boldsymbol{w}_0)p(\boldsymbol{w}_1^* \,|\, \boldsymbol{w}_0)p(\boldsymbol{w}_2 \,|\, \boldsymbol{w}_0) = p(\boldsymbol{w}_0)p(w^* \,|\, \boldsymbol{w}_0)p(\boldsymbol{w}_1 \,|\, \boldsymbol{w}_0, w^*)p(\boldsymbol{w}_2 \,|\, \boldsymbol{w}_0),
\end{aligned}
$$

therefore $p_0(\boldsymbol{w}^*)/p_1(\boldsymbol{w}^*) = p(\boldsymbol{w}_2 \,|\, \boldsymbol{w}_0^*)/p(\boldsymbol{w}_2 \,|\, \boldsymbol{w}_0)$; then by Jensen's inequality

$$
\begin{aligned}
KL(p\|p_1) - KL(p\|p_0) &= \int \left\{ \log\left(\frac{p(\boldsymbol{w}^*)}{p_1(\boldsymbol{w}^*)}\right) - \log\left(\frac{p(\boldsymbol{w}^*)}{p_0(\boldsymbol{w}^*)}\right) \right\} p(\boldsymbol{w}^*) d\boldsymbol{w}^* \\
&= \int \log\left(\frac{p_0(\boldsymbol{w}^*)}{p_1(\boldsymbol{w}^*)}\right) p(\boldsymbol{w}^*) d\boldsymbol{w}^* = \int \log\left(\frac{p(\boldsymbol{w}_2 \,|\, \boldsymbol{w}_0^*)}{p(\boldsymbol{w}_2 \,|\, \boldsymbol{w}_0)}\right) p(\boldsymbol{w}^*) d\boldsymbol{w}^* \\
&= \int \log\left(\frac{p(\boldsymbol{w}_2 \,|\, \boldsymbol{w}_0^*)}{p(\boldsymbol{w}_2 \,|\, \boldsymbol{w}_0)}\right) p(\boldsymbol{w}_1, \boldsymbol{w}_2, \boldsymbol{w}_0^*) d\boldsymbol{w}_1 d\boldsymbol{w}_2 d\boldsymbol{w}_0^* \\
&= \int \left\{ \int \log\left(\frac{p(\boldsymbol{w}_2 \,|\, \boldsymbol{w}_0^*)}{p(\boldsymbol{w}_2 \,|\, \boldsymbol{w}_0)}\right) p(\boldsymbol{w}_1, \boldsymbol{w}_2 \,|\, \boldsymbol{w}_0^*) d\boldsymbol{w}_1 d\boldsymbol{w}_2 \right\} p(\boldsymbol{w}_0^*) d\boldsymbol{w}_0^* \geq 0.
\end{aligned}
\tag{6}
$$

Intuitively, this shows that there is a penalty associated to positioning reference locations at higher levels of the treed hierarchy. Increasing the size of the reference set at the root augments the conditioning sets at all its children; since this is not true when the increase is at a branch level, the KL divergence of p_0 from p is smaller than the divergence of p_1 from the same density. In other words there is a cost of branching in $\mathcal{G}$ which must be justified by arguments related to computational efficiency. The above proposition also suggests populating near-root branches with locations of sparsely-observed outcomes. Not doing so in highly imbalanced settings may result in possibly too restrictive spatial conditional independence assumptions.

Proposition 2 *Consider the same setup as Proposition 1 and let p_2 be the density of a SpamTree such that $\eta(S_{12} \cup \{\boldsymbol{s}^*\}) = \boldsymbol{v}_2$. Let H_p be the conditional entropy of base process p. Then $H_p(w^* \,|\, \boldsymbol{w}_0, \boldsymbol{w}_2) < H_p(w^* \,|\, \boldsymbol{w}_0, \boldsymbol{w}_1)$ implies $KL(p\|p_2) < KL(p\|p_1)$.*

The density of the new model is

$$p_2(\boldsymbol{w}^*) = p(\boldsymbol{w}_0)p(\boldsymbol{w}_1 \,|\, \boldsymbol{w}_0)p(\boldsymbol{w}_2^* \,|\, \boldsymbol{w}_0) = p(\boldsymbol{w}_0)p(\boldsymbol{w}_1 \,|\, \boldsymbol{w}_0)p(\boldsymbol{w}_2 \,|\, \boldsymbol{w}_0)p(w^* \,|\, \boldsymbol{w}_0, \boldsymbol{w}_2).$$

Then, noting that $p(\boldsymbol{w}_1^* \,|\, \boldsymbol{w}_0) = p(\boldsymbol{w}_1 \,|\, \boldsymbol{w}_0)p(w^* \,|\, \boldsymbol{w}_0, \boldsymbol{w}_1)$, we get $\frac{p_1(\boldsymbol{w}^*)}{p_2(\boldsymbol{w}^*)} = \frac{p(w^* \,|\, \boldsymbol{w}_0, \boldsymbol{w}_1)}{p(w^* \,|\, \boldsymbol{w}_0, \boldsymbol{w}_2)}$ and

$$\begin{aligned} KL(p\|p_2) - KL(p\|p_1) &= \int \log p(w^* \,|\, \boldsymbol{w}_0, \boldsymbol{w}_1)p(\boldsymbol{w}^*)d\boldsymbol{w}^* - \int \log p(w^* \,|\, \boldsymbol{w}_0, \boldsymbol{w}_2)p(\boldsymbol{w}^*)d\boldsymbol{w}^* \\ &= H_p(w^* \,|\, \boldsymbol{w}_0, \boldsymbol{w}_2) - H_p(w^* \,|\, \boldsymbol{w}_0, \boldsymbol{w}_1). \end{aligned}$$

While we do not target the estimation of these quantities, this result is helpful in designing SpamTrees as it suggests placing a new reference location $\boldsymbol{s}^*$ in the reference subset *least* uncertain about the realization of the process at $\boldsymbol{s}^*$. We interpret this as justifying recursive domain partitioning on $\mathcal{S}$ in spatial contexts in which local spatial clusters of locations are likely less uncertain about process realization in the same spatial region. In the remainder of this article, we will consider a given reference set $\mathcal{S}$ which typically will be based on a subset of observed locations; the combinatorial problem of selecting an optimal $\mathcal{S}$ (in some sense) is beyond the scope of this article. If $\mathcal{S}$ is not partitioned, it can be considered as a set of knots or "sensors" and one can refer to a large literature on experimental design and optimal sensor placement (see e.g. Krause et al., 2008, and references therein). It might be possible to extend previous work on adaptive knot placement (Guhaniyogi et al., 2011), but this will come at a steep cost in terms of computational performance.

3. Bayesian Spatial Regressions Using SpamTrees

Suppose we observe an l-variate outcome at spatial locations $\boldsymbol{\ell} \in \mathcal{D} \subset \Re^d$ which we wish to model using a spatially-varying regression model:

$$y_j(\boldsymbol{\ell}) = \boldsymbol{x}_j(\boldsymbol{\ell})^\top \boldsymbol{\beta}_j + \sum_k z_{jk}(\boldsymbol{\ell})w(\boldsymbol{\ell}, \boldsymbol{\xi}_k) + \varepsilon_j(\boldsymbol{\ell}), \quad j = 1, \ldots, l, \tag{7}$$

where $y_j(\boldsymbol{\ell})$ is the j-th point-referenced outcome at $\boldsymbol{\ell}$, $\boldsymbol{x}_j(\boldsymbol{\ell})$ is a $p_j \times 1$ vector of spatially referenced predictors linked to constant coefficients $\boldsymbol{\beta}_j$, $\varepsilon_j(\boldsymbol{\ell}) \overset{iid}{\sim} N(0, \tau_j^2)$ is the measurement error for outcome j, and $z_{jk}(\boldsymbol{\ell})$ is the k-th (of q) covariates for the j-th outcome modeled with spatially-varying coefficient $w(\boldsymbol{\ell}, \boldsymbol{\xi}_k)$, $\boldsymbol{\ell} \in \mathcal{D}$, $\boldsymbol{\xi}_k \in \Xi$. This coefficient $w(\boldsymbol{\ell}, \boldsymbol{\xi}_k)$ corresponds to the k-th margin of a q-variate Gaussian process $\{\boldsymbol{w}(\boldsymbol{\ell}) : \boldsymbol{\ell} \in \mathcal{D}\}$ denoted as $\boldsymbol{w}(\boldsymbol{\ell}) \sim GP(\boldsymbol{0}, \boldsymbol{C}_{\boldsymbol{\theta}}(\cdot, \cdot))$ with cross-covariance $\boldsymbol{C}_{\boldsymbol{\theta}}$ indexed by unknown parameters $\boldsymbol{\theta}$ which we omit in notation for simplicity. A valid cross-covariance function is defined as $\boldsymbol{C}_{\boldsymbol{\theta}} : \mathcal{D} \times \mathcal{D} \to \mathcal{M}_{q\times q}$, where $\mathcal{M}_{q\times q}$ is a subset of the space of all $q \times q$ real matrices $\Re^{q\times q}$. It must satisfy $\boldsymbol{C}(\boldsymbol{\ell}, \boldsymbol{\ell}') = \boldsymbol{C}(\boldsymbol{\ell}', \boldsymbol{\ell})^\top$ for any two locations $\boldsymbol{\ell}, \boldsymbol{\ell}' \in \mathcal{D}$, and $\sum_{i=1}^n \sum_{j=1}^n \boldsymbol{z}_i^\top \boldsymbol{C}(\boldsymbol{\ell}_i, \boldsymbol{\ell}_j)\boldsymbol{z}_j > 0$ for any integer n and finite collection of points $\{\boldsymbol{\ell}_1, \boldsymbol{\ell}_2, \ldots, \boldsymbol{\ell}_n\}$ and for all $\boldsymbol{z}_i \in \Re^q \setminus \{\boldsymbol{0}\}$.

We replace the full GP with a Gaussian SpamTree for scalable computation considering the q-variate multivariate Gaussian process $\boldsymbol{w}(\cdot)$ as the base process. Since the (i, j)-th entry of $\boldsymbol{C}(\boldsymbol{\ell}, \boldsymbol{\ell}')$ is $\boldsymbol{C}(\boldsymbol{\ell}, \boldsymbol{\ell}')_{i,j} = \mathrm{Cov}(w_i(\boldsymbol{\ell}), w_j(\boldsymbol{\ell}'))$, i.e. the covariance between the i-th and j-th elements of $\boldsymbol{w}(\boldsymbol{\ell})$ at $\boldsymbol{\ell}$ and $\boldsymbol{\ell}'$, we can obtain a covariance function on the augmented domain $\boldsymbol{C}^* : \mathcal{D}^* \times \mathcal{D}^* \to \Re$ as $\boldsymbol{C}^*((\boldsymbol{\ell}, \boldsymbol{\xi}), (\boldsymbol{\ell}', \boldsymbol{\xi}')) = \boldsymbol{C}(\boldsymbol{\ell}, \boldsymbol{\ell}')_{i,i'}$ where $\boldsymbol{\xi}$ and $\boldsymbol{\xi}'$ are the

locations in Ξ of variables i and j, respectively. Apanasovich and Genton (2010) use a similar representation to build valid cross-covariances based on existing univariate covariance functions; their approach amounts to considering $\boldsymbol{\xi}$ or $\|\boldsymbol{\xi}-\boldsymbol{\xi}'\|$ as a parameter to be estimated. Our approach can be based on any valid cross-covariance as we may just set $\Xi = \{1, \dots, q\}$. Refer to e.g. Genton and Kleiber (2015) for an extensive review of cross-covariance functions for multivariate processes. Moving forward, we will not distinguish between $\boldsymbol{C}^*$ and $\boldsymbol{C}$. The linear multivariate spatially-varying regression model (7) allows the l outcomes to be observed at different locations; we later consider the case $l = q$ and $\boldsymbol{Z}(\boldsymbol{\ell}) = I_q$ resulting in a multivariate space-varying intercept model.

3.1 Gaussian SpamTrees

Enumerate the set of nodes as $\boldsymbol{V} = \{\boldsymbol{v}_1, \dots, \boldsymbol{v}_{m_V}\}$, $m_V = m_S + m_U$ and denote $\boldsymbol{w}_i = w(\eta^{-1}(\boldsymbol{v}_i))$, $\boldsymbol{C}_{ij}$ as the $n_i \times n_j$ covariance matrix between $\boldsymbol{w}_i$ and $\boldsymbol{w}_j$, $\boldsymbol{C}_{i,[i]}$ the $n_i \times J_i$ covariance matrix between $\boldsymbol{w}_i$ and $\boldsymbol{w}_{[i]}$, $\boldsymbol{C}_i$ the $n_i \times n_i$ covariance matrix between $\boldsymbol{w}_i$ and itself, and $\boldsymbol{C}_{[i]}$ the $J_i \times J_i$ covariance matrix between $\boldsymbol{w}_{[i]}$ and itself. A base Gaussian process induces $\widetilde{p}(\boldsymbol{w}_{\mathcal{S}}) = \prod_{j:\{\boldsymbol{v}_j \in \boldsymbol{A}\}} N(\boldsymbol{w}_j \mid \boldsymbol{H}_j \boldsymbol{w}_{[j]}, \boldsymbol{R}_j)$, where

$$\boldsymbol{H}_j = \boldsymbol{C}_{j,[j]} \boldsymbol{C}_{[j]}^{-1} \quad \text{and} \quad \boldsymbol{R}_j = \boldsymbol{C}_j - \boldsymbol{C}_{j,[j]} \boldsymbol{C}_{[j]}^{-1} \boldsymbol{C}_{[j],j}, \tag{8}$$

implying that the joint density $\widetilde{p}(\boldsymbol{w}_{\mathcal{S}})$ is multivariate normal with covariance $\widetilde{\boldsymbol{C}}_{\mathcal{S}}$ and precision matrix $\widetilde{\boldsymbol{C}}_{\mathcal{S}}^{-1}$. At $\mathcal{U}$ we have $\widetilde{p}(\boldsymbol{w}_{\mathcal{U}} \mid \boldsymbol{w}_{\mathcal{S}}) = \prod_{j:\{\boldsymbol{v}_j \in \boldsymbol{B}\}} N(\boldsymbol{w}_j \mid \boldsymbol{H}_j \boldsymbol{w}_{[j]}, \boldsymbol{R}_j)$, where $\boldsymbol{H}_j$ and $\boldsymbol{R}_j$ are as in (8). All quantities can be computed using the base cross-covariance function. Given that the $\widetilde{p}$ densities are Gaussian, so will be the finite dimensional distributions.

The treed graph $\mathcal{G}$ leads to properties which we analyze in more detail in Appendix B and summarize here. For two nodes $\boldsymbol{v}_i, \boldsymbol{v}_j \in \boldsymbol{V}$ denote the *common descendants* as $\text{cd}(\boldsymbol{v}_i, \boldsymbol{v}_j) = (\{\boldsymbol{v}_i\} \cup \text{Ch}[\boldsymbol{v}_i]) \cap (\{\boldsymbol{v}_j\} \cup \text{Ch}[\boldsymbol{v}_j])$. If $\boldsymbol{v}_i \in \text{Pa}[\boldsymbol{v}_j]$ denote $\boldsymbol{H}_{i \to j}$ and $\boldsymbol{H}_{\setminus i \to j}$ as the matrix obtained by subsetting $\boldsymbol{H}_j$ to columns corresponding to $\boldsymbol{v}_i$, or to $\text{Pa}[\boldsymbol{v}_j] \setminus \{\boldsymbol{v}_i\}$, respectively. Similarly define $\boldsymbol{w}_{[i \to j]} = \boldsymbol{w}_i$ and $\boldsymbol{w}_{[\setminus i \to j]}$. As a special case, if the tree depth is $\delta = 1$ and $\{\boldsymbol{v}_j\} = \text{Pa}[\boldsymbol{v}_i]$ then $\text{cd}(\boldsymbol{v}_i, \boldsymbol{v}_j) = \{\boldsymbol{v}_i\}$, $\boldsymbol{H}_{i \to j} = \boldsymbol{H}_j$, and $\boldsymbol{w}_{[i \to j]} = \boldsymbol{w}_{[j]}$. Define $\boldsymbol{\mathcal{H}}$ as the matrix whose (i, j) block is $\boldsymbol{\mathcal{H}}_{ij} = \boldsymbol{O}_{n_i \times n_j}$ if $\boldsymbol{v}_j \notin \text{Pa}[\boldsymbol{v}_i]$, and otherwise $\boldsymbol{\mathcal{H}}_{ij} = \boldsymbol{H}_{j \to i}$.

3.1.1 Precision Matrix

The (i, j) block of the precision matrix at both reference and non-reference locations $\widetilde{\boldsymbol{C}}^{-1}$ is denoted by $\widetilde{\boldsymbol{C}}^{-1}(i, j)$, with $i, j = 1, \dots, m_V$ corresponding to nodes $\boldsymbol{v}_i, \boldsymbol{v}_j \in \boldsymbol{V}$ for some i, j; it is nonzero if $\text{cd}(\boldsymbol{v}_i, \boldsymbol{v}_j) = \emptyset$, otherwise:

$$\begin{aligned} \widetilde{\boldsymbol{C}}^{-1}(i, j) &= \sum_{\boldsymbol{v}_k \in \text{cd}(\boldsymbol{v}_i, \boldsymbol{v}_j)} (\boldsymbol{I}_{ki} - \boldsymbol{H}_{i \to k})^\top \boldsymbol{R}_k^{-1} (\boldsymbol{I}_{kj} - \boldsymbol{H}_{j \to k}) \\ &= \sum_{\boldsymbol{v}_k \in \text{cd}(\boldsymbol{v}_i, \boldsymbol{v}_j)} (\boldsymbol{I}_{ki} - \boldsymbol{\mathcal{H}}_{ki})^\top \boldsymbol{R}_k^{-1} (\boldsymbol{I}_{kj} - \boldsymbol{\mathcal{H}}_{kj}), \end{aligned} \tag{9}$$

where $\boldsymbol{I}_{ij}$ is the (i,j) block of an identity matrix with $n_{\mathcal{S}} + n_{\mathcal{U}}$ rows and is nonzero if and only if $i = j$. We thus obtain that the number of nonzero elements of $\widetilde{\boldsymbol{C}}^{-1}$ is

$$\mathrm{nnz}(\widetilde{\boldsymbol{C}}^{-1}) = \sum_{i=1}^{m_V} \left(2n_i J_i + n_i^2 \mathbf{1}\{\boldsymbol{v}_i \in \boldsymbol{V}\}\right), \tag{10}$$

where $n_i = |\eta^{-1}(\boldsymbol{v}_i)|$, $J_i = |\eta^{-1}(\mathrm{Pa}[\boldsymbol{v}_i])|$, and by symmetry $(\widetilde{\boldsymbol{C}}^{-1}(i,j))^\top = \widetilde{\boldsymbol{C}}^{-1}(j,i)$.

If $\delta > 1$, the size of $\boldsymbol{C}_{[i]}$ is larger for nodes $\boldsymbol{v}_i$ at levels of the treed hierarchy farther from $\boldsymbol{A}_{M_\delta}$. However suppose $\boldsymbol{v}_i, \boldsymbol{v}_j$ are such that $\mathrm{Pa}[\boldsymbol{v}_j] = \{\boldsymbol{v}_i\} \cup \mathrm{Pa}[\boldsymbol{v}_i]$. Then computing $\boldsymbol{C}_{[j]}^{-1}$ proceeds more cheaply by recursively applying the following:

$$\boldsymbol{C}_{[j]}^{-1} = \begin{bmatrix} \boldsymbol{C}_{[i]}^{-1} + \boldsymbol{H}_i^\top \boldsymbol{R}_i^{-1} \boldsymbol{H}_i & -\boldsymbol{H}_i^\top \boldsymbol{R}_i^{-1} \\ -\boldsymbol{R}_i^{-1} \boldsymbol{H}_i & \boldsymbol{R}_i^{-1} \end{bmatrix}. \tag{11}$$

3.1.2 Induced Covariance

Define a path from $\boldsymbol{v}_k$ to $\boldsymbol{v}_j$ as $\mathcal{P}_{k\to j} = \{\boldsymbol{v}_{i_1}, \ldots, \boldsymbol{v}_{i_r}\}$ where $\boldsymbol{v}_{i_1} = \boldsymbol{v}_k$, $\boldsymbol{v}_{i_r} = \boldsymbol{v}_j$, and $\boldsymbol{v}_{i_h} \in \mathrm{Pa}[\boldsymbol{v}_{i_{h+1}}]$. The longest path $\widetilde{\mathcal{P}}_{k\to j}$ is such that if $\boldsymbol{v}_k \in \boldsymbol{A}_{r_k}$ and $\boldsymbol{v}_j \in \boldsymbol{A}_{r_j}$ then $|\widetilde{\mathcal{P}}_{k\to j}| = r_j - r_k + 1$. The shortest path $\bar{\mathcal{P}}_{k\to j}$ is the path from $\boldsymbol{v}_k$ to $\boldsymbol{v}_j$ with minimum number of steps. We denote the longest path from the root to $\boldsymbol{v}_j$ as $\widetilde{\mathcal{P}}_{0\to j}$; this corresponds to the full set of ancestors of $\boldsymbol{v}_j$, and $\mathrm{Pa}[\boldsymbol{v}_j] \subset \widetilde{\mathcal{P}}_{0\to j}$. For two nodes $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ we have $(\mathrm{Pa}[\boldsymbol{v}_i] \cap \mathrm{Pa}[\boldsymbol{v}_j]) \subset (\widetilde{\mathcal{P}}_{0\to i} \cap \widetilde{\mathcal{P}}_{0\to j})$. We define the *concestor* between $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ as $\mathrm{con}(\boldsymbol{v}_i, \boldsymbol{v}_j) = \arg\max_{\boldsymbol{v}_k \in \boldsymbol{V}} \{k : \mathcal{P}_{k\to i} \cap \mathcal{P}_{k\to j} \neq \emptyset\}$ i.e. the last common ancestor of the two nodes.

Take the path $\widetilde{\mathcal{P}}_{M_\delta \to j}$ in $\mathcal{G}$ from a node at $\boldsymbol{A}_{M_\delta}$ leading to $\boldsymbol{v}_j$. After defining the cross-covariance function $\boldsymbol{K}_i(\boldsymbol{\ell}, \boldsymbol{\ell}') = \boldsymbol{C}_{\boldsymbol{\ell},\boldsymbol{\ell}'} - \boldsymbol{C}_{\boldsymbol{\ell},[i]} \boldsymbol{C}_{[i]}^{-1} \boldsymbol{C}_{[i],\boldsymbol{\ell}'}$ and denoting $\boldsymbol{K}_i(\boldsymbol{\ell}, s) = \boldsymbol{K}_i(\boldsymbol{\ell}, \eta^{-1}(\boldsymbol{v}_s))$ we can write

$$\boldsymbol{w}_j = \sum_{s=i_{M_\delta}}^{i_{r-1}} \boldsymbol{K}_s(j,s) \boldsymbol{K}_s^{-1}(s,s) \boldsymbol{e}_s + \boldsymbol{e}_j, \tag{12}$$

where for $s > i_{M_\delta}$ the $\boldsymbol{e}_s$ are independent zero-mean GPs with covariance $K_s(\boldsymbol{\ell}, \boldsymbol{\ell}')$ and we set $K_{i_{M_\delta}}(\boldsymbol{\ell}, \boldsymbol{\ell}') = \boldsymbol{C}(\boldsymbol{\ell}, \boldsymbol{\ell}')$ and $\boldsymbol{e}_{i_{M_\delta}} = \boldsymbol{w}_{i_{M_\delta}} \sim N(0, \boldsymbol{C}_{i_{M_\delta}})$. Take two locations $\boldsymbol{\ell}, \boldsymbol{\ell}'$ such that $\boldsymbol{v}_i = \eta(\boldsymbol{\ell}), \boldsymbol{v}_j = \eta(\boldsymbol{\ell}')$ and let $\boldsymbol{v}_z = \mathrm{con}(\boldsymbol{v}_i, \boldsymbol{v}_j)$; if $\mathrm{Pa}[\boldsymbol{v}_i] \cap \mathrm{Pa}[\boldsymbol{v}_j] \neq \emptyset$ then the above leads to

$$Cov_{\widetilde{p}}(\boldsymbol{w}(\boldsymbol{\ell}), \boldsymbol{w}(\boldsymbol{\ell}')) = \sum_{s \in \mathrm{Pa}[\boldsymbol{v}_i] \cap \mathrm{Pa}[\boldsymbol{v}_j]} \boldsymbol{K}_s(\boldsymbol{\ell}, s) \boldsymbol{K}_s^{-1}(s,s) \boldsymbol{K}_s(s, \boldsymbol{\ell}') + \mathbf{1}\{\boldsymbol{\ell} = \boldsymbol{\ell}'\} \boldsymbol{K}_j(\boldsymbol{\ell}, \boldsymbol{\ell}'), \tag{13}$$

where $\boldsymbol{K}_z(\boldsymbol{\ell}, \boldsymbol{\ell}') = \boldsymbol{C}(\boldsymbol{\ell}, \boldsymbol{\ell}')$. If $\mathrm{Pa}[\boldsymbol{v}_i] \cap \mathrm{Pa}[\boldsymbol{v}_j] = \emptyset$ take the shortest paths $\bar{\mathcal{P}}_{z\to i} = \{i_1, \ldots, i_{r_i}\}$ and $\bar{\mathcal{P}}_{z\to j} = \{j_1, \ldots, j_{r_j}\}$; setting $\boldsymbol{F}_{i_h} = \boldsymbol{C}_{i_h, i_{h-1}} \boldsymbol{C}_{i_{h-1}}^{-1}$ we get

$$Cov_{\widetilde{p}}(\boldsymbol{w}(\boldsymbol{\ell}), \boldsymbol{w}(\boldsymbol{\ell}')) = \boldsymbol{F}_{i_{r_i}} \cdots \boldsymbol{F}_{i_1} \boldsymbol{C}_z \boldsymbol{F}_{j_1}^\top \cdots \boldsymbol{F}_{j_{r_j}}^\top. \tag{14}$$

In particular if $\delta = M$ then $\mathrm{Pa}[\boldsymbol{v}_i] \cap \mathrm{Pa}[\boldsymbol{v}_j] \neq \emptyset$ for all i, j and only (13) is used, whereas if $\delta = 1$ then the only scenario in which (13) holds is $\{\boldsymbol{v}_z\} = \mathrm{Pa}[\boldsymbol{v}_i] \cap \mathrm{Pa}[\boldsymbol{v}_j]$ in which case the two are equivalent. In univariate settings, the special case in which $\delta = M$, and hence $M_\delta = 0$, leads to an interpretation of (12) as a basis function decomposition; considering all

leaf paths $\mathcal{P}_j$ for $\boldsymbol{v}_j \in \boldsymbol{B}$, this leads to an MRA (Katzfuss, 2017; Katzfuss and Gong, 2019). On the other hand, keeping other parameters constant, $\delta < M$ and in particular $\delta = 1$ may be associated to savings in computing cost, leading to a trade-off between graph complexity and size of reference subsets; see Appendix B.5.

3.1.3 Block-sparse Cholesky Decompositions

In recent work Jurek and Katzfuss (2020) consider sparse Cholesky decompositions of covariance and precision matrices for treed graphs corresponding to the case $\delta = M$ above in the context of space-time filtering; their methods involve sparse Cholesky routines on reverse orderings of $\widetilde{\boldsymbol{C}}^{-1}$ at the level of individual locations. In doing so, the relationship between Cholesky decompositions and $\mathcal{G}$, $\widetilde{\boldsymbol{C}}^{-1}$ and the block structure in $\mathcal{S}$ remains somewhat hidden, and sparse Cholesky libraries are typically associated to bottlenecks in MCMC algorithms. However we note that a consequence of (9) is that it leads to a direct algorithm, for any δ, for the block-decomposition of any symmetric positive-definite matrix $\boldsymbol{\Lambda}$ conforming to $\mathcal{G}$, i.e. with the same block-sparse structure as $\widetilde{\boldsymbol{C}}^{-1}$. This allows us to write $\boldsymbol{\Lambda} = (\boldsymbol{I} - \boldsymbol{L})^\top \boldsymbol{D}(\boldsymbol{I} - \boldsymbol{L})$ where $\boldsymbol{I}$ is the identity matrix, $\boldsymbol{L}$ is block lower triangular with the same block-sparsity pattern as $\boldsymbol{\mathcal{H}}$ above, and $\boldsymbol{D}$ is block diagonal symmetric positive-definite. In Appendix B.2.3 we outline Algorithm 4 which *(i)* makes direct use of the structure of $\mathcal{G}$, *(ii)* computes the decomposition at blocks of reference and non-reference locations, and *(iii)* requires no external sparse matrix library, in particular no sparse Cholesky solvers. Along with Algorithm 5 for the block-computation of $(\boldsymbol{I} - \boldsymbol{L})^{-1}$, it can be used to compute $\boldsymbol{\Lambda}^{-1} = (\widetilde{\boldsymbol{C}}^{-1} + \boldsymbol{\Sigma})^{-1}$ where $\boldsymbol{\Sigma}$ is a block-diagonal matrix; it is thus useful in computing the Gaussian integrated likelihood.

3.2 Estimation and Prediction

We introduce notation to aid in obtaining the full conditional distributions. Write (7) as

$$\boldsymbol{y}(\boldsymbol{\ell}) = \boldsymbol{X}(\boldsymbol{\ell})\boldsymbol{\beta} + \boldsymbol{Z}(\boldsymbol{\ell})\boldsymbol{w}(\boldsymbol{\ell}) + \boldsymbol{\varepsilon}(\boldsymbol{\ell}), \tag{15}$$

where $\boldsymbol{y}(\boldsymbol{\ell}) = (\{y_j(\boldsymbol{\ell})\}_{j=1}^l)^\top$, $\boldsymbol{\varepsilon}(\boldsymbol{\ell}) = (\{\varepsilon_j(\boldsymbol{\ell})\}_{j=1}^l)^\top \sim N(\boldsymbol{0}, \boldsymbol{D}_\tau)$, $\boldsymbol{D}_\tau = \text{diag}(\tau_1^2, \ldots, \tau_l^2)$, $\boldsymbol{X}(\boldsymbol{\ell}) = \text{b.diag}\{\boldsymbol{x}_j(\boldsymbol{\ell})^\top, j = 1, \ldots, l\}$, $\boldsymbol{\beta} = (\boldsymbol{\beta}_{p_1}^\top, \ldots, \boldsymbol{\beta}_{p_j}^\top)^\top$. The $l \times q$ matrix $\boldsymbol{Z}(\boldsymbol{\ell}) = (\boldsymbol{z}_j(\boldsymbol{\ell})^\top, j = 1, \ldots, l)$ with $\boldsymbol{z}_j(\boldsymbol{\ell})^\top = (z_{jk}(\boldsymbol{\ell}), k = 1, \ldots, q)$ acts a design matrix for spatial location $\boldsymbol{\ell}$. Collecting all locations along the j-th margin, we build $\mathcal{T}_j = \{\boldsymbol{\ell}_1^{(j)}, \ldots, \boldsymbol{\ell}_{N_j}^{(j)}\}$ and $\mathcal{T} = \cup_j \mathcal{T}_j$. We then call $\boldsymbol{y}^{(j)} = (y_j(\boldsymbol{\ell}_1^{(j)}), \ldots, y_j(\boldsymbol{\ell}_{N_j}^{(j)}))^\top$ and $\boldsymbol{\varepsilon}^{(j)}$ similarly, $\boldsymbol{X}^{(j)} = (\boldsymbol{x}_j(\boldsymbol{\ell}_1^{(j)}), \ldots, \boldsymbol{x}_j(\boldsymbol{\ell}_{N_j}^{(j)}))^\top$, $\boldsymbol{w}^{(j)} = (\boldsymbol{w}(\boldsymbol{\ell}_1^{(j)}, \boldsymbol{\xi})^\top, \ldots, \boldsymbol{w}(\boldsymbol{\ell}_{N_j}^{(j)}, \boldsymbol{\xi})^\top)^\top$ and $\boldsymbol{Z}^{(j)} = \text{b.diag}\{\boldsymbol{z}_j(\boldsymbol{\ell}_s^{(j)})^\top\}_{s=1}^{N_j}$. The full observed data are $\boldsymbol{y}, \boldsymbol{X}, \boldsymbol{Z}$. Denoting the number of observations as $n = \sum_{j=1}^l N_j$, $\boldsymbol{Z}$ is thus a $n \times qn$ block-diagonal matrix, and similarly $\boldsymbol{w}$ is a $qn \times 1$ vector. We introduce the diagonal matrix $\boldsymbol{D}_n$ such that $\text{diag}(\boldsymbol{D}_n) = (\tau_1^2 \boldsymbol{1}_{N_1}^\top, \ldots, \tau_l^2 \boldsymbol{1}_{N_l}^\top)^\top$.

By construction we may have $\eta(S_i) = \boldsymbol{v}_i$ and $\eta(S_j) = \boldsymbol{v}_j$ such that $(\boldsymbol{\ell}, \boldsymbol{\xi}) \in S_i$ and $(\boldsymbol{\ell}', \boldsymbol{\xi}') \in S_j$ where $\boldsymbol{\ell}' = \boldsymbol{\ell}$, $\boldsymbol{\xi} \neq \boldsymbol{\xi}'$ and similarly for non-reference subsets. Suppose $\mathcal{A} \subset \mathcal{D} \times \Xi$ is a generic reference or non-reference subset. We denote $\bar{\mathcal{A}} \subset \mathcal{D} \times \Xi$ as the set of all combinations of spatial locations of $\mathcal{A}$ and variables i.e. $\bar{\mathcal{A}} = \mathcal{A}\big|_{\mathcal{D}} \times \mathcal{A}\big|_{\Xi}$ where $\mathcal{A}\big|_{\mathcal{D}} \subset \mathcal{D}$ is the set of unique spatial locations in $\mathcal{A}$ and $\mathcal{A}\big|_{\Xi}$ are the unique latent variable coordinates.

By subtraction we find $\mathcal{A}_- = \bar{\mathcal{A}} \setminus \mathcal{A}$ as the set of locations whose spatial location is in $\mathcal{A}$ but whose variable is not. Let $\boldsymbol{y}(\bar{\mathcal{A}}) = \boldsymbol{y}(\mathcal{A}) = (\{\boldsymbol{y}(\boldsymbol{\ell}), \boldsymbol{\ell} \in \mathcal{A}\big|_{\mathcal{D}}\})^\top$, $\boldsymbol{X}(\bar{\mathcal{A}}) = \boldsymbol{X}(\mathcal{A}) = \text{b.diag}\{\boldsymbol{X}(\boldsymbol{\ell})^\top, \boldsymbol{\ell} \in \mathcal{A}\big|_{\mathcal{D}}\}$; values corresponding to unobserved locations will be dealt with by defining $\widetilde{\boldsymbol{D}}_n(\mathcal{A})$ as the diagonal matrix obtained from $\boldsymbol{D}_n$ by replacing unobserved outcomes with zeros. Denote $\boldsymbol{Z}(\bar{\mathcal{A}}) = \text{b.diag}\{\boldsymbol{Z}(\boldsymbol{\ell}), \boldsymbol{\ell} \in \mathcal{A}\big|_{\mathcal{D}}\}$ and $\boldsymbol{w}(\bar{\mathcal{A}})$ similarly. If $\mathcal{A}$ includes L unique spatial locations then $\boldsymbol{y}(\bar{\mathcal{A}})$ is a $L\ l \times 1$ vector and $\boldsymbol{X}(\mathcal{A})$ is a $L\ l \times pl$ matrix. In particular, $\boldsymbol{Z}(\bar{\mathcal{A}})$ is a $L\, l \times Lql$ matrix; the subset of its columns with locations in $\mathcal{A}$ is denoted as $\boldsymbol{Z}(\mathcal{A})$ whereas at other locations we get $\boldsymbol{Z}(\mathcal{A}_-)$. We can then separate the contribution of $\boldsymbol{w}(\mathcal{A})$ to $\boldsymbol{y}(\mathcal{A})$ from the contribution of $\boldsymbol{w}(\mathcal{A}_-)$ by writing $\boldsymbol{y}(\mathcal{A}) = \boldsymbol{X}(\mathcal{A})\boldsymbol{\beta} + \boldsymbol{Z}(\mathcal{A}_-)\boldsymbol{w}(\mathcal{A}_-) + \boldsymbol{Z}(\mathcal{A})\boldsymbol{w}(\mathcal{A}) + \boldsymbol{\varepsilon}(\mathcal{A})$, using which we let $\widetilde{\boldsymbol{y}}(\mathcal{A}) = \boldsymbol{y}(\mathcal{A}) - \boldsymbol{X}(\mathcal{A})\boldsymbol{\beta} - \boldsymbol{Z}(\mathcal{A}_-)\boldsymbol{w}(\mathcal{A}_-)$.

With customary prior distributions $\boldsymbol{\beta} \sim N(\boldsymbol{0}, \boldsymbol{V}_\beta)$ and $\tau_j^2 \sim Inv.Gamma(a_\tau, b_\tau)$ along with a Gaussian SpamTree prior on $\boldsymbol{w}$, we obtain the posterior distribution as

$$p(\boldsymbol{w}, \boldsymbol{\beta}, \{\tau_j^2\}_{j=1}^l, \boldsymbol{\theta} \,|\, \boldsymbol{y}) \propto p(\boldsymbol{y} \,|\, \boldsymbol{w}, \boldsymbol{\beta}, \{\tau_j^2\}_{j=1}^l) p(\boldsymbol{w} \,|\, \boldsymbol{\theta}) p(\boldsymbol{\theta}) p(\boldsymbol{\beta}) \prod_{j=1}^{l} p(\tau_j^2). \tag{16}$$

We compute the full conditional distributions of unknowns in the model, save for $\boldsymbol{\theta}$; iterating sampling from each of these distributions corresponds to a Gibbs sampler which ultimately leads to samples from the posterior distribution above.

3.2.1 Full Conditional Distributions

The full conditional distribution for $\boldsymbol{\beta}$ is Gaussian with covariance $\boldsymbol{\Sigma}^*_{\boldsymbol{\beta}} = (\boldsymbol{V}_{\boldsymbol{\beta}}^{-1} + \boldsymbol{X}^\top \boldsymbol{D}_n^{-1} \boldsymbol{X})^{-1}$ and mean $\mu^*_{\boldsymbol{\beta}} = \boldsymbol{\Sigma}_{\boldsymbol{\beta}} \boldsymbol{X}^\top \boldsymbol{D}_n^{-1} (\boldsymbol{y} - \boldsymbol{Z}\boldsymbol{w})$. For $j = 1, \ldots, l$, $p(\tau_j^2 \,|\, \boldsymbol{\beta}, \boldsymbol{y}, \boldsymbol{w}) = Inv.Gamma(a^*_{\tau,j}, b^*_{\tau,j})$ where $a^*_{\tau,j} = a_\tau + N_j/2$ and $b^*_{\tau,j} = b_\tau + \frac{1}{2}\boldsymbol{E}^{(j)\top}\boldsymbol{E}^{(j)}$ with $\boldsymbol{E}^{(j)} = \boldsymbol{y}^{(j)} - \boldsymbol{X}^{(j)}\boldsymbol{\beta}_j - \boldsymbol{Z}^{(j)}\boldsymbol{w}^{(j)}$.

Take a node $\boldsymbol{v}_i \in \boldsymbol{V}$. If $\boldsymbol{v}_i \in \boldsymbol{A}$ then $\eta^{-1}(\boldsymbol{v}_i) = S_i$ and for $\boldsymbol{v}_j \in \text{Ch}[\boldsymbol{v}_i]$ denote $\widetilde{\boldsymbol{w}}_j = \boldsymbol{w}_j - \boldsymbol{H}_{\setminus i \to j} \boldsymbol{w}_{[\setminus i \to j]}$. The full conditional distribution of $\boldsymbol{w}_i$ is $N(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$, where

$$\begin{aligned} \boldsymbol{\Sigma}_i^{-1} &= \boldsymbol{Z}(S_i)^\top \boldsymbol{D}_n(S_i)^{-1} \boldsymbol{Z}(S_i) + \boldsymbol{R}_i^{-1} + \boldsymbol{F}_i^{(c)} \\ \boldsymbol{\Sigma}_i^{-1}\boldsymbol{\mu}_i &= \boldsymbol{Z}(S_i)^\top \boldsymbol{D}_n(S_i)^{-1} \widetilde{\boldsymbol{y}}(S_i) + \boldsymbol{R}_i^{-1} \boldsymbol{H}_i \boldsymbol{w}_{[i]} + \boldsymbol{m}_i^{(c)} \\ \boldsymbol{F}_i^{(c)} = \sum_{j:\{\boldsymbol{v}_j \in \text{Ch}[\boldsymbol{v}_i]\}} \boldsymbol{H}_{i\to j}^\top \boldsymbol{R}_j^{-1} \boldsymbol{H}_{i \to j} \qquad & \boldsymbol{m}_i^{(c)} = \sum_{j:\{\boldsymbol{v}_j \in \text{Ch}[\boldsymbol{v}_i]\}} \boldsymbol{H}_{i\to j}^\top \boldsymbol{R}_j^{-1} \widetilde{\boldsymbol{w}}_j \end{aligned} \tag{17}$$

If $\boldsymbol{v}_i \in \boldsymbol{B}$ instead $\boldsymbol{\Sigma}_i = (\boldsymbol{Z}(U_i)^\top \boldsymbol{D}_n(U_i)^{-1} \boldsymbol{Z}(U_i) + \boldsymbol{R}_i)^{-1}$ and $\boldsymbol{\mu}_i = \boldsymbol{\Sigma}_i (\boldsymbol{Z}(U_i)^\top \boldsymbol{D}_n(U_i)^{-1} \widetilde{\boldsymbol{y}}(U_i) + \boldsymbol{R}_i^{-1} \boldsymbol{H}_i \boldsymbol{w}_{[i]})$. Sampling of $\boldsymbol{w}$ at nodes at the same level r proceeds in parallel given the assumed conditional independence structure in $\mathcal{G}$. It is thus essential to minimize the computational burden at levels with a small number of nodes to avoid bottlenecks. In particular computing $\boldsymbol{F}_i^{(c)}$ and $\boldsymbol{m}_i^{(c)}$ can become expensive at the root when the number of children is very large. In Algorithm 3 we show that one can efficiently sample at a near-root node $\boldsymbol{v}_i$ by updating $\boldsymbol{F}_i^{(c)}$ and $\boldsymbol{m}_i^{(c)}$ via message-passing from the children of $\boldsymbol{v}_i$.

3.2.2 Update of $\boldsymbol{\theta}$

The full conditional distribution of $\boldsymbol{\theta}$—which may include $\boldsymbol{\xi}_j$ for $j = 1, \ldots, q$ or equivalently $\delta_{ij} = \|\boldsymbol{\xi}_i - \boldsymbol{\xi}_j\|$ if the chosen cross-covariance function is defined on a latent domain of variables—is not available in closed form and sampling a posteriori can proceed

Initialize: $\ell = 0$;
for $r \in \{0, \dots, M\}$ **do**
  for $j : \{\boldsymbol{v}_j \in \boldsymbol{V}_r\}$ **do** `// [parallel for]`
    Compute $\boldsymbol{R}_j^{-1} = (\boldsymbol{C}_j - \boldsymbol{C}_{j,[j]}\boldsymbol{C}_{[j]}^{-1}\boldsymbol{C}_{[j],j})^{-1}$ and $|\boldsymbol{R}_j^{-1}|$;
    $\ell = \ell + \frac{1}{2}\log|\boldsymbol{R}_j^{-1}| - \frac{1}{2}(\boldsymbol{w}_j - \boldsymbol{H}_j\boldsymbol{w}_{[j]})^\top \boldsymbol{R}_j^{-1}(\boldsymbol{w}_j - \boldsymbol{H}_j\boldsymbol{w}_{[j]})$;
    if $Ch[\boldsymbol{v}_j] \neq \emptyset$ **then**
      Identify $\boldsymbol{v}_i \in \mathrm{Ch}[\boldsymbol{v}_j]$ such that $\boldsymbol{v}_i \in \boldsymbol{V}_{r+1}$;
      Compute and store $\boldsymbol{C}_{[i]}^{-1}$ (possibly via (11));
Result: $\exp(\ell) \propto p(\boldsymbol{w} \,|\, \boldsymbol{\theta}) = \prod_i N(\boldsymbol{w}_i \,|\, \boldsymbol{H}_i\boldsymbol{w}_{[i]}, \boldsymbol{R}_i)$.

Algorithm 1: Computing $p(\boldsymbol{w} \,|\, \boldsymbol{\theta})$.

Input: $\boldsymbol{C}_{[j]}$ for all j from Algorithm 1;
$\boldsymbol{W}_e = \bigcup_{r \text{ is even}} \boldsymbol{V}_r$; $\boldsymbol{W}_o = \bigcup_{r \text{ is odd}} \boldsymbol{V}_r$;
for $i \in \{e, o\}$ **do**
  for $j : \{\boldsymbol{v}_j \in \boldsymbol{W}_i\}$ **do** `// [parallel for]`
    Sample $\boldsymbol{w}_j \sim N(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$ using (17);
    Let $\mathrm{Pa}[\boldsymbol{v}_j] = \{\boldsymbol{v}_p\}$, then $\boldsymbol{m}_p^{(c)} = \boldsymbol{H}_j^\top \boldsymbol{R}_j^{-1}\boldsymbol{w}_j$ and $\boldsymbol{F}_p^{(c)} = \boldsymbol{H}_j^\top \boldsymbol{R}_j^{-1}\boldsymbol{H}_j$;
Result: sample from $p(\boldsymbol{w}_j \,|\, \boldsymbol{w}_{-j}, \boldsymbol{y}, \boldsymbol{\beta}, \boldsymbol{\theta}, \boldsymbol{\tau})$ for all $\boldsymbol{v}_j \in \boldsymbol{V}$.

Algorithm 2: Sampling from the full conditional distribution of $\boldsymbol{w}_i$ when $\delta = 1$.

Input: $\boldsymbol{C}_{[j]}$ for all j from Algorithm 1
Initialize: for all i, $\boldsymbol{m}_i^{(c)} = \boldsymbol{0}_{n_i \times 1}$ and $\boldsymbol{F}_i^{(c)} = \boldsymbol{O}_{n_i \times n_i}$;
for $r \in \{M, \dots, 0\}$ **do**
  for $j : \{\boldsymbol{v}_j \in \boldsymbol{V}_r\}$ **do** `// [parallel for]`
    Sample $\boldsymbol{w}_j \sim N(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$ using (17);
    for $p : \{\boldsymbol{v}_p \in Pa[\boldsymbol{v}_j]\}$ **do**
      $\boldsymbol{m}_p^{(c)} = \boldsymbol{m}_p^{(c)} + \boldsymbol{H}_{p\to j}^\top \boldsymbol{R}_j^{-1}\boldsymbol{w}_j$;
      $\boldsymbol{F}_p^{(c)} = \boldsymbol{F}_p^{(c)} + \boldsymbol{H}_{p\to j}^\top \boldsymbol{R}_j^{-1}\boldsymbol{H}_{p\to j}$;
Result: sample from $p(\boldsymbol{w}_j \,|\, \boldsymbol{w}_{-j}, \boldsymbol{y}, \boldsymbol{\beta}, \boldsymbol{\theta}, \boldsymbol{\tau})$ for all $\boldsymbol{v}_j \in \boldsymbol{V}$.

Algorithm 3: Sampling from the full conditional distribution of $\boldsymbol{w}_j$ when $\delta = M$.

via Metropolis-Hastings steps which involve accept/reject steps with acceptance probability $\alpha = \min\{1, \frac{p(\boldsymbol{w}\,|\,\boldsymbol{\theta}')p(\boldsymbol{\theta}')q(\boldsymbol{\theta}\,|\,\boldsymbol{\theta}')}{p(\boldsymbol{w}\,|\,\boldsymbol{\theta})p(\boldsymbol{\theta})q(\boldsymbol{\theta}'\,|\,\boldsymbol{\theta})}\}$. In our implementation, we adaptively tune the standard deviation of the proposal distribution via the robust adaptive Metropolis algorithm (RAM; Vihola, 2012). In these settings, unlike similar models based on DAG representations such as NNGPs and MGPs, direct computation via $p(\boldsymbol{w}\,|\,\boldsymbol{\theta}) = \prod_i N(\boldsymbol{w}_i\,|\,\boldsymbol{H}_i\boldsymbol{w}_{[i]}, \boldsymbol{R}_i)$ is inefficient as it requires computing $\boldsymbol{C}_{[i]}^{-1}$ whose size grows along the hierarchy in $\mathcal{G}$. We thus outline Algorithm 1 for computing $p(\boldsymbol{w}\,|\,\boldsymbol{\theta})$ via (11). As an alternative we can perform the update using ratios of $p(\boldsymbol{y}\,|\,\boldsymbol{\beta},\boldsymbol{\theta},\boldsymbol{\tau}) = \int p(\boldsymbol{y}\,|\,\boldsymbol{w},\boldsymbol{\beta},\boldsymbol{\tau})p(\boldsymbol{w}\,|\,\boldsymbol{\theta})d\boldsymbol{w} = N(\boldsymbol{y}\,|\,\boldsymbol{X}\boldsymbol{\beta}, \boldsymbol{Z}\widetilde{\boldsymbol{C}}\boldsymbol{Z}^\top + \boldsymbol{D}_n)$ using Algorithms 4 and 5 outlined in Appendix B.2.3 which require no sparse matrix library.

3.2.3 Graph Coloring for Parallel Sampling

An advantage of the treed structure of $\mathcal{G}$ is that it leads to fixed graph coloring associated to parallel Gibbs sampling; no graph coloring algorithms are necessary (see e.g. Molloy and Reed, 2002; Lewis, 2016). Specifically, if $\delta = M$ (full depth) then there is a one to one correspondence between the $M+1$ levels of $\mathcal{G}$ and graph colors, as evidenced by the parallel blocks in Algorithms 1 and 3. In the case $\delta = 1$, $\mathcal{G}$ is associated to only two colors alternating the odd levels with the even ones. This is possible because the Markov blanket of each node at level r, with r even, only includes nodes at odd levels, and vice-versa.

3.2.4 Prediction of the Outcome at New Locations

The Gibbs sampling algorithm will iterate across the above steps and, upon convergence, will produce samples from $p(\boldsymbol{\beta}, \{\tau_j^2\}_{j=1}^q, \boldsymbol{w}\,|\,\boldsymbol{y})$. We obtain posterior predictive inference at arbitrary $\boldsymbol{\ell} \in \mathcal{D}$ by evaluating $p(\boldsymbol{y}(\boldsymbol{\ell})\,|\,\boldsymbol{y})$. If $\boldsymbol{\ell} \in \mathcal{S} \cup \mathcal{U}$, then we draw one sample of $\boldsymbol{y}(\boldsymbol{\ell}) \sim N(\boldsymbol{X}(\boldsymbol{\ell})^\top\boldsymbol{\beta} + \boldsymbol{Z}(\boldsymbol{\ell})^\top\boldsymbol{w}(\boldsymbol{\ell}), \boldsymbol{D}_n(\boldsymbol{\ell}))$ for each draw of the parameters from $p(\boldsymbol{\beta}, \{\tau_j^2\}_{j=1}^l, \boldsymbol{w}\,|\,\boldsymbol{y})$. Otherwise, considering that $\eta(\boldsymbol{\ell}) = \boldsymbol{v}_j \in \boldsymbol{B}$ for some j, with parent nodes $\mathrm{Pa}[\boldsymbol{v}_j]$, we sample $\boldsymbol{w}(\boldsymbol{\ell})$ from the full conditional $N(\boldsymbol{\mu}_{\boldsymbol{\ell}}^*, \boldsymbol{\Sigma}_{\boldsymbol{\ell}}^*)$, where $\boldsymbol{\Sigma}_{\boldsymbol{\ell}}^* = (\boldsymbol{Z}(\boldsymbol{\ell})\boldsymbol{D}_n(\boldsymbol{\ell})^{-1}\boldsymbol{Z}(\boldsymbol{\ell})^\top + \boldsymbol{R}_{\boldsymbol{\ell}}^{-1})^{-1}$ and $\boldsymbol{\mu}_{\boldsymbol{\ell}}^* = \boldsymbol{\Sigma}_{\boldsymbol{\ell}}^*(\boldsymbol{Z}(\boldsymbol{\ell})\boldsymbol{D}^{-1}(\boldsymbol{y}(\boldsymbol{\ell}) - \boldsymbol{X}(\boldsymbol{\ell})^\top\boldsymbol{\beta}) + \boldsymbol{R}_{\boldsymbol{\ell}}^{-1}\boldsymbol{H}_{\boldsymbol{\ell}}\boldsymbol{w}_{[j]})$, then draw $\boldsymbol{y}(\boldsymbol{\ell}) \sim N(\boldsymbol{X}(\boldsymbol{\ell})^\top\boldsymbol{\beta} + \boldsymbol{Z}(\boldsymbol{\ell})^\top\boldsymbol{w}(\boldsymbol{\ell}), \boldsymbol{D}_n)$.

3.2.5 Computing and Storage Cost

The update of τ_j^2 and $\boldsymbol{\beta}$ can be performed at a minimal cost as typically $p = \sum_{j=1}^l p_j$ is small; almost all the computation budget must be dedicated to computing $p(\boldsymbol{w}\,|\,\boldsymbol{\theta})$ and sampling $p(\boldsymbol{w}\,|\,\boldsymbol{y},\boldsymbol{\beta},\boldsymbol{\tau}^2)$. Assume that reference locations are all observed $\mathcal{S} \subset \mathcal{T}$ and that all reference subsets have the same size i.e. $|S_i| = N_s$ for all i. We show in Appendix B.5 that the cost of computing SpamTrees is $O(nN_s^2)$. As a result, SpamTrees compare favorably to other models specifically in not scaling with the cube of the number of samples. δ does not impact the computational order, however, compared to $\delta = M$, choosing $\delta = 1$ lowers the cost by a factor of M or more. For a fixed reference set partition and corresponding nodes, choosing larger δ will result in stronger dependence between leaf nodes and nodes closer to the root—this typically corresponds to leaf nodes being assigned conditioning sets that span larger distances in space. The computational speedup corresponding to choosing $\delta = 1$ can effectively be traded for a coarser partitioning of $\mathcal{S}$, resulting in large conditioning sets that are more local to the leaves.

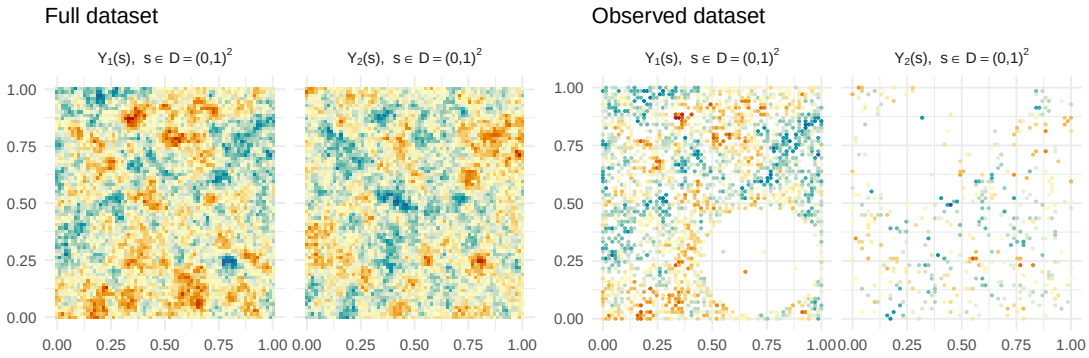

Figure 3: Left half: *Full data set* – a bivariate outcome is generated on 4,900 spatial locations. Right half: *Observed data set* – the training sample is built via independent subsampling of each outcome.

4. Applications

We consider Gaussian SpamTrees for the multivariate regression model (15). Consider the spatial locations $\boldsymbol{\ell}, \boldsymbol{\ell}' \in \mathcal{D}$ and the locations of variables i and j in the latent domain of variables $\boldsymbol{\xi}_i, \boldsymbol{\xi}_j \in \Xi$, then denote $\boldsymbol{h} = \|\boldsymbol{\ell} - \boldsymbol{\ell}'\|$, $\Delta = \delta_{ij} = \|\boldsymbol{\xi}_i - \boldsymbol{\xi}_j\|$, and

$$C(\boldsymbol{h}, \Delta) = \frac{\exp\left\{-\phi\|\boldsymbol{h}\| / \exp\left\{\frac{1}{2}\beta\log(1+\alpha\Delta)\right\}\right\}}{\exp\left\{\beta\log(1+\alpha\Delta)\right\}}.$$

For $j = 1, \ldots, q$ we also introduce $C_j(\boldsymbol{h}) = \exp\{-\phi_j\|\boldsymbol{h}\|\}$. A non-separable cross-covariance function for a multivariate process can be defined as

$$\mathrm{Cov}(w(\boldsymbol{\ell}, \boldsymbol{\xi}_i), w(\boldsymbol{\ell}', \boldsymbol{\xi}_j)) = \boldsymbol{C}_{ij}(\boldsymbol{h}) = \begin{cases} \sigma_{i1}^2 C(\boldsymbol{h}, \delta_{ij}) + \sigma_{i2}^2 C_i(\boldsymbol{h}) & \text{if } i = j \\ \sigma_{i1}\sigma_{j1} C(\boldsymbol{h}, \delta_{ij}) & \text{if } i \neq j, \end{cases} \tag{18}$$

which is derived from eq. (7) of Apanasovich and Genton (2010); locations of variables in the latent domain are unknown, therefore $\boldsymbol{\theta} = \{\sigma_{i1}, \sigma_{i2}, \phi_i\}_{i=1,\ldots,q} \cup \{\delta_{ij}\}_{i=1,\ldots,q}^{j<i} \cup \{\alpha, \beta, \phi\}$ for a total of $3q + q(q-1)/2 + 3$ unknown parameters.

4.1 Synthetic Data

In this section we focus on bivariate outcomes ($q = 2$). We simulate data from model (15), setting $\boldsymbol{\beta} = \boldsymbol{0}$, $\boldsymbol{Z} = I_q$ and take the measurement locations on a regular grid of size 70×70 for a total of 4,900 spatial locations. We simulate the bivariate spatial field by sampling from the full GP using (18) as cross-covariance function; the nuggets for the two outcomes are set to $\tau_1^2 = 0.01$ and $\tau_2^2 = 0.1$. For $j = 1, 2$ we fix $\sigma_{j2} = 1, \alpha = 1, \beta = 1$ and independently sample $\sigma_{j1} \sim U(-3, 3), \phi_j \sim U(0.1, 3), \phi \sim U(0.1, 30), \delta_{12} \sim Exp(1)$, generating a total of 500 bivariate data sets. This setup leads to empirical spatial correlations between the two outcomes smaller than 0.25, between 0.25 and 0.75, and larger than 0.75 in absolute value in 107, 330, and 63 of the 500 data sets, respectively. We introduce misalignment and make the outcomes imbalanced by replacing the first outcome with missing values at $\approx 50\%$ of the spatial locations chosen uniformly at random, and then repeating this procedure for the

second outcome keeping only $\approx 10\%$ of the total locations. We also introduce almost-empty regions of the spatial domain, independently for each outcome, by replacing observations with missing values at $\approx 99\%$ of spatial locations inside small circular areas whose center is chosen uniformly at random in $[0,1]^2$. As a result of these setup choices, each simulated data set reproduces some features of the real-world unbalanced misaligned data we consider in Section 4.2 at a smaller scale and in a controlled experiment. Figure 3 shows one of the resulting 500 data sets.

We consider SpamTrees with $\delta = 1$ and implement multiple variants of SpamTrees with $\delta = M$ in order to assess their sensitivity to design parameters. Table 1 reports implementation setups and the corresponding results in all cases; if the design variable "All outcomes at $\boldsymbol{\ell}$" is set to "No" then a SpamTree is built on the $\mathcal{D} \times \Xi$ domain. If it is set to "Yes" the DAG will be built using $\mathcal{D}$ only – in other words, the q margins of the latent process are never separated by the DAG if they are measured at the same spatial location. "Cherry pick same outcome" indicates whether the map $\eta(\cdot)$ should search for neighbors by first filtering for matching outcomes – refer to our discussion at Section 2.1.2. We mention here if the DAG is built using $\mathcal{D}$ only, then the nearest neighbor found via cherry picking will always include a realization of the same margin of $\boldsymbol{w}(\cdot)$. Finally, if SpamTree is implemented with "Root bias" then the reference set and the DAG are built with locations of the more sparsely observed outcome closer to root nodes of the tree as suggested by Proposition 1.

SpamTrees are compared with multivariate cubic meshed GPs (Q-MGPs; Peruzzi et al., 2020), a method based on stochastic partial differential equations (Lindgren et al., 2011) estimated via integrated nested Laplace approximations (Rue et al., 2009) implemented via `R-INLA` using a 15×15 grid and labeled spde-inla, a low-rank multivariate GP method (labeled lowrank) on 25 knots obtained via SpamTrees by setting $M = 1$ with no domain partitioning, and an independent partitioning GP method (labeled ind-part) implemented by setting $M = 1$ and partitioning the domain into 25 regions. Refer e.g. to Heaton et al. (2019) for an overview of low-rank and independent partitioning methods. All multivariate SpamTree variants, Q-MGPs, lowrank and ind-part use (18) as the cross-covariance function in order to evaluate their relative performance in estimating $\boldsymbol{\theta}$ in terms of root mean square error (RMSE) as reported in Table 1. We also include results from a non-spatial regression using Bayesian additive regression trees (BART; Chipman et al., 2010) which uses the domain coordinates as covariates in addition to a binary fixed effect corresponding to the outcome index. All methods were setup to target a compute time of approximately 15 seconds for each data set. We focused on comparing the different methods under computational constraints because (a) without constraints it would not be feasible to implement the methods for many large simulated spatial datasets; and (b) the different methods are mostly focused on providing a faster approximation to full GPs; if constraints were removed one would just be comparing the same full GP method.

Table 1 reports average performance across all data sets. All Bayesian methods based on latent GPs exhibit very good coverage; in these simulated scenarios, SpamTrees exhibit comparatively lower out-of-sample prediction errors. All SpamTrees perform similarly, with the best out-of-sample predictive performance achieved by the SpamTrees cherry picking based solely on spatial distance (i.e. disregarding whether or not the nearest-neighbor belongs to the same margin). Additional implementation details can be found in Appendix

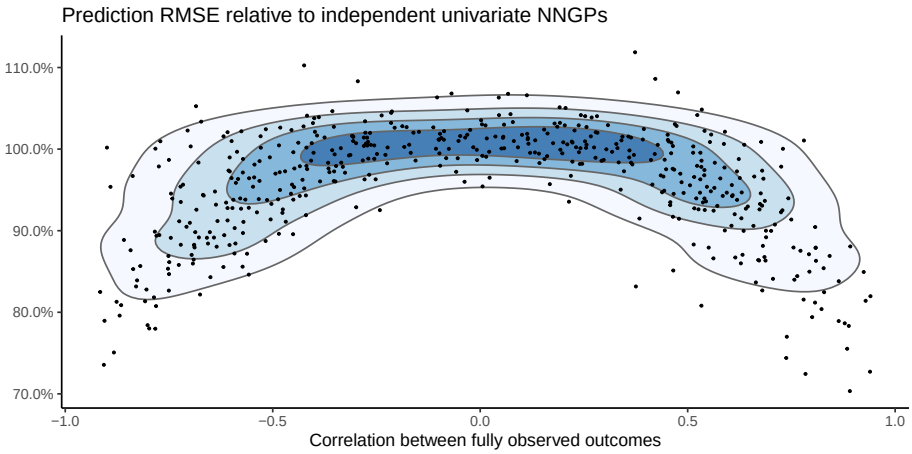

Figure 4: Predictive RMSE of the best-performing SpamTree of Table 1 relative to independent univariate NNGP models of the two outcomes, for different empirical correlations between the two outcomes in the full data. Lower values indicate smaller errors of SpamTrees in predictions.

C.2.1. Finally, we show in Figure 4 that the relative gains of SpamTrees compared to independent univariate NNGP model of the outcomes are increasing with the magnitude of the correlations between the two outcomes, which are only available due to the simulated nature of the data sets.

	All outcomes at $\boldsymbol{\ell}$	Cherry pick same outcome	Root bias	RMSE($\boldsymbol{y}$)	MAE($\boldsymbol{y}$)	COVG($\boldsymbol{y}$)	RMSE($\boldsymbol{\theta}$)
SpamTrees $\delta = M$	No	No	No	1.078	0.795	0.955	4.168
	No	No	Yes	**1.065**	**0.786**	0.955	4.138
	No	Yes	No	1.083	0.799	0.954	4.168
	No	Yes	Yes	1.085	0.799	0.954	4.138
	Yes	Yes	No	1.081	0.797	0.954	**4.080**
	Yes	Yes	Yes	1.087	0.801	0.954	4.188
SpamTrees $\delta = 1$	Yes	Yes	No	1.198	0.880	0.956	4.221
Q-MGP	Yes	–	–	1.125	0.819	**0.951**	4.389
IND-PART	Yes	–	–	1.624	1.229	0.948	8.064
LOWRANK	Yes	–	–	1.552	1.173	0.952	5.647
SPDE-INLA	Yes	–	–	1.152	0.862	0.913	
SpamTrees Univariate	–	–	–	1.147	0.846	0.953	
NNGP Univariate	–	–	–	1.129	0.832	0.952	
BART	–	–	–	1.375	1.036	0.488	

Table 1: Prediction and estimation performance on multivariate synthetic data. The four columns on the right refer to root mean square error (RMSE) and mean absolute error (MAE) in out-of-sample predictions, average coverage of empirical 95% prediction intervals, and RMSE in the estimation of $\boldsymbol{\theta}$.

4.2 Climate Data: MODIS-TERRA and GHCN

Climate data are collected from multiple sources in large quantities; when originating from satellites and remote sensing, they are typically collected at high spatial and relatively

low temporal resolution. Atmospheric and land-surface products are obtained via post-processing of satellite imaging, and their quality is negatively impacted by cloud cover and other atmospheric disturbances. On the other hand, data from a relatively small number of land-based stations is of low spatial but high temporal resolution. An advantage of land-based stations is that they measure phenomena related to atmospheric conditions which cannot be easily measured from satellites (e.g. precipitation data, depth of snow cover).

We consider the joint analysis of five spatial outcomes collected from two sources. First, we consider Moderate Resolution Imaging Spectroradiometer (MODIS) data from the Terra satellite which is part of the NASA's Earth Observing System. Specifically, data product `MOD11C3` v. 6 provides monthly Land Surface Temperature (LST) values in a 0.05 degree latitude/longitude grid (the Climate Modeling Grid or CMG). The monthly data sets cover the whole globe from 2000-02-01 and consist of daytime and nighttime LSTs, quality control assessments, in addition to emissivities and clear-sky observations. The second source of data is the Global Historical Climatology Network (GHCN) database which includes climate summaries from land surface stations across the globe subjected to common quality assurance reviews. Data are published by the National Centers of Environmental Information (NCEI) of the National Oceanic and Atmospheric Administration (NOAA) at several different temporal resolutions; daily products report five core elements (precipitation, snowfall, snow depth, maximum and minimum temperature) in addition to several other measurements.

We build our data set for analysis by focusing on the continental United States in October, 2018. The MODIS data correspond to 359,822 spatial locations. Of these, 250,874 are collected at the maximum reported quality; we consider all remaining 108,948 spatial locations as missing, and extract (1) daytime LST (`LST_Day_CMG`), (2) nighttime LST (`LST_Night_CMG`), (3) number of days with clear skies (`Clear_sky_days`), (4) number of nights with clear skies (`Clear_sky_nights`). From the GHCN database we use daily data to obtain monthly averages for precipitation (`PRCP`), which is available at 24,066 spatial locations corresponding to U.S. weather stations; we log-transform `PRCP`. The two data sources do not share measurement locations as there is no overlap between measurement locations in MODIS and GHCN, with the latter data being collected more sparsely—this is a scenario of complete spatial misalignment. From the resulting data set of size n =1,027,562 we remove all observations in a large 3×3 degree area in the central U.S. (from -100W to -97W and from 35N to 38N, i.e. the red area of Figure 5) to build a test set on which we calculate coverage and RMSE of the predictions.

We implement SpamTrees using the cross-covariance function (18). Considering that `PRCP` is more sparsely measured and following Proposition 1, we build SpamTrees favoring placement of GHCN locations at root nodes. We compare SpamTrees with a Q-MGP model built on the same cross-covariance function, and two univariate models that make predictions independently for each outcome. Comparisons with other multivariate methods are difficult due to the lack of scalable software for this data size which also deals with misalignment and imbalances across outcomes. Compute times per MCMC iteration ranged from 2.4s/iteration of the multivariate Q-MGP model, to 1.5s/iteration of the univariate NNGP model. The length of the MCMC chains (30,000 for SpamTrees and 20,000 for Q-MGP) was such that the total compute time was about the same for both models at less than 16 hours. Univariate models cannot estimate cross-covariances of multivariate outcomes and are thus associated to faster compute times; we set the length of their MCMC

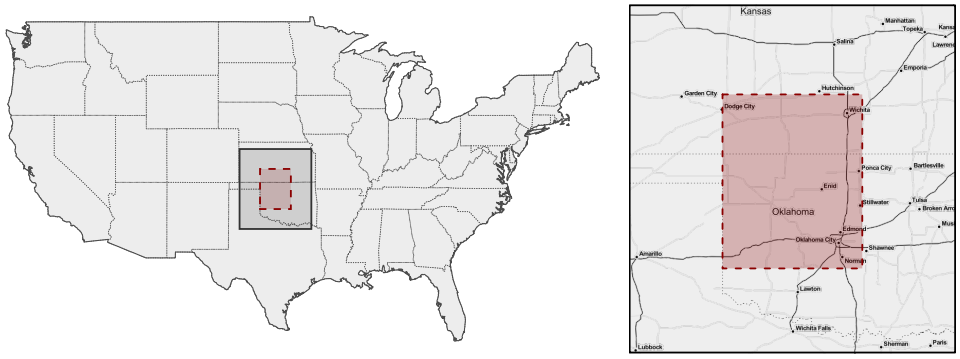

Figure 5: Prediction area

MODIS/GHCN variables		*Multivariate*		*Univariate*	
		SpamTree	Q-MGP	SpamTree	NNGP
`Clear_sky_days`	RMSE	1.611	1.928	1.466	1.825
	COVG	0.980	0.866	0.984	0.986
`Clear_sky_nights`	RMSE	1.621	1.766	2.002	2.216
	COVG	0.989	0.943	0.992	0.992
`LST_Day_CMG`	RMSE	1.255	1.699	1.645	1.666
	COVG	1.000	1.000	1.000	1.000
`LST_Night_CMG`	RMSE	1.076	1.402	0.795	1.352
	COVG	0.999	0.999	1.000	1.000
`PRCP`	RMSE	0.517	0.632	0.490	0.497
	COVG	0.972	1.000	0.969	0.958

Table 2: Prediction results over the 3×3 degree area shown in Figure 5

chains to 15,000 for a total compute time of less than 7 hours for both models. We provide additional details about the models we implemented at Appendix C.

Table 2 reports predictive performance of all models, and Figure 6 maps the predictions at all locations from SpamTrees and the corresponding posterior uncertainties. Multivariate models appear advantageous in predicting some, but not all outcomes in this real world illustration; nevertheless, SpamTrees outperformed a Q-MGP model using the same cross-covariance function. Univariate models perform well and remain valid for predictions, but cannot estimate multivariate relationships. We report posterior summaries of $\boldsymbol{\theta}$ in Appendix C.2.2. Opposite signs of σ_{i1} and σ_{j1} for pairs of variables $i, j \in \{1, \ldots, q\}$ imply a negative relationship; however, the degree of spatial decay of these correlations is different for each pair as prescribed by the latent distances in the domain of variables δ_{ij}. Figure 7 depicts the resulting cross-covariance function for three pairs of variables.

5. Discussion

In this article, we introduced SpamTrees for Bayesian spatial multivariate regression modeling and provided algorithms for scalable estimation and prediction. SpamTrees add

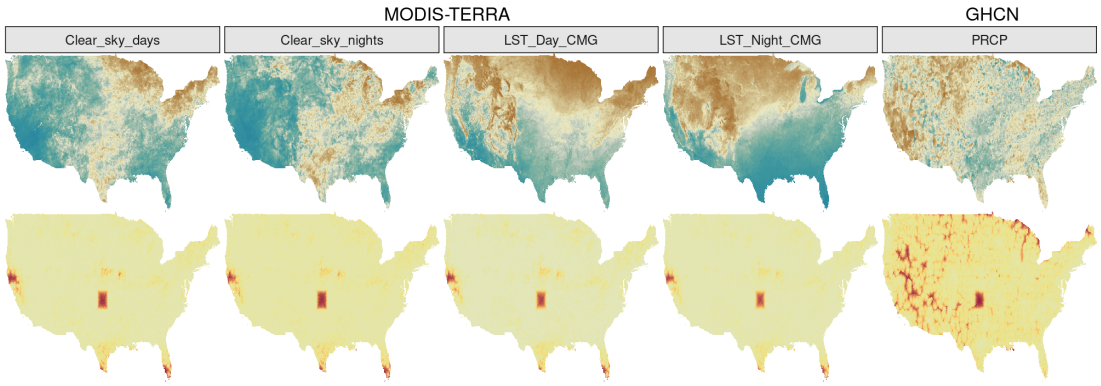

Figure 6: Predicted values of the outcomes at all locations (top row) and associated 95% uncertainty (bottom row), with darker spots corresponding to wider credible intervals.

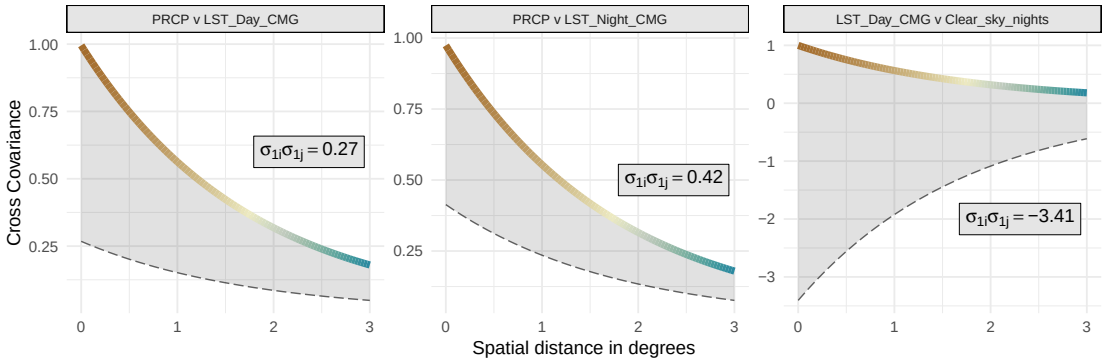

Figure 7: Given the latent dimensions δ_{ij}, the color-coded lines represent $C(\boldsymbol{h}, \delta_{ij})$ whereas $\boldsymbol{C}_{ij}(\boldsymbol{h}) = \sigma_{1i}\sigma_{1j}C(\boldsymbol{h}, \delta_{ij})$ is shown as a dashed grey line.

significantly to the class of methods for regression in spatially-dependent data settings. We have demonstrated that SpamTrees maintain accurate characterization of spatial dependence and scalability even in challenging settings involving multivariate data that are spatially misaligned. Such complexities create problems for competing approaches, including recent DAG-based approaches ranging from NNGPs to MGPs.

One potential concern is the need for users to choose a tree, and in particular specify the number of locations associated to each node and the multivariate composition of locations in each node. Although one can potentially estimate the tree structure based on the data, this would eliminate much of the computational speedup. We have provided theoretical guidance based on KL divergence from the full GP and computational cost associated to different tree structures. This and our computational experiments lead to practical guidelines that can be used routinely in tree building. Choosing a tree provides a useful degree of user-input to refine and improve upon an approach.

We have focused on sampling algorithms for the latent effects because they provide a general blueprint which may be used for posterior computations in non-Gaussian outcome models; efficient algorithms for non-Gaussian big geostatistical data sets are currently lacking and are the focus of ongoing research. SpamTrees can be built on larger dimensional inputs for general applications in regression and/or classifications; such a case requires special considerations regarding domain partitioning and the construction of the tree. In particular, when time is available as a third dimension it may be challenging to build a sparse DAG with reasonable assumptions on temporal dependence. For these reasons, future research may be devoted to building sparse DAG methods combining the advantages of treed structures with e.g. Markov-type assumptions of conditional independence, and applying SpamTrees to data with larger dimensional inputs.

Acknowledgments

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 856506). This project was also partially funded by grant R01ES028804 of the United States National Institutes of Health (NIH).

Appendix A. Kolmogorov consistency conditions for SpamTrees

We adapt results from Datta et al. (2016a) and Peruzzi et al. (2020). Let $w(\boldsymbol{s}), \boldsymbol{s} \in \mathcal{D}^*$ be the univariate representation in the augmented domain of the multivariate base process $\{\boldsymbol{w}(\boldsymbol{\ell}), \boldsymbol{\ell} \in \mathcal{D} \subset \Re^d\}$. Fix the reference set $\mathcal{S} \subset \mathcal{D}^*$ and let $\mathcal{L} = \{\boldsymbol{\ell}_1, \ldots, \boldsymbol{\ell}_n\} \subset \mathcal{D}^*$ and $\mathcal{U} = \mathcal{L} \setminus \mathcal{S}$. Then

$$\begin{aligned}\int \widetilde{p}(\boldsymbol{w}_{\mathcal{L}}) \prod_{\boldsymbol{\ell}_i \in \mathcal{L}} dw(\boldsymbol{\ell}_i) &= \int\int \widetilde{p}(\boldsymbol{w}_{\mathcal{U}} \mid \boldsymbol{w}_{\mathcal{S}}) \widetilde{p}(\boldsymbol{w}_{\mathcal{S}}) \prod_{\boldsymbol{s}_i \in \mathcal{S} \setminus \mathcal{L}} dw(\boldsymbol{s}_i) \prod_{\boldsymbol{\ell}_i \in \mathcal{L}} dw(\boldsymbol{\ell}_i) \\ &= \int \widetilde{p}(\boldsymbol{w}_{\mathcal{S}}) \left(\int \widetilde{p}(\boldsymbol{w}_{\mathcal{U}} \mid \boldsymbol{w}_{\mathcal{S}}) \prod_{\boldsymbol{\ell}_i \in \mathcal{U}} dw(\boldsymbol{\ell}_i) \right) \prod_{\boldsymbol{\ell}_i \in \mathcal{S}} dw(\boldsymbol{\ell}_i) = 1,\end{aligned}$$

hence $\widetilde{p}(\boldsymbol{w}_{\mathcal{L}})$ is a proper joint density. To verify the Kolmogorov consistency conditions, take the permutation $\mathcal{L}_\pi = \{\boldsymbol{\ell}_{\pi(1)}, \ldots, \boldsymbol{\ell}_{\pi(n)}\}$ and call $\mathcal{U}_\pi = \mathcal{L}_\pi \setminus \mathcal{S}$. Clearly $\mathcal{U}_\pi = \mathcal{L}_\pi \setminus \mathcal{S} = \mathcal{L} \setminus \mathcal{S} = \mathcal{U}$ and similarly $\mathcal{S} \setminus \mathcal{L}_\pi = \mathcal{S} \setminus \mathcal{L}$ so that

$$\begin{aligned}\widetilde{p}(\boldsymbol{w}_{\mathcal{L}_\pi}) &= \int \widetilde{p}(\boldsymbol{w}_{\mathcal{U}_\pi} \mid \boldsymbol{w}_{\mathcal{S}}) \widetilde{p}(\boldsymbol{w}_{\mathcal{S}}) \prod_{\boldsymbol{s}_i \in \mathcal{S} \setminus \mathcal{L}_\pi} dw(\boldsymbol{s}_i) \\ &= \int \widetilde{p}(\boldsymbol{w}_{\mathcal{U}} \mid \boldsymbol{w}_{\mathcal{S}}) \widetilde{p}(\boldsymbol{w}_{\mathcal{S}}) \prod_{\boldsymbol{s}_i \in \mathcal{S} \setminus \mathcal{L}} dw(\boldsymbol{s}_i) = \widetilde{p}(\boldsymbol{w}_{\mathcal{L}})\end{aligned}$$

implying

$$\widetilde{p}(\boldsymbol{w}(\boldsymbol{\ell}_1), \ldots, \boldsymbol{w}(\boldsymbol{\ell}_n)) = \widetilde{p}(\boldsymbol{w}(\boldsymbol{\ell}_{\pi(1)}), \ldots, \boldsymbol{w}(\boldsymbol{\ell}_{\pi(n)})).$$

Next, take a new location location $\boldsymbol{\ell}_0 \in \mathcal{D}^*$. Call $\mathcal{L}_1 = \mathcal{L} \cup \{\boldsymbol{\ell}_0\}$. We want to show that $\int \widetilde{p}(\boldsymbol{w}_{\mathcal{L}_1}) dw(\boldsymbol{\ell}_0) = \widetilde{p}(\boldsymbol{w}_{\mathcal{L}})$. If $\boldsymbol{\ell}_0 \in \mathcal{S}$ then $\mathcal{L}_1 \setminus \mathcal{S} = \mathcal{L} \setminus \mathcal{S} = \mathcal{U}$ and hence

$$\begin{aligned}\int \widetilde{p}(\boldsymbol{w}_{\mathcal{L}_1}) dw(\boldsymbol{\ell}_0) &= \int \left(\widetilde{p}(\boldsymbol{w}_{\mathcal{L}_1 \setminus \mathcal{S}} \mid \boldsymbol{w}_{\mathcal{S}}) \widetilde{p}(\boldsymbol{w}_{\mathcal{S}}) \prod_{\boldsymbol{s}_i \in \mathcal{S} \setminus \mathcal{L}_1} dw(\boldsymbol{s}_i) \right) dw(\boldsymbol{\ell}_0) \\ &= \int \widetilde{p}(\boldsymbol{w}_{\mathcal{U}} \mid \boldsymbol{w}_{\mathcal{S}}) \widetilde{p}(\boldsymbol{w}_{\mathcal{S}}) \prod_{\boldsymbol{s}_i \in \mathcal{S} \setminus \mathcal{L}} dw(\boldsymbol{s}_i) = \widetilde{p}(\boldsymbol{w}_{\mathcal{L}}).\end{aligned}$$

If $\boldsymbol{\ell}_0 \notin \mathcal{S}$ we have

$$\begin{aligned}\int \widetilde{p}(\boldsymbol{w}_{\mathcal{L}_1}) dw(\boldsymbol{\ell}_0) &= \int \left(\int \widetilde{p}(\boldsymbol{w}_{\mathcal{L}_1 \setminus \mathcal{S}} \mid \boldsymbol{w}_{\mathcal{S}}) \widetilde{p}(\boldsymbol{w}_{\mathcal{S}}) \prod_{\boldsymbol{s}_i \in \mathcal{S} \setminus \mathcal{L}_1} dw(\boldsymbol{s}_i) \right) dw(\boldsymbol{\ell}_0) \\ &= \int \left(\int \widetilde{p}(\boldsymbol{w}_{\mathcal{L} \setminus \mathcal{S} \cup \{\boldsymbol{\ell}_0\}} \mid \boldsymbol{w}_{\mathcal{S}}) \widetilde{p}(\boldsymbol{w}_{\mathcal{S}}) \prod_{\boldsymbol{s}_i \in \mathcal{S} \setminus \mathcal{L}} dw(\boldsymbol{s}_i) \right) dw(\boldsymbol{\ell}_0) \\ &= \int \left(\int \widetilde{p}(\boldsymbol{w}_{\{\boldsymbol{\ell}_0\}} \mid \boldsymbol{w}_{\mathcal{L} \setminus \mathcal{S}}, \boldsymbol{w}_{\mathcal{S}}) \widetilde{p}(\boldsymbol{w}_{\mathcal{L} \setminus \mathcal{S}} \mid \boldsymbol{w}_{\mathcal{S}}) \widetilde{p}(\boldsymbol{w}_{\mathcal{S}}) \prod_{\boldsymbol{s}_i \in \mathcal{S} \setminus \mathcal{L}} dw(\boldsymbol{s}_i) \right) dw(\boldsymbol{\ell}_0)\end{aligned}$$

$$= \int \widetilde{p}(\boldsymbol{w}_{\mathcal{L}\setminus\mathcal{S}} \mid \boldsymbol{w}_{\mathcal{S}})\widetilde{p}(\boldsymbol{w}_{\mathcal{S}}) \prod_{\boldsymbol{s}_i \in \mathcal{S}\setminus\mathcal{L}} dw(\boldsymbol{s}_i) \int \widetilde{p}(\boldsymbol{w}_{\{\boldsymbol{\ell}_0\}} \mid \boldsymbol{w}_{\mathcal{S}}) dw(\boldsymbol{\ell}_0)$$
$$= \int \widetilde{p}(\boldsymbol{w}_{\mathcal{L}\setminus\mathcal{S}} \mid \boldsymbol{w}_{\mathcal{S}})\widetilde{p}(\boldsymbol{w}_{\mathcal{S}}) \prod_{\boldsymbol{s}_i \in \mathcal{S}\setminus\mathcal{L}} dw(\boldsymbol{s}_i)$$
$$= \widetilde{p}(\boldsymbol{w}_{\mathcal{L}}).$$

Appendix B. Properties of Gaussian SpamTrees

Consider the treed graph $\mathcal{G}$ of a SpamTree. In this section, we make no distinction between reference and non-reference nodes, and instead label $\boldsymbol{V}_i = \boldsymbol{A}_i$ for $i = 0, \ldots, M-1$ and $\boldsymbol{V}_M = \boldsymbol{B}$ so that $\boldsymbol{V} = \{\boldsymbol{A}, \boldsymbol{B}\} = \{\boldsymbol{V}_0, \ldots, \boldsymbol{V}_{M-1}, \boldsymbol{V}_M\}$ and the $\boldsymbol{V}_M$ are the leaf nodes. Each $\boldsymbol{w}_i$ is $n_i \times 1$ and corresponds to $\boldsymbol{v}_i \in \boldsymbol{V}_r$ for some $r = 0, \ldots, M$ so that $\mathrm{Pa}[\boldsymbol{v}_i] = \{\boldsymbol{v}_{j_1}, \ldots, \boldsymbol{v}_{j_r}\}$ for some sequence $\{j_1, \ldots, j_r\}$, and $\eta^{-1}(\mathrm{Pa}[\boldsymbol{v}_i]) = \{S_{j_1}, \ldots, S_{j_r}\}$. Denote the h-th parent of $\boldsymbol{v}_i$ as $\mathrm{Pa}[\boldsymbol{v}_i](h)$.

B.1 Building the precision matrix

We can represent each conditional density $N(\boldsymbol{w}_i \mid \boldsymbol{H}_i \boldsymbol{w}_{[i]}, \boldsymbol{R}_i)$ as a linear regression on $\boldsymbol{w}_i$:

$$\boldsymbol{w}_0 = \boldsymbol{\omega}_0 \sim N(\boldsymbol{0}, \boldsymbol{R}_0), \quad \boldsymbol{w}_i = \sum_{\{j : \boldsymbol{v}_j \in \mathrm{Pa}[\boldsymbol{v}_i]\}} \boldsymbol{h}_{ij}\boldsymbol{w}_j + \boldsymbol{\omega}_i \ , \ \ i = 1, 2, \ldots, M, \tag{19}$$

where each $\boldsymbol{h}_{ij}$ is an $n_i \times n_j$ coefficient matrix representing the regression of $\boldsymbol{w}_i$ given $\boldsymbol{w}_{[i]}$, $\boldsymbol{\omega}_i \overset{ind}{\sim} N(\boldsymbol{0}, \boldsymbol{R}_i)$ for $i = 0, 1, \ldots, M$, and each $\boldsymbol{R}_i$ is an $n_i \times n_i$ residual covariance matrix. We set $\boldsymbol{h}_{ii} = \boldsymbol{O}$ and $\boldsymbol{h}_{ij} = \boldsymbol{O}$, where $\boldsymbol{O}$ is the matrix of zeros, whenever $j \notin \{j_1, \ldots, j_r\}$. Using this representation, we have $\boldsymbol{H}_i = [\boldsymbol{h}_{i,j_1}, \boldsymbol{h}_{i,j_2}, \ldots, \boldsymbol{h}_{i,j_r}]$, which is an $n_i \times J_i$ block matrix formed by stacking $\boldsymbol{h}_{i,j_k}$ side by side for $k = 1, \ldots, r$. Since $\mathrm{E}[\boldsymbol{w}_i \,|\, \boldsymbol{w}_{[i]}] = \boldsymbol{H}_i \boldsymbol{w}_{[i]} = \boldsymbol{C}_{i,[i]}\boldsymbol{C}_{[i]}^{-1}\boldsymbol{w}_{[i]}$, we obtain $\boldsymbol{H}_i = \boldsymbol{C}_{i,[i]}\boldsymbol{C}_{[i]}^{-1}$. We also obtain $\boldsymbol{R}_i = \mathrm{var}\{\boldsymbol{w}_i \,|\, \boldsymbol{w}_{[i]}\} = \boldsymbol{C}_{i,i} - \boldsymbol{C}_{i,[i]}\boldsymbol{C}_{[i]}^{-1}\boldsymbol{C}_{[i],i}$, hence all $\boldsymbol{H}_i$'s, $\boldsymbol{h}_{ij}$'s, and $\boldsymbol{R}_i$'s can be computed from the base covariance function.

In order to continue building the precision matrix, define the block matrix $\boldsymbol{\mathcal{H}} = \{\boldsymbol{h}_{ij}\}$. We can write

$$\boldsymbol{h}_{ij} = \begin{cases} \boldsymbol{O} & \text{if } \boldsymbol{v}_j \notin \mathrm{Pa}[\boldsymbol{v}_i] \\ (\boldsymbol{C}_{i,[i]}\boldsymbol{C}_{[i]}^{-1})(\cdot, h) = \boldsymbol{H}_i(\cdot, h) & \text{if } \boldsymbol{v}_j = \boldsymbol{v}_{j_h} \in \mathrm{Pa}[\boldsymbol{v}_i], \end{cases} \tag{20}$$

where $(\cdot, h)$ refers to the h-th block column. More compactly using the indicator function $\boldsymbol{1}\{\cdot\}$ we have $\boldsymbol{h}_{ij} = \boldsymbol{1}\{\exists h : \boldsymbol{v}_j = \mathrm{Pa}[\boldsymbol{v}_i](h)\}(\boldsymbol{C}_{i,[i]}\boldsymbol{C}_{[i]}^{-1})(\cdot, h)$. If we stack *all* the $\boldsymbol{h}_{ik}$ horizontally for $k = 0, \ldots, M_S - 1$, we obtain the $n_i \times n$ matrix $\boldsymbol{\mathcal{H}}(i, \cdot)$, which is i-th block row of $\boldsymbol{\mathcal{H}}$. Intuitively, $\boldsymbol{\mathcal{H}}(i, \cdot)$ is a *sparse* matrix with the coefficients linking the full $\boldsymbol{w}$ to $\boldsymbol{w}_i$, with zero blocks at locations whose corresponding node is $\boldsymbol{v}_j \notin \mathrm{Pa}[\boldsymbol{v}_i]$. The ith block-row of $\boldsymbol{\mathcal{H}}$ is of size $n_i \times n$ but only has r non-empty sub-blocks, with sizes $n_i \times n_j$ for $j \in \{j_1, \ldots, j_r\}$, respectively. Instead, $\boldsymbol{H}_i$ is a *dense* matrix obtained by dropping all the zero-blocks from $\boldsymbol{\mathcal{H}}(i, \cdot)$, and stores the coefficients linking $\boldsymbol{w}_{[i]}$ to $\boldsymbol{w}_i$. The two are linked as $\boldsymbol{H}_i \boldsymbol{w}_{[i]} = \boldsymbol{\mathcal{H}}(i, \cdot)\boldsymbol{w}$.

Since $\boldsymbol{w} = \boldsymbol{\mathcal{H}}\boldsymbol{w} + \boldsymbol{\omega}$, $\widetilde{\boldsymbol{C}} = \text{var}(\boldsymbol{w}) = (\boldsymbol{I} - \boldsymbol{\mathcal{H}})^{-1}\boldsymbol{R}(\boldsymbol{I} - \boldsymbol{\mathcal{H}})^{-\top}$, where $\boldsymbol{R} = \text{b.diag}\{\boldsymbol{R}_i\}$ and $\boldsymbol{I} - \boldsymbol{\mathcal{H}}$ is block lower-triangular with unit diagonal, hence non-singular. We find the precision matrix as $\widetilde{\boldsymbol{C}}^{-1} = (\boldsymbol{I} - \boldsymbol{\mathcal{H}})^\top \boldsymbol{R}^{-1}(\boldsymbol{I} - \boldsymbol{\mathcal{H}})$.

B.2 Properties of $\widetilde{\boldsymbol{C}}^{-1}$

When not ambiguous, we use the notation $\boldsymbol{X}_{ij}$ to denote the (i, j) block of $\boldsymbol{X}$. An exception to this is the (i, j) block of $\widetilde{\boldsymbol{C}}^{-1}$ which we denote as $\widetilde{\boldsymbol{C}}^{-1}(i, j)$. In SpamTrees, $\widetilde{\boldsymbol{C}}^{-1}(i, j)$ is nonzero if $i = j$ or if the corresponding nodes $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ are connected in the moral graph $\mathcal{G}^m$, which is an undirected graph based on $\mathcal{G}$ in which an edge connects all nodes that share a child. This means that either (1) $\boldsymbol{v}_i \in \text{Pa}[\boldsymbol{v}_j]$ or vice-versa, or (2) there exists $\boldsymbol{a}^*$ such that $\{\boldsymbol{v}_i, \boldsymbol{v}_j\} \subset \text{Pa}[\boldsymbol{a}^*]$. In SpamTrees, $\mathcal{G}^m = \mathcal{G}$. In fact, suppose there is a node $\boldsymbol{a}^* \in \boldsymbol{v}_{r^*}$ such that $\boldsymbol{a}^* \in \text{Ch}[\boldsymbol{v}_j] \cap \text{Ch}[\boldsymbol{v}_k]$, where $\boldsymbol{v}_j \in \boldsymbol{v}_{r_j}$ and $\boldsymbol{v}_k \in \boldsymbol{v}_{r_k}$. By definition of $\mathcal{G}$ there exists a sequence $\{i_1, \ldots, i_{r^*}\}$ such that $\text{Pa}[\boldsymbol{a}^*] = \{\boldsymbol{v}_{i_1}, \ldots, \boldsymbol{v}_{i_{r^*}}\} \supset \{\boldsymbol{v}_j, \boldsymbol{v}_k\}$, and furthermore $\text{Pa}[\boldsymbol{v}_{i_h}] = \{\boldsymbol{v}_{i_1}, \ldots, \boldsymbol{v}_{i_{h-1}}\}$ for $h \leq r^*$. This implies that if $j = k$ then $\boldsymbol{v}_j = \boldsymbol{v}_k$, whereas if $j > k$ then $\boldsymbol{v}_k \in \text{Pa}[\boldsymbol{v}_j]$, meaning that no additional edge is necessary to build $\mathcal{G}^m$.

B.2.1 Explicit derivation of $\widetilde{\boldsymbol{C}}^{-1}(i, j)$

Denote $\boldsymbol{R}^{-1} = \boldsymbol{R}^{-\frac{1}{2}}\boldsymbol{R}^{-\frac{\top}{2}}$, $\boldsymbol{U} = (\boldsymbol{I} - \boldsymbol{\mathcal{H}})^\top \boldsymbol{R}^{-\frac{1}{2}}$, and define the "common descendants" as $\text{cd}(\boldsymbol{v}_i, \boldsymbol{v}_j) = (\{\boldsymbol{v}_i\} \cup \text{Ch}[\boldsymbol{v}_i]) \cap (\{\boldsymbol{v}_j\} \cup \text{Ch}[\boldsymbol{v}_j])$. Then consider $\boldsymbol{a}_i \in \boldsymbol{A}, \boldsymbol{v}_j \in \boldsymbol{V}$ such that $\boldsymbol{a}_i \in \text{Pa}[\boldsymbol{v}_j]$ and denote as $\boldsymbol{H}_{i\to j}$ the matrix obtained by subsetting $\boldsymbol{H}_j$ to columns corresponding to $\boldsymbol{a}_i$ and note that $\boldsymbol{H}_{i\to j} = \boldsymbol{\mathcal{H}}_{ji}$. The (i, j) block of $\boldsymbol{U}$ is then

$$\boldsymbol{U}_{ij} = \begin{cases} \boldsymbol{O}_{n_i \times n_j} & \text{if } \boldsymbol{v}_j \notin \text{Ch}[\boldsymbol{v}_i] \\ \boldsymbol{I}_{n_i \times n_i} & \text{if } i = j \\ (\boldsymbol{I}_{ji} - \boldsymbol{H}_{i\to j})^\top \boldsymbol{R}_j^{-\frac{1}{2}} & \text{if } \boldsymbol{v}_j \in \text{Ch}[\boldsymbol{v}_i] \\ \qquad = (\boldsymbol{I}_{ji} - \boldsymbol{\mathcal{H}}_{ji})^\top \boldsymbol{R}_j^{-\frac{1}{2}} & \end{cases}$$

Then $\widetilde{\boldsymbol{C}}^{-1}(i, j) = \sum_k \boldsymbol{U}_{ik}\boldsymbol{U}_{jk}^\top$ and, as in (9), each block of the precision matrix is:

$$\begin{aligned} \widetilde{\boldsymbol{C}}^{-1}(i, j) &= \sum_{\boldsymbol{v}_k \in \text{cd}(\boldsymbol{v}_i, \boldsymbol{v}_j)} (\boldsymbol{I}_{ki} - \boldsymbol{H}_{i\to k})^\top \boldsymbol{R}_k^{-1}(\boldsymbol{I}_{kj} - \boldsymbol{H}_{j\to k}) \\ &= \sum_{\boldsymbol{v}_k \in \text{cd}(\boldsymbol{v}_i, \boldsymbol{v}_j)} (\boldsymbol{I}_{ki} - \boldsymbol{\mathcal{H}}_{ki})^\top \boldsymbol{R}_k^{-1}(\boldsymbol{I}_{kj} - \boldsymbol{\mathcal{H}}_{kj}) \end{aligned} \tag{21}$$

where $\text{cd}(\boldsymbol{v}_i, \boldsymbol{v}_j) = \emptyset$ implies $\widetilde{\boldsymbol{C}}^{-1}(i, j) = \boldsymbol{O}$ and $\boldsymbol{I}_{ij}$ a zero matrix unless $i = j$ as it is the (i, j) block of an identity matrix of dimension $n \times n$.

B.2.2 Computation of large matrix inverses

One important aspect in building $\widetilde{\boldsymbol{C}}^{-1}$ is that it requires the computation of the inverse $\boldsymbol{C}_{[i]}^{-1}$ of dimension $J_i \times J_i$ for all nodes with parents, i.e. at $r > 0$. Unlike models which achieve scalable computations by limiting the size of the parent set (e.g. NNGPs and their

blocked variant, or tessellated MGPs), this inverse is increasingly costlier when $\delta > 1$ for nodes at a higher-level of the tree as those nodes have more parents and hence larger sets of parent locations (the same conclusion holds for non-reference nodes). However, the treed structure in $\mathcal{G}$ allows one to avoid computing the inverse in $O(J_i^3)$. In fact, suppose we have a symmetric, positive-definite block-matrix $\boldsymbol{A}$ and we wish to compute its inverse. We write

$$\boldsymbol{A} = \begin{bmatrix} C & B \\ B^\top & D \end{bmatrix} \quad \boldsymbol{A}^{-1} = \begin{bmatrix} C^{-1} + C^{-1}BS^{-1}B^\top C^{-1} & -C^{-1}BS^{-1} \\ -S^{-1}B^\top C^{-1} & S^{-1} \end{bmatrix},$$

where $S = C - BD^{-1}B^\top$ is the Schur complement of D in $\boldsymbol{A}$. If C^{-1} was available, the only necessary inversion is that of S. In SpamTrees with $\delta > 1$, suppose $\boldsymbol{v}_i, \boldsymbol{v}_j$ are two nodes such that $\mathrm{Pa}[\boldsymbol{v}_j] = \{\boldsymbol{v}_i\} \cup \mathrm{Pa}[\boldsymbol{v}_i]$ – this arises for nodes $\boldsymbol{v}_j \in \boldsymbol{V}_r, r \geq M_\delta$. Regardless of whether $\boldsymbol{v}_j$ is a reference node or not, $\eta^{-1}(\mathrm{Pa}[\boldsymbol{v}_j]) = \{\mathcal{S}_i, \mathcal{S}_{[i]}\}$ and

$$\boldsymbol{C}_{[j]} = \begin{bmatrix} \boldsymbol{C}_{[i]} & \boldsymbol{C}_{[i],i} \\ \boldsymbol{C}_{i,[i]} & \boldsymbol{C}_i \end{bmatrix}, \quad \boldsymbol{C}_{[j]}^{-1} = \begin{bmatrix} \boldsymbol{C}_{[i]}^{-1} + \boldsymbol{C}_{[i]}^{-1}\boldsymbol{C}_{[i],i}S^{-1}\boldsymbol{C}_{i,[i]}\boldsymbol{C}_{[i]}^{-1} & -\boldsymbol{C}_{[i]}^{-1}\boldsymbol{C}_{[i],i}S^{-1} \\ -S^{-1}\boldsymbol{C}_{i,[i]}\boldsymbol{C}_{[i]}^{-1} & S^{-1} \end{bmatrix},$$

where the Schur complement of $\boldsymbol{C}_i$ is $S = \boldsymbol{C}_i - \boldsymbol{C}_{i,[i]}\boldsymbol{C}_{[i]}^{-1}\boldsymbol{C}_{[i],i} = \boldsymbol{R}_i$. Noting that $\boldsymbol{H}_i = \boldsymbol{C}_{i,[i]}\boldsymbol{C}_{[i]}^{-1}$ we write

$$\boldsymbol{C}_{[j]}^{-1} = \begin{bmatrix} \boldsymbol{C}_{[i]}^{-1} + \boldsymbol{H}_i^\top \boldsymbol{R}_i^{-1}\boldsymbol{H}_i & -\boldsymbol{H}_i^\top \boldsymbol{R}_i^{-1} \\ -\boldsymbol{R}_i^{-1}\boldsymbol{H}_i & \boldsymbol{R}_i^{-1} \end{bmatrix}. \tag{22}$$

B.2.3 Computing $(\widetilde{\boldsymbol{C}}^{-1} + \boldsymbol{\Sigma})^{-1}$ and its determinant without sparse Cholesky

Bayesian estimation of regression models requiring the computation of $(\boldsymbol{Z}\widetilde{\boldsymbol{C}}\boldsymbol{Z}^\top + \boldsymbol{D})^{-1}$ and its determinant use the Sherman-Morrison-Woodbury matrix identity to find $(\boldsymbol{Z}\widetilde{\boldsymbol{C}}\boldsymbol{Z}^\top + \boldsymbol{D})^{-1} = \boldsymbol{D}^{-1} - \boldsymbol{D}^{-1}\boldsymbol{Z}(\widetilde{\boldsymbol{C}}^{-1} + \boldsymbol{\Sigma})^{-1}\boldsymbol{Z}^\top\boldsymbol{D}^{-1}$, where $\boldsymbol{\Sigma} = \boldsymbol{Z}^\top\boldsymbol{D}^{-1}\boldsymbol{Z}$. A sparse Cholesky factorization of $\widetilde{\boldsymbol{C}}^{-1} + \boldsymbol{\Sigma}$ can be used as typically $\boldsymbol{\Sigma}$ is diagonal or block-diagonal, thus maintaining the sparsity structure of $\widetilde{\boldsymbol{C}}^{-1}$. Sparse Cholesky libraries (e.g. `Cholmod`, Chen et al., 2008), which are embedded in software or high-level languages such as MATLAB™ or the `Matrix` package for `R`, scale to large sparse matrices but are either too flexible or too restrictive in our use cases: (1) we know $\mathcal{G}$ and its properties in advance; (2) SpamTrees take advantage of block structures and grouped data. In fact, sparse matrix libraries typically are agnostic of $\mathcal{G}$ and heuristically attempt to infer a sparse $\mathcal{G}$ given its moralized counterpart. While this operation is typically performed once, *a priori* knowledge of $\mathcal{G}$ implies that reliance on such libraries is in principle unnecessary.

We thus take advantage of the known structure in $\mathcal{G}$ to derive direct algorithms for computing $(\widetilde{\boldsymbol{C}}^{-1} + \boldsymbol{\Sigma})^{-1}$ and its determinant. In the discussion below we consider $\delta = M$, noting here that choosing $\delta = 1$ simplifies the treatment as $\mathrm{cd}(\boldsymbol{v}_i, \boldsymbol{v}_j) = \{\boldsymbol{v}_i\}$ if $\boldsymbol{v}_i = \boldsymbol{v}_j$, and it is empty otherwise. We now show how (21) leads to Algorithm 4 for the decomposition of *any* precision matrix $\boldsymbol{\Lambda}$ which conforms to $\mathcal{G}$ – i.e. it has the same block-sparse structure as a precision matrix built as in Section B.1. Suppose from $\boldsymbol{\Lambda}$ we seek a block lower-triangular matrix $\boldsymbol{L}$ and a block diagonal $\boldsymbol{D}$ such that

$$\boldsymbol{\Lambda}_{ij} = \sum_{\boldsymbol{v}_k \in \mathrm{cd}(\boldsymbol{v}_i, \boldsymbol{v}_j)} (\boldsymbol{I}_{ki} - \boldsymbol{L}_{ki})^\top \boldsymbol{D}_k (\boldsymbol{I}_{kj} - \boldsymbol{L}_{kj}).$$

Start with $\boldsymbol{v}_i, \boldsymbol{v}_j$ taken from the leaf nodes, i.e. $\boldsymbol{v}_i, \boldsymbol{v}_j \in \boldsymbol{V}_M$. Then $\mathrm{cd}(\boldsymbol{v}_i, \boldsymbol{v}_j) = \emptyset$ and we set $\boldsymbol{L}_{ij} = \boldsymbol{L}(j,i)^\top = \boldsymbol{O} = \boldsymbol{\Lambda}_{ij}$. If $i = j$ then $\mathrm{cd}(\boldsymbol{v}_i, \boldsymbol{v}_i) = \{\boldsymbol{v}_i\}$ and

$$\sum_{\boldsymbol{v}_k \in \mathrm{cd}(\boldsymbol{v}_i, \boldsymbol{v}_i)} (\boldsymbol{I}_{ki} - \boldsymbol{L}_{ki})^\top \boldsymbol{D}_k (\boldsymbol{I}_{ki} - \boldsymbol{L}_{ki}) = (\boldsymbol{I}_{ii} - \boldsymbol{L}_{ii})^\top \boldsymbol{D}_i (\boldsymbol{I}_{ii} - \boldsymbol{L}(i,i))$$
$$= \boldsymbol{D}_i - \boldsymbol{D}_i \boldsymbol{L}_{ii} - \boldsymbol{L}_{ii}^\top \boldsymbol{D}_i + \boldsymbol{L}_{ii}^\top \boldsymbol{D}_i \boldsymbol{L}_{ii};$$

we then set $\boldsymbol{L}_{ii} = \boldsymbol{O}$ and get the i-th block of $\boldsymbol{D}_i$ simply setting $\boldsymbol{D}_i = \boldsymbol{\Lambda}_{ii}$. Proceeding downwards along $\mathcal{G}$, if $\boldsymbol{v}_j \in \boldsymbol{V}_{M-1} \cap \mathrm{Pa}[\boldsymbol{v}_i]$ we have $\boldsymbol{\Lambda}_{ij} = \boldsymbol{D}_i(\boldsymbol{I}_{ij} - \boldsymbol{L}_{ij}) = -\boldsymbol{D}_i \boldsymbol{L}_{ij}$ and thus set $\boldsymbol{L}_{ij} = -\boldsymbol{D}_i^{-1} \boldsymbol{\Lambda}_{ij}$. We then note that $\mathrm{cd}(\boldsymbol{v}_j, \boldsymbol{v}_j) = \{\boldsymbol{v}_j, \boldsymbol{v}_i\}$ and obtain $\boldsymbol{\Lambda}_{jj} = \boldsymbol{D}_j + \boldsymbol{L}_{ij}^\top \boldsymbol{D}_i \boldsymbol{L}_{ij}$ where $\boldsymbol{L}_{ij}$ and $\boldsymbol{D}_i$ have been fixed at the previous step; this results in $\boldsymbol{D}_j = \boldsymbol{\Lambda}_{jj} - \boldsymbol{L}_{ij}^\top \boldsymbol{D}_i \boldsymbol{L}_{ij}$.

Then, the s-th (of M) step takes $\boldsymbol{v}_j \in \boldsymbol{V}_{M-s} \cap \mathrm{Pa}[\boldsymbol{v}_i]$ and $\boldsymbol{v}_i \in \boldsymbol{V}_{M-s+1}$, implying $\mathrm{cd}(\boldsymbol{v}_i, \boldsymbol{v}_j) = \{\boldsymbol{v}_i\} \cup \mathrm{Ch}[\boldsymbol{v}_i]$. Noting that $\boldsymbol{F}^* = \sum_{\boldsymbol{v}_k \in \mathrm{Ch}[\boldsymbol{v}_i]} (\boldsymbol{I}_{ki} - \boldsymbol{L}_{ki})^\top \boldsymbol{D}_k (\boldsymbol{I}_{kj} - \boldsymbol{L}_{kj})$ has been fixed at previous steps since each $\boldsymbol{v}_k$ is at level $M - s + 2$, we split the sum in (21) and get

$$\boldsymbol{\Lambda}_{ij} - \boldsymbol{F}^* = \boldsymbol{D}_i(\boldsymbol{I}_{ij} - \boldsymbol{L}_{ij}) = -\boldsymbol{D}_i \boldsymbol{L}_{ij},$$

where $\boldsymbol{D}_i$ has been fixed at step $s-1$, obtaining $\boldsymbol{L}_{ij} = -\boldsymbol{D}_i^{-1}(\boldsymbol{\Lambda}_{ij} - \boldsymbol{F}^*)$; $\boldsymbol{D}_j$ can be found using the same logic. Proceeding until $M - s = 0$ from the leaves of $\mathcal{G}$ to the root, we ultimately fill each non-empty block in $\boldsymbol{L}$ and $\boldsymbol{D}$ resulting in $\boldsymbol{\Lambda} = (\boldsymbol{I} - \boldsymbol{L})^\top \boldsymbol{D} (\boldsymbol{I} - \boldsymbol{L})$. Algorithm 4 unifies these steps to obtain the block decomposition of any sparse precision matrix $\boldsymbol{\Lambda}$ conforming to $\mathcal{G}$ resulting in $\boldsymbol{\Lambda} = (\boldsymbol{I} - \boldsymbol{L})^\top \boldsymbol{D} (\boldsymbol{I} - \boldsymbol{L})$, where $\boldsymbol{L}$ is block lower triangular and $\boldsymbol{D}$ is block diagonal. This is akin to a block-LDL decomposition of $\boldsymbol{\Lambda}$ indexed on nodes of $\mathcal{G}$. Algorithm 5 complements this decomposition by providing a $\mathcal{G}$-specific block version of forward substitution for computing $(\boldsymbol{I} - \boldsymbol{L})^{-1}$ with $\boldsymbol{L}$ as above.

In practice, a block matrix with K^2 blocks can be represented as a K^2 array with rows and columns indexed by nodes in $\mathcal{G}$ and matrix elements which may be zero-dimensional whenever corresponding to blocks of zeros. The specification of all algorithms in block notation allows us to never deal with large (sparse) matrices in practice but only with small block matrices indexed by nodes in $\mathcal{G}$, bypassing the need for external sparse matrix libraries. Specifically we use the above algorithms to compute $\boldsymbol{\Lambda}^{-1} = (\widetilde{\boldsymbol{C}}^{-1} + \boldsymbol{\Sigma})^{-1}$ and its determinant: $\boldsymbol{\Lambda}^{-1} = (\boldsymbol{I} - \boldsymbol{L})^{-1} \boldsymbol{D}^{-1} (\boldsymbol{I} - \boldsymbol{L})^{-\top}$ and $|\boldsymbol{\Lambda}^{-1}| = \prod_{i=1}^{M_S} 1/|\widetilde{\boldsymbol{D}}_i|$. We have not distinguished non-reference and reference nodes in this discussion. In cases in which the non-reference set is large, we note that the conditional independence of all non-reference locations, given their parents, results in $\widetilde{\boldsymbol{C}}^{-1}(i,i)$ being diagonal for all $\boldsymbol{\ell} \in \mathcal{U}$ (i.e. $\eta(\boldsymbol{\ell}) = \boldsymbol{v}_i \in \boldsymbol{B}$). This portion of the precision matrix can just be stored as a column vector.

B.2.4 Sparsity of $\widetilde{\boldsymbol{C}}^{-1}$

We calculate the sparsity in the precision matrix; considering an enumeration of nodes by level in $\mathcal{G}$, denote $n_{ij} = |\eta^{-1}(\boldsymbol{v}_{ij})|$, $m_j = |\boldsymbol{V}_j|$, and $J_{ij} = |\eta^{-1}(\mathrm{Pa}[\boldsymbol{v}_{ij}])|$, and noting that by symmetry $(\widetilde{\boldsymbol{C}}^{-1}(i,j))^\top = \widetilde{\boldsymbol{C}}^{-1}(j,i)$, the number of nonzero elements of $\widetilde{\boldsymbol{C}}^{-1}$ is

$$\mathrm{nnz}(\widetilde{\boldsymbol{C}}^{-1}) = \sum_{j=0}^{M} \sum_{i=1}^{m_j} \left(2 n_{ij} J_{ij} + n_{ij}^2 \mathbf{1}\{j < M\} + n_{ij} \mathbf{1}\{j = M\} \right),$$

Input: $\boldsymbol{\Lambda}$ $n \times n$ precision matrix conforming to $\mathcal{G}$
Initialize $\boldsymbol{L} = \boldsymbol{O}_{n\times n}, \boldsymbol{D} = \boldsymbol{O}_{n\times n}$;
for $r \in \{M, \ldots, 0\}$ **do** `// top down from last level`
  for $j : \{\boldsymbol{v}_j \in \boldsymbol{V}_r\}$ **do** `// [parallel for]`
    $\boldsymbol{D}_{jj} = \boldsymbol{\Lambda}_{jj}$;
    for $p : \{\boldsymbol{v}_p \in Pa[\boldsymbol{v}_j]\}$ **do**
      $\boldsymbol{L}_{jp} = -\boldsymbol{D}_{jj}^{-1}\boldsymbol{\Lambda}_{jp}$;
      for $g : \{\boldsymbol{v}_g \in Pa[\boldsymbol{v}_j]\}$ **do**
        $\boldsymbol{\Lambda}_{pg} = \boldsymbol{\Lambda}_{pg} - \boldsymbol{\Lambda}_{jp}^{\top}\boldsymbol{L}_{jg}$;
        $\boldsymbol{\Lambda}_{gp} = \boldsymbol{\Lambda}_{pg}^{\top}$;
Result: Block-lower-triangular $\boldsymbol{L}$ with $\boldsymbol{L}_{ij} \neq \boldsymbol{O}$ if $\boldsymbol{v}_i \in \mathrm{Pa}[\boldsymbol{v}_j]$, and block-diagonal $\boldsymbol{D}$ such that $(\boldsymbol{I} - \boldsymbol{L})^{\top}\boldsymbol{D}(\boldsymbol{I} - \boldsymbol{L}) = \boldsymbol{\Lambda}$.

Algorithm 4: Precision matrix decomposition given treed graph $\mathcal{G}$ with M levels.

Input: $\boldsymbol{\Gamma} = \boldsymbol{I} - \boldsymbol{L}$ where $\boldsymbol{L}$ is as in Algorithm 4.
Initialize $\boldsymbol{\Delta}_{ij} = \boldsymbol{O}_{n_i,n_j}$ for all i, j such that $\boldsymbol{v}_j \in \mathrm{Pa}[\boldsymbol{v}_i]$;
for $r \in \{0, \ldots, M\}$ **do** `// bottom up from root of` $\mathcal{G}$
  for $j : \{\boldsymbol{v}_j \in \boldsymbol{V}_r\}$ **do** `// [parallel for]`
    for $p : \{\boldsymbol{v}_p \in \widetilde{\mathcal{P}}_{0\to[j]}\}$ **do**
      Set $\mathrm{chain}(\boldsymbol{v}_p, \boldsymbol{v}_j) = \{\boldsymbol{v}_p\} \cup \{\widetilde{\mathcal{P}}_{0\to[j]} \cap \widetilde{\mathcal{P}}_{0\to[p]}\}$;
      for $g : \{\boldsymbol{v}_g \in \mathrm{chain}(\boldsymbol{v}_p, \boldsymbol{v}_j)\}$ **do**
        $\boldsymbol{\Delta}_{jp} = \boldsymbol{\Delta}_{jp} - \boldsymbol{\Gamma}_{jg}\boldsymbol{\Delta}_{gp}$
Result: $\boldsymbol{\Delta} = \boldsymbol{\Gamma}^{-1}$.

Algorithm 5: Calculating the inverse of $\boldsymbol{I} - \boldsymbol{L}$ with $\boldsymbol{L}$ output from Algorithm 4.

where $n_{ij}\mathbf{1}\{j = M\}$ refers to the diagonal elements of the precision matrix at non-reference locations.

B.3 Properties of SpamTrees with $\delta = M$

We outline recursive properties of $\widetilde{\boldsymbol{C}}$ induced by $\mathcal{G}$ when $\delta = M$. In the case $1 < \delta < M$, these properties hold for nodes at or above level M_δ, using $\boldsymbol{A}_{M_\delta}$ as root. We focus on paths in $\mathcal{G}$. These can be represented as sequences of nodes $\{\boldsymbol{v}_{i_1}, \ldots, \boldsymbol{v}_{i_r}\}$ such that $\{\boldsymbol{v}_{i_j}, \ldots, \boldsymbol{v}_{i_k}\} \subset \mathrm{Pa}[\boldsymbol{v}_{i_{k+1}}]$ for $1 < j < k < r$. Take two successive elements of such a sequence, i.e. $\boldsymbol{v}_i, \boldsymbol{v}_j$ such that $\boldsymbol{v}_i \to \boldsymbol{v}_j$ in $\mathcal{G}$. Consider $\mathrm{E}[\boldsymbol{w}_j \,|\, \boldsymbol{w}_{[j]}] = \boldsymbol{H}_j\boldsymbol{w}_{[j]} = \boldsymbol{C}_{j,[j]}\boldsymbol{C}_{[j]}^{-1}\boldsymbol{w}_{[j]}$ and $\boldsymbol{R}_j = \mathrm{var}\{\boldsymbol{w}_j \,|\, \boldsymbol{w}_{[j]}\} = \boldsymbol{C}_{j,j} - \boldsymbol{C}_{j,[j]}\boldsymbol{C}_{[j]}^{-1}\boldsymbol{C}_{[j],j}$. By (22) we can write

$$
\begin{aligned}
\boldsymbol{H}_j\boldsymbol{w}_{[j]} &= \begin{bmatrix}\boldsymbol{C}_{j,[i]} & \boldsymbol{C}_{j,i}\end{bmatrix}\begin{bmatrix}\boldsymbol{C}_{[i]}^{-1} + \boldsymbol{H}_i^\top\boldsymbol{R}_i^{-1}\boldsymbol{H}_i & -\boldsymbol{H}_i^\top\boldsymbol{R}_i^{-1} \\ -\boldsymbol{R}_i^{-1}\boldsymbol{H}_i & \boldsymbol{R}_i^{-1}\end{bmatrix}\begin{bmatrix}\boldsymbol{w}_{[i]} \\ \boldsymbol{w}_i\end{bmatrix} \\
&= \begin{bmatrix}\boldsymbol{C}_{j,[i]} & \boldsymbol{C}_{j,i} - \boldsymbol{C}_{j,[i]}\boldsymbol{C}_{[i]}^{-1}\boldsymbol{C}_{[i],i}\end{bmatrix}\begin{bmatrix}\boldsymbol{C}_{[i]}^{-1} & \boldsymbol{O} \\ \boldsymbol{O} & (\boldsymbol{C}_{i,i} - \boldsymbol{C}_{i,[i]}\boldsymbol{C}_{[i]}^{-1}\boldsymbol{C}_{[i],i})^{-1}\end{bmatrix}\begin{bmatrix}\boldsymbol{w}_{[i]} \\ \boldsymbol{w}_i - \boldsymbol{C}_{i,[i]}\boldsymbol{C}_{[i]}^{-1}\boldsymbol{w}_{[i]}\end{bmatrix} \\
&= \begin{bmatrix}\boldsymbol{C}_{j,[i]}\boldsymbol{C}_{[i]}^{-1} & (\boldsymbol{C}_{j,i} - \boldsymbol{C}_{j,[i]}\boldsymbol{C}_{[i]}^{-1}\boldsymbol{C}_{[i],i})(\boldsymbol{C}_{i,i} - \boldsymbol{C}_{i,[i]}\boldsymbol{C}_{[i]}^{-1}\boldsymbol{C}_{[i],i})^{-1}\end{bmatrix}\begin{bmatrix}\boldsymbol{w}_{[i]} \\ \boldsymbol{w}_i - \boldsymbol{C}_{i,[i]}\boldsymbol{C}_{[i]}^{-1}\boldsymbol{w}_{[i]}\end{bmatrix}.
\end{aligned}
$$

Now define the covariance function $\boldsymbol{K}_i(\boldsymbol{\ell}, \boldsymbol{\ell}') = \boldsymbol{C}_{\boldsymbol{\ell},\boldsymbol{\ell}'} - \boldsymbol{C}_{\boldsymbol{\ell},[i]}\boldsymbol{C}_{[i]}^{-1}\boldsymbol{C}_{[i],\boldsymbol{\ell}'}$; recalling that the reference set is $\mathcal{S} = \cup_{i=0}^{M-1}\cup_{j=1}^{m_j} S_j$ we use a shorthand notation for these subsets: $\boldsymbol{K}_i(S_h, S_k) = \boldsymbol{K}_i(h,k)$. Also denote $\boldsymbol{e}_i = \boldsymbol{w}_i - \boldsymbol{C}_{i,[i]}\boldsymbol{C}_{[i]}^{-1}\boldsymbol{w}_{[i]}$ for all i. The above expression becomes

$$
\begin{aligned}
\boldsymbol{H}_j\boldsymbol{w}_{[j]} &= \begin{bmatrix}\boldsymbol{C}_{j,[i]}\boldsymbol{C}_{[i]}^{-1} & \boldsymbol{K}_i(j,i)\boldsymbol{K}_i^{-1}(i,i)\end{bmatrix}\begin{bmatrix}\boldsymbol{w}_{[i]} \\ \boldsymbol{e}_i\end{bmatrix} \\
&= \boldsymbol{H}_i\boldsymbol{w}_{[i]} + \boldsymbol{K}_i(j,i)\boldsymbol{K}_i^{-1}(i,i)\boldsymbol{e}_i;
\end{aligned}
$$

we can use this recursively on $\{\boldsymbol{v}_{i_0}, \boldsymbol{v}_{i_1}, \ldots, \boldsymbol{v}_{i_r}\}$ where $\boldsymbol{v}_{i_0} \in \boldsymbol{A}_0$ and $\boldsymbol{v}_{i_r} = \boldsymbol{v}_j$ and get

$$
\begin{aligned}
\boldsymbol{H}_j\boldsymbol{w}_{[j]} &= \sum_{s=i_1}^{i_{r-1}} \boldsymbol{K}_s(j,s)\boldsymbol{K}_s^{-1}(s,s)\boldsymbol{e}_s \\
\mathrm{E}[\boldsymbol{w}_j \,|\, \boldsymbol{w}_{[j]}] &= \sum_{s=i_1}^{i_{r-1}} \mathrm{E}_{\boldsymbol{e}_s}[\boldsymbol{w}_j \,|\, \boldsymbol{e}_s],
\end{aligned}
$$

where the expectations on the r.h.s. are taken with respect to the distributions of $\boldsymbol{e}_s$ which are Gaussian with mean zero and $\mathrm{var}\{\boldsymbol{e}_h\} = \boldsymbol{K}_h(h,h)$ – this is a compact expression of the conditionals governing the process as prescribed by $\mathcal{G}$. We can also write the above as $\mathrm{E}(\boldsymbol{w}_j \,|\, \boldsymbol{w}_{[j]}) = \sum_{s=i_0}^{i_r} \boldsymbol{K}_s(j,s)\boldsymbol{K}_s^{-1}(s,s)(\boldsymbol{w}_s - \mathrm{E}[\boldsymbol{w}_s \,|\, \boldsymbol{w}_{[s]}])$; using $\mathrm{E}[\boldsymbol{e}_h \,|\, \boldsymbol{w}_h, \boldsymbol{w}_{[h]}] = 0$, for $h < k$ we find

$$
\begin{aligned}
\mathrm{cov}\{\boldsymbol{e}_h, \boldsymbol{e}_k\} &= \mathrm{E}[\mathrm{cov}\{\boldsymbol{e}_h, \boldsymbol{e}_k \,|\, \boldsymbol{w}_h, \boldsymbol{w}_{[h]}\}] + \mathrm{cov}\{\mathrm{E}[\boldsymbol{e}_h \,|\, \boldsymbol{w}_h, \boldsymbol{w}_{[h]}], \mathrm{E}[\boldsymbol{e}_k \,|\, \boldsymbol{w}_h, \boldsymbol{w}_{[h]}]\} \\
&= \mathrm{cov}\{\mathrm{E}[\boldsymbol{e}_h \,|\, \boldsymbol{w}_h, \boldsymbol{w}_{[h]}], \mathrm{E}[\boldsymbol{e}_k \,|\, \boldsymbol{w}_h, \boldsymbol{w}_{[h]}]\} = 0.
\end{aligned}
$$

The above results also imply $\boldsymbol{C}_{j,[j]}\boldsymbol{C}_{[j]}^{-1}\boldsymbol{C}_{[j],j} = \sum_{s=i_0}^{i_r} \boldsymbol{K}_s(j,s)\boldsymbol{K}_s^{-1}(s,s)\boldsymbol{K}_s(s,j)$ and suggest an additive representation via orthogonal basis functions:

$$\boldsymbol{w}_j = \sum_{s=i_0}^{i_{r-1}} \boldsymbol{K}_s(j,s)\boldsymbol{K}_s^{-1}(s,s)\boldsymbol{e}_s + \boldsymbol{e}_j \tag{23}$$

Finally, considering the same sequence of nodes, recursively introduce the covariance functions $\boldsymbol{F}_0(r,s) = \boldsymbol{C}_{r,s}$ and for $j > 1$, $\boldsymbol{F}_j(r,s) = \boldsymbol{F}_{j-1}(r,s) - \boldsymbol{F}_{j\text{-}1}(r,j\text{-}1)\boldsymbol{F}_{j\text{-}1}^{-1}(j\text{-}1,j\text{-}1)\boldsymbol{F}_{j\text{-}1}(j\text{-}1,s)$. We get

$$\begin{aligned}
\boldsymbol{F}_{j+1}(r,s) &= \boldsymbol{F}_j(r,s) - \boldsymbol{F}_j(r,j)\boldsymbol{F}_j^{-1}(j,j)\boldsymbol{F}_j(j,s) \\
\text{using (22)} \quad &= \boldsymbol{F}_{j-1}(r,s) - \boldsymbol{F}_{j-1}(r,[j\text{-}1{:}j])\boldsymbol{F}_{j-1}^{-1}([j\text{-}1{:}j],[j\text{-}1{:}j])\boldsymbol{F}_{j-1}^{-1}([j\text{-}1{:}j],s) \\
&= \boldsymbol{C}(r,s) - \boldsymbol{C}(r,[0{:}j])\boldsymbol{C}^{-1}([0{:}j],[0{:}j])\boldsymbol{C}([0{:}j],s) \\
&= \boldsymbol{C}(r,s) - \boldsymbol{C}_{r,[j+1]}\boldsymbol{C}_{[j+1]}^{-1}\boldsymbol{C}_{[j+1],s}
\end{aligned}$$

which can be iterated forward and results in an additional recursive way to compute covariances in SpamTrees. Notice that while $\boldsymbol{K}_j$ is formulated using the inverse of $J_j \times J_j$ matrix $\boldsymbol{C}_{[j]}$, the $\boldsymbol{F}_j$'s require inversion of smaller $n_j \times n_j$ matrices $\boldsymbol{F}_{j-1}(j\text{-}1,j\text{-}1)$.

B.4 Properties of $\widetilde{C}$

B.4.1 $\delta = 1$

Choosing depth $\delta = 1$ results in each node having exactly 1 parent. In this case the path $\mathcal{P}_{k\to j} = \{\boldsymbol{v}_{i_1},\ldots,\boldsymbol{v}_{i_r}\}$ from $\boldsymbol{v}_k$ to $\boldsymbol{v}_j$, where $\boldsymbol{v}_{i_1} = \boldsymbol{v}_k$, $\boldsymbol{v}_{i_r} = \boldsymbol{v}_j$ and $\{\boldsymbol{v}_{i_h}\} = \mathrm{Pa}[\boldsymbol{v}_{i_{h+1}}]$, is unique, and there is thus no distinction between shortest and longest paths: $\mathcal{P}_{k\to j} = \bar{\mathcal{P}}_{k\to j} = \widetilde{\mathcal{P}}_{k\to j}$. Then denote $\dddot{\boldsymbol{H}}_{k\to j} = \boldsymbol{H}_{i_r}\cdot\boldsymbol{H}_{i_{r-1}}\cdots\boldsymbol{H}_{i_1}$. Let $\boldsymbol{v}_z$ be the *concestor* between $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ i.e. $\boldsymbol{v}_z = \mathrm{con}(\boldsymbol{v}_i,\boldsymbol{v}_j) = \arg\max_{\boldsymbol{v}_k\in\boldsymbol{V}}\{k : \mathcal{P}_{k\to i}\cap\mathcal{P}_{k\to j}\neq\emptyset\}$ and the associated paths $\mathcal{P}_{z\to i} = \{\boldsymbol{v}_{i_1},\ldots,\boldsymbol{v}_{i_{r_i}}\}$ and $\mathcal{P}_{z\to j} = \{\boldsymbol{v}_{j_1},\ldots,\boldsymbol{v}_{j_{r_j}}\}$ where $\boldsymbol{v}_{i_1} = \boldsymbol{v}_{j_1} = \boldsymbol{v}_z$, $\boldsymbol{v}_{i_{r_i}} = \boldsymbol{v}_i$ and $\boldsymbol{v}_{j_{r_j}} = \boldsymbol{v}_j$. Then we can write $\boldsymbol{w}_i = \boldsymbol{w}_{i_{r_i}} = \boldsymbol{H}_{i_{r_i}}\boldsymbol{w}_{i_{r_i-1}} + \boldsymbol{\nu}_{i_{r_i}}$ where $\boldsymbol{\nu}_{i_{r_i}} \sim N(\boldsymbol{0},\boldsymbol{R}_{i_{r_i}})$ and proceed expanding $\boldsymbol{w}_{i_{r_i-1}}$ to get $\boldsymbol{w}_{i_{r_i}} = \boldsymbol{H}_{i_{r_i}}(\boldsymbol{H}_{i_{r_i-1}}\boldsymbol{w}_{i_{r_i-2}} + \boldsymbol{\nu}_{i_{r_i-1}}) + \boldsymbol{\nu}_{i_{r_i}} = \boldsymbol{H}_{i_{r_i}}\boldsymbol{H}_{i_{r_i-1}}\boldsymbol{w}_{i_{r_i-2}} + (\boldsymbol{H}_{i_{r_i}}\boldsymbol{\nu}_{i_{r_i-1}} + \boldsymbol{\nu}_{i_{r_i}})$; continuing downwardly along the tree we eventually find $\boldsymbol{w}_i = \boldsymbol{H}_{i_{r_i}}\cdots\boldsymbol{H}_{i_1}\boldsymbol{w}_{i_1} + \widetilde{\boldsymbol{v}}_i = \dddot{\boldsymbol{H}}_{z\to i}\boldsymbol{w}_z + \widetilde{\boldsymbol{v}}_i$ where $\widetilde{\boldsymbol{v}}_i$ is independent of $\boldsymbol{w}_z$. After proceeding analogously with $\boldsymbol{w}_j$, take $\boldsymbol{\ell}_i,\boldsymbol{\ell}_j$ such that $\eta(\boldsymbol{\ell}_i) = \boldsymbol{v}_i$ and $\eta(\boldsymbol{\ell}_j) = \boldsymbol{v}_j$. Then

$$\mathrm{Cov}_{\widetilde{p}}(w(\boldsymbol{\ell}_i), w(\boldsymbol{\ell}_j)) = \dddot{\boldsymbol{H}}_{z\to i}(\boldsymbol{\ell}_i)\boldsymbol{C}_z\dddot{\boldsymbol{H}}_{z\to j}(\boldsymbol{\ell}_j)^\top, \tag{24}$$

where $\dddot{\boldsymbol{H}}_{z\to i}(\boldsymbol{\ell}_i) = \boldsymbol{C}(\boldsymbol{\ell}_i,S_i)\boldsymbol{C}_i^{-1}\dddot{\boldsymbol{H}}_{z\to[i]}$ and similarly for $\dddot{\boldsymbol{H}}_{z\to j}(\boldsymbol{\ell}_j)$.

B.4.2 $1 < \delta < M$

Take two nodes $\boldsymbol{v}_i,\boldsymbol{v}_j \in \boldsymbol{V}$. If $\mathrm{Pa}[\boldsymbol{v}_i]\cap\mathrm{Pa}[\boldsymbol{v}_j] \neq \emptyset$ then we apply the same logic as in B.4.3 using $\boldsymbol{v}_z = \mathrm{con}(\boldsymbol{v}_i,\boldsymbol{v}_j)$ as root. If $\mathrm{Pa}[\boldsymbol{v}_i]\cap\mathrm{Pa}[\boldsymbol{v}_j] = \emptyset$ and both nodes are at levels below M_δ then we use B.4.1. The remaining scenario is thus one in which $\boldsymbol{v}_i \in \boldsymbol{A}_r$, $r > M_\delta$ and

$\mathrm{Pa}[\boldsymbol{v}_i] \cap \mathrm{Pa}[\boldsymbol{v}_j] = \emptyset$. We take $\boldsymbol{v}_j \in \boldsymbol{A}_s$, $s < M_\delta$ for simplicity in exposition and without loss of generality. By (23)

$$\begin{aligned} \boldsymbol{w}_i &= \sum_{s=i_{M_\delta}}^{i_{r-1}} \boldsymbol{K}_s(i,s)\boldsymbol{K}_s^{-1}(s,s)\boldsymbol{e}_s + \boldsymbol{e}_i, \\ &= \sum_{s=i_{M_\delta+1}}^{i_{r-1}} \boldsymbol{K}_s(i,s)\boldsymbol{K}_s^{-1}(s,s)\boldsymbol{e}_s + \boldsymbol{C}_{ix}\boldsymbol{C}_x^{-1}\boldsymbol{w}_x + \boldsymbol{e}_i, \end{aligned} \tag{25}$$

where $\boldsymbol{v}_x \in \boldsymbol{A}_{M_\delta}$ is the parent node of $\boldsymbol{v}_i$ at level M_δ. The final result of (14) is then achieved by noting that the relevant subgraph linking $\boldsymbol{v}_x$ and $\boldsymbol{v}_j$ has depth $\delta_x = 1$ and thus $\mathrm{Cov}(\boldsymbol{w}_x, \boldsymbol{w}_j)$ can be found via B.4.1, then $\mathrm{Cov}(\boldsymbol{w}_i, \boldsymbol{w}_j) = \boldsymbol{C}_{ix}\boldsymbol{C}_x^{-1}\mathrm{Cov}(\boldsymbol{w}_x, \boldsymbol{w}_j) = \boldsymbol{F}_i\mathrm{Cov}(\boldsymbol{w}_x, \boldsymbol{w}_j)$. Notice that $\boldsymbol{F}_i$ directly uses the directed edge $\boldsymbol{v}_x \to \boldsymbol{v}_i$ in $\mathcal{G}$; for this reason the path between $\boldsymbol{w}_i$ and $\boldsymbol{w}_z = \mathrm{con}(\boldsymbol{w}_x, \boldsymbol{w}_j)$ is the actually the shortest path and we have $\boldsymbol{v}_z \to \cdots \to \boldsymbol{v}_x \longrightarrow \boldsymbol{v}_i$.

B.4.3 $\delta = M$

Take $\boldsymbol{v}_i, \boldsymbol{v}_j \in \boldsymbol{V}$ and the full paths from the root $\widetilde{\mathcal{P}}_{0\to i} = \{i_0, \ldots, i_{r_i}\}$ and $\widetilde{\mathcal{P}}_{0\to j} = \{j_0, \ldots, j_{r_j}\}$, respectively. Then using (23) we have

$$\begin{aligned} \boldsymbol{w}_i &= \sum_{s\in\widetilde{\mathcal{P}}_{0\to i}} \boldsymbol{K}_s(i,s)\boldsymbol{K}_s^{-1}(s,s)\boldsymbol{e}_s + \boldsymbol{e}_i \\ &= \sum_{s\in\widetilde{\mathcal{P}}_{0\to i}\cap\widetilde{\mathcal{P}}_{0\to j}} \boldsymbol{K}_s(i,s)\boldsymbol{K}_s^{-1}(s,s)\boldsymbol{e}_s + \sum_{s\in\widetilde{\mathcal{P}}_{0\to i}\setminus\widetilde{\mathcal{P}}_{0\to j}} \boldsymbol{K}_s(i,s)\boldsymbol{K}_s^{-1}(s,s)\boldsymbol{e}_s + \boldsymbol{e}_i \\ &= \sum_{s\in\widetilde{\mathcal{P}}_{0\to i}\cap\widetilde{\mathcal{P}}_{0\to j}} \boldsymbol{K}_s(i,s)\boldsymbol{K}_s^{-1}(s,s)\boldsymbol{e}_s + \widetilde{\boldsymbol{e}}_i \\ \boldsymbol{w}_j &= \sum_{s\in\widetilde{\mathcal{P}}_{0\to j}} \boldsymbol{K}_s(j,s)\boldsymbol{K}_s^{-1}(s,s)\boldsymbol{e}_s + \boldsymbol{e}_j \\ &= \sum_{s\in\widetilde{\mathcal{P}}_{0\to i}\cap\widetilde{\mathcal{P}}_{0\to j}} \boldsymbol{K}_s(j,s)\boldsymbol{K}_s^{-1}(s,s)\boldsymbol{e}_s + \sum_{s\in\widetilde{\mathcal{P}}_{0\to j}\setminus\widetilde{\mathcal{P}}_{0\to i}} \boldsymbol{K}_s(j,s)\boldsymbol{K}_s^{-1}(s,s)\boldsymbol{e}_s + \boldsymbol{e}_j \\ &= \sum_{s\in\widetilde{\mathcal{P}}_{0\to i}\cap\widetilde{\mathcal{P}}_{0\to j}} \boldsymbol{K}_s(j,s)\boldsymbol{K}_s^{-1}(s,s)\boldsymbol{e}_s + \widetilde{\boldsymbol{e}}_j, \end{aligned} \tag{26}$$

where $\mathrm{Cov}(\widetilde{\boldsymbol{e}}_i, \widetilde{\boldsymbol{e}}_j) = 0$. Then since $\boldsymbol{e}_s$ are independent and $\boldsymbol{e}_s \sim N(\boldsymbol{0}, \boldsymbol{K}_s(s,s))$ we find

$$\begin{aligned} \mathrm{Cov}_{\widetilde{p}}(\boldsymbol{w}_i, \boldsymbol{w}_j) = \mathrm{Cov}\Bigg(&\sum_{s\in\widetilde{\mathcal{P}}_{0\to i}\cap\widetilde{\mathcal{P}}_{0\to j}} \boldsymbol{K}_s(i,s)\boldsymbol{K}_s^{-1}(s,s)\boldsymbol{e}_s + \boldsymbol{e}_i, \\ &\sum_{s\in\widetilde{\mathcal{P}}_{0\to i}\cap\widetilde{\mathcal{P}}_{0\to j}} \boldsymbol{K}_s(j,s)\boldsymbol{K}_s^{-1}(s,s)\boldsymbol{e}_s + \boldsymbol{e}_j \Bigg) \\ = &\sum_{s\in\widetilde{\mathcal{P}}_{0\to i}\cap\widetilde{\mathcal{P}}_{0\to j}} \boldsymbol{K}_s(i,s)\boldsymbol{K}_s^{-1}(s,s)\boldsymbol{K}_s(s,j) + \mathbf{1}_{i=j}\{\boldsymbol{K}_i(i,i)\}. \end{aligned} \tag{27}$$

We conclude by noting that $\delta = M$ implies $\widetilde{\mathcal{P}}_{0\to i} \cap \widetilde{\mathcal{P}}_{0\to j} = \mathrm{Pa}[\boldsymbol{v}_i] \cap \mathrm{Pa}[\boldsymbol{v}_j]$; considering two locations $\boldsymbol{\ell}_i, \boldsymbol{\ell}_j \in \mathcal{D}^*$ such that $\eta(\boldsymbol{\ell}_i) = \boldsymbol{v}_i$ and $\eta(\boldsymbol{\ell}_j) = \boldsymbol{v}_j$ we obtain

$$\mathrm{Cov}_{\widetilde{p}}(w(\boldsymbol{\ell}_i), w(\boldsymbol{\ell}_j)) = \sum_{s:\{\boldsymbol{v}_s \in \mathrm{Pa}[\boldsymbol{v}_i] \cap \mathrm{Pa}[\boldsymbol{v}_j]\}} \boldsymbol{K}_s(\boldsymbol{\ell}_i, s)\boldsymbol{K}_s^{-1}(s,s)\boldsymbol{K}_s(s, \boldsymbol{\ell}_j) + \mathbf{1}_{\boldsymbol{\ell}_i = \boldsymbol{\ell}_j}\{\boldsymbol{K}_i(\boldsymbol{\ell}_i, \boldsymbol{\ell}_j)\}. \tag{28}$$

B.5 Computational cost

We make some assumptions here to simplify the calculation of overall cost: first, we assume that reference locations are all observed $\mathcal{S} \subset \mathcal{T}$, and consequently $\mathcal{U} = \mathcal{T} \setminus \mathcal{S}$. Second, we assume that all reference subsets have the same size i.e. $|S_i| = N_s$ for all i. Third, we assume all nodes have the same number of children at the next level in $\mathcal{G}$, i.e. if $\boldsymbol{v}_i \in \boldsymbol{A}_r$ with $r < M-1$, then $|\mathrm{Ch}[\boldsymbol{v}_i] \cap \boldsymbol{A}_{r+1}| = C$, whereas if $r = M-1$ then $|\mathrm{Ch}[\boldsymbol{v}_i]| = N_u$. Fourth, we assume that all non-reference subsets are singletons i.e. if $\boldsymbol{v}_i \in \boldsymbol{B}$ then $|U_i| = 1$. The latter two assumptions imply (5). We also fix $CN_s = N_u$. As a result, the number of nodes at level $r = 0, \ldots, M-1$ is C^r, therefore $|\boldsymbol{A}| + |\boldsymbol{B}| = \sum_{r=0}^{M-1} C^r + N_u C^{M-1} = \frac{C^M - 1}{C-1} + N_s C^M$. Then the sample size is $n = |\mathcal{T}| = |\mathcal{S}| + |\mathcal{U}| = N_s \frac{C^{M+1}-1}{C-1}$ hence $M \approx \log_C(n/N_s)$. Starting with $\delta = M$, the parent set sizes J_i for a node $\boldsymbol{v}_i \in \boldsymbol{A}_r$ grow with r as $J_i = rN_s$ and if $\boldsymbol{v}_i \in \boldsymbol{B}$ then $J_i = MN_s$. The cost of computing $p(\boldsymbol{w} \,|\, \boldsymbol{\theta})$ is driven by the calculation of $\boldsymbol{H}_j$, which is $O(r^2 N_s^3)$ for reference nodes at level r, for a total of $O(N_s^3 \sum_{r=0}^{M-1} C^r r^2)$. Since for common choices of C and M we have $\sum_{r=0}^{M-1} C^r r^2 N_s^3 \leq \sum_{r=0}^{M-1} C^{2r} N_s^3 = \frac{C^{2M}-1}{C^2-1} N_s^3 \approx C^M N_s^3 \approx \frac{n}{N_s} N_s^3 = n N_s^2$ then the cost for reference sets is $O(nN_s^2)$. Analogously for non reference nodes we get $O(C^M M^2 N_s^3)$ which leads to a cost of $O(nN_s^2)$. The cost of sampling $\boldsymbol{w}$ is mainly driven by the computation of the Cholesky factor of a $N_s \times N_s$ matrix at each of $\frac{C^M-1}{C-1}$ reference nodes, which amounts to $O(nN_s^2)$. For the $N_s C^M$ non-reference nodes the main cost is in computing $\boldsymbol{H}_i \boldsymbol{w}_{[i]}$ which is MN_s for overall cost $O(C^M M N_s^2)$ which is smaller than $O(nN_s^2)$. Obtaining $\boldsymbol{F}_i^{(c)}$ at the root of $\mathcal{G}$ is associated to a cost $O(N_s^2 \frac{C^M - C}{C-1})$ which is $O(nN_s)$ but constitutes a bottleneck if such operation is performed simultaneously to sampling; however this bottleneck is eliminated in Algorithm 3.

If $\delta = 1$ then the parent set sizes J_i for all nodes $\boldsymbol{v}_i \in \boldsymbol{V}$ are constant $J_i = N_s$; since the nodes at levels 0 to $M-1$ have C children, the asymptotic cost of computing $p(\boldsymbol{w} \,|\, \boldsymbol{\theta})$ is $O(N_s^3 \sum_{r=0}^{M-1} C^r) = O(N_s^3 \frac{C^M - 1}{C-1}) = O(nN_s^2)$. However there are savings of approximately a factor of M associated to $\delta = 1$ in fixed samples since $\sum_{r=1}^{M-1} C^r r^2 > \sum_{r=1}^{M-1} C^r r > \frac{MC^M - 1}{C-1} - \frac{C^{M+1}}{(C-1)^2} > \frac{MC^M - 1}{C-1} > M \sum_{r=0}^{M-1} C^r$. Fixing C and M one can thus choose larger N_s and smaller δ, or vice-versa.

The storage requirements are driven by the covariance at parent locations $\boldsymbol{C}_{[j]}$ for nodes $\boldsymbol{v}_j$ with $\mathrm{Pa}[\boldsymbol{v}_j] \neq \emptyset$ i.e. all reference nodes at level $r = 1, \ldots, M-1$ and non-reference nodes. Taking $\delta = M$, suppose $\boldsymbol{v}_i$ is the last parent of $\boldsymbol{v}_j$, meaning $\boldsymbol{v}_i \cup \mathrm{Pa}[\boldsymbol{v}_i] = \mathrm{Pa}[\boldsymbol{v}_j]$. Then $\boldsymbol{C}_{[j]} = \boldsymbol{C}(\{S_i, S_{[i]}\}, \{S_i, S_{[i]}\})$. If $\boldsymbol{v}_i \in \boldsymbol{A}_r$ then these matrices are of size $(r+1)N_s \times (r+1)N_s$; each of these is thus $O(r^2 N_s^2)$ in terms of storage. Considering all such matrices brings the overall storage requirement to $O(\sum_{r=0}^{M-1} C^r r^2 N_s^2)$ which is $O(nN_s)$ using analogous arguments as above. For $\delta = 1$ we apply similar calculations as above. The same number of $\boldsymbol{H}_j$ and $\boldsymbol{R}_j$ must be stored but these are smaller in size and therefore do not affect the overall storage

requirements. The design matrix $\boldsymbol{Z}$ is stored in blocks and never as a large (sparse) matrix implying a storage requirement of $O(nq)$.

Appendix C. Implementation details

Building a SpamTree DAG proceeds by first constructing a base-tree $\mathcal{G}_1$ at depth $\delta = 1$ and then adding edges to achieve the desired depth level. The base tree $\mathcal{G}_1$ is built from the root by branching each node $\boldsymbol{v}$ into $|\text{Ch}[\boldsymbol{v}]| = c^d$ children where d is the dimension of the spatial domain and c is a small integer. The spatial domain $\mathcal{D}$ is partitioned recursively; after setting $\mathcal{D}$, each recursive step proceeds by partitioning each coordinate axis of $D_i \subset \mathcal{D}$ into c intervals. As a consequence $D_i = \cup_j D_{ij}$ and $D_{ij} \cap D_{ij'} = \emptyset$ if $j \neq j'$. This recursive partitioning scheme is used to partition the reference set $\mathcal{S}$ which we consider as a subset of the observed locations. Suppose we wish to associate node $\boldsymbol{v}$ to approximately n_S locations where $n_S = k^d$ for some k. Start from the root i.e. $\boldsymbol{v} \in \boldsymbol{A}_0$. Then take $\mathcal{S}_0 = \mathcal{S}$ and partition it via parallel partitioning of each coordinate axis into k intervals. Collect 1 location from each subregion to build S_0. Then set $\eta(S_0) = \boldsymbol{v}_0$ and $\mathcal{S}_1 = \mathcal{S} \setminus \mathcal{S}_0$. Then, take $\{D_{1j}\}_j$ such that $\cup_j D_{1j} = D_0 = \mathcal{D}$. We find S_{1j} via axis-parallel partitioning of $\mathcal{S}_1 \cap D_{1j}$ into k^d regions and selecting one location from each partition, as above, and setting $\mathcal{S}_2 = \mathcal{S} \setminus \{\mathcal{S}_0 \cup \mathcal{S}_1\}$. All other reference subsets are found by sequentially removing locations from the reference set, and proceeding analogously as above. This stepwise procedure is stopped when either the tree reaches a predetermined height M, or when there is an insufficient number of remaining locations to build reference subsets of size n_S. The remaining locations are assigned to the leaf nodes via η_B as defined in Section 2.1 in order to include at least one neighboring realization of the process from the same variable.

One specific issue arises when multivariate data are imbalanced, i.e. one of the margins is observed at a much sparser grid, e.g. in Section 4.2 `PRCP` is collected at a ratio of 1:10 locations compared to other variables. In these cases, if locations were chosen uniformly at random to build the reference subsets then the root nodes would be associated via η to reference subsets which likely do not contain such sparsely observed variables. This scenario goes against the intuition of 1 suggesting that a naïve approach would result in poor performance at the sparsely-observed margins. To avoid such a scenario, we bias the sampling of locations to favor those at which the sparsely-observed variables are recorded. As a result, in Section 4.2 near-root nodes are associated to reference subsets in which all variables are balanced; the imbalances of the data are reflected by imbalanced leaf nodes instead.

The source code for SpamTrees is available at `https://CRAN.R-project.org/package=spamtree` and can be installed as an `R` package. The `spamtree` package is written in C++ using the Armadillo library for linear algebra (Sanderson and Curtin, 2016) interfaced to `R` via `RcppArmadillo` (Eddelbuettel and Sanderson, 2014). All matrix operations are performed efficiently by linkage to the LAPACK and BLAS libraries (Blackford et al., 2002; Anderson et al., 1999) as implemented in OpenBLAS 0.3.10 (Zhang, 2020) or the Intel Math Kernel Library. Multithreaded operations proceed via OpenMP (Dagum and Menon, 1998).

C.1 On the dependencies on sparse Cholesky libraries

SpamTrees do not require the use of external Cholesky libraries because Cholesky-like algorithms can be written explicitly by referring to the treed DAG and its edges. This is unusual for DAG-based models commonly used in geostatistical settings. For example, an NNGP model uses neighbors to build a DAG. The DAG can be used to fill the L and D matrices leading to $LDL^\top = C^{-1}$, where C^{-1} is the sparse precision matrix of the latent process. When one then adds measurement error in a regression setting and marginalizes out the latent process, the goal is to find the Cholesky decomposition of $C^{-1} + \tau^2 I_n$. The original NNGP DAG used for L and D is not useful for this purpose – hence the need for NNGP to use sparse Cholesky libraries in collapsed samplers (Finley et al., 2019). On the other hand, with SpamTrees we can still look at the original DAG to "update" L and D by using Algorithm 4. We included it in the Appendix as our software package implements the Gibbs sampler in the main article, which does not involve Cholesky decompositions of large sparse precision matrices.

Furthermore, one of the initial steps in Cholesky algorithms for sparse symmetric positive-definite matrices involves finding "good" reordering rows and columns. These reorderings simplify the (undirected) graphical model that corresponds to the sparsity structure in the matrix. Once a simple-enough graphical model is found heuristically, it is used for the decomposition. On the other hand, the sparse Cholesky algorithm for SpamTrees can be written explicitly using the underlying DAG, without any intermediate step, because it is fixed and with a convenient treed structure. It might be possible to write software that outperforms excellent libraries such as `CHOLMOD` (Chen et al., 2008) at decomposing the matrices needed in collapsed sampling algorithms for SpamTrees.

Regarding a more general perspective about dependencies on well established software libraries, we should clarify that the software for SpamTree *does* take advantage of highly efficient libraries such as BLAS/LAPACK provided in Intel MKL 2019.5, OpenMP (Dagum and Menon, 1998) for parallelizing the algorithms as described in the main article. These libraries are optional and our code does not strictly depend on them. For example, noting that it is considerably more difficult to compile OpenMP code on Macs, one can just disable OpenMP and let the Accelerate BLAS (rather than OpenBLAS or Intel MKL) deal with all matrix algebra for Apple computers, at the cost of some performance in big data settings. Our code also has R package dependencies for data pre-processing, but these are peripheral to the proposed methods and algorithms. Any improvement in the upstream libraries we used for coding `spamtree` will positively impact the performance of our software.

C.2 Applications

C.2.1 Simulated datasets

SpamTrees with full depth are implemented by targeting reference subsets of size $n_S = 25$ and tress with $c = 4$ additional children for each branch. The tree is built starting from a 2×2 partition of the domain, hence there are 4 root nodes with no parents in the DAG. The cherry-pickying function η is set as in Section 2.1; with these settings the tree height is $M = 3$. For SpamTrees with depth $\delta = 1$ we build the tree with reference subsets of size $n_S = 80$ and $c = 4$. Multivariate Q-MGPs are implemented via axis-parallel partitioning using 57 intervals along each axis. The multivariate SPDE-INLA method was implemented

i	σ_{i1}	σ_{i2}	ϕ_i
LST_Day_CMG	-0.8936 $-0.9499, -0.8401$	8.3285 $7.8071, 8.9614$	0.2174 $0.1854, 0.2460$
LST_Night_CMG	-1.4104 $-1.4794, -1.3530$	7.3927 $7.0730, 7.7230$	0.0968 $0.0883, 0.1054$
Clear_sky_days	0.9189 $0.8695, 0.9708$	3.3133 $3.3033, 3.4523$	0.5790 $0.5303, 0.6185$
Clear_sky_nights	3.8138 $3.7138, 3.9306$	0.9603 $0.9194, 1.0114$	6.2129 $5.7944, 6.6519$
PRCP	-0.3009 $-0.3348, -0.2702$	0.6897 $0.6466, 0.7200$	0.1832 $0.1655, 0.2051$

α
0.1012 $0.0696, 0.1248$
β
0.1654 $0.1258, 0.2203$
ϕ
0.5715 $0.5326, 0.6079$

δ_{ij}	LST_Day_CMG	LST_Night_CMG	Clear_sky_days	Clear_sky_nights
LST_Night_CMG	0.1279 $0.0608, 0.2328$			
Clear_sky_days	1.7295 $1.6639, 1.7962$	1.5371 $1.3765, 1.7059$		
Clear_sky_nights	0.0307 $0.0221, 0.0395$	1.1156 $0.8964, 1.3194$	1.5035 $1.2670, 1.7380$	
PRCP	0.2436 $0.2039, 0.2878$	1.3151 $0.9149, 1.7000$	0.0572 $0.0490, 0.0643$	0.7677 $0.4010, 1.1468$

Figure 8: Posterior means and 95% credible intervals for components of $\boldsymbol{\theta}$ for SpamTrees.

following the examples in Krainski et al. (2019), Chapter 3, setting the grid size to 15×15 to limit the compute time to 15 seconds when using 10 CPU threads. BART was implemented on each dataset via the `wbart` function in the R package `BART`; the set of covariates for BART was built using the spatial coordinates in addition to a binary variable representing the output variable index (i.e. taking value 1 whenever y_i is of the first outcome variable, 0 otherwise).

C.2.2 MODIS-TERRA and GHCN

The implemented SpamTrees are built with 36 root nodes and $c = 6$ additional children for each level of the tree, 25 reference locations for each tree node, and for up to $M = 5$ levels of the tree and $\delta = 5$ (i.e. full depth). The non-reference observed locations are linked to leaves via cherry-pickying as in Section 2.1. Multivariate models were run on an AMD Epyc 7452-based virtual machine with 256GB of memory in the Microsoft Azure cloud; the SpamTree R package was set to run on 20 CPU threads, on R version 4.0.3 linked to the Intel Math Kernel Library (MKL) version 2019.5-075. The univariate models were run on an AMD Ryzen 5950X-based dedicated server with 128GB of memory, on 16 threads, R version 4.1.1 linked to Intel MKL 2019.5-075. The univariate NNGP model was implemented using R package `spNNGP` (Finley et al., 2020) using 20 neighbors for all outcomes and a "latent" algorithm. The univariate SpamTree was implemented on each outcome with full depth, $c = 16$ additional chidren for each level of the tree, and 25 reference locations for each node.

The MGP model was implemented via the development package at `github.com/mkln/meshgp` targeting a block size with 4 spatial locations, resulting in an effective average block dimension of 20. Caching was unavailable due to the irregularly spaced `PRCP` values. Fewer MCMC iterations were run compared to SpamTrees to limit total runtime to less than 16h.

References

S. Ambikasaran, D. Foreman-Mackey, L. Greengard, D. W. Hogg, and M. O'Neil. Fast direct methods for Gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(2):252–265, 2016. doi:10.1109/TPAMI.2015.2448083.

E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users' Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, third edition, 1999.

T. V. Apanasovich and M. G. Genton. Cross-covariance functions for multivariate random fields based on latent dimensions. *Biometrika*, 97:15–30, 2010. doi:10.1093/biomet/asp078.

S. Banerjee. High-dimensional Bayesian geostatistics. *Bayesian Analysis*, 12(2):583–614, 2017. doi:10.1214/17-BA1056R.

S. Banerjee. Modeling Massive Spatial Datasets Using a Conjugate Bayesian Linear Modeling Framework. *Spatial Statistics*, in press, 2020. doi:10.1016/j.spasta.2020.100417.

S. Banerjee, A. E. Gelfand, A. O. Finley, and H. Sang. Gaussian predictive process models for large spatial data sets. *Journal of the Royal Statistical Society, Series B*, 70:825–848, 2008. doi:10.1111/j.1467-9868.2008.00663.x.

S. Banerjee, A. O. Finley, P. Waldmann, and T. Ericsson. Hierarchical spatial process models for multiple traits in large genetic trials. *Journal of American Statistical Association*, 105(490): 506–521, 2010. doi:10.1198/jasa.2009.ap09068.

L. S. Blackford, A. Petitet, R. Pozo, K. Remington, R. C. Whaley, J. Demmel, J. Dongarra, I. Duff, S. Hammarling, G. Henry, et al. An updated set of basic linear algebra subprograms (BLAS). *ACM Transactions on Mathematical Software*, 28(2):135–151, 2002.

Y. Chen, T. A. Davis, W. W. Hager, and S. Rajamanickam. Algorithm 887: CHOLMOD, Supernodal Sparse Cholesky Factorization and Update/Downdate. *ACM Trans. Math. Softw.*, 35(3), 2008. doi:10.1145/1391989.1391995.

H. A. Chipman, E. I. George, and R. E. McCulloch. BART: Bayesian additive regression trees. *Annals of Applied Statistics*, 4(1):266–298, 2010. doi:10.1214/09-AOAS285.

T. M. Cover and J. A. Thomas. *Elements of information theory*. Wiley Series in Telecommunications and Signal Processing. Wiley Interscience, 1991.

N. Cressie and G. Johannesson. Fixed Rank Kriging for Very Large Spatial Data Sets. *Journal of the Royal Statistical Society, Series B*, 70:209–226, 2008. doi:10.1111/j.1467-9868.2007.00633.x.

L. Dagum and R. Menon. OpenMP: an industry standard api for shared-memory programming. *Computational Science & Engineering, IEEE*, 5(1):46–55, 1998.

A. Datta, S. Banerjee, A. O. Finley, and A. E. Gelfand. Hierarchical nearest-neighbor Gaussian process models for large geostatistical datasets. *Journal of the American Statistical Association*, 111:800–812, 2016a. doi:10.1080/01621459.2015.1044091.

A. Datta, S. Banerjee, A. O. Finley, N. A. S. Hamm, and M. Schaap. Nonseparable dynamic nearest neighbor gaussian process models for large spatio-temporal data with an application to particulate matter analysis. *The Annals of Applied Statistics*, 10:1286–1316, 2016b. doi:10.1214/16-AOAS931.

D. Eddelbuettel and C. Sanderson. RcppArmadillo: Accelerating R with high-performance C++ linear algebra. *Computational Statistics and Data Analysis*, 71:1054–1063, March 2014. doi:10.1016/j.csda.2013.02.005.

J. Eidsvik, B. A. Shaby, B. J. Reich, M. Wheeler, and J. Niemi. Estimation and prediction in spatial models with block composite likelihoods. *Journal of Computational and Graphical Statistics*, 23: 295–315, 2014. doi:10.1080/10618600.2012.760460.

M. A. Ferreira and H. K. Lee. *Multiscale Modeling: A Bayesian Perspective*. Springer Publishing Company, Incorporated, 1st edition, 2007. ISBN 0387708979, 9780387708973.

A. O. Finley, A. Datta, B. D. Cook, D. C. Morton, H. E. Andersen, and S. Banerjee. Efficient Algorithms for Bayesian Nearest Neighbor Gaussian Processes. *Journal of Computational and Graphical Statistics*, 28:401–414, 2019. doi:10.1080/10618600.2018.1537924.

A. O. Finley, A. Datta, and S. Banerjee. R package for Nearest Neighbor Gaussian Process models. 2020. arXiv:2001.09111.

E. B. Fox and D. B. Dunson. Multiresolution Gaussian processes. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, page 737–745, Red Hook, NY, USA, 2012. Curran Associates Inc. https://dl.acm.org/doi/10.5555/2999134.2999217.

R. Furrer, M. G. Genton, and D. Nychka. Covariance Tapering for Interpolation of Large Spatial Datasets. *Journal of Computational and Graphical Statistics*, 15:502–523, 2006. doi:10.1198/106186006X132178.

A. Gelfand, P. Diggle, M. Fuentes, , and P. Guttorp. *Handbook of Spatial Statistics*. CRC Press, Boca Raton, FL, 2010.

M. G. Genton and W. Kleiber. Cross-Covariance Functions for Multivariate Geostatistics. *Statistical Science*, 30:147–163, 2015. doi:10.1214/14-STS487.

C. J. Geoga, M. Anitescu, and M. L. Stein. Scalable Gaussian process computations using hierarchical matrices. *Journal of Computational and Graphical Statistics*, 29:227–237, 2020. doi:10.1080/10618600.2019.1652616.

E. Gilboa, Y. Saatçi, and J. P. Cunningham. Scaling multidimensional inference for structured gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(2): 424–436, 2015. doi:10.1109/TPAMI.2013.192.

R. B. Gramacy and D. W. Apley. Local Gaussian Process Approximation for Large Computer Experiments. *Journal of Computational and Graphical Statistics*, 24:561–578, 2015. doi:10.1080/10618600.2018.1537924.

R. B. Gramacy and H. K. H. Lee. Bayesian Treed Gaussian Process Models With an Application to Computer Modeling. *Journal of the American Statistical Association*, 103:1119–1130, 2008. doi:10.1198/016214508000000689.

R. Guhaniyogi, A. O. Finley, S. Banerjee, and A. E. Gelfand. Adaptive Gaussian predictive process models for large spatial datasets. *Environmetrics*, 22:997–1007, 2011. doi:10.1002/env.1131.

J. Guinness. Permutation and grouping methods for sharpening gaussian process approximations. *Technometrics*, 60(4):415–429, 2018. doi:10.1080/00401706.2018.1437476.

M. J. Heaton, A. Datta, A. O. Finley, R. Furrer, J. Guinness, R. Guhaniyogi, F. Gerber, R. B. Gramacy, D. Hammerling, M. Katzfuss, F. Lindgren, D. W. Nychka, F. Sun, and A. Zammit-Mangion. A case study competition among methods for analyzing large spatial data. *Journal of Agricultural, Biological and Environmental Statistics*, 24(3):398–425, Sep 2019. doi:10.1007/s13253-018-00348-w.

H. Huang and Y. Sun. Hierarchical low rank approximation of likelihoods for large spatial datasets. *Journal of Computational and Graphical Statistics*, 27(1):110–118, 2018. doi:10.1080/10618600.2017.1356324.

M. Jurek and M. Katzfuss. Hierarchical sparse cholesky decomposition with applications to high-dimensional spatio-temporal filtering, 2020. arXiv:2006.16901.

M. Katzfuss. A multi-resolution approximation for massive spatial datasets. *Journal of the American Statistical Association*, 112:201–214, 2017. doi:10.1080/01621459.2015.1123632.

M. Katzfuss and W. Gong. A class of multi-resolution approximations for large spatial datasets. *Statistica Sinica*, 30:2203–2226, 2019. doi:10.5705/ss.202018.0285.

M. Katzfuss and J. Guinness. A general framework for Vecchia approximations of Gaussian processes. *Statistical Science*, 36(1):124–141, 2021. doi:10.1214/19-STS755.

C. G. Kaufman, M. J. Schervish, and D. W. Nychka. Covariance Tapering for Likelihood-Based Estimation in Large Spatial Data Sets. *Journal of the American Statistical Association*, 103: 1545–1555, 2008. doi:10.1198/016214508000000959.

E. T. Krainski, V. Gómez-Rubio, H. Bakka, A. Lenzi, D. Castro-Camilo, D. Simpson, F. Lindgren, and H. Rue. *Advanced Spatial Modeling with Stochastic Partial Differential Equations Using R and INLA*. CRC Press/Taylor and Francis Group, 2019.

A. Krause, A. Singh, and C. Guestrin. Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research*, 8: 235–284, 2008. http://www.jmlr.org/papers/v9/krause08a.html.

L. Lauritzen, S. *Graphical Models*. Clarendon Press, Oxford, UK, 1996.

R. Lewis. *A guide to graph colouring*. Springer International Publishing, 2016. doi:10.1007/978-3-319-25730-3.

F. Lindgren, H. Rue, and J. Lindström. An explicit link between Gaussian fields and Gaussian Markov random fields: the stochastic partial differential equation approach. *Journal of the Royal Statistical Society: Series B*, 73:423–498, 2011. doi:10.1111/j.1467-9868.2011.00777.x.

J. Loper, D. Blei, J. P. Cunningham, and L. Paninski. General linear-time inference for Gaussian processes on one dimension, 2020. arXiv:2003.05554.

K. H. Low, J. Yu, J. Chen, and P. Jaillet. Parallel Gaussian process regression for big data: Low-rank representation meets Markov approximation. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, page 2821–2827, 2015. http://hdl.handle.net/1721.1/116273.

M. Molloy and B. Reed. *Graph colouring and the probabilistic method*. Springer-Verlag Berlin Heidelberg, 2002. doi:10.1007/978-3-642-04016-0.

K. R. Moran and M. W. Wheeler. Fast increased fidelity approximate Gibbs samplers for Bayesian Gaussian process regression, 2020. arXiv:2006.06537.

D. Nychka, S. Bandyopadhyay, D. Hammerling, F. Lindgren, and S. Sain. A multiresolution gaussian process model for the analysis of large spatial datasets. *Journal of Computational and Graphical Statistics*, 24:579–599, 2015. doi:10.1080/10618600.2014.914946.

M. Peruzzi, S. Banerjee, and A. O. Finley. Highly scalable Bayesian geostatistical modeling via meshed Gaussian processes on partitioned domains. *Journal of the American Statistical Association*, 2020. in press. doi:10.1080/01621459.2020.1833889.

Z. C. Quiroz, M. O. Prates, and D. K. Dey. Block Nearest Neighboor Gaussian processes for large datasets, 2019. arXiv:1604.08403.

J. Quiñonero-Candela and C. E. Rasmussen. A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research*, 6:1939–1959, 2005. https://www.jmlr.org/papers/volume6/quinonero-candela05a/quinonero-candela05a.pdf.

H. Rue and L. Held. *Gaussian Markov Random Fields: Theory and Applications*. Chapman & Hall/CRC, 2005. doi:10.1007/978-3-642-20192-9.

H. Rue, S. Martino, and N. Chopin. Approximate Bayesian inference for latent Gaussian models by using integrated nested Laplace approximations. *Journal of the Royal Statistical Society: Series B*, 71:319–392, 2009. doi:10.1111/j.1467-9868.2008.00700.x.

C. Sanderson and R. Curtin. Armadillo: a template-based C++ library for linear algebra. *Journal of Open Source Software*, 1:26, 2016.

H. Sang and J. Z. Huang. A full scale approximation of covariance functions for large spatial data sets. *Journal of the Royal Statistical Society, Series B*, 74:111–132, 2012. doi:10.1111/j.1467-9868.2011.01007.x.

E. Snelson and Z. Ghahramani. Local and global sparse Gaussian process approximations. In *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics*, volume 2 of *Proceedings of Machine Learning Research*, pages 524–531, 2007. http://proceedings.mlr.press/v2/snelson07a.html.

M. L. Stein. Limitations on low rank approximations for covariance matrices of spatial data. *Spatial Statistics*, 8:1–19, 2014. doi:doi:10.1016/j.spasta.2013.06.003.

M. L. Stein, Z. Chi, and L. J. Welty. Approximating likelihoods for large spatial data sets. *Journal of the Royal Statistical Society, Series B*, 66:275–296, 2004. doi:10.1046/j.1369-7412.2003.05512.x.

Y. Sun, B. Li, and M. Genton. Geostatistics for large datasets. In J. Montero, E. Porcu, and M. Schlather, editors, *Advances and Challenges in Space-time Modelling of Natural Events*, pages 55–77. Springer-Verlag, Berlin Heidelberg, 2011. doi:10.1007/978-3-642-17086-7.

A. V. Vecchia. Estimation and model identification for continuous spatial processes. *Journal of the Royal Statistical Society, Series B*, 50:297–312, 1988. doi:10.1111/j.2517-6161.1988.tb01729.x.

M. Vihola. Robust adaptive Metropolis algorithm with coerced acceptance rate. *Statistics and Computing*, 22:997–1008, 2012. doi:10.1007/s11222-011-9269-5.

L. Wu, A. Miller, L. Anderson, G. Pleiss, D. Blei, and J. Cunningham. Hierarchical inducing point Gaussian process for inter-domain observations. In *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2021. arXiv:2103.00393.

X. Zhang. *An Optimized BLAS Library Based on GotoBLAS2.*, 2020. URL https://github.com/xianyi/OpenBLAS/.